



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

NÁZEV PRÁCE

THESIS TITLE

SEMESTRÁLNÍ PROJEKT

TERM PROJECT

AUTOR PRÁCE

AUTHOR

Bc. VOJTĚCH KULIČKA

VEDOUCÍ PRÁCE

SUPERVISOR

Ing. JOZEF MLÍCH

BRNO 2010

Abstrakt

Výtah (abstrakt) práce v českém jazyce.

Abstract

Výtah (abstrakt) práce v anglickém jazyce.

Klíčová slova

Klíčová slova v českém jazyce.

Keywords

Klíčová slova v anglickém jazyce.

Citace

Vojtěch Kulička: Thesis title, semestrální projekt, Brno, FIT VUT v Brně, 2010

Thesis title

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana inženýra Jozefa Mlícha

.....

Vojtěch Kulička
January 10, 2011

Poděkování

Zde je možné uvést poděkování vedoucímu práce a těm, kteří poskytli odbornou pomoc.

© Vojtěch Kulička, 2010.

Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Contents

1	Introduction	2
2	Makneto	4
3	Telepathy	6
3.1	Telepathy basic terminology	7
3.2	Empathy	11
4	Existing solutions with whiteboard and VoIP	13
5	Conclusion	15

Chapter 1

Introduction

Human is a social creature and likes to chat, share feelings and ideas. At first we managed to do so by making simple sounds. Those sounds later on developed into words. Then much later the human race started to feel the need to record what we were thinking. We made up symbols and started to write. As the society grew and spread, we wanted to communicate with people from other tribes and villages. At first we would travel and use spoken words, but as the distances grew we figured we can have our thoughts delivered in writing. Mail was born. In 1844 telegraph was invented by Samuel Morse followed by telephone in 1874 by Alexander Graham Bell. And finally in 1969 the Internet was created. All of these inventions aimed to provide means of communication to satisfy the needs of the evolving society.

In the early days of the Internet email was the main means of communication. And just like regular mail people would have their electronic mailboxes to which the emails were delivered. Email was a huge step forward for it provided a way to almost instantly deliver text from one place to another regardless of the distance for free. The main disadvantage of email is that people had to check their mailboxes read new mail and then reply. It is just neither fast nor convenient enough for team cooperation when team members are far apart. For those and other purposes like chatting Instant Messenger programs were introduced.

An IM program offers realtime communication between two people via text messages that are delivered from one user to another instantly. Instant messengers became very popular and started adding on features like multiuser chat, various games and most importantly VoIP (Voice over IP) support. VoIP capable IM like skype have become extremely popular at first for making it possible for people to call each other for free over the internet. Later video conferencing capability was added, so you could talk and see you colleague at the same time. One more thing comes in extremely handy when working in a team - a whiteboard.

At this time there is no usable IM providing VoIP and shared whiteboard for GNU/Linux. This thesis aims to add VoIP support to an existing XMPP client with shared board called Makneto. Makneto was created by Jaroslav Řezník as a master's thesis in 2008. At this point it is using iris library for XMPP communication. The shared board data is also transferred over XMPP. One of the goals of this thesis is to port Makneto to telepathy, which is now a very reliable and robust library for communication for numerous protocols.

The following chapter gives a detailed description of the current version of program Makneto. We will find out about it's architecture, strengths and weaknesses.

Chapter three is about Telepathy communication framework, how it works, what it consists of. There is also a description of a IM client application Empathy based on Telepathy.

Current audio and video streaming protocols are listed and discussed in chapter five. Based on this chapter a suitable protocol is chosen for the implementation. Finally the implementantion of VoIP into Makneto is in chapter five.

Chapter 2

Makneto

Makneto is an instant messenger with a shared board capability. It was written by Jaroslav Řezník as his master's thesis in 2008. Makneto is written in C++ using Qt version 4 and KDE 4 libraries. Qt is a application framework written in C++. Qt extends C++ and instead of callback function uses a concept of signals and slots, which is as opposed to callback functions type-safe. It runs on all major computer platforms like GNU Linux, Microsoft Windows and Mac OS X. Moreso it is supported by mobile device platforms such as Symbian and Microsoft Windows Mobile. There is also a port to Google Android called Lighthouse. Qt was developed by company named Trolltech and was available under two licenses. First was a commercial license allowing companies to write proprietary applications for a fee. Second was GNU General Public License, which was of course free. Thanks to the commercial license the Qt documentation is one of the best among any software available under GNU GPL. In june 2008 Troltech was acquired by Nokia. Nokia decided to make the source code available so that anyone could contribute. In January 2009 with the release of Qt 4.5 another licensing option was added - Lesser General Public License.[\[10\]](#)[\[?\]](#)

Besides Qt Makneto utilizes functionality provide by KDE 4[\[8\]](#) libraries, which makes Makneto unable to run on a different operating system than GNU Linux. KDE is a desktop environment created by Matthias Ettricht in 1996 for he felt the need for a good quality window manager for Linux. KDE is currently in version 4, which brought great features. There is a new desktop called Plasma, which allows users to display widgets called plasmoids directly on the desktop. That allows users to have a TODO list, calendar, translator, weather forecast, system monitor and much more right in front of them on their desktops without having launch any of those to get the information they need. It makes users' work eesier and more efficient. KDE and GNOME are the most common window manager in Linux. GNOME offers stable useful environment and is influenced mostly by large businesses using it. KDE is more flexible and works more with the look and feel.

Makneto communicates using XMPP/Jabber protocol implemented in Qt-based C++ library called Iris. All of the Iris is primarily used by Psi instant messenger. It's development is still quite active and it supports all of Jabber's key features. Iris' downside is very poor documentation. It's wiki is very brief and all to all gives one example. The only way to find out how it works is browsing through Psi code.[\[?\]](#)

Makneto's shared whiteboard is based an official extension of XMPP SVGWB by Joonas Govenius[\[2\]](#). SVGWB is implemented in Psi and Makneto utilizes their code. SVGWB defines all the necessary actions like whiteboard session initianion including invitation and mainly how to send and receive information about the grahical objects using XMPP text messages. The actual graphical object representation is defined by SVG[\[3\]](#)(Scalable Vector

Graphics) by W3C. SVG describes graphics objects using vectors in XML format. Makneto uses just a subset of SVG called SVG Tiny[4] as it does not need all of the features of SVG. Tiny SVG describes two-dimensional vector graphics and raster graphics and multimedia. To sum up Makneto's whiteboard features, it allows you to draw lines, rectangles, ellipses, circles, sketch using a paintbrush and input images in jpg and png formats. Graphical objects are resizeable, can be rotated and copied.

Makneto runs in one window (see figure ??). User's contact list is on a panel on the left hand side. The whiteboard and chat session are initiated through the contact list. Makneto handles more session at once each in a separate tab. Using the application is very comfortable although it from dies time to time.

Chapter 3

Telepathy

Telepathy is a modular communications framework for building real-time communication applications. It supports numerous communication protocols as pluggable backends e.g XMPP/Jabber(telepathy-gabble), SIP(telepathy-sofiasip), MSN(telepathy-butterfly) etc. Each of telepathy's components runs in a separate process as desktop service and communicates via D-Bus. The components are shared by telepathy clients. For example if there are two clients using XMPP they both use the same instance of telepathy-gabble. To get a better idea of how this concept works take a look at figure 3.1.[9]

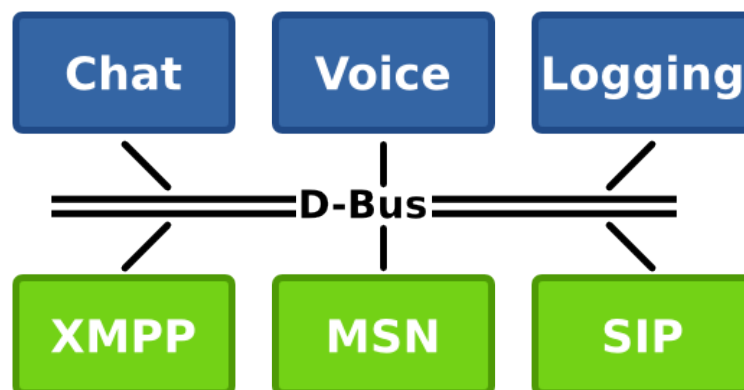


Figure 3.1: Telepathy architecture[9]

There are several features making telepathy very useful as a communications framework.

- **Robustness** - all the components are independent. If one crashes, others will not be affected
- **Ease of development** - the components can be replaced without having to stop the service
- **Language independence** - since telepathy components use D-Bus for communication among themselves, any language that has D-Bus binding might be used to write them
- **Desktop independence** - D-Bus is present in both main Linux window managers

GNOME and KDE, so the same telepathy components could be backend for appropriate frontends.

- **Code reuse** - the client applications do not have to worry about protocol specifics, which are handled by Telepathy. The client can use more protocols by making no or small alterations to the code.
- **Connection reuse** - more than one Telepathy client can use the same connection simultaneously:

3.1 Telepathy basic terminology

Telepathy is a very powerful framework and as such it is also complicated. To successfully write programs using Telepathy we need to know what telepathy consists of and what it is based on.

D-Bus

D-Bus is a kind of inter-process communication. It allows two applications running in different processes, written in different programming language communicate. More so these applications may communicate directly, without having to go through message bus daemon. There two types of D-Bus. First is a system bus used for events such as “USB device disconnected” or “printer out of paper.” Second type is per-user-login-session bus, which is used by user applications. D-Bus low level API is represented by libdbus and it requires XML parser(libxml or expat) to work. Higher level language bindings such as Qt, GLib, Java etc. are built on top of libdbus and offer more convenient way of using D-Bus, although they add more dependencies.[\[7\]](#)[\[9\]](#)

Each process that wants to communicate over D-Bus will need to use most the following depending on it's nature:

- **Unique name** - is an unique id(e.g. 2.1) assigned by D-Bus daemon to the client application. Unique name is similar to a public IP address.
- **Wellknown name** - is similar to a DNS name. If a process wants to make a service available to other processes it requests a wellknown name. If another process wants to access the service it uses the wellknown name to do so. Wellknown name might look like this: org.freedesktop.Telepathy.ChannelDispatcher.
- **Object path** - is a path to an object that is exported by process running a service.
- **Interface** - is a way of requesting a service using signals or methods. Each D-Bus client must register at least one interface and each interface provides at least one method or signal. Every interface needs to have to name like a wellknown name.
- **Method** - is implemented in the object specified by object path and exposed in the interface for that object for other processes to use.
- **Signal** - is a D-Bus signal client process can connect to it's callback function. If a signal is invoked the callbacked function is called.
- **Property** - is used for exposing D-Bus object's properties. To do so the object must implement org.freedesktop.DBus.Properties interface.

The following figure 3.2 shows an example of two programs connected to D-Bus to be able to communicate with each other. Program B provides a service called `org.freedesktop.foo.Bar` (wellknown name) and it's id is 1.3. Program A does not provide any service and thus does not need any wellknown name. It just needs an id(1.2) to use other programs' services.[9]

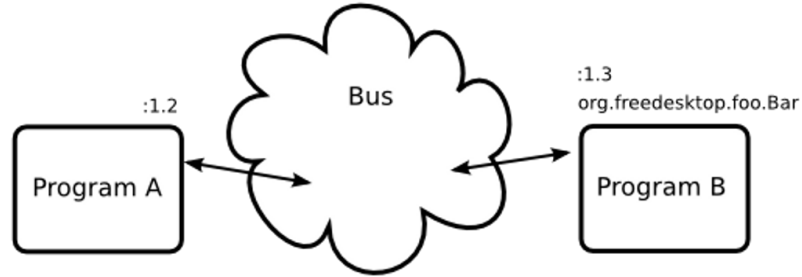


Figure 3.2: D-Bus id and wellknown name example[9]

The figure 3.3 shows an overview of all of the terms described above in a simple diagram.

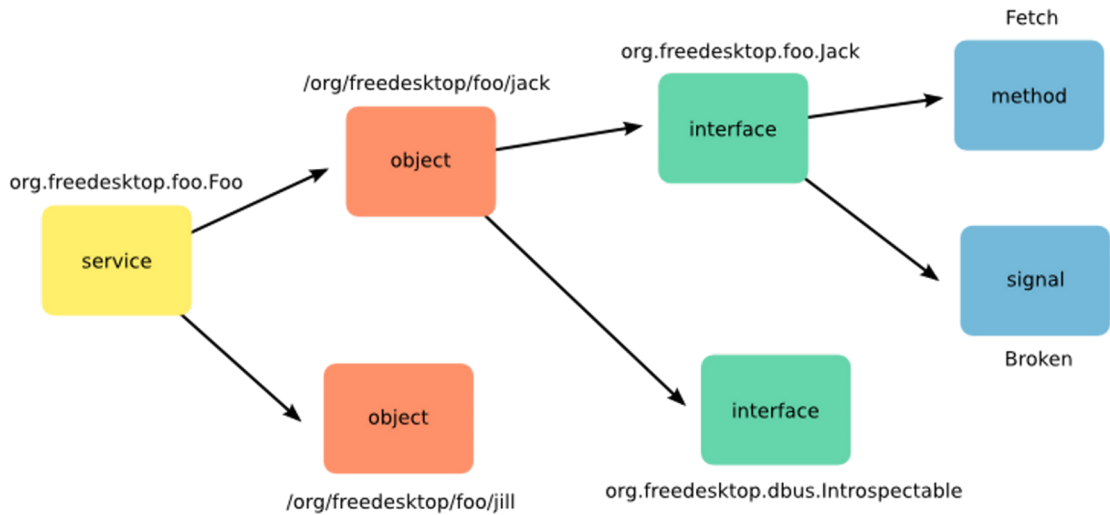


Figure 3.3: D-Bus architecture[9]

D-Bus is a key component of Telepathy framework. Telepathy supports many protocols all of which might provide different capabilities. For example IRC does not support avatars while XMPP does. Eventhough avatar feature is supported by XMPP protocol it might not be supported by the server we are connected to or by the opposite client in case of peer-to-peer connection. The available features are exposed by D-Bus Properties Interface. That is an easy way of determining the protocol, server or client capabilities.[7]

Mission Control

Mission Control is a Telepathy component that implements Account Manager and Channel Dispathter and it's primary purpose is to encapsulate those two.[9]

Account Manager

Account Manager is responsible for handling accounts(e.g. XMPP, ICQ, MSN etc.). It is accessible by well-known name on D-Bus - org.freedesktop.Telepathy.AccountManager. A client application first creates an account using the CreateAccount() method, supplying it with ConnectionManager, protocol and display name. Account Manager creates and then as long as the account is active maintains Connection to that account. To create a Connection a Connection Manager is called. An account may be valid or invalid. Valid accounts may establish a Connection, whereas invalid can't. The list of valid and invalid accounts is accessible via ValidAccounts and InvalidAccounts properties respectively.[9]

Account

When the Account Manager's CreateAccount() is called it returns an Account object. Account object registers with D-Bus and has an object path /org/freedesktop/Telepathy/Account/CM/PRO CM stands for Connection Manager(e.g. gabble, salut, butterfly etc.), PROTOCOL is substitution for a protocol name and ACCDN is Account Display Name. The Account object implements org.freedesktop.Telepathy.Account interface. Features supported by this interface depend on the protocol used and the server-side software. The Account settings are done via org.freedesktop.Telepathy.Properties interface. All available attributes can be obtained by calling the GetAll() method provided by the Properties Interface. The GetAll() method is very convenient for it returns all the properties at once in single D-Bus call. Similarly there is a method for setting all properties at once - Account interface's UpdateParameters() method. The following tables lists all properties of the Account object available. Table ?? shows properties that can be configured and table ?? shows properties just for reading. Some features are available via specified interface, e.g. avatar. Avatar used to be a property, but now it has it's own interface. The Interfaces property of the account lists all interfaces of additional features.[9]

Connection Manager

Connection Manager supplies Account Manager with Connections. It is not directly used by the client program. Account Manager requests connection for active accounts.

Connection Manager is a protocol-dependent Telepathy component. Different protocols need different Connection Managers. For example if the client application wants to communicate using XMPP/Jabber it has to use Telepathy-gabble and for MSN Telepathy-butterfly is required. Some Connection Managers can communicate via more than one protocol, for example Telepathy-haze. To see what protocols are supported there is ListProtocols() method implemented by the Connection Manager.[9]

Connection

Connection represents an active protocol session. It is associated with an Account and is created by Connection Manager based on a request of the Account Manager. Connection implements org.freedesktop.Telepathy.Connection interface and additional interfaces depending on the protocol. List additional interfaces available can be retrieved by checking the Interfaces property. The most common interfaces are listed below[9]:

- **Contacts** - used to get as much information about a contact as asked in one D-Bus call.

- **Aliasing** - serves for setting aliases for contacts and checking if the contacts have changed their alias themselves.
- **Avatars** - interface to one of the most popular protocol features. Allows users to set their avatars and retrieve other users' avatars.
- **ContactCapabilities** - retrieves capabilities of contacts' Clients to see what features they support. Checks for example for VoIP or file transfer support.
- **Location** - lets user publish his or her current location as well as find out his or her contacts' whereabouts.

Channel Dispatcher

This component handles Channels incoming from active Connections of valid Accounts. Channel Dispatcher monitors available or activatable Telepathy Clients through D-Bus. Clients register with user's session D-Bus and provide a CLIENT_NAME.client file. Both of those serve as a way to publish Client's properties including a channel filter. The Channel Dispatcher based on these properties knows what kind of a client it is and what type of channels it is interested in (channel filter). If the Client is running then the properties are acquired via the Client interface. If the client is not running and is activatable then the .client file is used by Channel Dispatcher to pre-look up the properties and if they match the incoming Channel, the Client is activate. So providing the .client file only makes sense for activatable Clients.

When a Channel comes in from one of the Connections Channel Dispatcher notifies appropriate Clients. There are three kinds of clients - Observer, Approver and Handler (see 3.1). The Channel is dispatched to all Observers and all Approvers with a matching channel filter. The Approvers choose Handler to handle the Channel. Should the Client fail, Channel Dispatcher may recover from such error and look for another Handler.^[9]

Channel

Channel allows the local client to exchange various kind of data with a remote server. It is associated with a Connection and always implements at least two interfaces. The first is org.freedesktop.Telepathy.Channel and the second depends on the Channel type. Channels for text messaging will be of type Text and will implement org.freedesktop.Telepathy.ChannelType.Text interface. The following list shows most common types of Channels^[9]:

- **ContactList** - used to get information of contacts in user's contact list.
- **Text** - designed for exchanging text messages.
- **Call** - used for VoIP and video calls.
- **FileTransfer** - Channel for sending and receiving files.
- **ContactSearch** - is used when a user wants to find a contact on a server.

Channels are created using two methods - CreateChannel() and EnsureChannel(). These methods are implemented by both Channel Dispatcher and Connection. When calling either of those methods on Channel Dispatcher the resulting Channel will go through the

procedure of looking for handler as described above. When using directly the connection the calling application must handle the Channel itself as the Channel Dispatcher will not interfere. It is also possible to supply the Channel Dispatcher with a preferred handler and thus achieve the same effect. It is better to use the Channel Dispatcher for if the client should fail it may dispatch the Channel to another handler.[9]

Both `CreateChannel()` and `EnsureChannel()` methods provide a Channel. The difference between the two is that `CreateChannel()` creates actual new Channel whereas `EnsureChannel()` will attempt to reuse an existing Channel with the same properties. If there is no Channel to be reused or it is being used by a different application than it creates a new Channel, just like `CreateChannel()`. Typically `CreateChannel()` is used for `FileTransfer` and `ContactSearch` and `EnsureChannel()` for `Text`, `StreamedMedia` and `ContactLists` Channels.[9]

Client

Client is an application that wants to use Telepathy. It needs to register a well-known name in `org.freedesktop.Telepathy.Client` namespace, e.g. Empathy registers `org.freedesktop.Telepathy.Client.Empathy`. Then it provides a `.client` file where purpose of which is described above. Telepathy defines three types of clients - Observer, Approver and Handler. All of these need to provide appropriate channel filter, e.g. Observer provides `ObserverChannelFilter`. Based on the published filter the Channel Dispatcher dispatches an incoming Channel to the Client or not.[9]

Observers are called upon a creation of a new Channel. They monitor Channels and provide the acquired information to user. The observers have different functions based on the type of observed Channel, e.g. Text Channel observer might serve as a logger and FileTransfer observer as a file transfer progress monitor. Observer is must not interfere except for when the user interaction like hitting the cancel button in a file transfer progress window.[9]

Approver is a Telepathy Client that is supposed to accept the incoming Channel and decide, which Handler it is dispatched to. The Channel Dispatcher provides Approvers with a list of possible Handlers. Approver notifies the user of a new Channel and lets him or her decide whether to accept or reject it. Similarly the user is allowed to choose which Handler will handle the Channel. Handler might also be chosen by the Approver itself. Approver does not call methods just like the Observer. Calling methods is up to the Handlers. For example if there is an incoming file transfer the Approves lets user decide whether to accept it or not, but the `AcceptFile` method will be called by the chosen Handler.[9]

The last client is Handler. Handler does all the interaction with the Channel. A typical example of a Handler is chat-window. It displays messages and allows the user to send text messages back.[9]

The figure 3.4

3.2 Empathy

Empathy is a multiprotocol instant messaging application based on Telepathy. In the terms described above Empathy is a Telepathy client. It is written in python using Telepathy-python bindings. Empathy registers a well-known name with D-Bus and communicates with Telepathy components to provide the communication services. Empathy supports text messaging, file transfer, voice and video calls over various protocols. Supported protocol include Google Talk, XMPP/Jabber, MSN, IRC, AIM Facebook, Yahoo!, Gau Gadu,

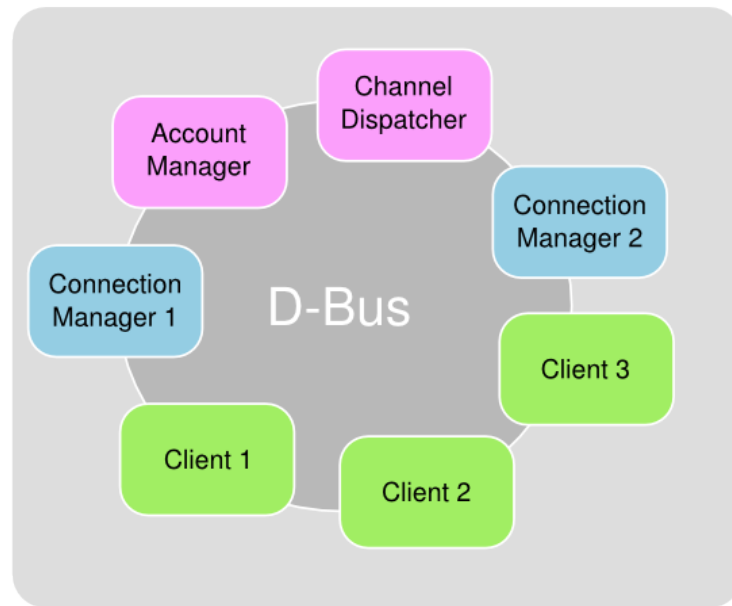


Figure 3.4: Telepathy components registered with user session D-Bus.^[9]

and ICQ. For some of those protocols like Google Talk and XMPP voice and video calls implemented. The support of the protocols depend on Telepathy Connection managers installed. Additional functionality includes sharing users' whereabouts among themselves, automatic reconnection when internet connection is reestablished and automatic changes of presence to away and extended away.^[?]

The current stable version of Empathy is 2.32.2 and it is a default communication application in GNOME releases since version 2.24 instead of Pidgin. Empathy also replaced Ekiga - program for voice calls and video-call. It became an ultimate free communication tool. Empathy's GUI is takes after Gossip, which is an older IM application for GNOME.

Chapter 4

Existing solutions with whiteboard and VoIP

There is a number of existing communication applications offering voice calls and shared whiteboard. Some more popular, more advanced, more user-friendly, offering more features or better support than others. Some of those program support video call and some even conference calls. We shall go over the existing solutions that implement both shared whiteboard and voice calls.

skype

Skype[5] is the most popular and most used common VoIP application of all. It is available for all major computer platforms (Microsoft Widnows, Mac OS X and GNU Linux) as well as mobile platforms like Android and even Apple's iOS. Although skype is available for Linux it is not well supported. The latest version of skype for Linux is 2.1.0.81 Beta while skype for Windows is of version 5.1. Finally skype is built-in on more and more TVs.

Skype communicates using Skype protocol, which is proprietary. Recently much effort is being put in reverse engineering the skype protocol. The motivation for this is simple. Skype is used by tens of millions of users every day, but the support for Linux is at this point almost nonexistent. Linux users have had enough and plan to create an open source client capable of communicating with skype. The wikipedia skype protocol page[11] is filling up with details. While most of IM programs utilize client-server communication scheme, skype uses peer-to-peer model. The skype network consists of nodes, supernodes and login servers. Nodes are clients. Each client keeps addresses of a number of supernodes. Supernodes are clients with good-enough bandwith and public IP address. Supernodes forward traffic to clients behind NAT, PAT or restrictive filters. This decentralized architecture seems to be working well although there was an outage on December 22nd 2010. Skype officials claim it was due to lack of supernodes[1].

Features of skype include calling landlines and cell phones and sending text messageses and vice versa - SkypeOut and SkypeIn. The feature list continues with voice and video calls, multi-user chat, conference calls, voice mail and screen sharing. The newest and long expected feature of video conference was introduced in may of 2010 in skype 5.0.

Though closed program, skype provides an API for developers who want to create extensions called skype extras. Skype extras include a wide range of utilities that might be plugged into skype. The extras uSeeToo, TalkAndWrite, WhiteBoardMeeting and Sketch

Pad all provide shared whiteboard each in their own way.

Windows Live Messenger

Windows Live Messenger[12] was first released in July 1999 as MSN Messenger and offered just text messaging with users of AOL Instant Messenger[6]. Due to AOL's constant effort to block Microsoft from it's network Microsoft gave in and removed the feature. Since then MSN Messenger could only connect to MSN Messenger Service. In 2001 with the release of Windows XP, MSN Messenger 4.6 came out with voice call support. The last version of MSN Messenger, version 7.5, featured video calls. Windows Live Messenger 8.0 was released in june 2006 and that was the end of the name MSN Messenger.

Windows Live Messenger utilizes client-server model and communicates using Microsoft Notification protocol over TCP. The first 7 versions of MSNP were disclosed to public, but since version 8 the details have been kept a secret. MSNP does not use encryption so eventhough MSNP's description was not published it was not hard to put it together using packet sniffers.

The live Messenger is available for Windows, Mac OS X and recently was integrated into Microsoft's game console Xbox 360. It features social network integration, offline messaging, games and applications, voice and video calls and standard IM features. An interesting feature is Multiple points of presence allowing user to be connected on two devices. Shared whiteboard is available as an extra application and is not capable of multi-user session.

Chapter 5

Conclusion

Závěrečná kapitola obsahuje zhodnocení dosažených výsledků se zvlášť vyznačeným vlastním přínosem studenta. Povinně se zde objeví i zhodnocení z pohledu dalšího vývoje projektu, student uvede náměty vycházející ze zkušeností s řešeným projektem a uvede rovněž návaznosti na právě dokončené projekty.

Bibliography

- [1] Terry Brock. Skype outage today.
http://blogs.skype.com/business/2010/12/skype_outage_today.html.
- [2] Joonas Govenius. Telepathy wiki, channels.
<http://xmpp.org/extensions/inbox/wb.html>, 2006-06-19.
- [3] Kolektiv autorů. Scalable vector graphics(svg). <http://www.w3.org/Graphics/SVG/>.
- [4] Kolektiv autorů. Svg tiny 1.2 specification. <http://www.w3.org/TR/SVGTiny12/>.
- [5] Skype.com. Skype. <http://www.skype.com>.
- [6] WWW stránky. Aol instant messenger. <http://www.aim.com>.
- [7] WWW stránky. D-bus. <http://www.freedesktop.org/wiki/Software/dbus>.
- [8] WWW stránky. Kde. <http://www.kde.org/>.
- [9] WWW stránky. Telepathy wiki.
<http://telepathy.freedesktop.org/doc/book/index.html>.
- [10] WWW stránky. Wikipedia - qt(framework).
[http://en.wikipedia.org/wiki/Qt_\(framework\)](http://en.wikipedia.org/wiki/Qt_(framework)).
- [11] WWW stránky. Wikipedia, skype protocol.
http://en.wikipedia.org/wiki/Skype_Protocol.
- [12] WWW stránky. Windows live messenger 2011.
<http://explore.live.com/windows-live-messenger>.