Konzeption und Implementierung einer Teilnehmerverfolgung für verteilte aktive Dokumente auf Basis Peer-to-Peer-basierter File Sharing Protokolle

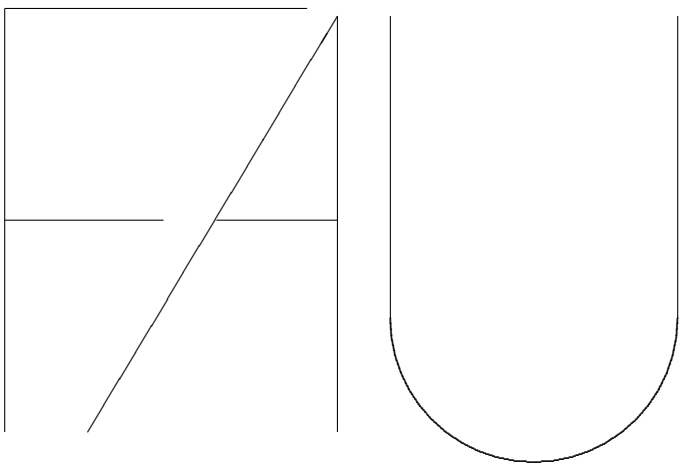
Bachelorarbeit

Hristiyan Pehlivanov

Lehrstuhl für Informatik 6
(Datenmanagement)

Department Informatik
Technische Fakultät

Friedrich Alexander-
Universität
Erlangen-Nürnberg



Konzeption und Implementierung einer Teilnehmerverfolgung für verteilte aktive Dokumente auf Basis Peer-to-Peer-basierter File Sharing Protokolle

Bachelorarbeit im Fach Informatik

vorgelegt von

Hristiyan Pehlivanov

geb. 22.02.1988 in Russe

angefertigt am

Department Informatik
Lehrstuhl für Informatik 6 (Datenmanagement)
Friedrich-Alexander-Universität Erlangen-Nürnberg

Betreuer: Univ.-Prof. Dr.-Ing. habil. Richard Lenz
Dipl.-Inf. Christoph P. Neumann

Beginn der Arbeit: 03.05.2010
Abgabe der Arbeit: 30.09.2010

Erklärung zur Selbständigkeit

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass diese Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Der Universität Erlangen-Nürnberg, vertreten durch den Lehrstuhl für Informatik 6 (Datenmanagement), wird für Zwecke der Forschung und Lehre ein einfaches, kostenloses, zeitlich und örtlich unbeschränktes Nutzungsrecht an den Arbeitsergebnissen der Bachelorarbeit einschließlich etwaiger Schutzrechte und Urheberrechte eingeräumt.

Erlangen, den 30.09.2010

_____

(Hristiyan Pehlivanov)

Kurzfassung

Konzeption und Implementierung einer Teilnehmerverfolgung für verteilte aktive Dokumente auf Basis Peer-to-Peer-basierter File Sharing Protokolle

Diese Arbeit beschäftigt sich mit der Konzeption und mit der Implementierung von einem Teilsystem für die Verfolgung von Teilnehmern innerhalb eines verteilten und dezentralisierten Workflows. Der Workflow is basiert auf aktiven Dokumenten, die die Verbreitung von medizinischer Information in einer dynamischen Gruppe von Teilnehmern unterstützen. Ein elektronisches System, das aus lose gekoppelten, autonomen, heterogenen Teilsystemen besteht, ist im Rahmen vom Gesundheitswesen vorgestellt. Der Akzent fällt auf die Komponente, die für die Organisation der Teilnehmer in einem Overlay-Netzwerk und für die Sammlung von Routing-Informationen verantwortlich ist. Diese Komponente ist basiert auf Peer-to-Peer-Technologie, damit Selbstorganisation und Ad-hoc-Verbindung ermöglicht werden.

Abstract

Design and Implementation of a Participant Tracker for Distributed Active Documents Based on Peer-to-Peer-Based File Sharing Protocols

This thesis deals with the design and the implementation of a subsystem for tracking participants within a distributed and decentralized workflow. The workflow is based on active documents, that support the distribution of medical data in a dynamic group of participants. An electronic system, that combines the activity-oriented and the content-oriented workflow paradigms, is presented in the context of healthcare. The focus is set on the component responsible for organizing the participants in an overlay network and for gathering routing information about them. This component is based on peer-to-peer technology, in order to enable self-organization and to provide ad-hoc ability.

Contents

List of Figures

List of Abbreviations

API	Application Programming Interface
DHCP	Dynamic Host Configuration Protocol
DHT	Distributed Hash Table
DNS	Domain Name System
DoS	Denial-of-Service
HTTP	Hypertext Transfer Protocol
JXTA	Juxtapose, meaning side-by-side; not an acronym
MIME	Multipurpose Internet Mail Extensions
P2P	Peer-to-Peer
PDA	Personal Digital Assistant
PKI	Public Key Infrastructure
SHA	Secure Hash Algorithm
TCP	Transmission Control Protocol
TTL	Time To Live
UML	Unified Modelling Language
UUID	Universally Unique Identifier
URI	Uniform Resource Identifier

URL Uniform Resource Locator

XML Extended Markup Language

1 Introduction

Healthcare IT support demands inter-institutional processes which implies many participants contributing to the workflow. Such organization requires complex communication between the contributors and continuous updating of documents. This thesis presents an electronic system for managing the treatment of a patient, focusing on the module that is responsible for tracking the participants in the treatment workflow.

1.1 Motivation and Challenges

Paper-based information exchange can no longer meet the demands of the healthcare sector, where many participants from different institutions are involved in the treatment of a patient. Inadequate distribution of information, due to the rapid advance of medicine and the specialization of physicians, is a basis for medical errors. In this context neither activity-oriented nor content-oriented workflows can realize the complex behavior of a workflow with unknown sets of actors and institutions [NL10]. Furthermore, the system has to be distributed amongst all the participants, while ensuring them with autonomy at the same time.

Project α-Flow combines the traditional activity-oriented and content-oriented paradigms to provide a model, where workflows are presented as documents shared between the participants. The aim of α-Flow is to present a concept for an electronic system, that offers document-based workflow with loosely coupled heterogeneous systems at the participating sites [NL09]. The documents will allow access, viewing and editing, while being active software agents at the same time.

1.2 Objectives

With a distributed and dynamic system, that evolves continuously, it is necessary amongst others to keep track of new members and changing information about known ones. This thesis focuses on the component from α-Flow, that is responsible for tracking the participants. It is called α-Tracker and its main purpose is to create a network, to which every participant can connect. This network should have ad-hoc behaviour, the ability to self-organize itself and fault tolerance with failing or unreachable nodes. The α-Tracker will gather routing information about participants of the network and provide it to other components of α-Flow for transferring data between members. Due to the importance of providing the latest information about a patient it has to be guaranteed, that every member of the network can reach every other member, who is currently online. In addition, a protocol for synchronizing nodes, who were offline, has to be defined.

2 Methods

In this thesis a possible design for a participant tracker in the context of distributed active documents is outlined. It is part of the α-Flow project and chapter 3 describes the concepts of this system. The following chapter 4 analyses the requirements to which the application has to correspond in order to fulfil its functions. The aspired behaviour predisposes the use of peer-to-peer technology as the most appropriate solution. Chapter 5 offers an overview of this technology and chapter 6 compares it to other possible solutions.

As a result of the evaluation of existing P2P technologies, the JXTA protocol, which chapter 7 describes in depths, has been chosen for the design and implementation. This protocol offers concepts to solve the typical problems presented to a dynamic and interchangeable network. It also gives the developer freedom to easily include his own services and extend the existing ones. The chapter explains the architecture of JXTA along with its basic building blocks and concepts for supporting the network.

A proposed solution is outlined in chapter 8. This solution is called α-Tracker and how it fulfils its functions is explained in chapter 9. UML diagrams have been used to describe the different interactions withing the tracking system. Chapter 10 presents some aspects, that have emerged during the implementation with JXTA, and chapter 11 proposes some future extensions of the α-Tracker. The last chapter 12 summarizes the whole thesis and gives an overview of the accomplished results.

3 Basics

This chapter explains the basic concepts of the α-Flow project. The workflow is based on active documents called α-Docs, that are organized in α-Episodes. These documents have different α-Cards for saving the information regarding content and organization. They are furthermore divided into artifacts. The term α refers to the active-properties associated with the documents.

3.1 Active Documents

An active document is a document that allows a direct interaction with itself. These documents represent the workflow schemata and function as software agents, that extend the documents to not just accessing, viewing and editing content, but to coordination between the participants too. The aim is to enable this without corrupting the workflow [NL10].

3.2 α-Episodes and α-Docs

A distributed process characterized by a particular goal and constructed of a variety of distributed activities is called an α-Episode [NL09]. There can be many different α-Episodes in the α-Flow depending on the treatment of the patient. Every α-Episode is represented by an α-Doc.

An α-Doc is the top level architecture in an α-Episode and every participant in the workflow receives a replica of the α-Doc responsible for this α-Episode. How exactly the replicas will be distributed is still a subject of future work.

The second level of architecture consists of α-Cards, which every α-Doc has as children (see fig. 3.1). The α-Cards are divided into content and coordination α-Cards. At

present, there are two coordination α-Cards and their existence is compulsive for every α-Doc. They are Treatment Structure Artifact (TSA) and Collaboration Resource Artifact (CRA). TSA provides information about the relationships between the different content cards and about the workflow schemata, while CRA holds information about all the participants in an α-Episode. There are currently no constraints for the number of content α-Cards that is allowed. The α-Tracker aims at providing information for the CRA. Due to the distributed nature of the application these cards are shared and updated amongst all the participants. α-Cards are used both for structuring content documents and consolidate coordination information [NL09].

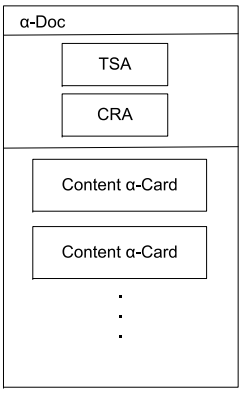


Figure 3.1: α-Doc structure

3.3 α-Flow Components

The α-Flow project is still under development, however some components can already be outlined as separate parts of the project. Currently there are seven basic components - the α-Editor, the α-Properties, the α-Model, the α-Injector, the α-VerVarStore, α-Institutions and α-Tracker. The α-Tracker is connected with the α-Properties and may be in future with the α-Editor. The α-Editor allows the user to view and edit the document. The α-Properties lie underneath the editor and are responsible for most of the coordination logic. Drools is a platform, that is a part of α-Properties. Drools Pipelines are used for incoming event streams.

4 Requirements Analysis

In the course of this thesis a system will be designed that enables users to connect to each other, organize themselves in groups and exchange information in order to participate in a common workflow. This system should have no single point of failure, because a very dynamic behaviour is expected, where new users will be frequently added to the network, existing ones will vanish or be offline and routing information will change. Due to the distribution of the system amongst users on the edges of the Internet, it should be able to connect them in spite of network differences and obstacles like firewall or Network Address Translation (NAT) devices.

The following functional requirements have been outlined in order to enable the tracking of participants from different institutions. They have been motivated by a workflow, within a continuously changing and evolving group of participants.

- Ad-Hoc Network - An overlay ad-hoc network should be created, where participants in an α-Flow self-organize themselves in groups, while enabling everyone to join and leave dynamically without endangering the stability of the network. A mechanism should guarantee that new members can discover everyone else and initiate contact.
- Routing Information - Provide the IP address and the port number of any known member. This information will be utilized in order to build Drools Pipelines to other participants and transfer payloads.
- List with Online Nodes - Provide a list with all participants, that are currently online. This list can be used together with the address information to send notifications to all other members of the group regarding local data changes. In future the editor could use this list to visualize the status of the nodes in the group.
- Synchronization - Due to the nature of active documents, it can not be expected, that the participants will be always online or that they will not change their network

address. Thus a protocol for synchronizing offline nodes, when they come online again, has to be defined.

5 Peer-to-Peer Technology Overview

Peer-to-peer is an architecture for building distributed networks where all participants act both as a client and as a server and provide a part of their resources to the others. Such networks consist of nodes with equal responsibilities that join and leave freely and do not rely on some central service for coordination. This ad-hoc concept offers improved scalability and robustness. The three principles that define an application as peer-to-peer are sharing resources, decentralization and self-organization [AH].

The main difference between the client-server model and peer-to-peer (P2P) is that in the first case, only servers provide resources, while a client consumes them. The peer-to-peer behaviour where two sides connect directly to each other can be found in many well-established concepts like E-Mail servers or DNS servers, but it has become popular after Napster introduced its file sharing service at the end of the 90s [AH].

5.1 P2P Architectures

P2P systems can be viewed as an overlay network on top of the Internet and information is usually exchanged over the underlying Internet Protocol (IP). The most common functions of P2P applications are instant messaging, file sharing and distributed computing. There are three approaches in which existing protocols create a network for this services [BS].

- Centralized Network – A central service is used to coordinate and organize the peers. They connect to the central server to receive information about other peers, after which they can connect directly to these peers. This approach is simple to implement and it solves some problems like query routing and indexing. However, it brings disadvantages of the server-client model, because there is a single point of failure and limited scalability.

- Decentralized Network – There is no central service and nodes have to organize themselves and care for the distribution of queries and resources on their own. This makes the network very scalable and independent, but queries give no guarantee that all nodes will be reached and flooding the network results in a sizable bandwidth consumption.
- Semi-centralized (Hybrid) Network – The hybrid network splits the central service provided by the centralized network in so called super nodes. The super nodes provide infrastructure support like routing and query distribution.

5.2 Current Technology

5.2.1 Centralized Approach

The first P2P protocol that gained popularity is Napster. It solved the main problems of peer-to-peer networks by introducing a central server, which is responsible for the response to queries, indexing and discovery of new peers. This server holds a database with all known peers and the content they are offering. It provides information about endpoint routing, that peers can use in order to connect directly to each other. However, the big disadvantage of this concept with a single point of failure has led to the end of the network, once the central server was closed due to copyright violations and the peers could not support the network on their own [AH].

The main idea of this approach was adopted by the Bittorrent protocol. It too uses a central server for answering queries, indexing and peer discovery (see fig. 5.1[1]), but it develops the concept further by introducing torrent files. They are stored on the server, called tracker, and contain the Uniform Resource Locator (URL) of the tracker as well as Secure Hash Algorithm (SHA-1) hashes. Shared files are divided into blocks, which the torrent file describes and verifies with the hashes. Once a peer has retrieved a torrent file from the tracker, it is informed by him about other peers, that have the same torrent and join their group called a swarm. Every member of this swarm sends and receives blocks of the file at the same time [Hin04].

1 Source: `http://computer.howstuffworks.com/bittorrent2.htm`

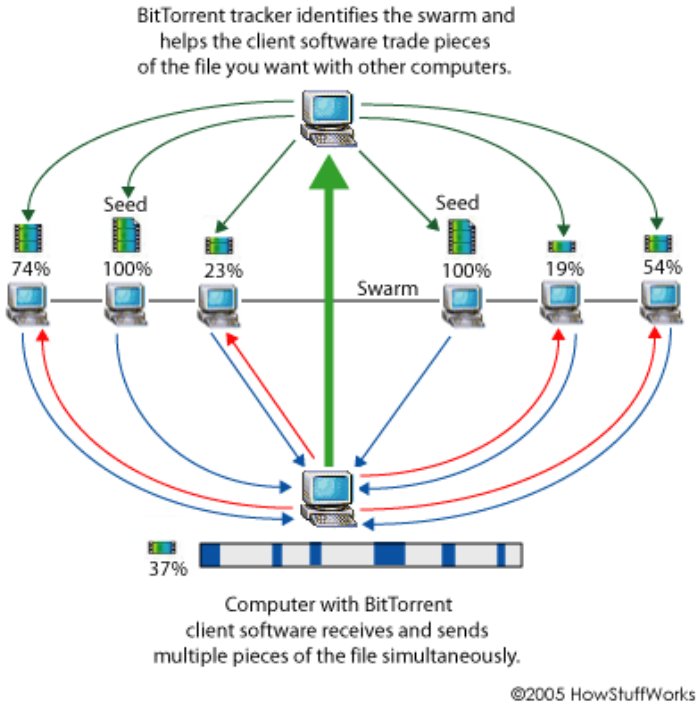


Figure 5.1: Bittorent network

5.2.2 Decentralization with DHT

In order to overcome the need of a central coordination service some protocols implement a Distributed Hash Table (DHT). A DHT is a decentralized distributed system that provides a service for retrieving a value only with the key assigned to it. The table is distributed amongst many nodes, which minimizes the expenses of maintaining it when nodes join, leave or fail and allows the table to scale to extremely large numbers [GBL+03].

Chord is a protocol and algorithm used in DHT. The hash table stores a key for every node and arranges them in a circle where a node has a list with a number of its successors and predecessors in that circle. If the node can not resolve a query on its own, it sends messages the other nodes. Routing information about newly joined or departed nodes is updated with periodical messages. The main features of the Chord protocol are provable correctness and provable performance [SMLN+02].

Another protocol that offers provable consistency and performance and adds resistance to Denial-of-Service (DoS) attacks is Kademlia. It is utilized for decentralized networks too. The protocol uses a special algorithm for distributing the list of known nodes in buckets based on the distance between two points. This distance is computed with XOR metric and the buckets are sorted with least-recently seen policy, which optimizes the distribution of messages, because there is an overview of the closest and most reliable nodes [MM02]. The Kad network is an example that implements the Kademlia protocol adding DHT to it [YL09].

A different approach to DHT is the protocol Tapestry. It focuses on efficiency and minimizing message latency through constructing locally optimized routing tables. They are created from initialization and are being continuously maintained in order [ZHS+04].

5.2.3 The Purely Decentralized Approach of Gnutella

Protocol Gnutella provides a simpler solution for a decentralized peer-to-peer network. In order for a peer to join the network, it has to connect to a known Gnutella host to receive a list with the addresses of other hosts. When the peer has this list to get started with, it can be independent and discover new hosts on his own. In this way every peer knows about a little part of the network and after a query message is sent to the known peers, they propagate it onwards to other peers they are aware of. The query can be flooded very fast through the network (see fig. 5.2[1]) and in the case of a hit, peers connect directly to exchange content. Every message has Time To Live (TTL) to prevent indefinite propagating. Nevertheless, the flooding results in a sizable bandwidth consumption and does not guarantee that every peer in the network will be reached [Ber03].

5.2.4 The Semi-centralized Approach of JXTA

Protocol Juxtapose (JXTA) offers a compromise between completely decentralized and completely centralized network, by deploying super nodes, that distribute the central service on many reliable nodes. This semi-centralized concept implements a DHT, where

1 Source: `http://computer.howstuffworks.com/file-sharing3.htm`

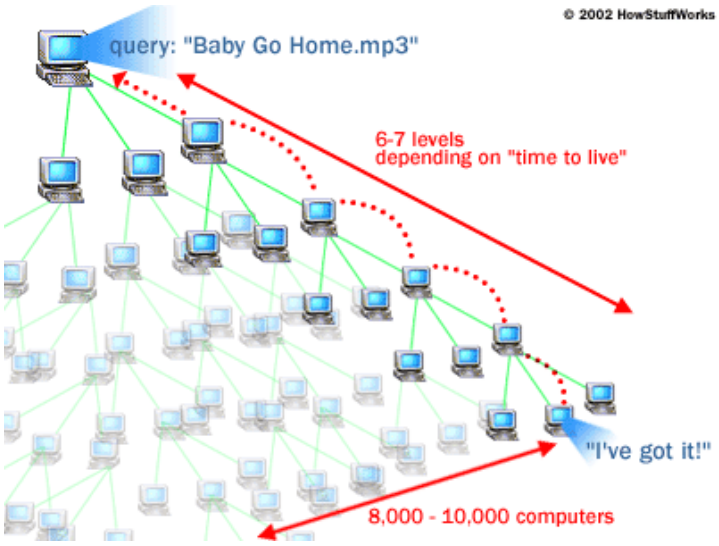


Figure 5.2: Flooding in a Gnutella network

every super node knows about a part of the network and about other super nodes [TAP03]. This enables the flooding of queries through the network in a more optimized way than the one used by Gnutella, because only super peers forward them to other peers. At the same time it resembles the centralized server approach of Napster and Bittorrent, because edge peers discover each other through the service created by the super peers. Edge peers rely on the super nodes to distribute queries, index the network, provide routing information and discover new members. In this way the single point of failure is avoided, while enabling the support of the network with a relatively small number of reliable nodes. Furthermore, with the utilization of the HTTP protocol and relay peers, that are super peers forwarding messages, JXTA also adds a feature, that lacks in other protocols - bypassing firewalls, NAT and other barriers on the Internet [TAD+02].

5.3 Summary

Peer-to-Peer technology has become extremely popular in the last ten years because it enables many users to share their resources over a network, that is cheap and very adaptable to changes. The ability to self-organize without the need of a central service offers robustness and scalability which is an important feature in the dynamic world of

the Internet, where users join and leave continuously. P2P networks have often no single point of failure and unlike the traditional client-server network, the bigger they become the better the performance gets.

However, the lack of central coordination has its disadvantages which is suggested by the fact, that two of the most popular protocols, Napster and Bittorrent, rely on such service. No protocol so far has provided a solution for propagating queries and messages through the network, that can match the server-client efficiency. The fact that everybody can share its resources leads to bad reliability, because nodes leave the network or fail. It also provides great difficulties to ensure security. Furthermore, the connection itself is hard to be guaranteed when so different configurations of hardware, software and network topology are spread across the edges of the Internet.

Many of the current P2P protocols, like Chord, Tapestry and Kademlia, focus on optimizing the latency, routing and the propagation of messages when the network consists of a very large number of nodes. This is of little importance for this project, because the network will not reach such gigantic scale. The centralized service of Napster and Bittorrent is not robust and scalable enough, while the approach of Gnutella lacks the ability to guarantee, that every node can reach everyone else. It is also very important to enable connecting under any circumstances and to bypass obstacles like firewall or NAT. JXTA offers this feature and provides tools to implement a semi-centralized network with DHT, where the discovery of every available peer is guaranteed. Therefore, it is the most suited protocol for the purposes of this project.

6 Possible Designs for the α-Tracker Network

This chapter describes the designs, that have been considered for implementing the prototype of the α-Tracker. Before deciding for P2P technology, the server-based concepts of E-Mail and Extensible Messaging and Presence Protocol (XMPP) were regarded as possible solutions. Their advantages and disadvantages, as well as those of a centralized and a semi-centralized P2P network, are compared in this chapter.

6.1 E-Mail

E-Mail is the most trivial way to exchange content, like sending e-mails with files attached or just plain text. This may meet the everyday needs of most users, however for this project it presents more problems than solutions.

The main disadvantage of E-Mail is the lack of guarantee for delivery. Certain modifications of the protocol fix this issue, but with the standard protocols the user sending an E-Mail has no way of knowing, that it has reached the intended recipient. Furthermore, E-Mail servers use the store-and-forward principle, which results in end-to-end latencies measured in minutes [MSA05].

Account management is another problem, when working with active documents and E-Mail. A different account for every α-Doc-Replica would have to be created, because they function as independent agents and this does not correspond to the ad-hoc requirement for the network. Other issues regarding E-Mail are spam, viruses and malware[1].

1 Malicious software

6.2 XMPP

The XMPP technology, started with the name Jabber[1], is very similar to E-Mail. It relies on the same client-server model, where the various local servers transfer messages between themselves until they reach the recipient [Dod00]. There is no main server, which makes the system scalable and fault tolerant.

The main advantage of XMPP over E-Mail is the elimination of intermediate servers, which enables delivering messages within seconds and the user is also immediately notified, if the delivery has failed. XMPP offers real time presence information with the publish-subscribe pattern [MSA05]. Knowing, whether your communication partner is online, can fulfill one of the main requirements - providing a list with all online participants.

The disadvantage of this protocol for this project is the account management. Every XMPP client needs a separate account, in order to connect to the server, which does not correspond to the ad-hoc requirement for the α-Tracker network. This presents the problem already explained in chapter 6.1 about α-Doc-Replicas and accounts.

6.3 Purely Decentralized P2P Network

The characteristics of P2P networks, where participants join and leave constantly, match at best the bahaviour of α-Doc-Replicas. The network growth and exact organization can not be predicted, due to the distribution of documents amongst many participant from different institutions and the constantly evolving workflows.

A completely decentralized network, where every α-Doc-Replica represents a peer, is the best case scenario for this project (see fig. 6.1). The peers will form a group, in which messages will be flooded on the basis of the Gnutella principle. After the desired content or member of the group is discovered, peers will connect directly to each other to exchange information.

The problem in this case arises from the fact, that active documents are mostly offline. So most of the time peers will not have the ability to support the network with query distribution and indexing, nor share their content with other participants in an α-Episode.

1 The term “XMPP” refers to the core XML streaming protocols contributed by the Jabber Software Foundation to the Internet Standards Process and subsequently published as RFCs 3920 and 3921.

Furthermore, even the discovery of new members can not be guaranteed, because it will require knowledge about existing peers in the network, in order to discover the rest of it. When these known peers are offline, new members will not have anyone to connect to.

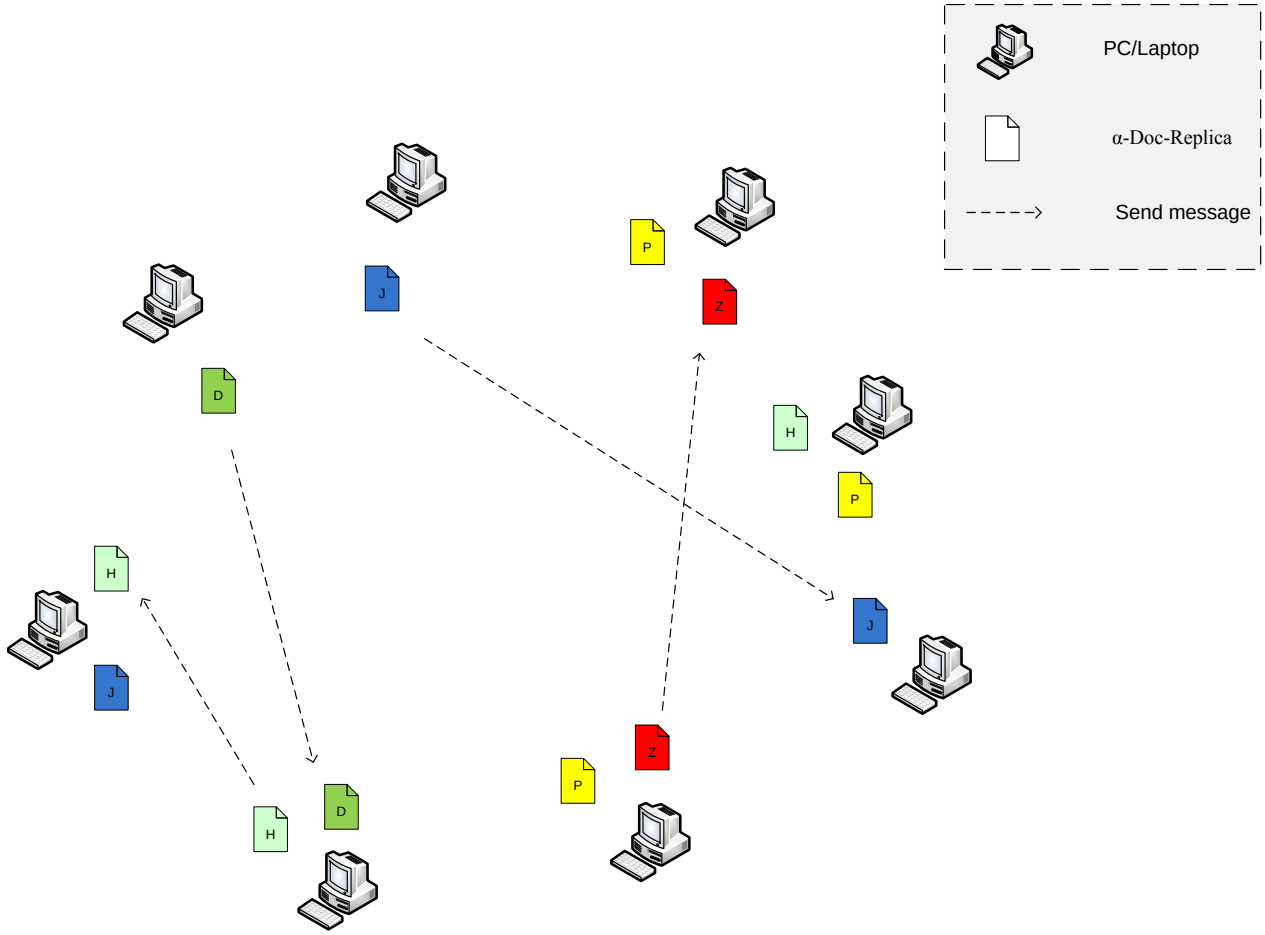


Figure 6.1: A completely decentralized P2P network

6.4 Partially Centralized P2P Network

A partially centralized P2P network combines the scalability and robustness of P2P technology with the dependability of a central service for organization. This model can

solve the problems of a purely decentralized P2P network presented in chapter 6.3, while ensuring that members can join and leave freely.

Due to the fact, that edge peers, who are mostly offline, can not support the network on their own, super peers will be deployed (see fig. 6.2). They will care for indexing content, distributing queries and discovering new peers in the network. They will also function as a repository for content, so every participant in a workflow will be able to receive the latest information, even when the other members of his group are offline.

This model is very similar to XMPP, because the super peers function as servers, that communicate with each other to transfer messages. The advantage of the P2P network compared to XMPP is that members do not need an account, in order to connect to the super peer, but instead can use the ad-hoc principle. This makes it simpler to let the network organize itself and expand according to the current needs.

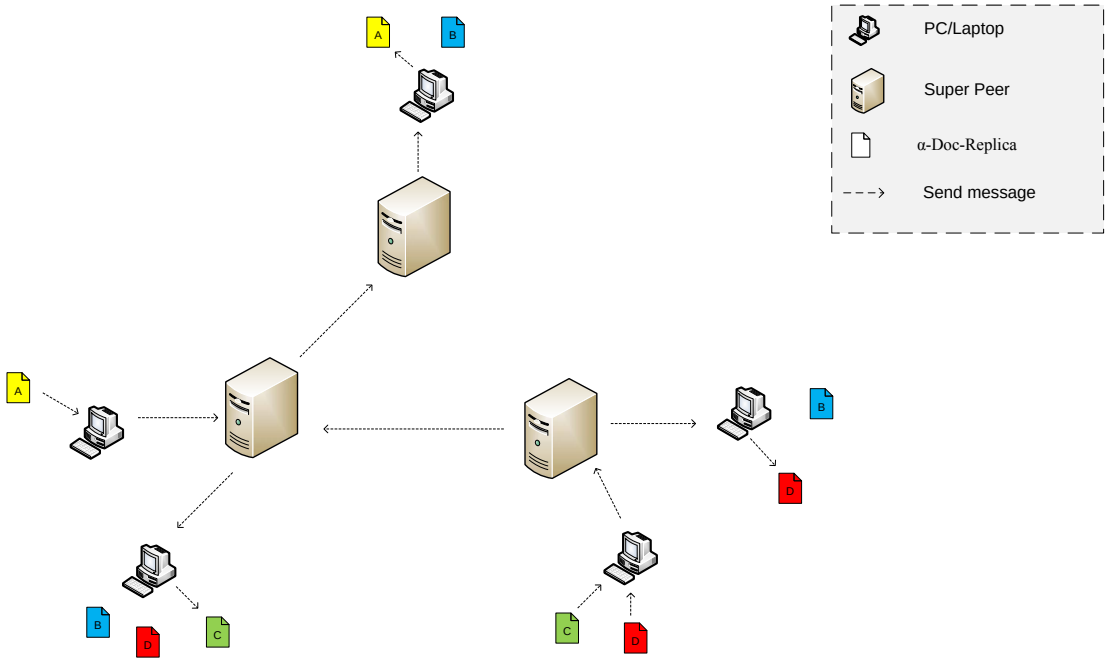


Figure 6.2: A partially centralized P2P network

6.5 Summary

The current E-Mail protocols do not offer enough flexibility and reliability to meet the demands of α-Flow. The XMPP protocol overcomes many of their disadvantages, enabling instant delivery, confirmation for delivery and real time presence information. However, the need of accounts does not allow ad-hoc behaviour, which is one of the main requirements for the network. A completely decentralized P2P network offers the ad-hoc feature, but due to the strong offline characteristic of α-Doc-Replicas it can neither guarantee that every member of the network can be reached nor that the latest information from a particular member can be acquired. This is not acceptable for the distribution of medical data. This is why the thesis focuses on the semi-centralized approach for P2P networking and the next chapter describes in depth the JXTA protocol, which is a powerful tool for such tasks.

7 JXTA

JXTA is a collaborative open-source project, which aims at providing a set of XML[1]-based standardized protocols for developing P2P applications. It was initially specified by Sun Microsystems, but has grown into a strong supportive community with many contributors. Its main objective was to define a platform for developers to create interoperable P2P applications. In order to achieve this the platform was not based on one specific software language, but was designed through a process, that has resulted in a specification of the major points of the system and the corresponding implementation information. The entire JXTA system is modelled with a small number of protocols for providing services, which are simple to implement and to integrate into an existing system. Furthermore, it is language, platform and network independent. JXTA is not an Application Programming Interface (API) and does not require the utilization of a certain computer language or a certain platform. There are language bindings for Java Standard Edition, C/C++, C# and Java Micro Edition, however their use is not compulsory [JXT07a]. In future new languages can evolve and provide their own bindings and new platforms can implement the protocols in a completely new way. A JXTA application can operate over TCP[2]/IP, HTTP[3], Bluetooth or any other kind of communication protocol. Peers located on completely different networks can communicate freely with each other. Thus, JXTA enables the deployment of applications and services on continuously evolving groups of collaborators.

1 Extended Markup Language
2 Transmission Control Protocol
3 Hypertext Transfer Protocol

7.1 JXTA Protocols

There are six protocols that define JXTA divided into two categories [JXT07b]:

- JXTA Core Protocols
  - Peer Resolver Protocol (PRP) – Provides query/response interface. A Resolver Query Message is send to contact other peers of the same peer group, who can then answer with a Resolver Response Message. These queries are the basic for finding information about peers, JXTA structure, contents, etc. Every peer defines listeners for message handling. The Endpoint Service provides Endpoint Addresses, which specify the handlers. PRP does not guarantee, that all or in fact any peers will be reached. It only offers a best effort attempt and maximizes the chances of receiving a response.
  - Endpoint Routing Protocol (ERP) – Provides request/query interface for finding routes to other peers. Endpoints are described with Uniform Resource Identifiers (URIs). They represent Endpoint Addresses, which can contain both physical and virtual addresses. Extra routing information is added to the message by the endpoints. A Route Query Message is sent with the Resolver Query Message and the contacted peer responds with a Route Response Message within the Resolver Response Message.
- JXTA Standard Protocols
  - Peer Discovery Protocol (PDP) – Provides a discovery service for finding peer resources in the network represented as advertisements. Such resource can be anything described with an advertisement like peers, pipes or modules. It is a high level service used for optimizing the discovery process thanks to knowledge about the group topology. PDP sends Discovery Query Messages and receives Discovery Response Messages. A response to the query is optional, so the service does not guarantee the discovery of existing resources and offers just a best effort attempt.
  - Rendezvous Protocol (RVP) – Provides a service for propagation of messages within a peer group in a controlled way. To increase efficiency some peers function as Rendezvous Peers (RdvPeers), all of the RdvPeers form a PeerView.

This protocol is divided into three parts. The PeerView Protocol is optional and defines how the RdvPeers organize themselves. The Rendezvous Lease Protocol is also optional and enables non-RdvPeers to receive propagated messages. The Rendezvous Propagation Protocol is the only one mandatory and it manages the propagation of the messages.

- Peer Information Protocol (PIP) – PIP is an optional protocol, that provides a way to obtain status information about a peer. It is layered upon the Peer Resolve Protocol and uses PIP Query Messages and PIP Response Messages to contact peers and receive information about them.
- Pipe Binding Protocol (PBP) – Provides a virtual channel in the form of a pipe between two endpoints, so peers can exchange information. It is layered upon the Endpoint Protocol. Pipe Resolver Messages are used for both query and response.

7.2 JXTA Architecture

The Architecture of JXTA is divided into three layers [JXT07a] (see fig. 7.1 [1]). The line between these layers is not strict and some peers may see a service, while others perceive it as an application.

JXTA Core

The core of JXTA encapsulates the minimal functions that are needed for a P2P network and contains the basic components for building any P2P application. The core enables discovery, transport, creation of peers and peer groups and security mechanisms.

Services Layer

The services layer provides functions that are not absolutely essential for every P2P application, but nevertheless will be employed often enough. Such services are searching and indexing, directory, storage system, file sharing, distributed file system, resource aggregation and renting, protocol translation, authentication and Public Key Infrastructure (PKI).

1 Source: JXTA ProgGuide 2.0

Application Layer

The application layer includes the developer's implementation. It pulls peers together for a common function like instant messaging, file sharing, content management, distributed computations or other.

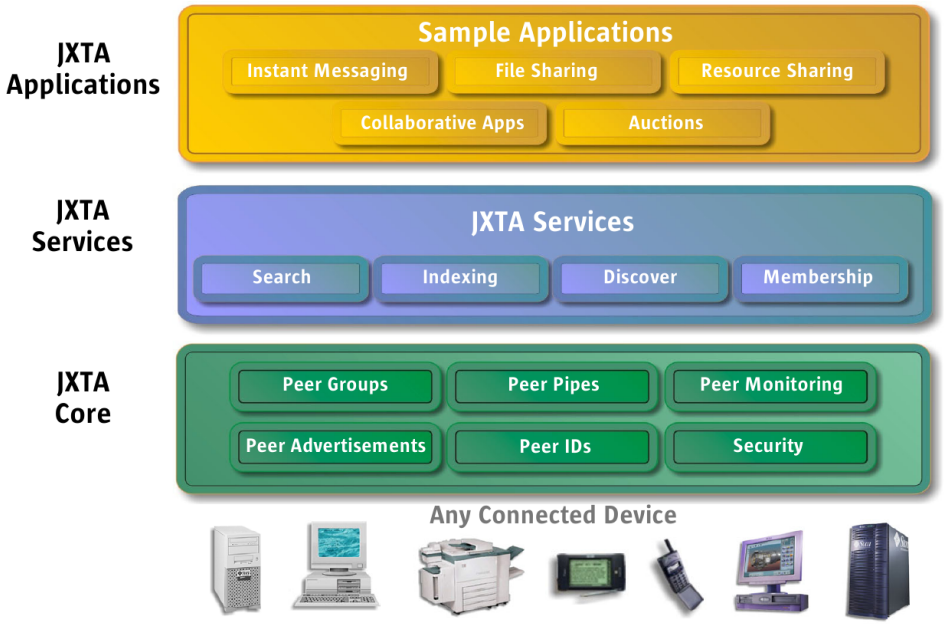


Figure 7.1: Layer organization in JXTA

7.3 Peers

In JXTA a peer is any virtual device that implements one or more of the JXTA protocols. This allows a wide range of devices like sensors, phones, PDAs[1], PCs, servers or supercomputers and one machine can host unlimited number of peers [JXT07a]. Every peer is characterized by a Universally Unique Identifier (UUID) assigned to him during the peer creation. This enables recognition even after the peer has changed its address.

Every peer publishes one or more network addresses as peer-endpoints. These endpoints are used for a direct point-to-point communication. When such connection is not possible

1 Personal Digital Assistant

due to boundaries like firewall, NAT, proxies or just physical differences between the networks, other peers can function as a bridge for the communication.

There are three basic configurations for a JXTA peer [JXT07a].

- Full-Edge Peers – They implement all core and standard services of JXTA. The majority of peers are full-edge, for example phones, PCs and servers.
- Minimal-Edge Peers – These are usually sensors, that implement only the core services of JXTA. They need a proxy to receive access to the network and to use standard services, which are above the core layer.
- Super-Peers – Peers that provide additional resources for supporting the network. They have three key functions and can implement one or more of them. In practice, when a peer has enough resources to be a super-peer, it is reasonable to give it both relay and rendezvous functions.
  - Relay – Stores messages, so it can forward them to peers who have no direct connection to the network, because of a firewall or NAT.
  - Rendezvous – Stores advertisements from other peers and resources, maintains a global advertisement index. Helps edge peers to broadcast their queries through the network and find peers they do not know.
  - Proxy – Gives minimal edge-peers access to the network. It handles requests and responds to queries on their behalf.

Peers are normally configured to discover each other and to organize themselves in peer groups. Every peer has to be a member of at least one peer group, but there is no upper limit. During startup a peer joins the World Peer Group per default, whose initial scope of operations and services it can use to discover or create other groups.

7.4 Peer Groups

A peer group is a set of peers who have agreed upon common services. Every peer group has an unique group ID and can define its own membership policy. This policy can range from completely open to the demand of secured membership credentials.

There are two types of peer groups in JXTA – the World Peer Group and user peer groups. Every user peer group is a derivation of the World Peer Group. The peer

joins this standard group as default in order to have access to its services and discover other resources [OTG02]. Then every peer can freely create new peer groups or discover existing ones and join them. The retirement from a group membership is also free, the World Peer Group is the only one that can not be left. Several reasons have led to the abstraction of peer groups [OTG02]:

Secure Environment

Every peer group can define its own security policy sometimes as complex as key cryptography. The access to the peer group's resources is limited and peers can communicate only with peers from the same peer group, which means that content can be securely published and accessed. All peers belong to the World Peer Group and every peer can communicate with everyone else, but thanks to the user groups, a more precise collection of peers can be addressed.

Scoping Environment

Peer Groups enable the specialization of regions from the network for a common task.

Monitoring Environment

The peer group enables peers to monitor other members of the group, their interactions, traffic introspection, accountability and traceability.

7.5 Advertisements

Advertisements are one of the basic building blocks in JXTA. They are meta-data structures based on XML, which carry the information about services and resources. Every advertisement contains the ID of the advertised resource plus other hierarchically structured elements needed for the complete description of this resource [OTG02]. Due to the XML format advertisements are platform independent and can be exchanged even between applications based on different computer languages. Every peer can create its own custom advertisements in order to describe some unique contents or modules it has to offer. They can also be included in messages exchanged between peers.

Advertisements are divided into three basic categories – peer advertisements, peer group advertisements and everything else. They can be pulled from super peers or from known edge peers and when an advertisement is received, every peer defines its own method

for handling it. Only core advertisements containing resolver and route information are stored automatically in the local cache. All others, including advertisements about peers, groups and pipes, should be explicitly saved by the application. They can be later retrieved from the cache and serve for the usage of resources, without the need to discover them again. Every advertisement has life time and expiration time. After the life time the advertisement will not be valid any more. And after the expiration time the advertisement will be deleted from the cache and the application will attempt to refresh it from the source. If an advertisement is republished, its life and expiration time are updated accordingly.

7.6 Pipes

In JXTA a pipe is a virtual channel between two peers. It is dynamically bound to an end-point of each peer during creation, which allows a fault-tolerant behaviour in an unreliable and interchangeable peer-to-peer network. An end-point is the logical abstraction of an address on the network for sending and receiving data. Any address that can support unreliable package-based messages can become an end-point [OTG02]. Connectivity is established independent of the end-point location and can be achieved as long as the peers have a direct connection or can communicate over a relay peer. A pipe can send any type of data including XML, string, binary and Java objects.

In order for two peers to create this connection, their end-points have to share the same peer group. All peers belong to he World Peer Group, but the difference is in which Pipe Service will resolve the pipe. Two peers can maintain many pipes between each other in different peer groups due to security reasons and every pipe will be resolved by the Pipe Service of the corresponding group.

The default Pipe Service offered by JXTA sends messages unidirectionally and asynchronously. This service enables two modes of transfer – point-to-point and propagated [OTG02].

Point-to-point Pipes

The IP address and the port chosen for communication are described in a JXTA Endpoint Address. The sending peer binds an output pipe to its endpoint address and to the one of the receiving peer, where the receiving peer regards this as

input pipe. This virtual channel, build between two JXTA Endpoints, is used for transferring data from one peer to another.

Propagated Pipes

For the propagated pipe to function, the sending peer binds the pipe to its JXTA Endpoint and to the JXTA Endpoints of many receiving peers. This is not the publish/subscribe method of XMPP, but a simple multicast from one peer to many others simultaneously.

Unidirectional pipes are the low-level of JXTA pipe abstraction. The developer can acquire a bidirectional and reliable communication with JxtaSocket and JxtaBiDiPipe, which JXSE[1] provides as the high-level abstraction implemented on top of the pipe primitives. They ensure message sequencing, delivery and security.

7.7 Messages

Messages are the basic block of information exchange between peers. The specification of the JXTA protocols consists in the set of messages exchanged. A message defines a package, which is divided into sections. Each section can have different MIME[2] type allowing peers to interpret the message on their own and to extract just the information they need [OTG02]. Any kind of data can be packed in the message like string, binary, Java objects or C structures. It is then converted into binary wire format for more efficient use of the underlying transfer protocols.

7.8 Network Organization

The ad-hoc nature of a peer-to-peer network means that peers, who are constantly joining and leaving it, present ever changing routing information. The only common denominator are the JXTA protocols, so the network relies on special types of peers to organize itself and sustain its functionality – rendezvous and relay peers [JXT07a].

1 JXTA for Java SE/EE 5.0
2 Multipurpose Internet Mail Extensions

The main task of rendezvous peers is to propagate queries and to store advertisements about other peers and their resources. An edge peer that sends a query can contact them and if they do not have the desired information, they propagate the query through the network. They ensure, that peers, who do not see the whole network, can too find all available members. Rendezvous peers also maintain a global index of each other and build a form of DHT. They can be used as super nodes in hybrid P2P networks to guarantee the scalability of the system and that new peers can be discovered or reached.

Relay peers are used for forwarding messages and not queries. In JXTA they correspond to routers and gateways. All the barriers that surround the different networks call for the use of such special peers that can bypass them. Firewalls are one of the main obstacle for sending information through the Internet and because they filter TCP/IP and almost everything else, JXTA can use HTTP to create the connection. The only condition for this connection is that the relay peer outside the firewall and the edge peer inside the firewall know about each other. The drawback is that HTTP allows only communication initiated by the client. So the peer inside the firewall has to contact the relay peer in order to receive its messages. Problems created by NAT, proxy servers and DHCP[1] are solved on the same way. When the IP address or port number of a peer changes, this peer has to contact the relay peer first and then receive the messages stored for him.

7.9 Security

Security is a main issue in P2P technology, because nodes join and leave all the time and take part in forwarding information on behalf of other nodes. JXTA offers different methods for ensuring the security of a network. One of them is a Membership Protocol, which allows peer groups to restrict the access to the group by requiring a set of credentials from new peers. Another possibility is the encryption of the communication between peers. For this purpose the Java binding of JXTA offers an interface that can encrypt the messages before they are sent. A digital signature can also be added to the messages, in order to guarantee the identity of the peer, who sends them [OTG02].

1 Dynamic Host Configuration Protocol

7.10 Summary

The JXTA protocol is a powerful tool for building semi-centralized P2P networks. Its super peer concept offers a simple method for building a DHT and enabling the self-organization of the network. In addition, it defines a way for bypassing barriers on the Internet - something, that most other protocols neglect. Furthermore, the Java binding of the protocol offers an interface for ensuring the security of the network. Therefore, JXTA is utilized for implementing the α-Tracker based on the proposed solution, which is outlined in the next chapter.

8 Proposed Solution

Based on the advantages and disadvantages of the possible designs (see chapter 6) and the study of the current peer-to-peer technology (see chapters 5 and 7), a partially centralized P2P network using protocol JXTA will be implemented to fulfill the requirements presented in chapter 4. This chapter outlines a solution for creating the α-Tracker, that will be responsible for building and supporting the network, tracking the participants and synchronizing them.

8.1 JXTA Edge- and Super-Peers

An edge peer is represented by the α-Doc-Replica, which every participant receives. They are JXTA full-edge peers, which do not connect to each other directly, but to a JXTA super peer. Every institution will deploy such super peer, that is responsible for the communication between edge peers. It is used for its rendezvous functions and is going to be online all the time. This guarantees, that the whole network can be reached over the super peer and makes possible the utilization of the JXTA features, that deal with firewalls, NAT and DHCP. The super peer can also be used as a repository for the latest updates in an α-Flow.

8.2 α-Doc-Coordinator

In order to use the JXTA super peers as a repository, a mechanism for storing information should be defined. Due to the complex interactions between the artifacts of an α-Card, the α-Tracker can only deliver information needed for the transfer of data and it lets the other components decide what to do with the data. This is why the super peer itself saves only a copy of every α-Doc, called α-Doc-Coordinator. It is placed for

synchronization on the machine where the super peer is running (see fig. 8.1). When an edge peer wants to commit its changes before going offline, it requests a connection to the α-Doc-Coordinator from its super peer. The super peer is used only as a middle man for providing routing information for the transmission. Edge peers, that were offline, can later use the α-Doc-Coordinator to receive the latest information and synchronize with the peers, that are now offline.

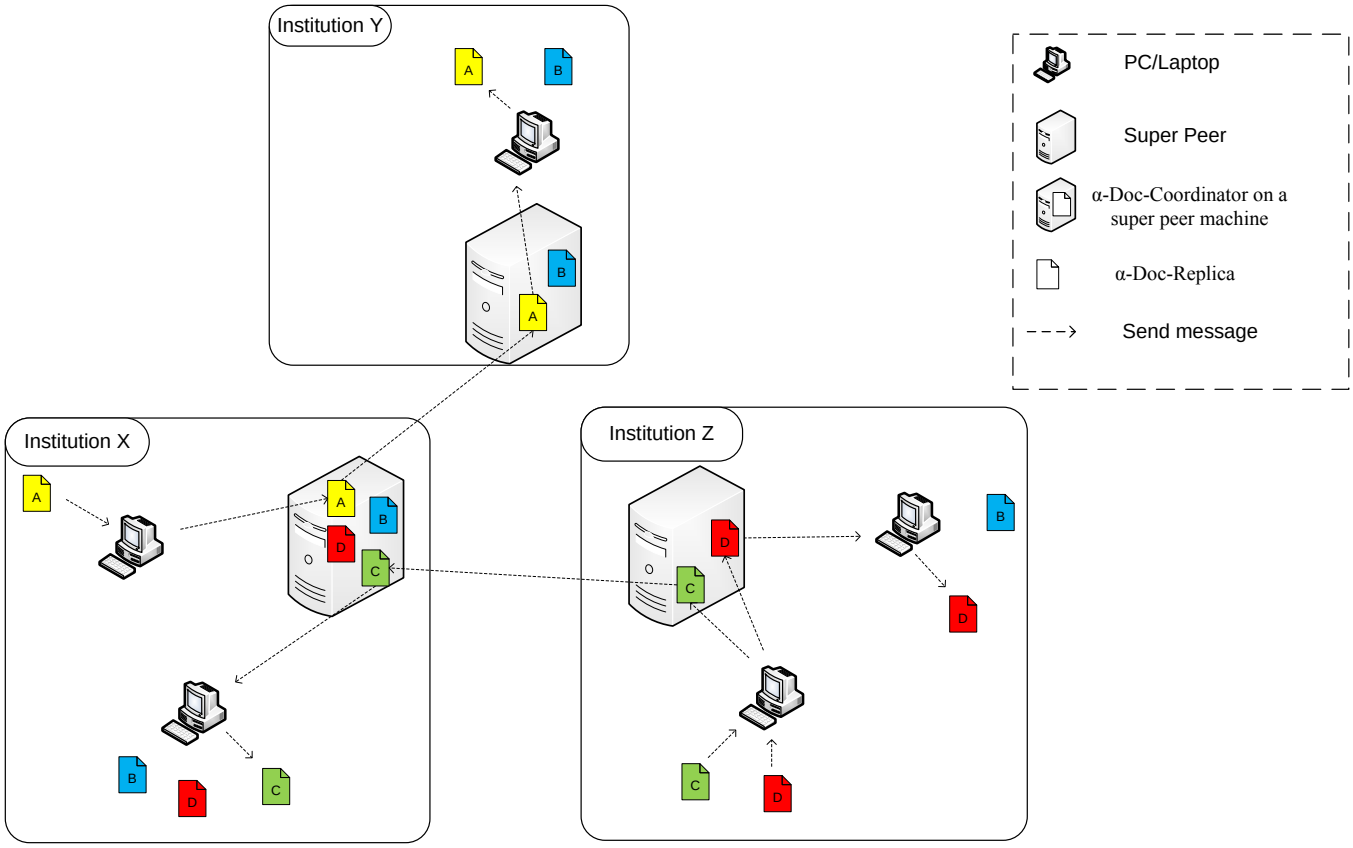


Figure 8.1: Network with α-Doc-Coordinators on the super peer machines

8.3 Communication

The JXTA edge peers are mostly offline and are not able to keep track of the current address information about other edge peers, who change their IP and port frequently. Therefore, it is more effective to use the JXTA super peer to initiate contact with them. The super peer is responsible for delivering the list with all online peers as well as information about the address of other edge peers needed for connection. The JXTA edge peers do not communicate directly with each other, but rely on their JXTA super peers completely.

8.4 Peer Discovery

Every JXTA edge peer needs the IP address and the port number of its JXTA super peer. Apart from that, edge peers do not need any other information, because everything else is taken over by the super peers. After startup, an edge peer connects automatically to its super peer and the advertisements, that describe it, will be stored there. This way the JXTA super peers build a DHT of the network. If a new JXTA super peer is introduced in the network, it has to contact just one other super peer to receive information about the whole network.

8.5 Summary

This chapter outlines a solution for building the α-Tracker network based on a partially centralized organization using edge- and super-peers. This method ensures the scalability of the network and offers a reliable service, that is available all the time, while enabling the edge peers to be mostly offline. The communication through the network is taken over by super peers. α-Doc-Coordinators enable the synchronization between peers, who are not online at the same time.

9 Design of the α-Tracker System

This chapter presents the design of the proposed solution from chapter 8. This is a prototype of the α-Tracker designed to fulfill the requirements listed in chapter 4. It is assumed, that every institution will deploy one super peer and the network will have at least one such peer. This super peer is continuously online on a machine, which has a public unchangeable IP address. Furthermore, no hardware or software component prevents the connection to this machine from the local or any external network.

9.1 Class Structure of the Prototype

Although there are two types of peers and respectively a prototype for each of them, their structure is so similar, that they can be regarded as one prototype with some extra functions attached to the JXTA super peer (see fig. 9.1). Each peer is started as a thread, which is defined in a primary class responsible for joining the α-Tracker network and initializing listeners for incoming messages. In order to improve the readability and the maintainability of the code, all other functions are encapsulated in separate classes and can be called by creating an object of the class. The strategy design pattern was used for this purpose.

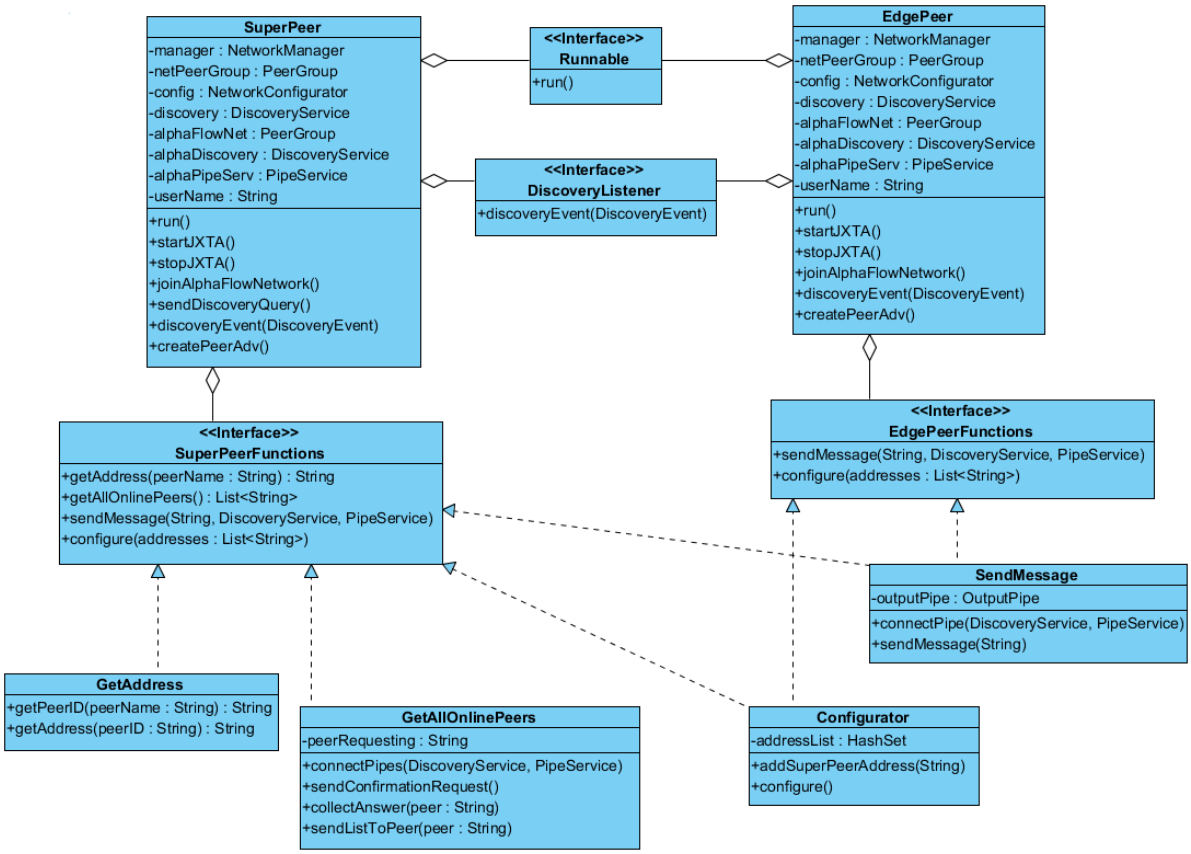


Figure 9.1: Class structure of the prototype

9.2 Control Interface

In order to control the peer several public methods are defined, that invoke the methods encapsulated in separate classes (see fig. 9.2). The methods create an object of the class, and if necessary, results are returned. This is the case of providing routing information or a list with all online peers.

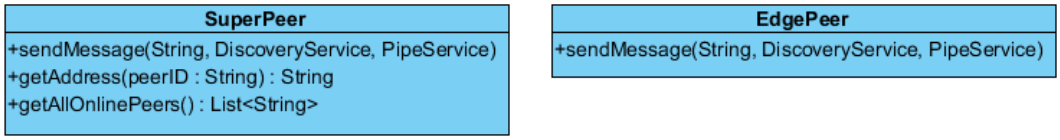


Figure 9.2: Interface for invoking functions

9.3 JXTA Messages

JXTA Messages, together with advertisements, are the instruments for exchanging information between peers. In this prototype JXTA messages are used only for confirmation, which edge peers are currently online. In the future they can be used for sending content payloads with JXTA pipes. There are a couple of steps needed for sending a JXTA message to another peer. This is why these steps are encapsulated in the class *SendMessage.java*. When an object of this class is created, the message and the recipient are passed as parameters. The methods in the class then find the correct pipe advertisement in the cache, open an output pipe to the peer, send the message and close the pipe.

9.4 α-Doc-Coordinator

As already explained in chapter 8.2 the α-Doc-Coordinators are used as a repository for synchronization between peers that are offline. The functions needed for this synchronization are getting the latest updates from the α-Doc-Coordinator and committing local updates in it, so they are visible by other peers. JXTA edge peers have access only to the JXTA super peer from their institution, so every institution taking part in an α-Episode needs an α-Doc-Coordinator of its own. As no machine can support so many running active documents, only the JXTA super peer will be online all the time and α-Doc-Coordinators can be started for a short time on demand by an JXTA edge peer.

For an edge peer to get the latest information from its α-Doc-Coordinator, the edge peer has to send a request to the super peer to start the coordinator (see fig. 9.3). Afterwards the super peer will inform the edge peer, on which port it can connect to the coordinator. The port number changes every time as explained in chapter 10.3.

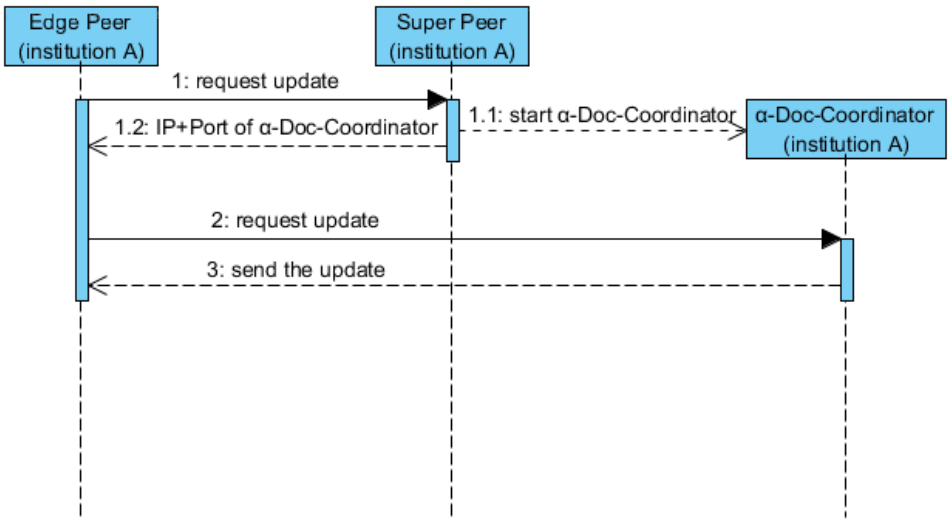


Figure 9.3: An edge peer gets the latest information from its α-Doc-Coordinator

If a JXTA edge peer wants to commit its content changes before going offline, the same procedure for starting the coordinator is executed. Once the edge peer and its α-Doc-Coordinator connect, other components can synchronize the content (see fig. 9.4). The α-Doc-Coordinator will then request from its super peer a list with all other super peers, contact them and inform them, that new information is available. At this point the super peers can synchronize their α-Doc-Coordinators on the same principle and every institution will have the latest update of an α-Doc.

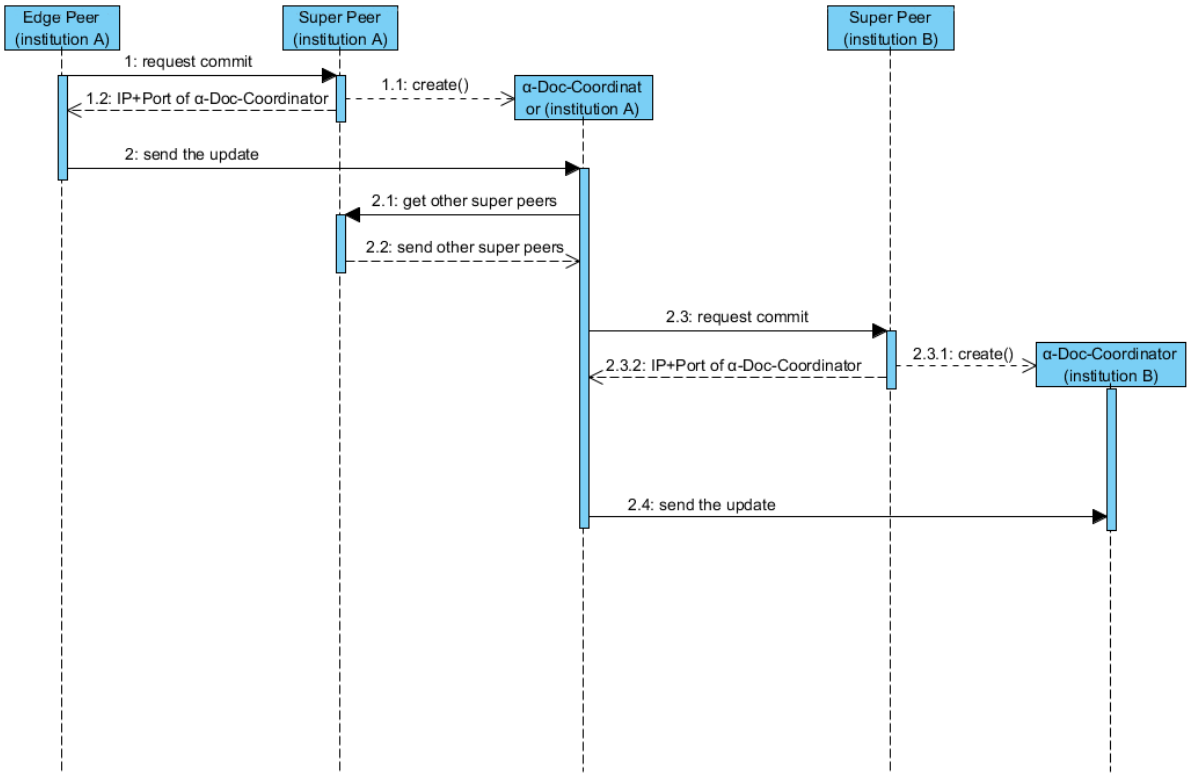


Figure 9.4: An edge peer commits local changes to its α-Doc-Coordinator

9.5 Provide a List with All Online Peers

When an edge peer requests such list from its super peer, the super peer has the task of creating it. For the list to be created, the online status of all known edge peers is checked by sending them JXTA messages and collecting the answers (see fig. 9.5). The super peer creates a worker thread, so it can handle new requests, while the list is being created. However, the super peer knows just the edge peers from its institution, thus the message will be send to all edge and super peers alike. When an edge peer receives a message demanding a confirmation about its online status, it answers directly to its super peer, because it is the only one that could send the message. If a super peer receives the same message, it starts a worker thread, that creates the list for its local institution and sends it to the requesting super peer. When all lists from institutions are received, the

initial worker thread combines them and sends the complete list to the edge peer, that has requested it in the first place.

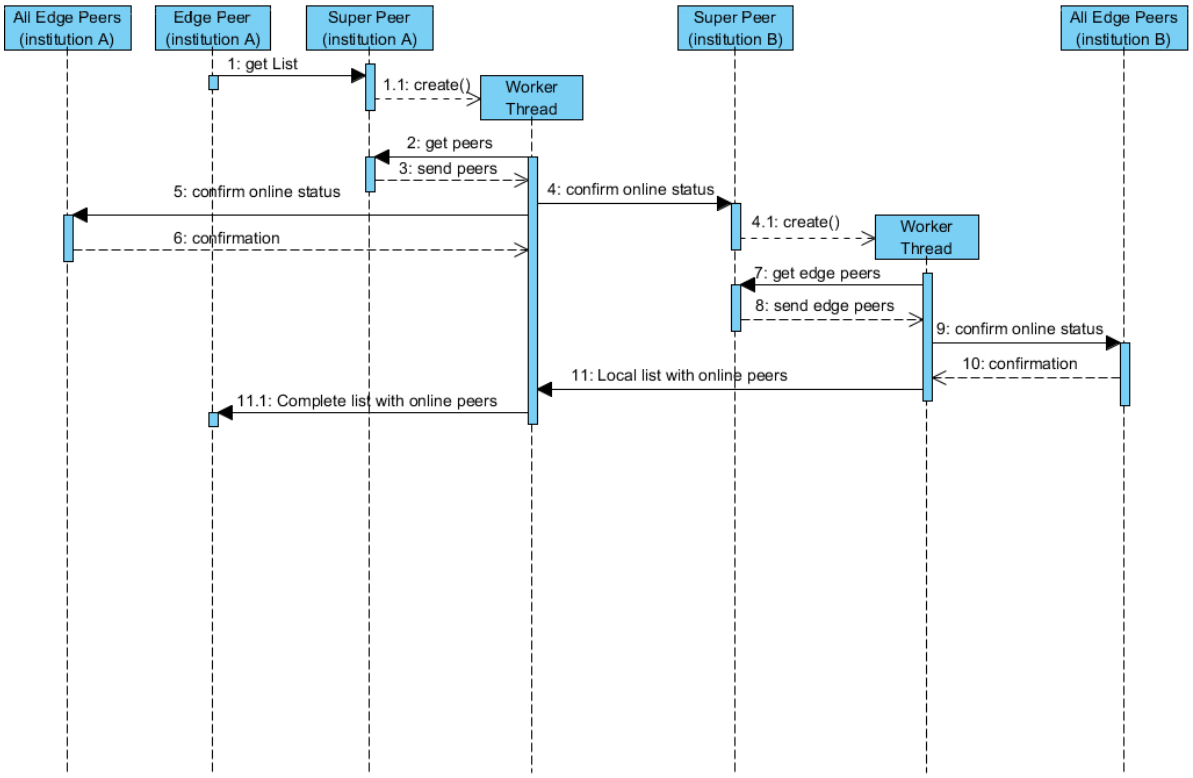


Figure 9.5: Creating a list with all peers currently online

9.6 Provide IP and Port of a Particular Peer

The routing information about known peers is saved automatically by JXTA in Route Advertisements when the peers are discovered. If the routing information has changed, the advertisement is updated at the next contact with the peer. When the exact address of an edge peer is requested from the super peer, the Route Advertisement of this edge peer is retrieved from the cache of the super peer, so the IP address and port number can be obtained from it.

9.7 Summary

This chapter details the prototype design of the component called α-Tracker. The structure of the prototype is presented and the interactions between peers for acquiring information and exchanging content are described with the help of UML diagrams. Furthermore, it is explained how the collaboration between JXTA edge peers, JXTA super peers and α-Doc-Coordinators fulfills the system requirements.

10 Implementation Issues

The following chapter overviews some aspects, that have emerged during the implementation. Chapter 10.1 describes an early attempt to build a purely decentralized network with JXTA and the disadvantages, that have resulted from this design. The rest of this chapter presents some JXTA principles, that need to be observed.

10.1 Building Purely Decentralized P2P Network with JXTA

Project JXTA is designed for partially centralized P2P networks, so building a purely decentralized one is not trivial. A very good example, that can be used as a basis for this purpose is the Gnutella protocol (see 5.2).

Firstly, all nodes in this JXTA network have to be JXTA super nodes, in order to forward queries through the network to the nodes they know. For a new node to join the network, it should know the IP address and port number of some of the existing ones, so it has someone to connect to. This information can be added to the configuration at startup as described in 10.2. The main disadvantage is that the exact address information for at least several other nodes, should be available. This is not trivial, because every JXTA peer requires separate port (see 10.3) and with many JXTA peers running on the same machine, ports should be assigned dynamically.

When every node knows some of the other members of the network, like the hosts in Gnutella, queries can be flooded from one node to another. This is a predisposition for a sizeable bandwidth consumption, but there is no other method for reaching every single node. This is why the TTL property of Gnutella queries should not be used in our case. Since participants exchange medical information, there can be no compromises with acquiring the latest available updates about the patient's condition.

10.2 Peer Configuration

When a peer is started for the first time, JXTA assigns a PID[1] to him. This PID is used only by JXTA and is saved in a XML-file together with everything else about the configuration of this peer. The configuration includes, which protocols should be used (HTTP, TCP, multicast), the ports to be opened and a list with the addresses of known super peers. This configuration file is retrieved by every start of the peer, so the same PID can be assigned to it.

The class *Configurator.java* is called automatically upon peer start and adjust the list with known super peers, if necessary. The design foresees that each edge peer will know only the super peer of its institution, but other super peers can be added too. For a super peer to join the super group and respectively become a part of the DHT of our network, it should know the IP address of at least one other super peer, in order to connect to it and retrieve the information about the rest of the group. This address needs to be given to the configurator, when starting the peer. Unfortunately, new IP addresses can not be dynamically added to the list in the configuration, while the peer is running.

10.3 Dynamic Port Assignment

Every JXTA peer requires a separate port for communication. One participant can take part in many α-Flows and respectively receive many α-Doc-Replicas. Therefore, the port number, which every α-Doc-Replica receives, can not be foreseen and should be assigned dynamically according to the currently available ports. This is necessary for the α-Doc-Coordinators on the super peer machine as well.

10.4 Starting α-Doc-Coordinator

One of the main principles of JXTA is that every user group is a derivation of the World Peer Group. When a peer wants do join a new group, it has to use the basic scope of services of the World Peer Group, in order to discover or create this new group. However,

1 Peer ID

the design of JXTA has defined the World Peer Group as singleton and there can be only one instance of it in a Java Virtual Machine. This results in a failure, when a second peer within the same virtual machine is started. Therefore, the JXTA super peer can not start the α-Doc-Coordinator on its own. A solution for this issue has to be found in future, in order to enable starting α-Doc-Coordinators on demand.

11 Discussion

The following chapter gives an overview about how this module of the α-Flow project can be extended to comprise more functions. JXTA can ensure a secure channel for transferring data and a collaboration with Drools can create a secure pipeline, that is able to bypass barriers. There are also several possibilities to improve the performance of the existing prototype with new cache management for edge peers or the introduction of headless α-Doc-Coordinators.

11.1 Combining Drools Pipelines and JXTA Pipes

Communication protocols, like TCP and UDP[1], are almost always blocked by firewalls. To ensure that data can always be transfered between participants it should be transferred over the HTTP protocol. JXTA can build pipes using HTTP. A collaboration between JXTA and Drools can be created, where JXTA bypasses the firewall and delivers the data to the Drools Pipeline for input. For this purpose the interface of α-Properties responsible for building pipelines should be overwritten, in order to include JXTA pipes too. This way JXTA cares for the physical transportation and Drools for inserting the data in α-Properties.

11.2 Security of the α-Flow Network

A security mechanism should be incorporated in the α-Flow network, due to sensitivity of medical information. As mentioned in 7.9, JXTA offers protocols for this purpose. Encryption interfaces are also included in the Java binding of the protocol. In future,

1 User Datagram Protocol

the introduction of a membership protocol with a set of credentials can ensure, that only authorized participants can join the network. The information exchanged between peers can be encrypted, in order to prevent third parties from accessing it. The encryption can be overtaken by JXTA, if the collaboration explained in 11.1 is realized.

11.3 Headless α-Doc-Coordinator

α-Doc-Coordinators are used only as a repository for the latest information. All components, for example the α-Editor, that do not take part in tasks connected with tracking, exchanging information and saving content can be stripped off the α-Doc-Coordinator in order to optimize its performance. This version of an α-Doc-Replica is called *headless*, because it has only the most essential functions.

11.4 Cache Management Optimization for JXTA Edge Peers

The cache, that every JXTA peer creates for its needs, takes about 10 MB of space on the HDD. This can become problematic for a participant, who takes part in many α-Flows and has to reserve this space for every α-Doc-Replica. The present cache management system was introduced in JXTA to optimize searching for advertisements, when the peer group is very large. In this prototype an edge peer communicates only with its super peer, so the cache can be abandoned completely. The edge peer knows the address of the super peer and connects to it at startup relying on the super peer to deliver information about the network from its cache. However, a JXTA peer can not function without advertisements, which are saved only in the cache. This is why cache creation can be regarded as a temporal need during the peer is functioning. At startup it is created so the peer can save advertisements there, but at shut down it is deleted.

12 Conclusion

Modern treatment of patients, where many participants from different institution are involved in the workflow, requires more than the traditional paper-based information exchange and coordination between the participants. The α-Flow model offers an electronic document-based solution, which combines both activity- and content-oriented workflows. The documents, called α-Docs, function as active software agents building a system, that is distributed amongst all participants giving them autonomy and enabling them to organize themselves according to the evolving workflow.

The organization of participants in such dynamic network can be implemented with peer-to-peer technology, which brings the ad-hoc ability - so important for a workflow with unknown set of actors and institutions. In this thesis a design has been presented based on the semi-centralized model for building P2P networks and a component responsible for tracking the participants, called α-Tracker, has been implemented. Protocol JXTA was utilized for the implementation, because it offers good tools for organizing the network with super peers and for enabling communication through barriers.

With the introduction of the α-Tracker to the α-Flow system, participants can form a group, synchronize their content through a common repository and exchange routing information needed for transferring data. Due to the deployment of one super peer per institution a reliable network is created, which can expand dynamically and organize itself. Participants can join or leave the network at any time, having the guarantee, that they can reach any other of its members.

Bibliography

[AH] Karl Aberer and Manfred Hauswirth. An overview on peer-to-peer information systems.

[Ber03] J.E. Berkes. Decentralized peer-to-peer network architecture: Gnutella and freenet, 2003.

[BS] Timothy Biron and Andrew Sprouse. Peer-to-peer systems term paper.

[Dod00] Catherine Dodson. Jabber technical white paper, 2000.

[GBL+03] Indranil Gupta, Ken Birman, Prakash Linga, Al Demers, and Robbert van Renesse. Kelips: Building an efficient and stable p2p dht through increased memory and background overhead. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, 2003.

[Hin04] Abram Hindle. Analysis of the p2p bittorrent protocol, 2004.

[JXT07a] Jxta java™ standard edition v2.5: Programmers guide, 2007.

[JXT07b] Jxta v2.0 protocol specification, 2007.

[MM02] Petar Maymounkov and David Mazieres. Kademlia: A peer-to-peer information system based on the xor metric, 2002.

[MSA05] Ralph Meijer and Peter Saint-Andre. Jabber, e-mail and beyond, 2005.

[NL09] Christoph P. Neumann and Richard Lenz. alpha-Flow: A Document-based Approach to Inter-Institutional Process Support in Healthcare. In *Proc of the 3rd Int'l Workshop on Process-oriented Information Systems in Healthcare (ProHealth '09) in conjunction with the 7th Int'l Conf on Business Process Management (BPM'09)*, Ulm, Germany, September 2009.

[NL10] Christoph P. Neumann and Richard Lenz. The alpha-Flow Use-Case of Breast Cancer Treatment – Modeling Inter-Institutional Healthcare Workflows by Active Documents. In *Proc of the 8th Int'l Workshop on Agent-based Computing for Enterprise Collaboration (ACEC) at the 19th Int'l Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE 2010)*, Larissa, Greece, June 2010.

[OTG02] Scott Oaks, Bernard Traversat, and Li Gong. *JXTA in a nutshell*. O'Reilly Media, 2002.

[SMLN+02] Ion Stoica, Robert Morris, David Liben-Nowell, David Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications, 2002.

[TAD+02] Bernard Traversat, Mohamed Abdelaziz, Dave Doolin, Mike Duigou, Jean-Christophe Hugly, and Eric Pouyoul. Project jxta-c: Enabling a web of things. In *Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03)*, Project JXTA, Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, CA 94303 USA, Bernard.Traversat@Sun.Com, 2002. IEEE Computer Society.

[TAP03] Bernard Traversat, Mohamed Abdelaziz, and Eric Pouyoul. Project jxta: A loosely-consistent dht rendezvous walker, 2003.

[YL09] Jie Yu and Zhoujun Li. Active measurement of routing table in kad. In *Proceedings of the 6th IEEE Conference on Consumer Communications and Networking Conference*, pages 1252–1256, Piscataway, NJ, USA, 2009. IEEE Press.

[ZHS+04] Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph, and John D. Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. In *IEEE Journal on Selected Areas in Communications*, volume 22, pages 41–53, 2004.