

An Application Platform for Mobile Ad-hoc Networks

Gerd Kortuem, Jay Schneider

Wearable Computing Laboratory
Department of Computer Science
University of Oregon
{kortuem,jay}@cs.uoregon.edu

Abstract. This paper describes *Proem*, an application development platform that provides a complete solution for developing and deploying peer-to-peer systems for mobile ad-hoc networks.

1 Introduction

Advances in wireless technology and mobile computing have provided a major impetus toward development of Mobile Ad-hoc Networks (MANET) and Personal-Area Networks (PAN). These networks are self-organizing networks that are comprised of wireless nodes that cooperate in order to dynamically establish communication. Any device with a microprocessor, whether highly mobile or stationary, is a potential node in an ad-hoc network. This includes mobile and wearable devices, desktop computers, ubiquitous devices embedded in buildings, and motor vehicles. Examples of ad-hoc networks are Bluetooth [1], Genuity's BodyLAN [2], Zimmermann intra-body network [3] and networks following the emerging IEEE 802.15 standard [4].

The combination of personal mobile devices and wireless ad-hoc networks allows the creation of highly dynamic, decentralized, self-organizing peer-to-peer (P2P) systems comprised of large numbers of mobile (and stationary) devices. In such a system, mobile hosts continuously change their physical location and establish peering relationships among each other based on proximity. Mobile ad-hoc networks create opportunities for a range of novel and interesting P2P applications including collaborative computing, mobile patient monitoring, distributed command and control systems and ubiquitous computing. In particular, Personal-Area Networks enable proximity-aware applications in support of face-to-face collaboration [15,16].

Our research addresses the following four questions:

1. What applications are useful for mobile ad-hoc networks?
2. What are the characteristics of these applications?
3. What are the requirements for mobile ad-hoc middleware?
4. What abstractions and support functions are needed by developers of mobile P2P applications?

In this paper, we focus on the later two questions. For a discussion of applications for ad hoc networks see [16].

The result of our research is *Proem*, a P2P platform for mobile ad-hoc networks. The key benefits of *Proem* are:

- High-level development support
- Platform independence
- Interoperability
- Extensibility

The *Proem* platform has been used as instructional tool in an advanced Software Engineering course on Peer-to-Peer Computing [5] at the University of Oregon; a pre-release version is available at <http://wearables.cs.uoregon.edu/proem>. An early version of *Proem* was described in [6]. A detailed description of several mobile P2P applications and a discussion why such applications are fundamentally different from Internet-based P2P applications can be found in [16]; this paper also contains a discussion of design rationale and platform requirements for *Proem*.

This paper is organized as follows:

In the next section we present the goals and architecture of our P2P platform *Proem*. Chapter 3 provides a brief discussion of our experiences in using *Proem* as instructional and research platform. We conclude with an outlook on future research directions.

2 *Proem*: An Application Platform for Mobile Ad-hoc Networks

2.1 The Vision and Goals of *Proem*

Proem is an open computing platform that provides a complete solution for developing and deploying peer-to-peer applications for mobile ad-hoc networks.

The motivation for developing *Proem* arose from our experiences in implementing a series of mobile applications for face-to-face collaboration [6,15,16]. Over time we identified enough commonalities among these applications to merit the development of a generic software development platform. The objectives for *Proem* include:

- **Adaptability:** *Proem* is designed to facilitate rapid and timely response to changes in the operating environment, for example regarding connectivity and resource availability.
- **Versatility:** *Proem* is an infrastructure for building diverse mobile applications ranging from ad-hoc meeting support to chat, instant messaging, distributes search and file sharing.
- **Platform Independence:** *Proem* is designed to be independent of programming languages, system platforms and networking platforms.
- **Built-in security:** Many interesting ad-hoc P2P applications must guarantee privacy, confidentiality, and integrity of transmitted and stored data and thus require strong security measures (the current release of *Proem* does not include security).
- **Development Support:** The most prominent goals in designing *Proem* is to provide a simple yet powerful development platform that facilitates the implementation of mobile P2P applications.

2.2 Important Concepts and Terminology

Before we go into details of the *Proem* platform we must first clarify some concepts that are fundamental for its understanding.

Entities

Proem defines four fundamental entity types:

- **Peer:** A peer is any autonomous, mobile host or device taking part in a peer-to-peer relationship.
- **Individual:** An individual is a person who owns and uses one or more peers. We assume mobile devices to be personal devices that are not routinely shared. Any person might use any number of peers, but each peer belongs to only one individual.
- **Data Space:** A data space is a collection of data items that are cooperatively owned and managed by a set of peers. Data spaces are stored in a replicated fashion on all peers that share them.
- **Community:** A community is a set of entities (peers, individuals, data spaces and other communities). Each entity can be a member of several communities (including none) and each community can contain members of different type. Communities can be used to define access rights to data and functionality or simply as a way to group entities. Examples of common communities include:
 - The set of peers owned by a particular user
 - A set of individuals who are friends and who grant each other special access rights
 - The set of data spaces related to a particular project
 - The set of all entities related to a particular project: individuals, peers, data spaces.

The concept of communities is different from the notion of groups as commonly defined in distributed systems. A community is an open set of entities. Membership is not controlled by the owner of the group (which might be one particular member or the collective of all members), but can be passed on by any member to any other entity. As a consequence it is impossible to determine the complete set of members of a community: no single authority controls membership and members can join at any time. Membership is conferred upon an entity by passing along a secret membership token cryptographically signed by the conferring entity. Membership tokens are unique to each community.

Communities represent realms of trust. In order to 'prove' membership in a society, an entity has to produce the membership token signed by a minimal number of members that are known as such to the verifying entity. Entities can arbitrarily set the number of signatures they require; it is entirely up to the verifying entity to accept or reject a presented membership token as proof. Communities only provide a mechanism for trust, policies can be defined by individual peers or applications.

Protocols and Messages

At the core of *Proem* lie a set of communication protocols that define the syntax and semantics of messages that peers can exchange. The definition and use of peer protocols guarantees interoperability between implementations of the *Proem* system on different hardware and software platforms.

Proem defines four protocols, one low-level transport protocol and three higher-level protocols.

The ***Proem* Transport Protocol** is a connectionless asynchronous communication protocol. Data is passed from peer to peer in one atomic unit. The *Proem* Protocol uses XML for representation of messages and can be implemented on top of a variety of existing protocols such as TCP/IP, UDP or HTTP. When an unreliable transport protocol is used messages may be delivered more than once, may not arrive at all, or may arrive in a different order than sent. The reception of a message is not acknowledged unless explicitly specified by the protocol.

The *Proem* core application protocols are:

- The **Presence Protocol** contains messages that allow peers to announce their presence and the availability of entities throughout a network. The primary message type of the presence protocol is profiles.
- The **Data Protocol** contains messages that allow peers to share and synchronize data by means of data spaces.

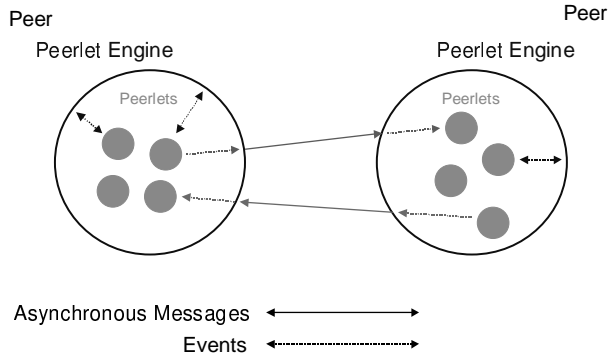


Figure 1.

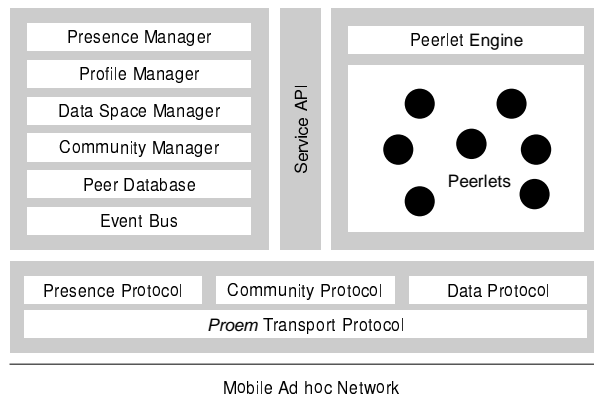


Figure 2.

- The **Community Protocol** contains messages for applying for, granting and verifying community membership. In addition to these built-in protocols, application developers can define their own application-specific protocols.

2.3 Proem Application Environment

The *Proem* application environment is a collection of tools, APIs and runtime structures for developing and deploying applications within the Proem framework. The two major components of the *Proem* application environment are the *Proem Runtime System* and the *Peerlet Development Kit*.

Peerlets and Peerlet Engine

The Proem application environment is based around the notion of *peerlets*. Peerlets are simple structured peer-to-peer applications that follow an event-based programming model. Peerlets are the locus of computation and function as communication end-points. They are hosted in the *Peerlet Engine*, which is responsible for the instantiation, execution and termination of peerlets. Peerlets are designed as drop-in modules and can be added to and removed from the peerlet engine at runtime.

Peerlets react to and communicate via events. The peerlet engine fires events to peerlets as a reaction to changes in its internal state or as reaction to messages received by remote peers. Peerlets are notified of and handle events asynchronously. Figure 1 shows the relationship between peers, peerlet engine, peerlets, messages and events.

Proem Runtime System

The *Proem Runtime System* is a proof of concept implementation of a peer that ‘speaks’ the *Proem* protocols. The overall architecture is shown in Figure 2. It consists of three components:

- The *protocol stack* implements the four *Proem* protocols.
- The *peerlet engine* controls the execution of peerlets.
- A set of *services* provides peerlets with access to commonly used functionality. Facilities provided by services include mechanisms for naming, data management, event-logging as well as management of user and trust related information.

The *Proem Services* are implemented by a number of system components. These are:

- The *presence manager* is responsible for announcing a peer’s presence and for discovering nearby peers. The meaning of “nearby” depends on the current network topology and includes all peers that are reachable either directly or indirectly.
- The *data space manager* is responsible for persistent storage of data spaces, as well as access control.
- The *community manager* keeps track of a peer’s membership in communities and performs validation checks of other peers’ community memberships.
- The *peer database* maintains a persistent log of encounters with other peers and allows peerlets to store custom meta-information on peers. This enables peerlets to determine when and how often a particular peer has been encountered in the past.
- The *event bus* enables event-based communication among systems components including peerlets. It provides a *publish-and-subscribe* model that allows anonymous exchange of data. System components and peerlets can announce the availability of data item and express interest in data by subscribing to update events. Events are also used by the presence manager to inform peerlets about the appearance and disappearance of peers.

All of these components are themselves implemented as peerlets. This enables independent developers to change the implementation of system components by replacing a system peerlet with a peerlet of their own design.

Peerlet Development Kit

The *Peerlet Development Kit* (PDK) is a set of high-level Java APIs for the rapid development of peerlets that can be executed by the peerlet engine. The PDK provides an extensive collection of Java interfaces and classes for naming, communication, data management, and event handling.

Some important classes and interfaces are: Peer, Peerlet, *ProemEvent*, *EncounterEvent*, *ProemProtocol*, *ProemMessage*, *ProemService*, *DataSpace*, *DataItem*, *Community*.

3 Experiences

The main goal of the *Proem* platform is to provide high-level support for mobile P2P application developers. The early experiences of using *Proem* in an advanced Software Engineering course at the University of Oregon and as a platform for various research projects are encouraging. Students who had no prior knowledge of ad hoc networks, wireless communication and P2P computing and started with nothing more than the *Proem* Programmers Manual were able to complete application design and implementation within a short timeframe (5 weeks for projects that were completed as part of a course, 2 weeks for individual research projects). According to our informal observations the *Proem* platform provides the necessary framework for guiding inexperienced programmers, but has also proven very useful for advanced developers and projects.

4 Conclusion and Future Research Directions

Peer-to-peer systems for mobile ad-hoc networks introduce a number of new issues related to naming, discovery, communication and security. In particular, the highly dynamic nature of ad-hoc networks requires an agile system architecture that can monitor and react to changes in the environment in a timely and efficient manner. In order to simplify the task of the application developer we need a P2P platform that facilitates the development of adaptive mobile P2P applications. We view the *Proem* mobile P2P platform as a step in this direction.

Our future work will focus on extensions to the *Proem* platform in the following areas:

First, we are working on a tighter integration of *Proem* with services provided by underlying ad-hoc networks.

Second, we are in the process of specifying and implementing a security architecture for *Proem*. One of the focal points will be the development of a fully decentralized trust mechanisms using a public key infrastructure and the use of reputations. Work in this area has already begun [14].

References

- [1] Bluetooth Consortium web site, <http://www.bluetooth.org>
- [2] Phillip Carvey. BodyLAN. IEEE Circuits and Devices, V4 No 12 July 1996
- [3] T. G. Zimmerman. Personal Area Networks: Near-field intrabody communication. IBM Systems Journal Vol. 35, No. 3&4, 1996
- [4] IEEE 802.15 Working Group for PANs, <http://grouper.ieee.org/groups/802/15/index.html>
- [5] Department of Computer Science, University of Oregon. Advanced Software Engineering on Peer-to-Peer Computing, Spring 2001. Course home page: www.cs.uoregon.edu/classes/cis650
- [6] Gerd Kortuem, Zary Segall, Thaddeus G. Cowan Thompson. Close Encounters: Supporting Mobile Cooperation Through Interchange of User Profiles. 1st International Conference on Handheld and Ubiquitous Computing (HUC), September 1999, Karlsruhe, Germany.
- [7] Napster. <http://www.napster.com/>
- [8] Gnutella Protocol, version 0.4. <http://dss.clip2.com/GnutellaProtocol04.pdf>
- [9] Gene Kan. Gnutella. In Andy Oram (ed.) Peer-to-Peer: Harnessing the Power of Disruptive Technologies. March 2001.
- [10] I. Clarke, O. Sandberg, B. Wiley, and T.W. Hong, Freenet: A Distributed Anonymous Information Storage and Retrieval System in Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability, LNCS 2009, ed. by H. Federrath. Springer: New York (2001).
- [11] Groove Networks. <http://groove.net/>
- [12] Michael Nidd, Timeliness of Service Discovery in DEAPspace. Proc. of the 2000 International Workshop on Parallel Processing
- [13] J. J. Kistler and M. Satyanarayanan. Disconnected Operation in the Coda File System. ACM Transactions on Computer Systems, 10(1):3, February 1992
- [14] Jay Schneider, Gerd Kortuem, Joe Jager, Steve Fickas, Zary Segall. Disseminating Trust Information in Wearable Communities. 2nd International Symposium on Handheld and Ubiquitous Computing (HUC2K), Sept 25-27, 2000, Bristol, England.
- [15] Gerd Kortuem, Jay Schneider, Jim Suruda, Steve Fickas, Zary Segall. When Cyborgs Meet: Building Communities of Cooperating Wearable Agents. Proceedings Third International Symposium on Wearable Computers, 18-19 October, 1999, San Francisco, Ca.
- [16] Kortuem, G., Schneider, J., Preuitt, D., Cowan Thompson, T.G., Fickas, S., Segall Z. When Peer-to-Peer comes Face-to-Face: Collaborative Peer-to-Peer Computing in Mobile Ad-hoc Networks. 2001 International Conference on Peer-to-Peer Computing (P2P2001), Aug. 2001, Linköping, Sweden.