Survey of software architecture V0.0

Name of system: Telepathy
Reviewer: Oleksii Kononenko
Date: 21.10.2011
Author of software: Robert McQueen
Author of book chapter: Danielle Madeley
Five star rating of book chapter: 3 *** Some good ideas. Reasonably clear

Purpose of system: The telepathy is the framework that can be used by developers to create software for on-line communication, which may involve text (instant messaging), file, voice (VoIP) and video transfer (videoconferencing). Using components through the D-Bus mechanism Telepathy makes development process simpler and allows code reuse.

The unique feature of Telepathy, as it is stated in the chapter, is that it offers a look at communication as a service that many other applications can talk to. This approach makes possible to build a complicated process of communication from one application, such as initiating an instant message communication from your email client, and then, for instance, file transfer from it.

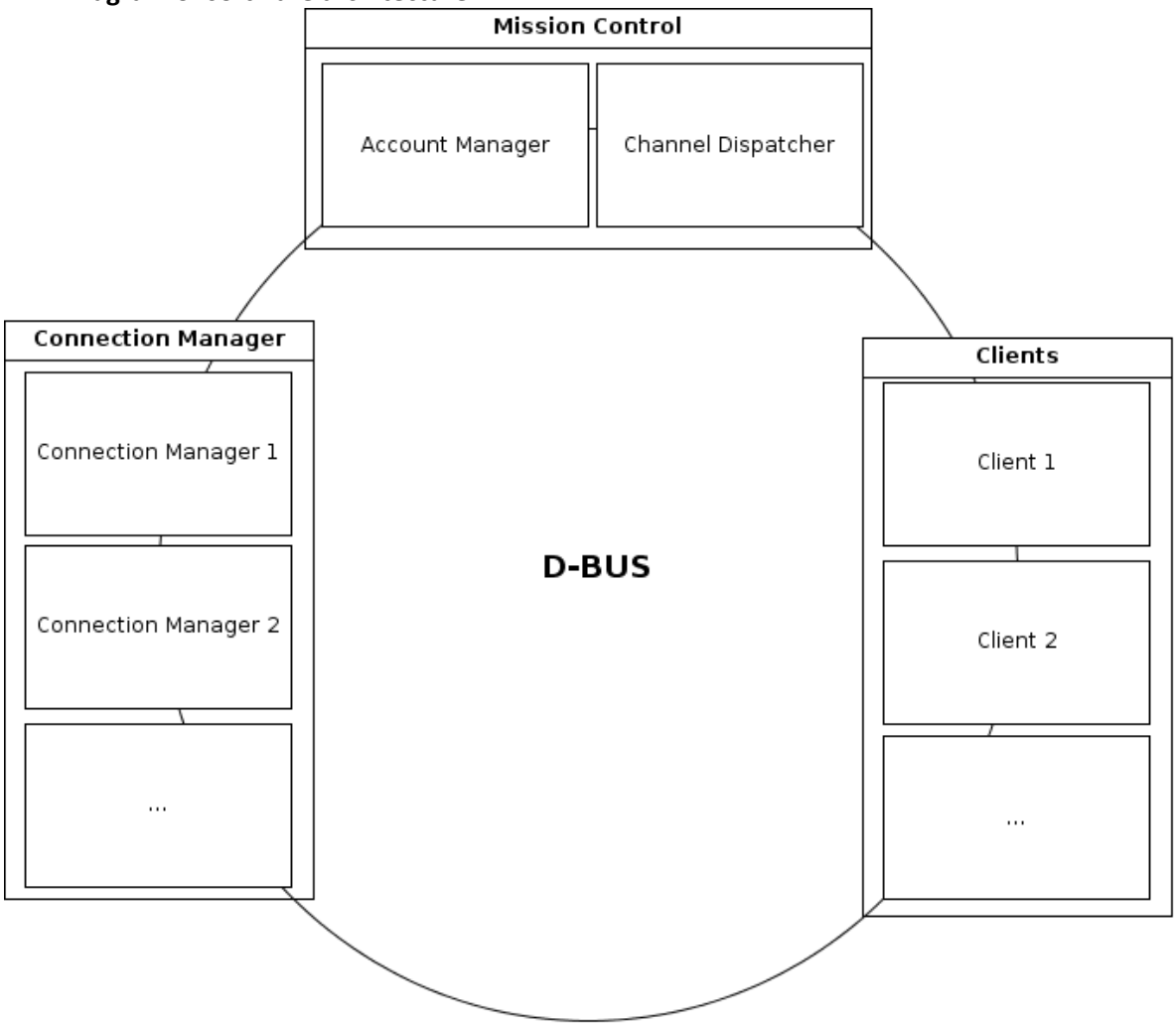
This idea (communication as a service) was found interesting by user community, and now Telepathy is included as a package in many official Linux distributions, such as Debian (lenny or later), Fedora, Ubuntu (feisty or later), and Gentoo.

Basic metrics

KLOC: 744 KLOC (1043 KLOC with blanks and comments)
Project start-up: 2005
Number of major releases: Current version is 6.0, so I guess there six major releases
Number of developers: 25 people can actually commit changes to the repository
Size of user community or number of installations: I've found no exact data about either size of user community or number of downloads. The main reason for this I guess is that Telepathy is a framework and that it's included with Linux destributions.
Major stakeholders: Linux community; also some Nokia phones use ideas of Telepathy
Use of concurrency: Because of the nature of D-Bus, all method calls that go through it are asynchronous
Implementation language: C (the main one)
Supporting software: D-Bus

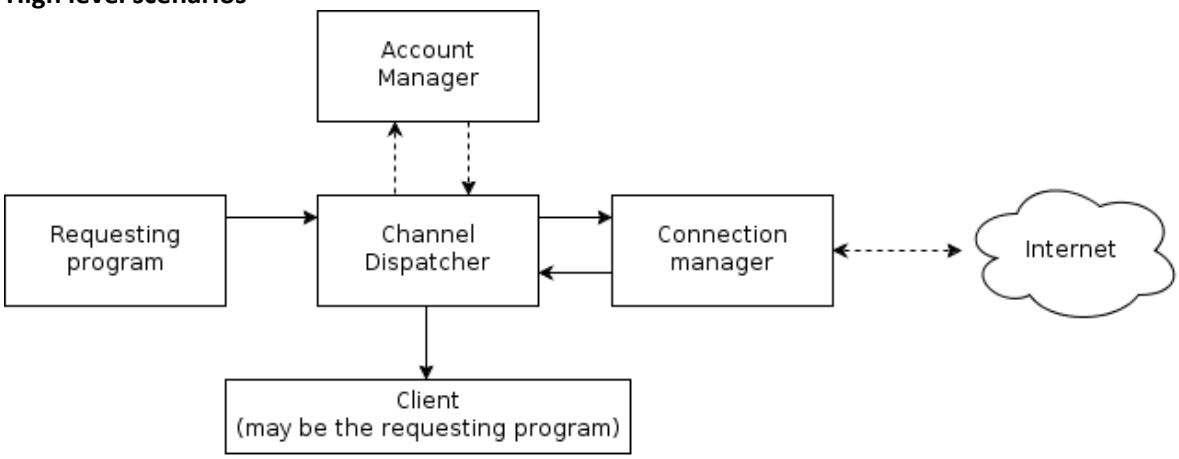
High level architecture

Diagram of software architecture



Each box here represents a module in the Telepathy. All these modules can be grouped into three categories – Mission Control (Account Manager and Cannel Dispatcher), Connection Manager (contains several Connection managers), and Clients (contains all clients of the framework). The circle here represents the bus (D-Bus), which is used by Telepathy; each box is connected to the bus.

High level scenarios



Let's take a look on a task of requesting channel, which is one of the fundamentals tasks in Telepathy.

- the requesting program makes a request to the Channel dispatcher; this request at least has information about channel type, target handle type and target
- if the request does not have information which connection to use, it will talk to the Account manager to find out this info
- the channel dispatcher talks to the specific connection manager and request the channel
- if the connection does not exist, the connection manager creates it
- the connection manager verify policy regarding number of allowed channels (single or multiple) for the connection, and return either an existing channel (for single channel policy) or a new one
- the Channel dispatcher find the appropriate host (client) for this channel; it may be the requesting program or another client

Data structures or algorithms: Anything special
Control flow and/or data key to the architecture if any: Because the Telepathy is built on D-Bus Message Bus, usually there are no direct calls between modules. Modules broadcast their calls to others, and other modules listen for incoming calls and identify whether they are "interested"

Architectural style: Message Bus Architectural Style and Modular design. The Telepathy is built on D-Bus (a message bus system) – all components of the software use D-Bus to communicate with each-other. Also the inner structure of Telepathy is clearly separated into modules with different set of features and purposes.

Major evolutionary changes: There are no evolutionary changes in terms of architecture. As it is stated in the chapter, developers have seen some problems or weak sides of the software and have tried to address them. The main problem that is stressed in the chapter is the huge traffic in D-Bus; and while many changes were made to address this issue, none of them are affected the overall architecture.

Performance bottlenecks: There were weak points in earlier versions of software such as traffic issue, which heavily slowdown user experience, and pseudo-synchronous API, which might freeze a caller; but all of them were solved with the evolution of Telepathy.

Real time: After evolutionary changes in Telepathy, there are no parts in it that might be critical for fast response. It seems to me that now all delays can appear only in network layer, which it is not the part of the framework.

Notation for architecture: In the book the author used UML-style diagrams to make some points clear

Methodology: No information

Appendix:
Kruchten's eight context attributes applied to Brown/Wilson systems

1. **Size:** L
2. **Criticality:** Lo
3. **Age of system:** M
4. **Rate of change:** Lo
5. **Business model:** open source
6. **Stable architecture:** Lo
7. **Team distribution:** UnK
8. **Governance:** UnK [Lo (?)]