# PROVIDING PEER-TO-PEER FEATURES TO EXISTING CLIENT-SERVER CSCW SYSTEMS

Bernd Eßmann
*Heinz Nixdorf Institute*
*University of Paderborn*
*Fürstenallee 11, 31102 Paderborn*

Holger Funke
*University of Paderborn*
*Fürstenallee 11, 31102 Paderborn*

Keywords: CSCW, CSCL, Peer-to-Peer, JXTA, Mobile Computing, Spontaneous Collaboration.

Abstract: Developers of classical client-server CSCW systems are facing a true dilemma: They created a working cooperation environment for many scenarios of cooperative work. Since users get independent from fixed places by using mobile devices interconnected by ad-hoc networks, the support of mobility becomes an important topic of CSCW. Furthermore, while client-server architectures do not work well in dynamic networks, P2P systems enter the field of CSCW. But, is it a good approach to discard the well working client-server system in order to implement a brand-new P2P system from scratch? Our approach is extending our CSCW platform step-by-step with P2P abilities without loosing the advantages of client-server computing. This paper describes the first step of wrapping the RMI-based communication protocol into an industry standard P2P protocol called JXTA.

## 1 INTRODUCTION

Client-server architectures found in classical CSCW systems always need central servers available in the used network infrastructure. But in mobility supporting networks like Multi-hop Ad-hoc Networks (MANETs) (Perkins, 2001) the network structure changes dynamically and not predictable. The server becomes an unreliable single point of failure because its accessibility can not be guaranteed.

Thus Peer-to-Peer (P2P) architectures are used for getting rid of this single points of failure. Our approach is to build an hybrid architecture without obligatory dependencies between client-server and P2P environment. Servers are used for providing the cooperation environment when available and a P2P architecture stands in for providing vital functionality when not. This hybrid architecture provides the advantages of both worlds for supporting the new scenarios of mobile cooperation as well as classical CSCW scenarios.

Our first step to provide this hybrid cooperation environment is extending our existing client-server CSCW platform *open*sTeam with P2P features by enabling the stateful communication protocol to be used in P2P communication. We will show that the basic features of many classical CSCW platforms can be transformed to work with P2P systems.

## 2 RELATED WORK

*Speakeasy* today known as *Obje* is a framework for communication in heterogeneous network environments with several devices and applications (Edwards et al., 2002b). The implemented approach is called *recombinant computing*, which means providing fixed domain-independent communication interfaces and mobile code. *Casca* is an CSCW application using the Speakeasy technology for cooperative work (Edwards et al., 2002a). It allows discovering and selecting possible partners for collaborative work but provides no semantic or object-oriented document structure like virtual knowledge spaces.

Groove[1] is implemented as a P2P CSCW platform. Users are enabled to work together in a shared workspace. Looking at the architecture for synchronizing the distributed workspaces Groove still requires servers.

[1] http://www.groove.net



Eßmann B. and Funke H. (2005).
PROVIDING PEER-TO-PEER FEATURES TO EXISTING CLIENT-SERVER CSCW SYSTEMS.
In *Proceedings of the Seventh International Conference on Enterprise Information Systems*, pages 271-274
DOI: 10.5220/0002528502710274

Magi Enterprise[2] is a commercial product supporting teamwork by allowing communication and a fast exchange of information. For this purpose, virtual workspaces accessible for all team members are constructed by enabling Microsoft products with P2P functionality.

Swifff[3] is a P2P framework providing the necessary infrastructure for managing knowledge in dynamic network environments (Eßmann et al., 2004). Since its still in heavy research many mechanisms of sophisticated CSCW environments like found in sTeam are missing.

## 3 STEAM GOES P2P

On every client access to the sTeam server the proprietary *COAL* protocol is used to transform the messages into byte sequences that are being sent to the server. The server transforms the byte sequence into a message which can be processed. The answer to the client runs accordingly (Bopp, 2003). The main objective for enabling sTeam for P2P communication is to transform this stateful protocol for stateless communication as used in P2P networks.

The open-source project JXTA is a peer-to-peer-platform consisting of a collection of different protocols. These protocols allow every device within a network – a mobile phone, a PDA, a PC or a server – to communicate and cooperate according to a P2P network. JXTA intends to provide methods that enable every node in a network to gain access to all data, contents and other nodes even when hid behind a firewall (Groth, 2002). All these methods are comprised in an open code library (Wilson, 2002).

JXTA was choosen because it not just allows to find other peers in the network but also provides the possibility to communicate in any heterogeneous network. JXTA builds new virtual networks on the basis of already existing physically. Figure 1 depicts such a mapped network (Traversat et al., 2003).

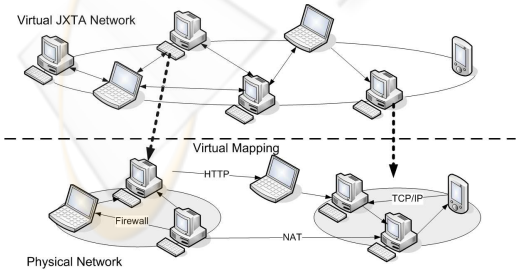


Figure 1: Mapping in a JXTA network.

[2]http://www.endeavors.com
[3]http://swifff.upb.de

For communication purposes JXTA provides pipes. Pipes are virtual channels for communication which interconnect peers with each other even if there is no direct physical connection. In this cases relay peers are being applied. A pipe can be installed between two peers (point-to-point) or between a single peer on the one end a hole group of peers on the other end (propagate). Communication that takes place by way of pipes always runs asynchronously and unidirectionally which makes pipes difficult to use for classical CSCW systems. Thus, JXTA offers only basic mechanisms for communcation purposes (Oaks et al., 2002).

## 4 IMPLEMENTATION

To use the P2P extension of our implementation it is necessary to establish a P2P network. The user can activate the extension automatically while starting the sTeam client. It is the possible to work in the classic mode of sTeam or in the new mode with P2P functions. When starting sTeam in P2P-mode methods will be invoked setting up the P2P network. With these methods the peers are able to search and find other peers of the same service type in their subnet. To use these connections in the future, connections are saved locally at each peer. During the setup of the network these calls are used very often. Once the network is established the frequency of these method invocations will be reduced to decrease the traffic overhead. Thus a new P2P network is established which is the base for the further process.

In the P2P network exist some peers with a special role. In the setup phase they scan their neighborhood for other peers in order to create a sTeam P2P network. In difference to the normal peers they own a direct connection to a sTeam server and adopt the role of an *agent peer*, communicating between the P2P network and the sTeam network. An agent peer offers other peers a service to interact with the connected sTeam server. The agent peer starts as a normal peer. Starting as a normal peer the peer recognizes its specific role during the runtime. Because of this self managing there is no matter to configure the agent peer. Because the agent peer acts like a proxy for the server, the server becomes a service in the P2P network. An example of P2P network can be found in figure 2.

The agent peer publishes its service by means of a Peer-Discovery-Protocol via an according advertisement that describes the respective service. As soon as the peers looking for this advertisement find it, they can make use of it. This process is similar to any other service in the JXTA network (Gradecki, 2002).

One important criterion of the implementation of

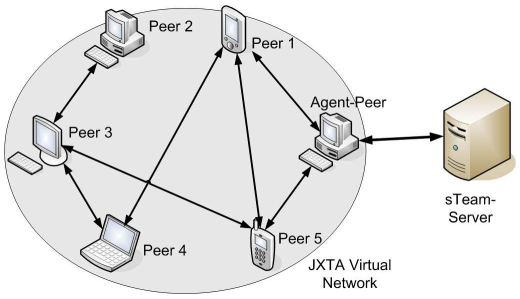


Figure 2: Network of sTeam-Peers.

the agent peer deals with the performance of the mesage conversion.

The communication between the two different networks is the core element of the P2P extension. Some methods are necessary for basic communication and file transfer. Further methods translate the messages between the two formats JXTA and COAL. All this methods are provided in the classes `JxtaMessage` and `JxtaMessageWrapper`.

Messages are exchanged between two different network types. In both networks there are different formats of messages available, on the one hand JXTA messages and on the other COAL messages. Therefore, the agent peer must be equipped with a facility altering the different message types. We call this facility `JxtaMessageWrapper`. Our implementation has a minimal transformation overhead, so that the transformation takes place very quickly. In fact, the user will not notice any difference between the delivery of a sTeam message in a classic client/server system and the delivery of a JXTA message which has to be transformed into a sTeam message on its way to the server.

A message transmitted by the COAL protocol used by sTeam is composed of a identifier, the message type, an object the message refers to and a number of attributes. Additionally the message ist tagged with a unique sequence number to assure the correct sequence of execution.

These message elements are extracted by the agent peer and transfered in a JXTA compatible format. The extracted elements generate the new JXTA-Message.

Figure 3 dispicts the way of a message from the client to the server and back again. The message is beeing generated as so far common by the `SteamConnector`, `MessageManager` and `ServerConnection`. After the message is finalized, there is no need to transfer it as a `OutputStream` to the server. Instead of this, the message will be passed to the `JxtaMessageWrapper` which carries the JXTA message to the P2P network. Arriving at the agent peer, the Message will be retranslated to the COAL format and delivered to the server. The back way takes place analogous.

The basic idea connecting the sTeam network in a fast way with the P2P network is wrapping COAL into the JXTA protocol. This allows to transmit messages in both networks. As said, wrapping between those protocols is applied by the agent peer. With a closer look on both networks, the following three problems appear:

- The communication of the COAL protocol in sTeam is synchronous compared to the asynchronous communication in JXTA networks.
- In JXTA the messages will arrive at the receiver in no order, but in COAL the order is well definded.
- sTeam provides events to the clients which also have to be processed in the correct order.

JXTA already provides a mechanism for synchronous communication called *bidirectional pipes*. Unfortunately they provide a very slow performance. In a simple measurement the factor 20 appeared as a average factor of slowdown in communication. Thus we implemented a new message type basing on the asynchronous concept of JXTA but extending it with synchronous features.

The sender transmits a message on a standard pipe and extends this message with a return address. On this return address the sender waits for a acknowledgment by the receiver. When receiving the message, the receiver creates a new pipe with the return address and sends a acknowledgment. This allows asynchronous communication with two standard pipes in a much faster way than with the bidirectional pipes.

To execute the messages in the right sequence in sTeam, messages in COAL have a sequence number. This concept is retained in the JXTA-Network. The wrapper of the COAL message uses the same sequence numbers as the COAL message; this way the well defined sequence of messages in sTeam is guaranteed.

The events send by the sTeam server are treated like any sTeam message. Events arrive as COAL messages from server to agent peer and are wrapped as JXTA messages and thus reach the peers. The sequence of the events is guaranteed by the same control sequence as used for any COAL message.

The fast message conversion and the automatic configuration of the P2P functionality lead to a high grade of transparency for the user. From the user's point of view it is of no importance whether he is connected to the server directly or via a P2P connection. Yet, it is a great advantage that he can profit by the flexibility offered by a P2P network. Although he still needs access to a sTeam server he may connect him through firewalls or from within networks with network address translation (NAT) making him usually unreachable from outside. While COAL would not allow this communication, this functionality is inherited from the JXTA protocol.

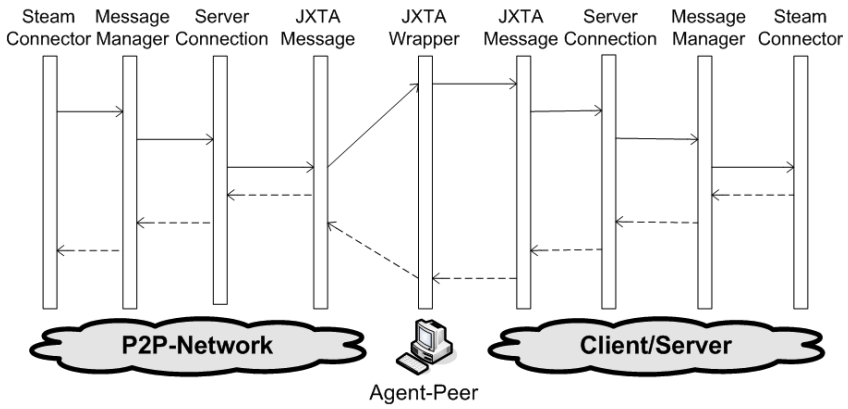


Figure 3: Message flow between two different networks.

## 5 CONCLUSION

In this paper we presented our approach opening a classical client-server CSCW system to the world of P2P. The first step of bringing P2P features to client-server CSCW systems is allowing their clients to communicate over P2P protocols. Classical CSCW systems feature special characteristics which have to be accounted to when wrapping them into stateless P2P protocols. We presented solutions for carrying the stateful communication of the client-server CSCW system sTeam into the stateless communication of JXTA's P2P world without loosing any functionality. Neither the client nor the server had to be changed in their core functionality.

Our future steps towards a fully mobility aware cooperation environment will be to equip the clients with basic features of the server. This will the clients transforming to real peers which can run standalone without any server in the network. The needed functionality includes e. g. management functionalities and a local repository for often needed cooperation objects.

Although it might be a possible future developing a sophisticated cooperation platform basing just on P2P technologies, our approach allows providing an always usable cooperation environment with evolving features for new cooperation scenarios. Thus we presented the first step toward an universal environment providing the technical infrastructure for cooperation support in everyday life situations.

## ACKNOWLEDGEMENTS

Bernd Eßmann is member of the postgraduate program 776 "Automatic Configuration in Open Systems" funded by the German Research Foundation (DFG) and the Heinz Nixdorf Institute.

## REFERENCES

Bopp, T. (2003). steam server documentation. http://www.open-steam.org/Dokumente/docs/-Programer_Documentation.pdf (31.01.2005).

Edwards, W. K., Newman, M. W., Sedivy, J. Z., Smith, T. F., Balfanz, D., Smetters, D. K., Wong, H. C., and Izadi, S. (2002a). Using speakeasy for ad hoc peer-to-peer collaboration. In *conference on Computer Supported Cooperative Work (CSCW'02)*, pages 256–265, New Orlando, Louisiana, USA. ACM Press, New York, NY, USA.

Edwards, W. K., Newman, M. W., Sedivy, J. Z., Smith, T. F., and Izadi, S. (2002b). Challenge: Recombinant computing and the speakeasy approach. In *International Conference on Mobile Computing and Networking (MOBICOM'02)*, pages 279–286, Atlanta, Georgia, USA. ACM Press, New York, NY, USA.

Eßmann, B., Hampel, T., and Slawik, J. (2004). A jxta-based framework for mobile cooperation in distributed knowledge spaces. In Karagianis, D. and Reimer, U., editors, *5th International Conference on Practical Aspects of Knowledge Management (PAKM'04)*, Lecture Notes in Artificial Intelligence, pages 11–22, Vienna, Austria. Springer Verlag.

Gradecki, J. D., editor (2002). *Mastering JXTA: Building Java Peer-to-Peer Applications*. Wiley Publishings, Indianapolis, USA.

Groth, T. (2002). Project jxta. In Schoder, D., Fischbach, K., and Teichmann, R., editors, *Peer-to-Peer*. Springer-Verlag, Berlin, Deutschland.

Oaks, S., Traversat, B., and Gong, L., editors (2002). *JXTA in a Nutshell*. O' Reilly, Sebastopol, USA.

Perkins, C. E. (2001). *Ad Hoc Networking*. Addison-Wesley, Boston, USA.

Traversat, B., Arora, A., Abdelaziz, M., Duigou, M., Haywood, C., Hugly, J.-C., Pouyoul, E., and Yeager, B. (2003). Project jxta 2.0 super-peer virtual network.

Wilson, B. J., editor (2002). *JXTA*. New Riders, Indianapolis, USA.