A Distributed Instant Messaging Architecture based on the Pastry Peer-To-Peer Routing Substrate

Henrik Lundgren, Richard Gold, Erik Nordström, Mattias Wiggberg
Department of Information Technology,
Uppsala University, Sweden.

ABSTRACT

Current Instant Messaging systems rely on architectures that include central, or only partly distributed, server(s). Although some systems can exchange messages peer-to-peer, the user registration and lookup are still based on a centralized solution. We have implemented a *fully distributed instant messaging system* by utilizing the Pastry peer-to-peer routing substrate. We use the Pastry object insertion functionality to insert user information when a user joins the network and a novel search engine to effectively perform distributed user lookup. Currently, our system runs in the FreePastry simulator and supports joining/leaving the network, searching for users and message exchange.

1. INTRODUCTION

The increasing availability of cable and DSL services has made it possible for users to be continuously connected to the Internet. Users now wants to announce their presence to their friends so that they can "meet" online and communicate. For a long time, e-mail has been the common way to exchange messages on the Internet. E-mail is a store-and-forward system where messages are stored at intermediate nodes (mail servers) and only retrieved at user request. With increased user presence, it is now popular to use Instant Messaging (IM). IM systems deliver messages, as the name implies, instantly i.e., messages are sent directly to the recipient without any intermediate storage and thus allow for real-time text-based communication.

There exist several IM system today and depending on their architectures, messages can be forwarded via dedicated servers or, preferably, directly between the communicating parties (peer-to-peer). Table 1 shows a side by side comparison of the most common IM systems today. Among all IM systems, none is both fully distributed and decentralized in terms of registration, lookup and message delivery. Although some of them, like Jabber and IRC (with DCC), have some of the desired characteristics, they all suffer from a single point of failure where a malfunction will close the service, either by making it impossible to find clients or deliver messages.

Peer-to-peer (p2p) systems provide powerful platforms for a variety of decentralized services, such as network storage and content distribution. Pastry [4] is a peer-to-peer object location and routing substrate based on a self-organizing overlay network.

Protocol	Reg./Lookup	Msg. exchange
AIM/OSCAR	CS	CS
ICQ	CS	p2p
IRC (DCC)	Distr. servers	CS/p2p
Jabber	CS	p2p
MSN IM	CS	CS
Yahoo	CS	CS

CS=Centralized Server, p2p=peer-to-peer

Table 1: Comparison of registration, lookup and message exchange mechanisms in different IM protocols.

We have designed a distributed instant messaging architecture (DIMA) and implemented a proof-of-concept prototype. The DIMA system runs on top of Pastry and includes a search engine for the p2p network. With our system it is possible to perform all necessary operations (join/leave, lookup, message exchange) without any centralized servers.

2. PASTRY

Pastry [4] is a generic peer-to-peer object location and routing scheme. It is a scalable, decentralized and self-organizing overlay network that automatically adapts to arrival, departure and failure of nodes. In this section we present a short overview of the Pastry design to provide the necessary background for understanding our DIMA system.

2.1 A Pastry Node

Each node joining the Pastry overlay network is assigned a random 128-bit node identifier (*nodeId*) within circular space 0 to $2^{128} - 1$. The distribution of nodeIds is assumed to be uniform, which can be achieved by basing the nodeIds on cryptographic hashes of the nodes public keys or IP addresses.

Each node maintains a routing table which is organized into $log_{16}N$ rows with 15 columns, where N is the number of nodes in the overlay. [1] The entries in a row n each refer to a node whose nodeId shares the n first digits but not the *n+1th* digit with the present node. Associated with each routing entry is the IP address of the closest node (e.g., according to round trip time) that have the appropriate prefix. If no such node is known, the entry is left empty.

[1] strictly, the number of rows and columns are $log_{2^b}N$ and $2^b - 1$ respectively, where normally b=4.

Each node maintains IP addresses for nodes in its *leaf set*. A leaf set contains the $L/2$ numerically closest larger nodeIds and the $L/2$ closest smaller nodeIds, relative to the present node (L typically has the value 16). The leaf set is used during message routing as described in the next subsection.

2.2 Routing

In Pastry, messages are routed according to a longest prefix match principle on the destination's Pastry key. If the key of a message is in range of the present node's leaf set, then the message is routed to the node whose nodeId is closest to the key. If the key is not covered by the node's leaf set, it looks up in the routing table a node whose nodeId shares a longer prefix with the key than its own nodeId and routes the message to this node. If there is no such node the message is routed to a node that shares the same length prefix with the present node, but is numerically closer to the destination address.

2.3 Node Arrival and Node Departure

A new node joining the overlay network has to initialize its state tables (i.e., the routing table and the leaf set) and then inform others of its presence. Assume a node with nodeId X. When node X wants to join the Pastry network it first locates a node in the network using for example "expanding ring IP multicast". Assume the found node has nodeId A. Now node X sends a special join message with *key* X to node A. As with normal messages, node A routes this message to node Z which is numerically closest to X. Each node along the path to node Z sends one row from its routing table to node X. The i:th node sends its i:th row. Node Z sends its leaf set to node X. Finally, node X informs any node that needs to be aware of its arrival.

As nodes may fail or depart without warning, neighboring nodes in the key space (i.e., nodes in the leaf set) periodically exchange keep-alive messages. If a node is unresponsive for a time T, it is considered to be failed and all the members of the failed node's leaf set update their leaf sets. Stale entries in the routing table are repaired by first trying to find a new route via its downstream node, and if not successful, starting its route table management mechanism.

2.4 A Pastry-based Store-and-Forward system

POST [2] is a messaging system based on Pastry which provides message storage, per-user meta-data (e.g., relating messages to user), and notification. In POST messages are objects that are being inserted into to the peer-to-peer network and "message storage"-nodes are responsible for notifying the recipient when it has any messages pending. Such a system is well-suited for email, but does not fulfill the requirements for being a true instant messaging system as it is a store-and-forward architecture.

3. DISTRIBUTED INSTANT MESSAGING ARCHITECTURE (DIMA)

We have designed a Distributed Instant Messaging Architecture (DIMA) using Pastry and SCAN [3]. The Pastry overlay network is used for node insertion and message routing. SCAN (Searching in Content Addressable Networks) is a search method for structured peer-to-peer networks, currently being developed at Uppsala University. The current version of SCAN is implemented on top of Pastry, but could equally well be implemented on top of other routing substrates like for example Chord [5]. We start this section with a short overview of SCAN.

3.1 SCAN

SCAN implements a search mechanism for Pastry networks that is capable of identifying the nodes holding the most relevant information. A classification of relevance of each node that is returned by the system is made on the basis of the number of keywords successfully matched.

3.1.1 Key Generation

The idea behind SCAN is that content is addressable. In SCAN, a Pastry key does not correspond to one physical host, but to a piece of content on a host. Therefore a host may insert more than one nodeId into the Pastry network. Content meta-data e.g., keywords, are encoded using the ASCII table, into a set of Pastry keys for NodeIds that are inserted into the network. A salt is added to the end of the Pastry key to make sure that similar/equal keywords do not generate the same Pastry key (figure 1). However, similar Pastry keys (generated from the same keyword) are inserted close to each other in the network since they share the same prefix.

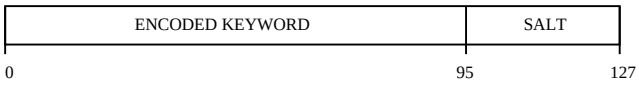

Figure 1: A Pastry key in SCAN.

This method of key generation will build a Pastry network structured according to meta-data keywords, i.e., keywords that are similar will be close in the key space and thus also correspond to nodeIds that are neighbors in the Pastry network. Each nodeId generated from the same content meta-data will also contain the full meta-data, so that it is possible to know which meta-data a nodeId was generated from.

For example, an mp3 file described by the meta-data word string `''Elvis Presley Teddybear''` will generate three Pastry keys inserted at different places in the Pastry network. But all three keys will correspond to nodeIds that points to the same content on a physical machine (IP address) and will all contain the full meta-data string `''Elvis Presley Teddybear''`.

3.1.2 Searching

When searching, Pastry keys are generated from a search string instead of content meta-data. But accept from that, key generation is done in the same way as when inserting content. Each search word will generate a separate key and using Pastry's routing, SCAN will route the closest matching nodeIds. From each of these nodeIds, the leaf set is returned. As a result, SCAN will end up with a list of nodeIds that have keys, and thus meta-data, that are similar (in a prefix sense) to each of the keys generated from the search string. In the next step SCAN will sort out those returned nodeIds that have a meta-data string that contains all search words. This is the search result, i.e., a list of nodeIds containing IP addresses of host that have content corresponding to all search words.

3.2 Instant Messaging using SCAN/Pastry

As Pastry can be seen as an overlay network layer, it is easy to implement instant messaging as long as you know the Pastry key of the peer you want to communicate with. As Pastry only provides node insertion and routing it would require a centralized lookup service to exist for peers to be able to find out the Pastry key of their IM buddies. However, with SCAN we have a way to implement buddy searching in a distributed way, without a central server. We use user specific information, like for example name, age and e-mail address, as the meta-data to put into the Pastry network. Using SCAN we can search for this information and retrieve a UID in form of a Pastry key (or IP address) to use for the communication channel.

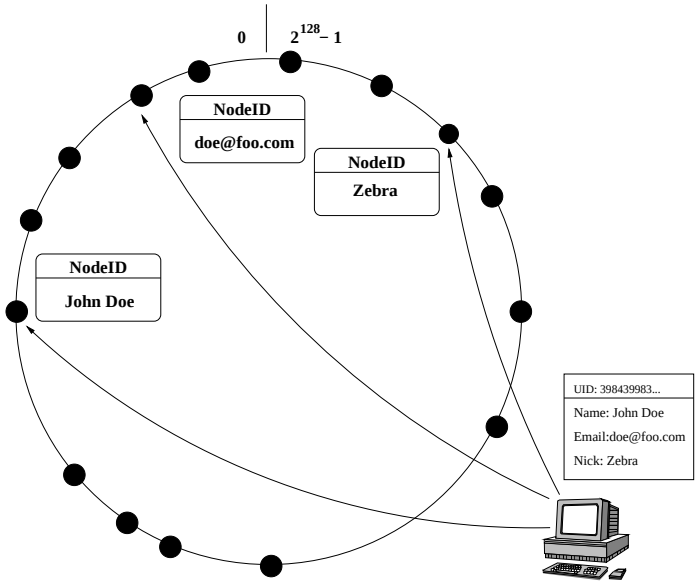

Figure 2: Inserting buddy information into the Pastry network.

Figure 2 shows a host inserting its buddy information into the circular key space of a Pastry network. Similar words will end up close to each other in the key space and thus in the network. One nodeId/key is used for routing the messages between users. This Pastry key is functionally the same as a UID.

3.3 Implementation

We have implemented DIMA running on top of a Pastry network. We use the FreePastry [1] implementation and the SCAN implementation for searching users in the network. FreePastry is an open-source Java implementation of Pastry. It can simulate a Pastry network running on one machine, but can also be deployed in a real network.

Currently, the implementation of DIMA provides a simple demonstration using the FreePastry simulation mode. It is possible to search for a buddy and, when located, send a message. The peer receiving the message will automatically respond back, proving that the routing and messaging works.

As DIMA does not have any centralized server, buddy lists have to be stored on the user machine. Similarly, peers have to keep track of buddies themselves and notify their buddies when signing off. To guard against failing nodes, the online buddy list should be kept as a soft state using "keep alive" messages.

4. CONCLUSIONS AND FUTURE WORK

This paper describes the first fully distributed instant messaging system. We call it DIMA and have built it on top of the Pastry p2p routing substrate. The strength of our system compared with existing IM systems is that it is a completely decentralized solution without the dependence of any central servers or service providers. A user simply joins the open Pastry overlay network and announces his presence to selected buddies. If a buddy is online, our SCAN search engine is always able to locate him provided it is given enough user information. Once buddies are located the Pastry substrate handles the message delivery. We have created a proof-of-concept implementation using FreePastry, to prove the merits of DIMA. This shows that it is possible to create a fully distributed instant messaging system that can be instantly deployed.

Currently, our implementation runs in simulation mode within FreePastry. We have ongoing work to extend FreePastry to deploy a real network. We will then deploy our DIMA and do performance measurements including parameters like latency and routing overhead. We also have an ongoing project investigating the performance of SCAN, both theoretically and practically. Considering our system's decentralized and self-organized nature we foresee that it will be attractive in ad hoc networks, e.g., at conferences or meetings where fixed infrastructure might not be available. It will be interesting to investigate the DIMA performance in such networks with dynamic topology.

5. REFERENCES

[1] P. Druschel, E. Engineer, R. Gil, Y.-C. Hu, S. Iyer, A. Ladd, A. Mislove, A. Nandi , A. Post, C. Reis, A. Singh, and R. Zhang *FreePastry implementation.* `http://www.cs.rice.edu/CS/Systems/Pastry/FreePastry/`

[2] A. Mislove, A. Post, C. Reis, P. Willmann, P. Druschel, D. Wallach, X. Bonnaire, P. Sens, J-M. Busca, and L. Arantes-Bezerra, *POST: A Secure, resilient, cooperative messaging system.* Submitted for publication.

[3] H. Olsson, *SCAN - Searching in Content Adressable Networks,* MSc. Thesis, 2003 – work in progress.

[4] A. Rowstrom and P. Druschel, *Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems.* In Proceedings of Middleware'01, November 2001.

[5] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, Chord: A scalable peer-to-peer lookup service for Internet applications, In Proceedings of ACM SIGCOMM'01, August 2001.