PeerWare: A Peer-to-Peer Middleware for Mobile TeamWork

Carlo Ghezzi, Gianpaolo Cugola, and Gian Pietro Picco
Dipartimento di Elettronica e Informazione—Politecnico di Milano
Piazza Leonardo da Vinci, 32, 20133 Milano, Italy.
Tel.: +39-2-23993493 Fax: +39-2-23993411
{ghezzi, cugola, picco}@elet.polimi.it

May 19, 2003

Abstract

The advantages of a peer-to-peer architecture reach well beyond the realm of Internet file sharing, becoming key in supporting enterprise processes and especially collaborative work involving mobile users. To support this view, we designed and experimented with PeerWare, a core communication middleware for TeamWork applications.

1 Peer-to-peer for collaborative applications

Collaborative work is intrinsically peer-to-peer in nature. Members of a team typically interact directly with each other, each of them is responsible of a given set of documents and carries along the subset of them relevant for discussion. On the other hand, most of currently available tools supporting collaboration exploit a rigid client-server architecture.

This results in an "architectural mismatch" between the external view provided by the application and its internal software architecture. The effect of such a mismatch is a lack of flexibility in carrying out the interactions, which must all be funneled through the server. This limitation becomes even more evident when mobility becomes part of the picture. People need to communicate and collaborate even while in movement, and independently from the place where they are. Nevertheless, in similar situations server access is often prevented by technical or administrative barriers.

As a consequence of the above considerations, we argue that a peer-to-peer approach holds significant advantages over traditional client-server architectures. When a peer-to-peer architecture is adopted, data and services are no longer gathered in a single point of accumulation. Instead, they are spread across all the nodes of the distributed system. Users may directly host the resources they want to share with others, with no need to publish them on some server.

Interestingly, these features are relevant not only in mobile scenarios but also in fixed ones, where the decentralized nature of a peer-to-peer architecture encompasses naturally the case of multisite or multicompany projects, whose cooperation infrastructure must span administrative boundaries, and is subject to security concerns.

Unfortunately, however, most of the peer-to-peer applications developed in the last years start from premises that are rather different from those outlined thus far. They target the Internet and aim at providing peer-to-peer computing over millions of nodes, with file sharing as their main application concern. The difference in perspective from the domain of collaborative work is made evident by their search capabilities, that typically do not guarantee to capture information about all matching files. Moreover, in most cases they do not take into consideration characteristics like security or the ability to support reactive interactions, which are crucial in cooperative applications for the enterprise.

Moreover, they bring peer-to-peer to an extreme, where the logical network of peers is totally fluid: none of them can be assumed to be fixed and contributing to the definition of a per-

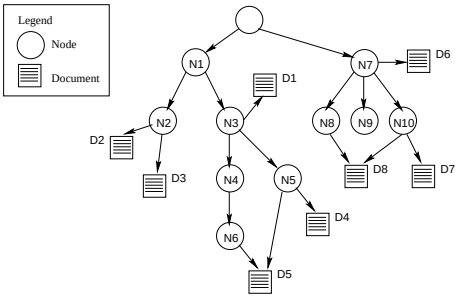

Figure 1: The data structure provided by PeerWare.

manent infrastructure. This radical view prevents access to the resources exported by non-connected peers, which is not acceptable in the enterprise domain we consider, where critical data is often required to be always available, independently of its owner.

2 PeerWare

Based on the above considerations, we developed PeerWare [1]: a peer-to-peer middleware for teamwork support specifically geared towards the enterprise domain.

PeerWare is both a *model* and an *incarnation* of this model into a middleware. In developing both, our first concerns were *minimality* and *flexibility*.

The model. The PeerWare coordination model exploits the notion of a *global virtual data structure* (GVDS) [2], which is a generalization of the Lime [3, 4] coordination model. Coordination among units is enabled through a data space that is transiently shared and dynamically built out of the data spaces provided by each accessible unit.

The data structure managed by PeerWare is a hierarchy of *nodes* containing *documents*, where a document may actually be accessible from multiple nodes, as shown in Figure 1. This structure resembles a standard file system, where directories play the role of nodes, files are the documents, and Unix-like hard links are allowed only on documents.

When a peer is isolated, it is given access only to its own tree (stored locally) of items(i.e., nodes and documents). However, when connectivity

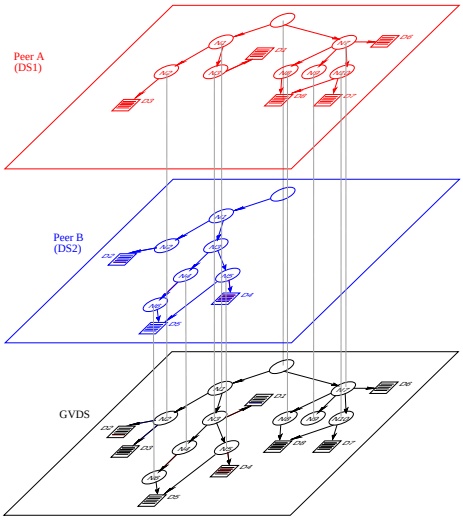

Figure 2: Building the GVDS in PeerWare.

with other peers is established, the peer has access to the virtual tree constructed by superimposing the trees contributed by all the peers in the system, as illustrated by Figure 2.

In search of minimality, PeerWare provides only three main operations to operate on the GVDS:

- the *execute* operation allows peers to execute an arbitrary piece of code on a selected set of items held by connected peers. The results produced by this execution are collected and given back to the caller;
- the *subscribe* operation allows peers to subscribe to events occurring on a selected set of items, while;
- the *publish* operation allows peers to notify the occurrence of events.

By exploiting these primitives peers can query the GVDS, as well as subscribe for events and receive the corresponding notifications. The hierarchical structure of the GVDS provides a natural scoping mechanism, thus leading to an efficient implementation of searches.

The middleware. Currently, the PeerWare model has been implemented in two middleware: one [1], developed in Java, has been used as the

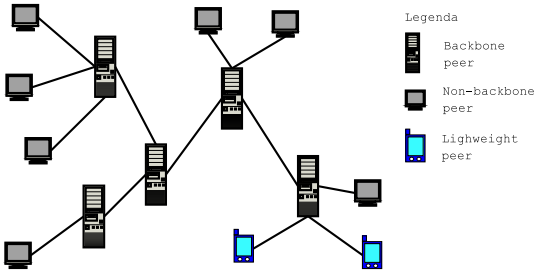

Figure 3: The PeerWare run-time architecture.

core of the motion platform [5, 6, 7], the other, developed in C# under the Microsoft .Net infrastructure, is the core of the PeerVerSy [8] configuration management tool.

Both implementation are tailored to the enterprise domain and distinguish between a set of permanently available *backbone peers*, and a fringe of *mobile peers*, which are allowed to connect and disconnect as required. To optimize routing, all these peers are connected to form an acyclic graph in which the mobile peers represent the leaves, as shown in Figure 3.

Access control and security are critical issues in the domain we target and they are addressed by two separate modules. One provides mechanisms to establish encrypted channels among peers and to manage the security information necessary to authenticate a peer. The other embeds the actual security policy that determines the capabilities of a given peer.

To increase flexibility, the functionality provided by the security modules and also by the repository holding local documents are sharply decoupled from the specific implementation provided for these functionalities. Thus, the security protocols, as well as the format of the security information used to perform authentication, and the repository effectively used can be changed easily, e.g., to adapt them to the common practice of a specific business environment.

3 Conclusions

Collaboration defines a scenario where interaction is intrinsically peer-to-peer. We exploited this idea by developing PeerWare, a peer-to-peer middleware explicitly tailored for collaboration. The model of collaboration adopted by PeerWare is minimal but expressive enough to support complex collaboration schemes, including both reactive and proactive interactions. Current incarnations of this model in a middleware have been tailored to the domain of enterprise-wide collaboration. We are working on a different implementation oriented toward more dynamic scenarios, including ad-hoc networks.

References

[1] "Peerware web site." `http://peerware.sourceforge.net`.

[2] A. Murphy, *Enabling the Rapid Development of Dependable Applications in the Mobile Environment*. PhD thesis, Washington University in St. Louis, MO, USA, August 2000.

[3] G.P. Picco, A. Murphy, and G.-C. Roman, "Lime: Linda Meets Mobility," in *Proc. of the 21st Int. Conf. on Software Engineering (ICSE'99)* (D. Garlan, ed.), (Los Angeles, CA, USA), pp. 368–377, ACM Press, May 1999.

[4] A. Murphy, G.P. Picco, and G.-C. Roman, "Lime: A Middleware for Physical and Logical Mobility," in *Proc. of the 21st Int. Conf. on Distributed Computing Systems (ICDCS-21)*, May 2001. To appear.

[5] "The web site of the motion project." `http://www.motion.softeco.it/pages/`.

[6] G. Cugola and G. Picco, "Peer-to-peer for collaborative applications," in *Proc. of the International Workshop on Mobile Teamwork Support*, (Vienna, Austria), IEEE press, July 2002.

[7] G. Reif, E. Kirda, H. Gall, G. Picco, G. Cugola, and P. Fenkam, "A web-based peer-to-peer architecture for collaborative nomadic working," in *Proc. of the 10th IEEE Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE)*, (Boston, MA, USA), IEEE Computer Society Press, June 2001.

[8] "Peerversy web site." `http://sourceforge.net/projects/peerversy/`.