SUPPORTING AD HOC COLLABORATIONS IN PEER-TO-PEER NETWORKS

Ismail Bhana, David Johnson, Nia Alexandrov
Advanced Computing and Emerging Technologies Centre, School of Systems Engineering,
The University of Reading
Reading, RG6 6AY, United Kingdom
{i.m.bhana, d.johnson, n.s.alexandrov}@reading.ac.uk

Abstract

Traditionally, applications and tools supporting collaborative computing have been designed only with personal computers in mind and support a limited range of computing and network platforms. These applications are therefore not well equipped to deal with network heterogeneity and, in particular, do not cope well with dynamic network topologies. Progress in this area must be made if we are to fulfill the needs of users and support the diversity, mobility, and portability that are likely to characterise group work in future. This paper describes a groupware platform called Coco that is designed to support collaboration in a heterogeneous network environment. The work demonstrates that progress in the development of a generic supporting groupware is achievable, even in the context of heterogeneous and dynamic networks. The work demonstrates the progress made in the development of an underlying communications infrastructure, building on peer-to-peer concept and topologies to improve scalability and robustness.

Keywords
Peer-to-peer, Collaborative Computing, Java, XML, JXTA, Coco

1 INTRODUCTION

In this paper we present a collaborative computing platform called Coco that provides a set of services supporting collaboration in distributed network environments. Coco is a peer-to-peer (P2P) system designed around a set of interoperable network protocols to enable connectivity in fully decentralised networks.

P2P networks may be beneficial to the design of cooperative and collaborative computing architectures, particularly in relation to:

Scalability – P2P systems are inherently scalable, enabling a collaborative system to scale with a dynamically changing user base [1].

Self-organisation and dynamic behaviour – P2P systems support the self-organisation of individuals into groups with shared interests, enabling users to form spontaneous and dynamic collaborations supported by the asynchronous underlying nature of the network [1].

Robustness, fault tolerance, and resilience to attack – since the organisation of peers is largely unpredictable (i.e. nondeterministic) and as there are no centralised mechanisms for the provision or management of services, peer-group services are highly resistant to forms of malicious attack (particularly denial-of-service attacks) to which client-server systems are notably vulnerable [2]. P2P networks therefore provide strength in numbers.

Dynamic – allowing users to form dynamic collaborations from locations that are not fixed to any physical endpoint address (such as an IP address), but to enable participants to move freely from endpoint to endpoint and device to device.

These properties are highly significant in supporting collaborative computing in a heterogeneous context. For instance, scalability is important in systems that are designed to cope with arbitrary numbers of users, especially when users’ interactions are dynamic and transient (i.e. within spontaneous sessions of interaction). Fault tolerance and high availability are very important in distributed CSCW platforms in order to maintain the consistency of state information (and therefore user experience) across the system. Most importantly, P2P systems utilise resources at the edge of the network where they are found in great abundance, providing collaborative platforms with access to greater bandwidth, processing power, and storage.

However, such systems are often very difficult to design and build. All of the apparent successes in this area have been built on proprietary communication protocols that are not interoperable and therefore provide insufficient flexibility for heterogeneous connectivity on a
large scale. Hence, this research project proposes to investigate the potential for leveraging the P2P paradigm for group applications in the area of CSCW and collaborative computing in the context of distributed heterogeneous networks.

Research in the area of collaborative computing and supporting middleware is currently both rich and diverse. A number of noteworthy projects exist in this area. Disciple [3] is a Java-based CSCW development and runtime platform that focuses on the areas of mobile and pervasive computing and promotes interoperability across a range of devices. Disciple supports synchronous real-time collaboration but has rejected conventional application-sharing approaches (i.e. where all applications share identical views) in favour of an abstraction based on the Model-View-Controller (MVC) model. This abstraction enables application data to be interpreted differently according to the capabilities of a particular device.

Rochester Institute of Technology (in the Anhinga Project) are developing a new object-oriented model called Many-to-Many Invocation (M2MI) [4] for building collaborative applications in the context of dynamic and ad hoc network environments. Similarly, Berkley Labs has developed the Pervasive Collaborative Computing Environment (PCCE) Project [5] – a collaborative framework that supports ad hoc collaboration between groups of participants that is designed to be cross platform.

At the level of collaborative middleware, NaradaBrokering is a resource brokering system that manages peer interactions using a system of cooperating brokers and events [6]. NaradaBrokering focuses on dynamic P2P networking with an emphasis on flexibility, fault tolerance, and performance (primarily in the area of Grid Computing) building on the publish-subscribe model and the notion of content-based routing.

2 OUR APPROACH

To make reasonable advances in this area, our research aims to develop a collaborative system that enables:

- Collaborators to form groups with shared interests and goals - The formation of groups of collaborators with mutual interests is a central theme in this project. Groups may be used to represent any community of users, from schools to universities and companies, or used to partition users according to some shared interest, such as clubs or discussion forums, and consist of arbitrary, dynamic and transient membership.

In addition, this research aims to enable collaborators to form secure domains of trust - as stipulated by English et al. [7] security in dynamic global collaborations is a major concern for the end user. The system should therefore enable users to form secure domains of trust and provide facilities for the creation and management of authenticated groups.

- Collaborators to form spontaneous sessions on demand - Whilst groups provide a persistent environment and scope in which users interact; sessions support synchronous real-time collaboration and communication between multiple participants over shared communication channels. Collaborators may be involved in a number of sessions with arbitrary membership.

- Awareness in the context of collaborative sessions - As Ackerman [8] suggests, collaborators prefer to be aware of both the presence and actions of others interacting in a shared space - where this awareness is used to guide the flow of work. Carstensen and Schmidt [9] stipulate that a collaborative computing middleware should provide facilities that support collaborative actors in “monitoring the state of affairs of the field of work.” Carstensen’s research underlines the importance of mutual awareness among actors.

- Sharing of arbitrary content and documents between users - Shared information, resources and content are important aspects of collaborative work that allow users to exchange information pertinent to some task or goal. Users should be able to share any type of document that is relevant to the task at hand in a scalable and robust manner, particularly in the context of heterogeneous and dynamic network environments.

The objective of this research is to develop collaborative methodologies that support the lifecycle of online workgroups, allowing such collaborations to be initialised, started, and stopped within the context of a particular task. The aim is to produce a development platform that manages the complexities of communication and synchronisation of collaborative workgroups on behalf of applications. Application developers will therefore be able to add support for collaboration to their applications with relatively little development effort via interfaces and APIs. The implementation will therefore provide an API that can be leveraged by developers to create collaboration-aware applications that are subsequently deployed within a runtime infrastructure that
is able to manage communication and interaction in a peer environment. The impetus is to develop a generic set of components and network services that will provide developers with a flexible groupware on which to build and deploy collaborative applications.

The goal of our research is to develop a supporting framework and tools that enable the creation of spontaneous multi-user collaborative sessions, allowing users to self-organise and communicate, share tasks, workloads, and content, and interact in a diverse network environment. Our aim is to support interaction in both ad hoc and continuous forms as well as in synchronous and asynchronous modes. Our approach focuses primarily on heterogeneous connectivity (i.e. how we can enable collaboration across both network and operating system platforms) and scalability (i.e. how collaborations can organically scale with the number of participants). We have therefore been building on two pivotal technologies: Java and JXTA [10].

Java was selected as the development and runtime environment as it offers a number of desirable features in relation to the design and implementation of a collaborative platform, including cross platform compatibility, portability, integrated security, multithreading, networking, and code mobility. However, it is understood that these features do not come without cost due to Java’s interpreted nature.

JXTA [11] consists of a set of interoperable protocols that standardise the ways in which peers discover resources and services, organise into peer groups, communicate, and monitor each other with the goal of supporting homogenous connectivity between peers. We determined JXTA to be the most suitable platform for the development of our underlying communications infrastructure due to the flexibility afforded by the JXTA protocol layer in terms of both network interoperability and scalability, allowing us to focus less on protocol issues and more on collaborative service/application development and design. Another advantage of JXTA is the beneficial attributes that it has been shown to exhibit in terms of fault tolerance, resilience, and scalability and research in this area is still ongoing [12].

3 CONCEPTS

Coco is built on two fundamental elements that sit on top of the network middleware: components and services. Components represent network objects that can be exchanged by peers in the process of communication. These objects are always specified using a platform neutral representation with XML. Coco services (currently exposed through a Java API) support interaction and communication within a P2P context, forming an underlying communications framework that can be built on by applications. The set of services supported by a particular peer therefore provides the network interface that peer has with other peers on the network. The component/services design approach is consistent with our desire to create a scalable and modular architecture that can be run on a range of devices supporting a variety of capabilities.

We have defined a set of components to include:

- **Groups** – the formation of groups of collaborators with mutual interests is a central theme in this project. Groups may be used to represent any community of users, from schools to universities and companies, or used to partition users according to some shared interest, such as clubs or discussion forums, and may consist of arbitrary, dynamic and transient membership. The Coco platform provides mechanisms that allow users to create and join arbitrary groups that act as the scope for collaboration.

- **Users** – a user by our definition is an individual running an instance of the Coco platform at particular point in time. User awareness within sessions is also supported enabling collaborators to see information relating to a user’s status at any particular time.

- **Sessions** – sessions allow users to form spontaneous collaborative groups. The purpose of a session is to offer an environment in which users can interact over a shared task. Every session has both a session state and session membership. The state of a session consists of the combined view of all participants in the session. There may be multiple sessions in a particular group and users may belong to multiple groups.

- **Channels** – channels provide the medium of communication and build on the underlying facilities provided by JXTA. The platform provides three types of communication channels: Unsecured one-to-one channels, secure one-to-one channels and shared multicast channels that are integral to communication within sessions.

- **Content** – content consists of any type of document, file, or network identifiable resource that can be shared or invoked over the network. The system is based on the manipulation of metadata representations of resources that contain information relating to a particular resource. As an integral part of the Coco service layer we have developed a fully distributed resource description and search system that can be used to describe and locate network resources in a generic way. Our implementation builds on the Resource Description Framework (RDF), a language for describing resources on the network using
The purpose of RDF is primarily to represent metadata about network (normally Web) resources. The Coco Metadata Content Service builds on RDF to describe resources in a P2P context using globally unique universal resource names (URNs).

Applications – session applications build on the facilities and components of the Coco platform using the service interfaces and APIs provided. Our design treats applications as a network resource using a representation that, like files and documents, can be shared between peers and incorporated into sessions, allowing the platform to search for and dynamically instantiate applications when a user wishes to join a particular session.

The architecture is designed to provide a number of services (currently exposed through a Java API) that support interaction and communication within a P2P context, forming an underlying communications framework that can be built on by applications.

Coco is not designed to access the facilities of the underlying network directly, but instead relies on the network abstraction provided by JXTA to propagate messages on its behalf. Equally, the Coco platform does not directly maintain network connections between peers, but builds on the JXTA virtualisation of the underlying network via pipes and endpoints.

4 ARCHITECTURE

Coco is designed to be both a development environment, through a set of component objects and a service API, and a runtime environment that manages the creation of multi-user collaborative sessions. The design of the Coco platform is based on a layered architecture, illustrated by figure 1 that builds on the features of JXTA and Java.

The Coco service layer exposes a set of pluggable network services that currently includes a distributed metadata resource description/search service (or content service), a file transfer service, a messaging service (supporting instant messaging as well as video and audio conferencing), and a presence monitoring service (called an interaction service) for monitoring the state of a collaborative session.

5 COCO FOR MOBILE ENVIRONMENTS

We are also developing a scaled down version of Coco, called MicroCoco, which enables users to access Coco services through mobile phones and PDAs.

Mobile devices have significant resource constraints compared to that of desktop and enterprise devices. In addition to this, these constraints vary across the range of devices. For example, a mobile phone tends to have less hardware capability than a PDA. In general, mobile devices have the following limitations:

- Significantly less processing power
- Limited memory
- Little or no persistent memory
- Very small screens with limited methods of interaction
- Constrained choice of communication protocol
- Lower network bandwidth and higher network latencies

Java’s solution to these hardware constraints is the Java 2 Platform Micro Edition (J2ME) [13]. J2ME provides an optimised Java Virtual Machine (JVM) that uses a subset of the full set of Java libraries. To account for the range of device capabilities, J2ME defines a set of configurations for a JVM and profiles that a developer uses. A configuration defines the features of a JVM and the core Java classes that are available on a particular category of device. A profile is a definition of the set of Java API’s that are available to a developer on a particular category of device. The most widely used configuration and profile is the Connected, Limited Device Configuration (CLDC) [14] and the Mobile Information Device Profile (MIDP) [15] and JXTA for J2ME (JXME) has been implemented using CLDC/MIDP.

JXME is a scaled down version of JXTA that supports a subset of the protocols used [16]. Currently, JXME uses a JXTA relay as a proxy to the JXTA peer network. This allows the bulk of the processing to be handled by a standard peer, such as a desktop PC. JXME peers send queries and messages to the relay and continually poll the peer for responses.

The use of a relay does not compromise the fundamentals of P2P, as illustrated by figure 2. This is because JXME peers do not have to maintain a static relationship with one single relay; the peers can change relays dynamically or can be connected to multiple relays.
As JXME has been designed to be interoperable with JXTA, MicroCoco is being built upon this platform. The services provided by MicroCoco are a subset of the full platform. This is because the hardware constraints allow a limited amount of bandwidth for communication with other full Coco peers. For example, a group of Coco peers may be able to participate in video conferencing; however, a MicroCoco peer running on a mobile phone may not necessarily have the bandwidth to deal with an incoming video stream. A MicroCoco peer could however participate in sessions that require more asynchronous communications such as instant messaging, content sharing and co-authoring.

Figure 2. JXTA and JXME Peer connectivity.

JXME has been designed for use on MIDP 1.0 devices. MIDP 1.0 devices and widely used and most Java-enabled mobile phones are MIDP 1.0 compliant. The JXME implementation can however be used in higher-end J2ME profiles such as the Personal Profile [17] that targets higher capability mobile devices such as PDAs. A MIDP 2.0 version of JXME is currently being developed that will allow for proxy-less P2P communication. Once a stable release of proxy-less JXME is available, a corresponding version of MicroCoco will be developed.

6 COCOSPACES: COCO IN ACTION

The CocoSpaces project builds on the Coco platform to provide a set of collaborative tools in an integrated collaborative environment. CocoSpaces is a collaborative application, developed in Java, which builds on the facilities of the Coco platform. The concept is that of the integration of virtual collaborative workspaces which are created and shared by users in a fully decentralised manner. CocoSpaces builds fully on the service set of the Coco API, enabling users to search for and join collaborative spaces, search within spaces for collaborators and content, and organise their spaces into manageable categories. The application supports instant messaging between participants and allows participants to share documents using the Coco services, as well as allowing collaborative applications to be plugged into workspaces and shared between participants.

7 CONCLUSIONS

The P2P approach has both strengths and weaknesses. The non-determinism and dynamic connectivity that characterises true P2P networks is a significant benefit in supporting spontaneous collaborations. In particular, our approach provides a number of distinct benefits in contrast with existing platforms in terms of:

- Self-organisation – enabling dynamic collaborations within secure user-defined domains.
- Heterogeneity – building on platform-agnostic technologies (i.e. XML, Java, JXTA) to support diverse devices, operating systems, and networks.
- Scalability – enabling large-scale collaborations overcoming common limitations of scale in collaborative tools.

Our experience indicates that the decentralised approach works very well for both messaging and content distribution. These examples demonstrate clearly that functionality which might traditionally have been provided by a central server can be offloaded to clients and pushed to the edges of the network, where there is a greater abundance of network resources (such as bandwidth and storage). P2P therefore affords greater scalability and robustness in these services.

8 REFERENCES


