

The PostIt Application

Raphaël Collet¹, Valentin Mesaros¹, and Celia Franco²

¹ Université catholique de Louvain, B-1348 Louvain-la-Neuve, Belgium. {raph, valentin}@info.ucl.ac.be

² INRIA Rocquencourt, 78153 Le Chesnay Cedex France. Celia.Franco@inria.fr

Abstract. The paper presents a collaborative application that demonstrates the functionalities of the peer-to-peer library P2PS. The application aims at sending messages with a limited lifetime to groups of users. We investigate a possible way to use P2PS for group communication, and show the appropriateness of the library for such applications.

1 Introduction

Distributed collaborative applications became part of our everyday life, but the development of such applications is still difficult. Robustness is hard to achieve, and efficiency is often poor. The classical client-server architecture revealed to be inappropriate in general for such applications. Advances in peer-to-peer computing give more insights on how to deploy applications that achieve better efficiency, and do not rely on single points of failures. This paper explores the peer-to-peer library P2PS, developed in Mozart at UCL and CETIC [1].

We develop PostIt, a distributed collaborative application to show the functionalities provided by P2PS. Users of PostIt share a virtual board for publishing short time lived texts. Each message is addressed to a group of users, and has a limited lifetime. We use the P2PS library as a group communication primitive, as it implements an efficient broadcast mechanism among the nodes of the peer-to-peer network. We also use the *lookup* facility in order to provide fast access to shared information.

To summarise, the paper explains *how* our application uses P2PS to achieve its aim, and *why* the library is appropriate for such an application.

Paper organization. Section 2 describes the application in general, its goals and user interface. Section 3 presents the P2PS library, and Sect. 4 gives our application's architecture, together with insights on how we plan to implement the functionalities on top of P2PS.

2 Definition of the Application

The main functionality of our application is to allow groups of users to communicate. The users exchange short messages with a limited lifetime. Messages can be sent to one or a subgroup of users. This section first defines more precisely the functionalities of the application, then describes its user interface.

2.1 Functionalities of PostIt

The PostIt application considers three kinds of entities, namely the user, the group of users, and the message.

A user is identified by a *name* in the application.

A group is a set of users. Each group is itself identified by a name. A given user may be member of several groups. The name of a user can be thought of as a group containing that user only.

A message is a short text sent by a user to another user or a group of users. A message is also called a *postit*. The message is given a duration by its sender, after which it is *expired*. Before this happens, the message is said to be *active*. Messages are composed of two parts: the sender, destination, and duration form the *header* of the postit, while its *body* is given by the text.

We now define how these entities interact with each other.

- A user can send a postit to another user or a group of users. In the latter case, the sender must be member of the group to which the postit is sent.
- A user can change the contents of a postit. The header can be changed by the original sender only. The body can be extended by any user who has received the postit. So one can only add some text, no deletion or change is possible. Body extensions are not ordered.
- The user must be informed of the sender, destination, body, and remaining duration of each postit before it expires. The user only sees postits of groups he is part of.
- When a user joins a group, he should receive a copy of all active postits in this group.
- Active postits should be saved in a file. This will give some robustness to the application.

2.2 The Graphical User Interface

We describe here the windows that show the currently active postits, notify postit arrivals, create new postits, and configure various options.

Figure 1 shows the main window. It presents a list of the active postits. The postit creation window can be incorporated in the main window if the users decides it. The main window provides three menu



Fig. 1. The main window, with the creation window included.

items, named *File*, *Edit*, and *Help*. The most important is *File*, which contains the main functionalities of the application.

Create opens the window for creating a new postit.

Save saves all postits received so far. Available only when the corresponding option below is active.

Mask hides a postit whose body is currently being shown. The hidden postit is still visible in the active list.

History shows all the postits received so far. This entry is available only when the corresponding option below is active.

Group allows the user to create a new group, join an existing group, or leave a group he is currently part of.

Configure allows to setup various options, such as:

- the message arrival notification, which may appear as a popup window, or a beep, or simply by a visual differentiation in the message list.
- the presence of the creation window inside the main window.
- whether all received postits are kept (for “Save” and “History” above).



Fig. 2. The postit creation window.

The postit creation window is shown in Fig. 2. The user enters the destination, either a group, or a user; the message lifetime; and the message body.

Figure 3 shows the reception window. It is composed of the message header, which is modifiable by the postit's creator only, and the message body, including all the message extensions. Since the message header cannot be modified by all users, we provide two variants of the modification window: the postit's creator window (see Fig. 4), and the other users' window (see Fig. 5).

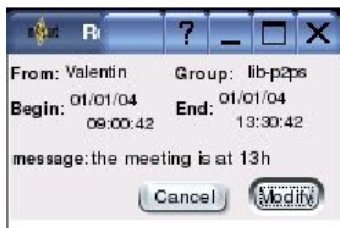


Fig. 3. The reception window when the "popup" option is active.



Fig. 4. The modification window for the user who sent the original postit.



Fig. 5. The modification window for the other users.

3 The Underlying P2P Platform

In this section we present the main functionality of the P2P platform, called P2PS [1], that we use as underlying infrastructure for our application. The P2PS library provides the developer with the possibility of building and working with P2P overlay applications, offering different P2P primitives and services. P2PS is providing the distributed peer-to-peer applications with a means to organize themselves in large scale structured overlay networks as well as providing them with management and communication primitives whose costs evolve logarithmically with the system size. For details on its API you can refer to the P2PS tutorial [1].

P2PS implements Tango [2], a peer-to-peer algorithm for structured P2P systems. Unlike unstructured P2P systems like Gnutella (gnutella.wego.com), whose overlay topology is ad-hoc, structured P2P systems organize their overlay by following well specified rules in order to improve overall efficiency. Tango provides overlay topology maintenance, self-organization, efficient data lookup, and fault-resilience. Tango is based on the idea of Distributed Hash Table – DHT, where key#value pairs are associated to nodes in the overlay network depending on the "distance" between the key *id* and the nodes' *ids*. Moreover, in order to achieve efficient data lookup, each node is connected to other $\log N$ nodes in the overlay network, where N represents the maximum number of nodes in the network.

The main functionality of the P2PS platform can be summarized as follows: network management primitives such as create, join and leave a network, communication primitives such as one-to-one, broadcast and multicast, and monitoring primitives.

3.1 Create a Network

This functionality provides the programmer with the possibility to create a P2P overlay network. It will create the first node of a network. What this actually means is the fact that an access point is created for this node. An access point represents an addressable entry point of a node. The access point can be published, thus allowing other nodes to connect and join the overlay network. Furthermore, at initialization a peer node is provided with message and event input streams on which messages from other nodes and respectively different node and network events will eventually be accessible.

3.2 Join/Leave a Network

When joining an overlay network, a peer node n needs to have the knowledge of an access point of another peer node p already present in the respective overlay network. The underneath protocols will actually join n to the network via the node p . The system will self-organize in order to guarantee overall efficiency and to remain resilient to node failures. As any other node in the overlay network, a new joined node will be associated an access point. Furthermore, once inside the network, a node may receive messages from other nodes from the network, and node and network events on the associated message and respectively event input streams.

Leaving an overlay network means implicitly disconnecting this node from all the other nodes it is connecting to in the overlay network. Underneath, a node will run a simple protocol to disconnect it from its neighbors. This will terminate the message and the event streams.

3.3 Message Sending and Receiving

P2PS provides end-to-end communication primitives. That is, sending messages from one peer node to another throughout the overlay network. Due to its organization, the system performs efficient key based routing. Thus, a message from a node s to a node d is routed throughout the overlay network according with the corresponding key lookup procedure, where d is considered a key. In P2PS the message sending and receiving are asynchronous, though the reliable send can be made synchronous. The provided communication primitives are: one-to-one best-effort and reliable, one-to-many best-effort: multicast and broadcast. Note that the multicast is explicit, i.e., one has to explicitly specify the destination nodes. In all cases, receiving a message at a node implies reading the message input stream at that node. The messages addressed to a node will appear on the associated message stream.

3.4 Monitoring

In a dynamic network, as in P2P overlay networks, being aware of the status and the changes with respect to the peer node and the network might be very useful for the upperlying application. P2PS provides a set of events on the event input stream associated with the peer node. These events indicate changes on the connections with the node's neighbors. Another way of monitoring the peer node is offered in the form of counters. For example, one can have information about the followings: the number of incoming and outgoing connections, the number of data and control messages sent by this node, the number of data and control messages forwarded by this node.

4 The Application's Architecture

The architecture of PostIt consists of three layered modules: the Graphical User Interface module (GUI), the Group Communication module, and the P2PS module. Each module only interacts with the modules right above or under it.

4.1 Functional Definition of the Modules

The GUI Module. This module mainly interacts with the Group Communication module to send and receive messages. Received messages are notified and shown, depending on the user's options. Messages are sent by the postit creation window. Other interactions are related to group management, when the user creates a new group or joins an existing group.

The Group Communication Module. This module is composed of the message manager and the group manager. The message manager contains a reception agent and a dispatch agent. The reception agent receives messages from the underlying layer and transmits them to the GUI. The dispatch accepts messages from the GUI, and queries the group manager in order to decide where to send it. The group manager basically consists in a table, that maps each group the user is part of to a communication channel. The table is updated when a new group is created, and when the user joins or leaves an existing group.

The P2PS Module. This module provides the communication primitives for the application. It allows to create a P2P network, join a P2P network, etc. We mainly use two communication primitives: one-to-one message send, and one-to-many broadcast send.

4.2 Mapping Groups to P2PS Networks

In this section we describe how we intend to use P2PS in order to form groups within PostIt. In PostIt we want to create groups that can be addressed by their names and where the group communication is efficient.

Recall that P2PS already provides group communication primitives: explicit multicast and broadcast. Nevertheless, while these primitives will be useful for our application, we still need more. That is, we need an implicit multicast method, where sending a message to a group does not require the knowledge of all members of that group. To this end, we chose to have a P2P network per group, where a node in a P2P system represents a PostIt process associated with a user. Moreover, a user (and consequently, the associated peer node) can be part of different groups. The same idea was proposed in [3].

Joining a group requires to find a P2P access point. It can be found by an external mean, for instance a web page updated manually or automatically by the users. Such an external source is necessary for the very first connection of a user. Once a user is connected, he can find access points for other groups via the application itself. For this it suffices that all users form one global group, like suggested in Fig. 6. The access point can then be found by performing a lookup inside the global P2P network.

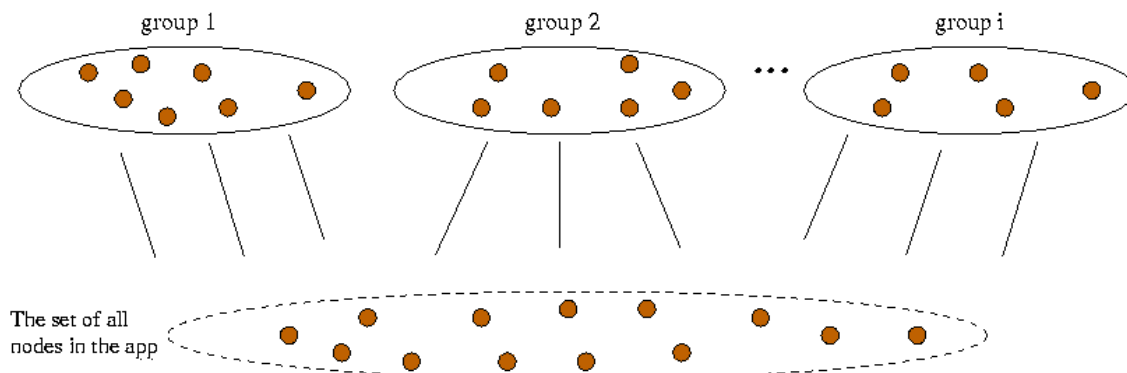


Fig. 6. Groups as P2P networks.

When a user joins a group, the application signals his presence by broadcasting a control message in the group's network. We propose to maintain the list of users of a group by a keep-alive mechanism. Every member of a group regularly broadcasts a control message in the group's network. A user from which no message has been received for a long period of time is considered to have left to group.

5 Conclusion

We have designed the collaborative application PostIt to be developed on top of a peer-to-peer network. The design is not specific to any peer-to-peer system, but the application's requirements make the P2PS

library quite appropriate for it. We have shown how to use P2PS to implement PostIt. The application's main functionality is to send messages to groups of users. P2PS provides enough flexibility to create as many P2P networks as groups. This allows to send messages to a group without having to know all the users in this group.

References

1. UCL, CETIC, Belgium: P2PS: Peer-to-Peer System Library. Available at http://www.mozart-oz.org/mogul/info/cetic_ucl/p2ps.html (2004)
2. Carton, B., Mesaros, V.: Improving the Scalability of Logarithmic-Degree DHT-based Peer-to-Peer Networks. In: Euro-Par. (2004) *To appear*.
3. Onana, L., El-Ansary, S., Brand, P., Haridi, S.: DKS: A Family of Low Communication, Scalable and Fault-Tolerant Infrastructures for P2P Applications. In: CCGRID. (2003)