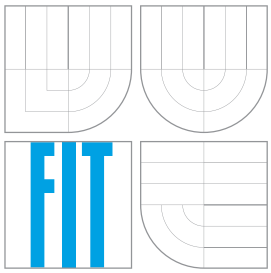
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



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 8, 2011

Poděkování

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

Contents

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 telepthone 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 messangers 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 number three focuses on XMPP/Jabber communication protocol. It talks about it's features and limitations.

Chapter four is about Telepathy communiation 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 six.

Chapter 2

Makneto

Chapter 3

XMPP

Chapter 4

Telepathy

Telepathy is a modular communications framework for building real-time communication applications. It supports numerous communication protocols as plugable 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 4.1.[2]

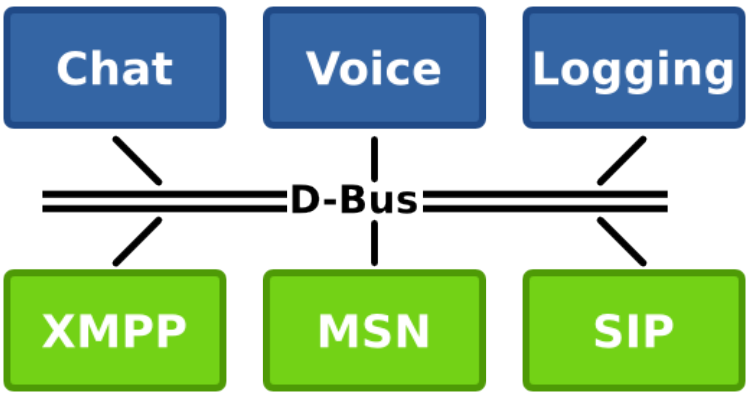


Figure 4.1: Telepathy architecture[2]

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:

4.1 Telepathy basic terminology

Telepathy is a very powerful framework and as such it is also complicated. To successfuly 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.[1][7]

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 requsets 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 impleneted 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 objet must implement org.freedesktop.DBus.Properties interface.

The following figure 4.2 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(wellkno 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.[7]

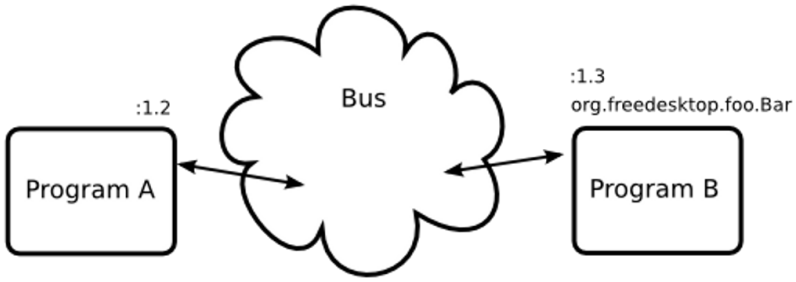


Figure 4.2: D-Bus id and wellknown name example[7]

The figure 4.3 shows an overview of all of the terms described above in a simple diagram.

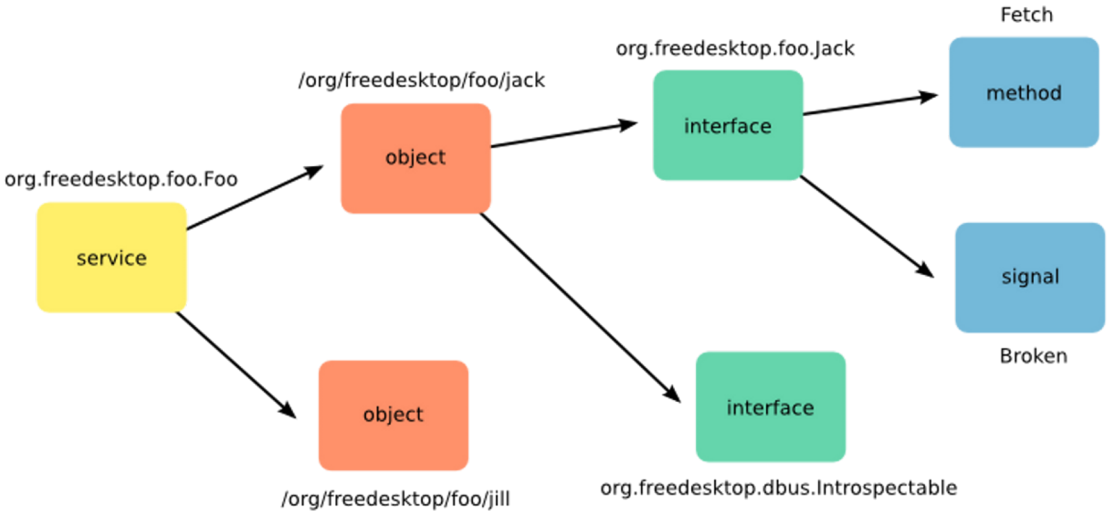


Figure 4.3: D-Bus architecture[7]

D-Bus is a key component of Telepathy framework. Telepathy supports many protocols all of which might provide different capabilites. 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.[1]

Mission Control

Mission Control is a Telepathy component that implements Account Manager and Channel Dispather and it's primary purpose is to encapsulate those two.[6][5]

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.[5][3]

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, butterly 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 propeties at once in signle 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 4.1 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.[5][4]

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.

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

Configurable properties			
Property	**Type**	**Description**	**Example**
DisplayName	String	Name of the account, used in the account object path. It is usualy displayed by the client application	“Bob”, “Work”
Icon	String	Name of an icon to be used for the account. If it is not present the system’s icon theme and if not it may be automatically chosen	“im-jabber” (jabber’s lighbulb), “im-icq” (icq’s green flower)
Enabled	Boolean	Whether or not the Account can go online. Enabled property is different from the valid property of the account with the Account Manager.	True
Nickname	String	This name will appear to other user’s that have this account in their contact list, unless they decide to change it on their side of course.	“Bobman”
AutomaticPresence	SimplePresence*	The status this account will have whenever it is brought online.	
RequestedPresence	SimplePresence*	The presence status we wish to change the current status to.	
ConnectAutomatically	Boolean	Whether or not a Connection should be requested by the Account Manager automatically.	True

Table 4.1: Configurable properties of Telepathy Account[4]

Nonconfigurable properties		
Property	**Type**	**Description**
Interfaces	DBus_Interface[]	A list of interfaces for extra features supported by the account, e.g. org.freedesktop.Telepathy.Account.Interface.Avatar
Valid	Boolean	Whether or not this account can be connected(brought online) by the Account Manager.
Connection	Connection	Object path to the connection associated with account if there is any. Otherwise it is set to: '/', e.g. /org/freedesktop/Telepathy/Connection/gabble/jabber/
ConnectionStatus	int	Status of the connection. That is if there is any.
ConnectionStatusReason	int	The reason why the last change of status of the connection occured.
CurrentPresence	SimplePresence	Current presence of the account.

Table 4.2: Configurable properties of Telepathy Account[4]

depending on the protocol. List additional interfaces available can be retrieved by checking the Interfaces property. The most common interfaces are listed below:

- **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 4.1). 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.

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:

- **ContactList** - used to get information of contacts in user's contact list.
- **Text** - designed for exhanging 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.

Both CreateChannel() and EnsureChannel() methods provide a Channel. The difference between the two is that CreateChannel() creates actual new Channel whereas EnsureChannel() will attemp 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(). Typicaly CreateChannel() is used for FileTransfer and ContactSearch and EnsureChannel() for Text, StreamedMedia and ContactLists Channels.

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.E
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. Oberver provides ObserverChannelFilter. Based on the published filter the Channel Dispatcher dispatches an incoming Channel to the Client or not.

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.

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.

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.

Handle

Telepathy handles represent objects like contact, roster, chatroom etc. It is a number, assigned by Connection the object is related to, that identifies the object. In addition to the numeric id handles also have a string id(e.g. chatroom could have a string id makneto@jabber-discussion.com). Depending on the protocol resource the handle represents it is of an apropriate type(Handle_type property). Handles with different types may have the same numeric id. When comparing handles it should be always done by comparing the numeric id's, assuming we are comparing handles of the same type.

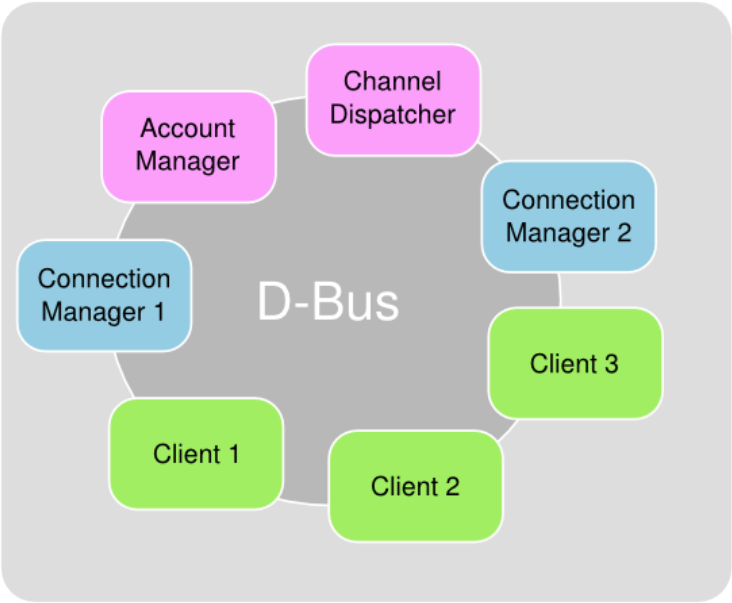


Figure 4.4: Telepathy components registered with user session D-Bus.[**?**]

The figure 4.4

4.2 Empathy

Empathy is a multiprotocol instant messaging application based on Telepathy. The current stable version of Empathy is 2.32.2 and it is a default communication application in current GNOME releases instead of Pidgin. It supports text messaging, file transfer, voice and video calls over various protocols. It's GUI is takes after Gossip, which is an older IM application for GNOME. Supported protocol include Google Talk, XMPP/Jabber, MSN, IRC, AIM Facebook, Yahoo!, Gau Gadu, Groupwise, ICQ and QQ. For some of those protocols like Google Talk and XMPP voice and video calls implemented. The suppport of the protocols depend on Telepathy Connection managers installed.[**?**]

Chapter 5

Audio and video streaming protocols

Chapter 6

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] WWW stránky. D-bus. http://www.freedesktop.org/wiki/Software/dbus.

[2] WWW stránky. Telepathy wiki.
http://telepathy.freedesktop.org/doc/book/index.html.

[3] WWW stránky. Telepathy wiki, account manager.
http://telepathy.freedesktop.org/doc/book/chapter.accounts.html.

[4] WWW stránky. Telepathy wiki, accounts.
http://telepathy.freedesktop.org/doc/book/sect.accounts.accounts.html.

[5] WWW stránky. Telepathy wiki, basic terminology.
http://telepathy.freedesktop.org/doc/book/sect.basics.terminology.html.

[6] WWW stránky. Telepathy wiki, mission control.
http://telepathy.freedesktop.org/wiki/Mission%20Control.

[7] WWW stránky. Telepathy wiki, using d-bus.
http://telepathy.freedesktop.org/doc/book/sect.basics.dbus.html.