Proem: A Peer-to-Peer Computing Platform  
for Mobile Ad-hoc Networks

Gerd Kortuem  
Wearable Computing Laboratory, Department of Computer Science  
University of Oregon  
kortuem@cs.uoregon.edu

Abstract. This paper describes Proem, a mobile computing 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 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 include Bluetooth [1], Genuity’s BodyLAN [2], Zimmermann intra-body network [3] and networks following the emerging IEEE 802.15 standard [4].

The combination of mobile devices and ad-hoc networks allows the creation of highly dynamic, self-organizing, mobile peer-to-peer (P2P) systems. In such systems, mobile hosts continuously change their physical location and establish peering relationships among each other based on proximity. Such an environment creates opportunities for a range of novel and interesting applications involving interaction between people who are co-located and organized in an unforeseeable ad hoc way. Examples include systems for the support of spontaneous meetings in office settings, task-oriented collaboration of emergency response teams, mobile patient monitoring, distributed command and control systems, and educational multi-user applications for use in classrooms.

Despite the fact that ad-hoc networking and P2P computing deal with similar issues, namely discovery, routing and information dissemination, there is not much overlap in the research. Most research on ad-hoc networks focuses on the lower layer of the protocol stack including the link layer, network layer and transport layer. On the other hand, current P2P systems are designed for an Internet-like network infrastructure in which stationary hosts are connected by high bandwidth links. The assumptions on which these P2P systems are built are no longer valid in ad-hoc networking environments.

Our research addresses the following four questions:
1. What are interesting applications 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 does a mobile ad-hoc application toolkit need to provide?

In this paper, we will focus on the latter two questions. Some applications are described in [6,14,15].

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
- Built-in support for communication, data sharing, event logging, security, and privacy.

Proem is used as instructional tool in an advanced Software Engineering course on Peer-to-Peer Computing [5] at the University of Oregon and is available at http://wearables.cs.uoregon.edu/proem. An earlier version of Proem was described in [6].

This paper is organized as follows: In Section 2, we identify challenges of mobile peer-to-peer systems. Section 3 presents the goals and overall architecture of the P2P platform Proem. Section 4 discusses results of using the Proem platform developing mobile applications. We conclude with an outlook on future research directions.

2 The Challenges of Mobile Peer-to-Peer Computing

Mobile ad hoc networking represents a radical departure from the currently dominant infrastructure model of wireless communication. Most wireless networks support untethered access for mobile devices by providing a wireless interface between the mobile devices and fixed network base-stations. The disadvantage of this configuration is that the fixed infrastructure limits device mobility and overall network deployability. As a result, infrastructure networks are not well
suited for rapid or temporary network deployment and for environments where it is difficult to achieve adequate base-
station coverage.

In line with the infrastructure model, mobile systems typically have been designed as client-server system in which thin
clients such as Personal Digital Assistants (PDAs) or data-capable cell-phones use wireless connections to gain access to
resources (data and services) provided by central servers. The emergence of wireless ad-hoc networks and powerful mobile
devices makes it possible to design mobile systems using a peer-to-peer architecture. A mobile peer-to-peer system consists
of mobile devices that interact during brief physical encounters in the real world thereby engaging in short-haul wireless
exchanges of data. Mobile P2P applications take advantage of resources -- storage, cycles, content, human presence --
provided by mobile (or stationary) devices in the immediate physical proximity.

A mobile P2P system inherits many of the features of ad-hoc networks:

- **Self-organizing**: as side effect of the movement of devices in physical space, the topology of a mobile P2P system
  constantly adjusts itself by discovering new communication links.
- **Fully decentralized**: each peer in a mobile peer-to-peer system is equally important and no central node exists.
- **Highly dynamic**: Since communication end-points can move frequently and independently of one another, mobile P2P
  systems are highly dynamic.
- **Low cost**: wireless ad hoc networks are built from low-cost transceivers and do not incur charges for provider access and
  air-time.

The unique character of mobile peer-to-peer systems represents a significant challenge for the designer. In the following,
we will discuss some of the challenges that are unique to mobile ad-hoc networks.

### 2.1 Challenges

#### Naming

Traditional (non mobile) P2P systems are characterized by an increasing decentralization and autonomy of hosts.
Because accessing these decentralized resources means operating in an environment of unstable connectivity and
unpredictable IP addresses, P2P systems often operate outside the DNS system. The same must be true for mobile P2P
systems. Additional reasons for not relying on the DNS system are:

- In ad-hoc networks, access to a central DNS server cannot be assumed
- Not all mobile devices support IP networking and thus do not have IP addresses
- Some mobile P2P applications, in particular for face-to-face collaboration, require the ability to identify not only peers,
  but also the people who run and use these peers.

#### Peer and Resource Discovery

One of the things that makes current P2P system so powerful is that they take advantage of resources -- storage, cycles,
content, human presence -- available at the edges of the Internet. In a mobile P2P system mobile peers take advantage of
resources provided by mobile peers that are physically close. Because of the unpredictable movement of mobile devices,
discovering resources becomes a challenge.

In ad-hoc networks, device discovery is part of the network; resource discovery, however, is the task of the peer system.
We need algorithms through which a peer can detect the presence of neighboring peers, share configuration and service
information with those peers, and notice when peers become unavailable. Resource discovery must be timely (in order to
detect moving peers) and efficient (so not to overload the network) [12]. In contrast to P2P systems that are targeted at
fixed networks, decentralization is not a mere option for mobile P2P networks, but a necessity.

#### Communication

Wireless data networks present a more constrained communication environment compared to wired networks. Because
of fundamental limitations of power, available spectrum, and mobility, wireless data networks tend to have less bandwidth,
more latency, less connectivity stability, and less predictable availability. Wireless ad-hoc networks add to these limitations
in two important respects:

First, in ad-hoc networks point-to-point communication is only possible between nodes that are within each other’s
transmission range. Depending on this range and the speed of mobile hosts, it might be necessary to set up and tear down
communication links with a very high frequency.

Second, the frequent failure and activation of links leads to increased network congestion while the network routing
algorithm reacts to topology changes [13]. This is because communication between arbitrary peers requires routing over
multiple-hop wireless paths whose end-points are likely to be moving independently of one another.

#### Data Sharing and Synchronization

A mobile ad hoc information system is basically a highly dynamic, decentralized distributed system with weakly
connected mobile hosts. In order to cooperate to the fullest extent peers need to be able to share and synchronize data. The
extreme decentralization and unpredictability of mobile ad hoc system together with the fact that peers always only establish pair-wise connections lead to the following conflicting requirements:

**High availability:** We want peers to be as autonomous as possible. They should be able to perform computations even in the absence of connections with other peers. Thus we need to employ a replicated object scheme where each peer maintains a local copy of each shared data object.

**Consistency:** A replicated object scheme introduces the problem that copies of a shared object can be updated independently and thus might become inconsistent over time. A synchronization mechanism must be employed that either prevents or reconciles inconsistencies.

**Timeliness:** Any solution to the consistency problem has to deal with the fact that data might be shared across a group of peers that rarely come together. Even more, there is no guarantee that a specific group of peers will ever meet at the same time, since interactions in an ad hoc system happen during situations of physical proximity and movement of hosts are unpredictable. Consequently, updates must be passed from peer to peer which can lead to a slow propagation of updates throughout the whole system.

**Security**

The security implications of mobile P2P systems, in which one can potentially track every movement of an individual as well as examine what they are doing, must be taken seriously. In wireless ad-hoc networks users may not even be aware to which devices they are connected. Someone in the next room or on the floor above may connect to someone else's mobile device and gain access to private data such as stored e-mail, and meeting schedules. Thus, not only must encryption be employed to avoid eavesdropping, but also robust authentication procedures need to be established for connecting both trusted and non-trusted devices with each other. This, however, is made difficult by the fact that this must occur in a completely decentralized environment with no or intermittent connection to a trusted authority. Possible solutions might include the use of reputations [14].

### 2.2 Conclusion

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 P2P applications and provides built-in support for naming, discovery, communication, and security. In the next section we will discuss such a platform.

### 3 Proem: A Mobile Peer-to-Peer Platform

#### 3.1 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]. Over time we identified enough commonalities among these applications to merit the development of a generic software platform. The objectives for Proem include:

- **Versatility:** Proem is an infrastructure for building diverse mobile applications ranging from ad-hoc meeting support to classical P2P applications like MP3 file sharing and instant messaging. In contrast to systems like Napster or Gnutella, which are designed for one particular purpose (mp3 file sharing), Proem provides the building blocks for a wide range of peer applications.

- **Interoperability:** Proem is designed to allow interoperability between heterogeneous hardware and software platforms.

- **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.

- **High-level 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.

#### 3.2 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.

---

1 The current release of the Proem software does not include security.
Entities

The Proem architecture defines the following 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.

Identities, Names and Profiles

Entities are identified by *names*. Names are expressed by Uniform Resource Identifiers (URI). Each entity can have one or more names. For example, a single peer can be known as `proem:peer:0101` or as `proem:peer:2222`. Multiple names provide for pseudo-anonymity: since there is no central name repository it is impossible to determine whether two different names refer to the same or two different identities. Each name, however, is unique and can only refer to one entity.

Proem provides a second way to refer to entities besides using explicit names. Entities can indirectly be referenced by *profiles*. A profile is an XML-based data structure for describing Proem entities, i.e. meta-data. A profile for an individual might contain his real name, address and email address while a peer profile could include a list of data spaces it has access to. Profiles function as intentional names and are used to advertise the presence of entities with particular attributes throughout a network.

Protocols and Messages

At the highest abstraction level, Proem simply can be viewed as 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.

Messages are the basic unit of communication between peers. Messages are addressed to and are sent from one peer to another. Messages are encoded as XML documents. This allows peers to implement the protocol in a manner best suited to its abilities and role. In particular, Proem peers can be implemented in any programming language and do not require a specific transport protocol. A collection of peers that communicates using the Proem Transport Protocol form a Proem network. Each host that implements the protocol can become part of a Proem network. In a Proem network, each peer operates independently and asynchronously of any other peer.
The Proem core 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.
- 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. This way, Proem can be extended to support MP3 file sharing or any other peer-to-peer application as an extension to the base Proem protocol.

### 3.3 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. It currently exists in form of a Java implementation [34]. 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. This includes 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.

---

**Figure 1.**

**Figure 2.**
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. 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 particular 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, Encounter, ProemEvent, ProemProtocol, ProemMessage, ProemService.

A pre-release version of the Peerlet Development Kit is available at http://wearables.cs.uoregon.edu/proem.

4 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 were starting 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.

5 Conclusion and Future Research Directions

Despite that fact that ad-hoc networking and P2P computing deal with similar issues, namely discovery, routing and information dissemination, there is not much overlap in the research. Most research on ad-hoc networks focuses on the lower layer of the protocol stack including the link layer, network layer and transport layer. On the other hand, current P2P systems are designed for an Internet-like network infrastructure in which stationary hosts are connected by high bandwidth links. The assumptions on which these P2P systems are built are no longer valid in dynamic ad-hoc networking environments. The unique characteristics of such networks require highly adaptable P2P systems that can react to changes in connectivity and resource availability in a timely and ongoing manner. We view the Proem mobile P2P platform as a step towards an integration of both research areas.

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