Information dissemination based on the en-passant communication pattern *

Daniel Görgen, Hannes Frey and Christian Hutter

University of Trier
Department of Computer Science
54286 Trier, Germany
E-mail: {goergen|frey|hutter}@syssoft.uni-trier.de

Abstract This work presents a communication pattern for high mobile ad hoc networks. En-passant communication uses the short interaction period of passing devices to efficiently synchronize the information of each device. This is achieved by creating peer-to-peer overlays of interest domains. Missing information is determined by exchanging profiles first. As example application `UbiQuiz` is presented, a mobile quiz application. It exchanges questions using the en-passant communication pattern. `UbiQuiz` has been implemented, tested and evaluated within a simulation and on real devices.

1 Introduction

Nowadays, a large variety of more and more powerful mobile devices such as Pocket PCs, PDAs, and smart-phones is available. Since most of these devices are equipped with wireless communication adapters like IEEE 802.11 or Bluetooth, it is reasonable to use these for communication with nearby devices. Communication in ad hoc networks can be classified in single hop communication, where devices communicate with neighboring devices only, and multi hop networks, where messages can pass several hops forwarded altruistically by others.

While it is possible to realize an end-to-end communication over a few hops in dense ad hoc networks by using ad hoc routing mechanisms such as topology based [11] or geographic [5] routing, end-to-end communication might fail when network density decreases and network size and mobility increases. In such sparse ad hoc networks infection-based mechanisms work well but degenerate to flooding when the network density increases and may lead to the well known broadcast storm problem [10]. Thus, message exchanging must be reduced to efficient broadcasting mechanisms, where the broadcasting property of wireless networks is utilized to reduce message forwards and not every message is forwarded to every other device. The most critical problem is to decide which message has

* This work is funded in part by DFG, Schwerpunktprogramm SPP1140 "Basissoftware für selbstorganisierende Infrastrukturen für vernetzte mobile Systeme", Microsoft Research Embedded Systems IFP (Contract 2003-210) and the Luxembourg Ministère de la Culture, de l'Enseignement Supérieur et de la Recherche.

to be forwarded to which device. Thus, the devices have to determine which information is of interest and which is already known to the communicating peer. Classifying devices by domains of interest is achieved by single hop peer-to-peer overlays, where only devices within the same overlay are detected via beaconing and are addressed by a local communication mechanism. The amount of data transferred can be further reduced by exchanging information profiles first. These profiles contain application specific information descriptions and all IDs of known information fitting to this profile. Thus only new information must be sent. This mobility driven information synchronization is termed *en-passant* communication.

Much work has been done in the area of multi hop ad hoc networks in recent years, but only a few real world applications have been implemented. One reason for this is that implementation and testing causes still a high effort since a critical mass of participating devices and test persons are needed. Starting the development with simulations and emulations can reduce the testing overhead and field trials can be reduced to a proof of concept only.

The `UbiQuiz` example is a quiz application helping students preparing for their exams. Questions are shared and exchanged with neighboring devices and disseminated within the ad hoc network. It is very probable, that en-passant communication can be used today in this field, as no connected multi hop ad hoc network is needed. Moreover, creating interest overlays perfectly fits to students behavior, as it is common that they meet fellow-students currently studying for the same exam.

This work is organized as follows: The next Section describes the `UbiQuiz` application and all its parts in detail. It starts with a short application overview, the description of peer-to-peer overlays and information synchronization with neighboring devices. This is followed by a discussion of the applications gaming part and the implementation issues starting with simulation and ending with the real application prototype. The field-trials and the results are presented in Section 3. Section 4 gives a short overview of the related work and finally Section 5 concludes this paper and gives an outlook to future work.

2 The UbiQuiz Application

`UbiQuiz` is a simple quiz game application in the manner of "Who wants to be a millionaire". The user has to answer questions with increasing difficulty by choosing one out of four possible answers. To get help with difficult questions, he is able to use jokers: Discard 50% of the answers keeping the correct one, call a person for help or ask the audience and display a statistic of the results.

`UbiQuiz` is mainly intended to help students preparing their exams. Therefore, it is possible to define different question categories, each covering one learning topic of different subjects, e.g. a lecture on distributed systems in computer science. Students and teachers are able to define own questions and categories in order to create a large and useful question pool. Devices running `UbiQuiz` are able to exchange questions using the en-passant communication pattern. `UbiQuiz` maps overlays to question categories. To reduce network load, only de-

vices interested in the same question categories will try to synchronize their question catalogs. This behavior is realized by creating ad hoc peer-to-peer overlays, where only devices interested in the same category are detected and addressed by local communication mechanisms. During the synchronization of question pools, the application aims at efficiently exchanging the missing questions causing a minimum of network load. This is achieved by only sending profiles containing a description of the needed subset and IDs of all known questions fitting to this subset.

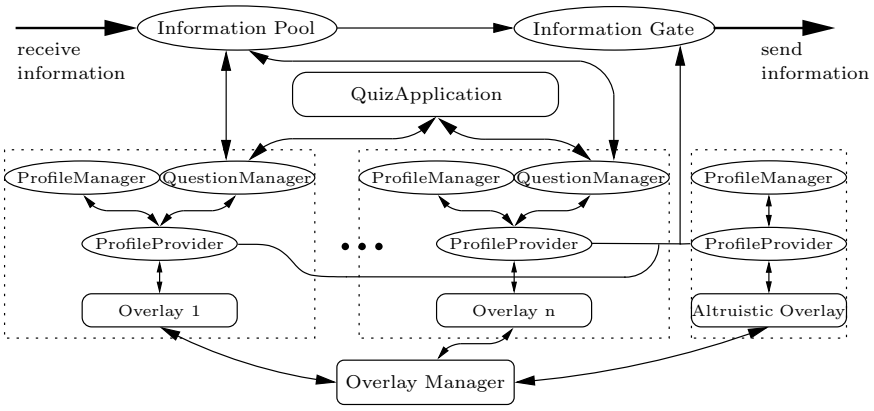

Figure 1. `UbiQuiz` P2P communication architecture overview

Figure 1 depicts a simplified overview of the application parts needed for the peer-to-peer ad hoc communication. Each device has one *InformationPool* which stores all known questions and one *InformationGate*, which delivers questions to interested neighboring devices. All received or newly created questions are directly passed to the InformationPool. For each question category the user is interested in, one overlay is created. All overlays are managed by the *OverlayManager*. On top of each overlay one *ProfileProvider* is used to send the user profile and all fitting question hashes to devices entering the overlay. The *ProfileManager* manages all known profiles of current neighbors. The *QuestionManager* provides questions to the quiz application and keeps track of all unanswered questions.

To increase the probability that newly created questions are disseminated within the network, each device also stores a fixed amount of questions from other categories altruistically and exchanges them with other devices in an *Altruistic Overlay* as in other overlays but with a lower priority. All devices running `UbiQuiz` are part of this overlay and also need a ProfileProvider and a ProfileManager for it.

2.1 Information Pool

All received and user created information is stored within the InformationPool. Other application parts are able to register handlers to find out about newly received, created or deleted information. Thus, the application is able to wait

for questions of a specific type, e.g. the question category and question difficulty. Additionally, all ProfileProviders are interested in new information in order to send it to interested neighbors. Each information item is addressed with an ID, containing a hash value generated when it is created and a creation timestamp. The hashes are currently generated using an MD5 hash function and are used in the same manner as in the rsync protocol [13], where the hash values are exchanged first to determine the missing information on each site.

2.2 Peer-to-peer Overlay Management

A single hop peer-to-peer overlay (see Figure 2) enables the application to find out about devices within the same overlay leaving or entering the communication range. Moreover, it enables the application to send unicasts, multicasts and broadcasts to devices within the overlay only. Thus, only devices which are possibly interested to communicate with each other are detected and are addressed by multicasts and broadcasts. As `UbiQuiz` overlays are mapped to question categories, questions are only shared with devices interested in the same question category.

The overlay management uses a periodic adaptive beaconing to broadcast the IDs of all overlays the device currently participates in. The beaconing interval is increased when the number of devices in the direct neighborhood increases. They are counted using the incoming beacon messages. This avoids, that too many beaconing messages are sent within dense networks and too much network bandwidth is used for this service. In sparse networks a smaller beaconing interval is needed to detect single passing devices much earlier. Applications can enter a specific overlay, thus being able to receive enter and leave events of other devices and receive multicast and broadcast messages of an overlay neighbor.

Broadcasts from other overlays are ignored, so that applications only have to consider messages which are of interest to them. Overlay multicasts are used to benefit from the broadcast capability of the network to send one message to all or a subset of all devices within the same local overlay. To increase reliability, in contrast to broadcasts, multicast messages are acknowledged by the receivers. This is achieved by adding all receiver addresses to the multicast message header. This information is also available to the receiving application, so that it can determine the other receivers.

2.3 Profile Based Information Dissemination

One of the most challenging problems when dealing with applications for mobile ad hoc networks is the goal of making all relevant information available to devices interested in them. Thus, it is essential to find efficient strategies for information dissemination.

`UbiQuiz` uses a profile-based approach, where each device determines a description for the subset of information it is interested in. In `UbiQuiz` these profiles currently only distinguish between different question difficulties and creation times, but can be extended easily. Thus, the application is able to request questions of specific difficulties where not enough unanswered questions are left and

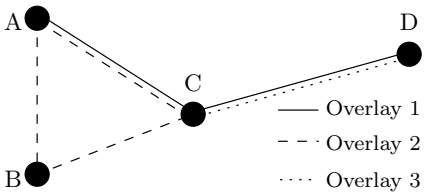

Figure 2. Devices A, B and C share overlay 2 and are able to communicate. A, C and D are in overlay 1, but A cannot address D directly. C and D are in overlay 3.

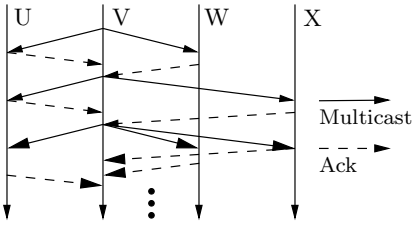

Figure 3. Device V sends sequentially information items to U,W and X. The next item is sent, after all receivers ACKs the item. The first item is multicasted to U and W, the second to U and X, etc.

request all new questions within a specific time delta. Due to the fact that this profile exchange sits on top of an overlay the profiles must only be exchanged with devices within the same question category. To ensure that the receiving devices only send information which is unknown to the profile sending devices, the profile message also contains a set of information keys fitting to the profile. With that, the receiving device can easily determine the set of information to be sent by calculating the difference between all locally known information fitting to the profile and all locally known keys of the other device. Profiles are exchanged by the ProfileProvider whenever another device enters the local overlay. When a profile changes, the device sends a profile update via broadcast to all neighboring devices. Only the difference between the last sent key set and the key set fitting to the new profile has to be included since all devices store the last profile and information keys within their local ProfileManager.

Another reason for storing profile information is that the application can send newly created or received information to interested devices when the local key set does not contain the new key and the information fits to the devices profile.

`UbiQuiz` has been designed to run in cooperative, mobile ad hoc networks. Therefore it features support for general information dissemination realized with an altruistic overlay. The user sacrifices some of his resources like storage capacities, CPU power and communication bandwidth to disseminate data he is not interested in.

The altruistic overlay will also be synchronized with other devices but its internal organization and the synchronization strategy differ from the ones applied to normal overlays. In contrast to the "unlimited" amount of local information stored for them, the size of the cache for the altruistic overlay is restricted and might be changed by the user during application runtime. Consequently the amount of information transmitted for this overlay is in the worst case limited by the size of the cache. When two devices come into communication range, a view of the altruistic cache is transmitted, the receiver computes the optimal cache by adding the content of its cache to the received one and sends the missing elements. Nevertheless it might still occur that transmitted information is

discarded. This can happen as the cache might have changed due to information received by other devices not being visible for the sending device.

The cache tries to keep recent data and also aims at being as altruistic as possible. This implies storing data from as many different overlays as possible. Therefore, the altruistic cache is internally organized as two independent caches, one using the timestamps and one using the overlays as an argument to the replacement algorithm. When an information key is deleted from the cache, the corresponding information item is deleted from the InformationPool.

All information data which needs to be sent to interested neighbors is sent by the InformationGate. This is mainly done in a FIFO manner: the information added first is sent first. Due to the fact, that one information can be of interest to more than one device, the FIFO order may be reorganized, so that information addressed to most devices is prioritized. To save network bandwidth, the information is sent via an overlay multicast, so that the information data must only be sent once because a broadcast medium is used (see Figure 3). After all receiving devices have acknowledged the message, the InformationGate sequentially sends the next messages until all information is sent or all receiving devices have left the communication range of the device.

It is possible that not all information is sent to the receiver, because it is always able to leave the communication range. Therefore, it can be necessary to send more important information first and unimportant information last. This ordering can be achieved by adding them in the correct order to the InformationGate, since it sends them in FIFO order. This strategy achieves that the maximum of relevant data can be exchanged even in short interaction periods.

The reception of the information by other devices is communicated to the ProfileManager, so that it does not have to send this information to the receiver again. Moreover, each receiving device can determine all other receivers and is also able to communicate this to its ProfileManager.

2.4 Playing the Game

During an `UbiQuiz` game, the QuestionManager provides questions for the current difficulty level. It keeps track of all answered questions so that it is able to provide only unanswered questions to the user. When it detects that it runs out of unanswered questions, it changes the current profile accordingly and communicates it to the ProfileProvider to announce this change to the neighboring devices. Thus it is possible to receive these questions before no unanswered question is left. To be up to date, the user can adjust the profile so that he always receives the newest questions with a definable age.

It is, of course, possible that no (unanswered) question for a difficulty level is left. In that case, the user can choose to answer a quetsion he answered in a prior game, use a question of a higher difficulty level or wait until a question for this difficulty level is received.

The game can be played offline when enough questions are stored within the QuestionPool. It is also possible to play together with users in the direct neighborhood. In this case, only simple singlehop ad-hoc communication is used.

The user is able to choose direct neighbors as a "telephone" or "audience" joker. All active users are listed – this information is already known by the overlay management – and the user is able to select one as telephone joker. Now the other user can answer the question and specify how reliable his answer is. By selecting the audience joker, neighboring devices of the same overlay are addressed via a multicast message and can help the user answering the question. All unicast answers are collected and an answer statistic is provided to the asking user. Another variant is to create a question "on demand". Neighbors are able to create new questions for the playing user who is waiting for receiving questions of the next difficulty step.

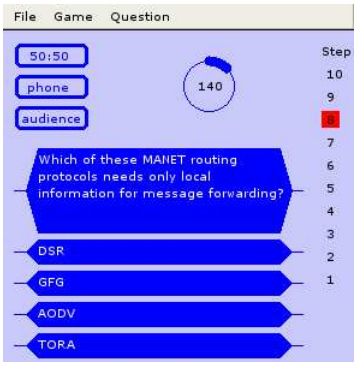

Figure 4. The `UbiQuiz` GUI.

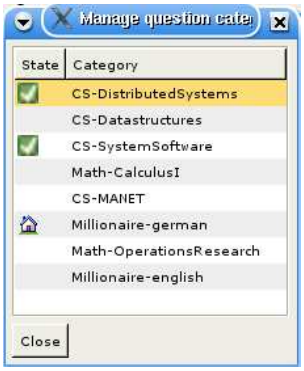

Figure 5. Manage question categories.

Beside playing the game, the user is able to manage the question categories he is currently able to see. (Figure 5). He can leave or enter each known category. It is also possible to load XML coded question libraries from the file system. These new questions are stored within the local QuestionPool and are disseminated in the ad hoc network from now on. The edit mode allows the user to create new questions for a specific question category and to create new question categories.

2.5 Implementation

The `UbiQuiz` application has been implemented in Java using a workbench for implementing and testing applications for mobile multihop ad hoc networks [6]. In a first step, all information management and exchanging components and protocols were implemented and tested within the workbench's simulator. The second step included the GUI and game logic development process and a testrun in the workbench hybrid mode (see Figure 6). There, real devices can be connected to the simulator and the application can be tested with real user behavior. For such application testing, the devices in the simulator can be moved by clicking on the visualization frame.

Finally, the implemented application and GUI code can be used without modification on an execution platform for real devices. Thus, the application can be tested and evaluated on a real hardware platform. The application not

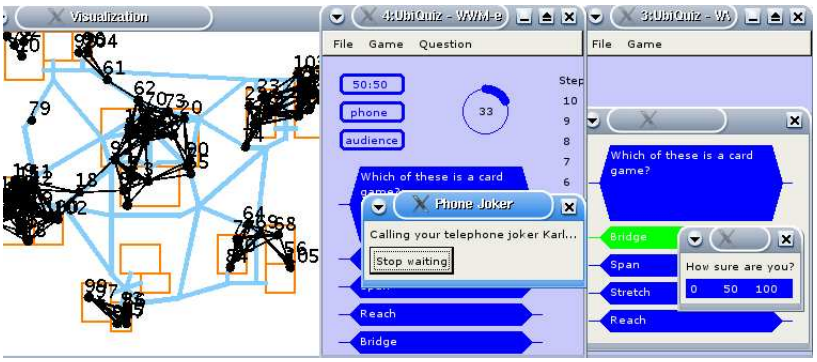

Figure 6. Two extern devices are connected to the simulator, playing `UbiQuiz` together. Devices are moved according to a path net mobility model.

only be used on notebooks as used for the field-trials, it is also possible to use it on PocketPCs or other small mobile devices, equipped with a WLAN adapter and providing a Java VM. The execution platform uses UDP unicasts and broadcasts over WLAN for communication. Since multicast communication is realized as broadcast within the lower layers in WLAN (no low level packaging and acknowledgment), the multicasts are mapped to broadcasts with unicast acknowledgments.

3 Evaluation of `UbiQuiz` in Field Trials

In order to estimate the practical applicability of the `UbiQuiz` application, the reference implementation has been investigated in both a static and a mobile "real world" usage scenario. All evaluation runs were performed by using a set of mobile devices consisting of four notebooks and two tablet PCs. Each device is equipped with an IEEE 802.11b interface. If possible, the interface was fixed to the maximum transfer rate and the power save function was disabled.

Each device was running the `UbiQuiz` application, while logging the times of message transmissions and message receipts. For each simulated scenario one device initiated the dissemination of a new set of `UbiQuiz` questions containing 5000 questions, while each question accounting for about $900 - 1000$ bytes of message size.

Data rate and error probability were of primary interest in order to judge the performance in both usage scenarios. However, in contrast to traditional evaluation methodology, performance was measured from an application point of view instead of sampling raw data traffic produced by the utilized network protocols. Thus, the presented empirical values were obtained by sampling the data rate and error probability in terms of average number of received question library entries per second and the average number of lost messages during transmission, respectively. The evaluation results presented in the following two sections reflect the typical properties of a wireless communication media. The data rate degrades for both, a growing number of devices utilizing the shared communication media, and an increased distance between sending and receiving device.

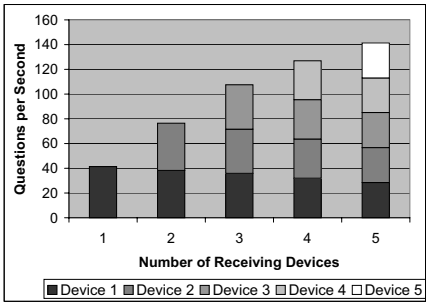

Figure 7. Question items per second.

Figure 8. The en-passant communication scenario.

3.1 Evaluating the One-to-Many Scenario

A typical real world application scenario of `UbiQuiz` may be some devices located at a public place, while each device is within the communication range of all others. In order to judge the performance of `UbiQuiz` for such a scenario, three independent trials have been conducted for an increasing number (2-6) of participating wireless devices.

Figure 7 depicts the average number of questions received per second as a function of the number of receiving devices. One can see that the number of received questions per device decreases, when the number of recipients increases. This is due to the application design which requires that acknowledgements of all current recipients have to be received before the next library entry is being sent. Figure 7 also shows that summing the average number of question receipts per device compensates that performance loss. This is due to the fact, that questions are broadcasted to all devices and only the acknowledgments are sent via unicast. Thus, only the small acknowledgement messages leads to a higher protocol overhead, the large question messages has only to be send once to all devices. Implementing such a multicast not within the application layer but in mac layer as the unicast acknowledge my further reduce this acknowledgement overhead.

3.2 Evaluating the En-passant Scenario

In a second application scenario, the quality of the `UbiQuiz` application has been investigated with respect to exploiting the limited communication window emerging from two devices passing each other causally as depicted in Figure 8. Again three independent evaluation trials have been conducted, while both test persons passed each other with "almost" the same moving speed. The distance of the starting points was about 150 meter, the meeting point was almost in the middle. In all three evaluation runs the total amount of received messages was about $1300 - 1500$ (≈ 14 MByte).

The curve progression of the number of transmitted question library entries per second and the number of lost messages was investigated to be nearly the same for all three evaluation runs. Thus, the data rate can be depicted as an average over all three evaluation runs without losing its main characteristics (see

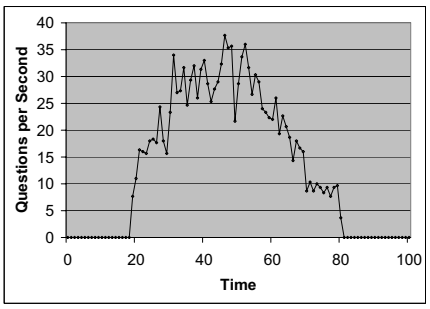

Figure 9. Data rate during en-passant communication.

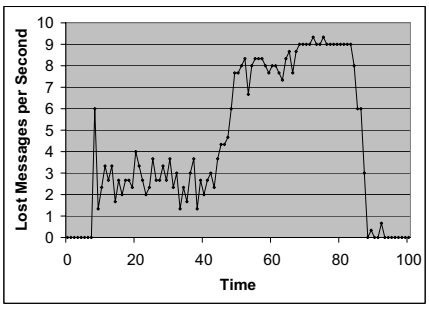

Figure 10. Message losses during en-passant communication.

Figure 9 and Figure 10). One can see that both devices get in contact for the first time at about 10 sec after starting the experiment. However, signal quality is bad at this time and messages get lost with a high probability. The first message was received successfully about 10 seconds later. The data rate increases and stays around 30 question per second between 30 and 60. The devices met about second 50. There, the data rate drops to 21, probably caused by antenna interference. After passing the meeting point the data rate start to decrease significantly. This is probably due to the test persons hiding the device antennas. The same can also be observed in Figure 9 where the number of lost messages is significantly higher as before the meeting point.

4 Related work

There are multiple strategies to disseminate information in mobile multi hop ad hoc networks. Beside several flooding algorithms like XCast [7], which uses controlled flooding, e.g. MobiGrid [3] discusses among other ideas the publish subscribe (PS) and the autonomous gossiping (AG) techniques. As `UbiQuiz`, AG also uses en-passant communication and exchanges profiles first. But it neither uses interest overlays, nor efficient information exchanging – date items are always forwarded to neighbors with fitting profiles. PS is used in [1] and works with multi-layer networks to transport the data to the interested parties. A problem when using PS is that a subscription to distant information sources is difficult. Therefore, `UbiQuiz` can be thought of a PS implementation with one hop communication. [4] as a representative of the AG technique broadcasts information items and interested clients can detect and request missing information. Obviously, this might consume a lot of communication bandwidth. Instead of creating information overlays as in `UbiQuiz` it is also possible to use tuple spaces. For instance, [9] spreads tuples in the network according to propagation rules. Thus, the data item "decides" to which device it migrates not the targeted device. XMIDDLE [15] synchronizes information with direct neighbors by using XML trees. Moreover, it is possible to link subtrees to devices were the information is stored, but this leads to a high coupling of devices. Moreover, information access is difficult in larger mobile ad hoc networks.

Many research has been done in the area of efficient flooding protocols [14]. One example are the SPIN [8] protocols which are intended to reduce message overhead in sensor networks. This is also achieved by using information meta-data, an application specific information description. New information is offered to all neighbors by sending the metadata and neighbors then request missing information. This causes much more overhead in a mobile environment, since all locally known metadata must be exchanged when a new device is detected in the neighborhood.

Even though the problem of information dissemination/retrieval in mobile ad hoc network has been researched for some time, up to now very few applications have been implemented. [2] is a multi-player adoption of the old single-player "Elite" game. But it still uses central servers and was developed to research social aspects of ubiquitous computer games. The approach made by [12] looks more promising as Opentrek enables rapid game prototyping and delivers easy discovery and integration of players into running games.

5 Conclusion and Further Work

This work introduced `UbiQuiz`, a real world gaming application for multi hop ad hoc networks. Moreover, it introduced an ad hoc information dissemination protocol for efficient information exchanging by reducing the amount of exchanged information to interest domains and profiles. The field-trials performed demonstrated on the one hand the usefulness of using the broadcasting capabilities of wireless communication and the necessity of reducing communication overhead by using interest domains. On the other hand it can be observed, that current system softwares and wireless communication techniques are still not prepared for multi hop ad hoc communication and that still much work has to be done in this area. One example is the lack of reliable communication mechanisms efficiently using the broadcasting facilities of wireless communication.

Since field trials are still very expensive, only a very small subset of necessary tests has been performed. More extensive tests are planned for the nearer future, including the development of a field-trial testbed and a management application helping to reduce the complexity of field-trials.

The `UbiQuiz` application is only one aspect of a larger m-learning environment and should be combined with other m-learning applications. For example combining questions with distributively created scripts and lecture slides disseminated over the ad hoc network to give answering hints is planned. Moreover, it is of course possible to realize more complex applications not only based on simple multiple choice questions.

Another application aspect is the possibility of editing and deleting questions. Currently editing is achieved by simply changing the creation timestamp and keeping the original hash value so that only the newest information survives. Deleting is achieved by propagating deletion information and storing this in a local deletion history. The current work is focused on a distributed information evaluation process, where the user is able to vote for questions. Thus, it is possible to keep a "good" question alive while a "bad" question expires and is deleted.

Finally the application should be put into practice by using it in the context of a system software lecture, since most students are already equipped with mobile devices having wireless communication adapters.

References

1. Emmanuelle Anceaume, Ajoy K. Datta, Maria Gradinariu, and Gwendal Simon. Publish/subscribe scheme for mobile networks. In *Proceedings of the second ACM international workshop on Principles of mobile computing*, 2002.
2. S. Bjork, J. Falk, R. Hansson, and P. Ljungstrand. Pirates! Using the Physical World as a Game Board. In *Conference on Human-Computer Interaction*, 2001.
3. Anwitaman Datta. MobiGrid: P2P Overlay and MANET Rendezvous - a Data Management Perspective. In *CAiSE 2003 Doctoral Symposium*, 2003.
4. Anwitaman Datta, Silvia Quarteroni, and Karl Aberer. Autonomous Gossiping: A self-organizing epidemic algorithm for selective information dissemination in mobile ad-hoc networks. In *International Conference on Semantics of a Networked World*, 2004.
5. Hannes Frey. Scalable geographic routing algorithms for wireless ad hoc networks. *IEEE Network*, 18(4):18–20, July 2004.
6. Hannes Frey, Daniel Görgen, Johannes K. Lehnert, and Peter Sturm. A java-based uniform workbench for simulating and executing distributed mobile applications. In *Proceedings of FIDJI 2003 International Workshop on scientific engineering of distributed Java applications*, Luxembourg, November 27–28 2003.
7. J. Koberstein, F. Reuter, and N. Luttenberger. The XCast Approach for Content-based Flooding Control in Distributed Virtual Shared Information Spaces - Design and Evaluation. In *1st European Workshop on Wireless Sensor Networks (EWSN)*, 2004.
8. J. Kulik, W. Rabiner, and H. Balakrishnan. Adaptive Protocols for Information Dissemination in Wireless Sensor Networks. In *MobiCom*, 1999.
9. Marco Mamei, Franco Zambonelli, and Letizia Leonardi. Tuples on The Air: a Middleware for Context-Aware Computing in Dynamic Networks. Technical report, University of Modena and Reggio Emilia, 2002.
10. S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu. The broadcast storm problem in a mobile ad hoc network. *Proc. of the 5th ACM/IEEE Int. Conf. on Mobile Computing and Networking*, pages 151–162, 1999.
11. Elizabeth M. Royer and Chai-Keong Toh. A review of current routing protocols for ad-hoc mobile wireless networks. *IEEE Personal Communications*, pages 46–55, April 1999.
12. Johan Sanneblad and Lars Erik Holmquist. Prototyping mobile game applications, 2002.
13. Andrew Tridgell and Paul Mackerras. The rsync algorithm. Technical report, Australian National University, 1998.
14. B. Williams and T. Camp. Comparison of broadcasting techniques for mobile ad hoc networks. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, pages 194–205, 2002.
15. Stefanos Zachariadis, Licia Capra, Cecilia Mascolo, and Wolfgang Emmerich. XMIDDLE: Information sharing middleware for a mobile environment. In *ACM Proc. Int. Conf. Software Engineering (ICSE02). Demo Presentation.*, Orlando, FL, USA, May 2002.