Telepathy

Real-time Communications Framework
Rationale

• A new approach to real-time communications
• Unifying IM, VOIP and collaboration
• A brief look at desktop clients...
The Unix Way

Do one thing and do it well

IM Client
The Unix Way

Do one thing and do it well

UI

Protocol
The Unix Way

six

Do one thing and do it well

Contact List  Chats  Logging

AIM  MSN  XMPP
The Unix Way

twelve
Do one thing and do it well?
Heading The Same Way

- Gaim users want voice calls...
Heading The Same Way

- Ekiga users want IM...
Heading The Same Way

- Gossip users want more protocols...
This Sucks

- Massive duplication of effort
- Fragmentation of APIs
- Integration suffers badly
- Few reusable components for embedded devices
The Big Idea

• Move away from the monolithic client
• Split stuff into separate processes
• Run protocols as services on the desktop
• Create a standard API for clients to use presence, messaging, media, etc...
The Big Idea

VOIP Client -> D-Bus IPC
Chat Client -> D-Bus IPC
Logger -> D-Bus IPC
SIP Backend -> D-Bus IPC
XMPP Backend -> D-Bus IPC
MSN Backend -> D-Bus IPC

Telepathy
Benefits

✔ Do one thing and do it well
✔ Re-usable components
✔ Interchangeable user interfaces
✔ Share connections between UI programs
✔ Language (and license) independence
✔ Only run what you need
What we're doing...

- Telepathy is a Freedesktop project
- **Massively Important**: well-documented D-Bus API
- Some protocol backends
- Libraries so you can use them
Specification

• Connection manager objects give you connection objects
• Connections have interfaces: presence, aliases, avatars...
• Connections give you channel objects
• Channels have a type: IM, VOIP/video, contact list...
• Channels have interfaces: properties, groups...
Backend Implementation: Gabble

- Jabber/XMPP backend by Collabora
- Implements IM, multi-user chat and roster channels, presence, aliases, avatars...
- Support Google Talk and Jingle signalling for voice and video calls
Backend Implementation: Sofia-SIP

- SIP backend by Nokia and Collabora
- Based on Nokia's Sofia-SIP stack
- Support for voice/video calls and SIMPLE messaging
- Recently open-sourced: http://tp-sofiasip.sourceforge.net/
Other Backends...

- Rendezvous: Salut
- IRC: Idle
- MSN: Butterfly
- AIM/ICQ (aka Oscar): Wilde
Stream Engine

- Separate service to handle voice/video/etc streams, independently from the UI
- Signal information over a Telepathy media channel
- Uses Google's libjingle for NAT traversal
- Uses Farsight & GStreamer 0.10 for the RTP streams and codecs
- I'm working on a library...
Stream Engine

- Contact
- RTP
- XMPP Backend
- D-Bus
- UI
- Sound

Telepathy
Libraries

- libtelepathy (sorry!)
- telepathy-python
- telepathy-glib released today!
Tapioca Project

- Guys from Nokia Technology Institute (INdT) in Brazil
- Similar goals to our project
- Now adopted our specification
- Producing client libraries for Qt, Glib & C#
Landell

• C# client based on Tapioca# and Gtk#
Mission Control

- Just released by Nokia, based on Glib and Gconf
- Stores your account settings
- Manage the presence of all your connections
- Handles incoming events
- See http://mission-control.sourceforge.net/
The Knights Who Say NIH

• I already tried making a shared library of the protocol code
• Loose coupling is essential for success
• Making a the client into a library doesn't fix everything
• We chose not to take eg Gaim's protocol code, but it would be cool...
Not Just for IM Clients

• Telepathy's API is for abstracting and sharing the protocol code itself
• Because D-Bus objects can be extended with interfaces, it's not a lowest common denominator abstraction
• The actual policy and behaviour of the client on top is not specified by the API
• So it's useful for other stuff too...
GNOME Integration: Gossip

Telepathy support now in 0.23 release!
GNOME Integration: Gossip

Voice & video support on its way...
GNOME Integration: Galago

- Christian Hammond's presence framework
- Hook in to Telepathy backends
- See your contacts throughout the desktop...
KDE Integration: Decibel

• “Houston” policy daemon & libraries
• Manage connection managers
• Provide account management
• APIs for common tasks
• Handle incoming events
• Based on Tapioca Qt libraries
KDE Integration: Kopete

- Can use Telepathy backends
- Provide protocols as another Telepathy backend
- Working on Qt libraries with Tapioca
Nokia 770

- IM and VOIP on the Nokia 770 based on Telepathy
- Uses Gabble, Stream Engine and Galago with Nokia UI for Chat, Call & Contacts
Nokia N800

- Jingle video calls on XMPP using Telepathy, a world first!
- INdT's Canola phone plugin provides alternative UI for placing video calls
One Laptop Per Child

• Using Telepathy for presence & messaging

• Video calls too!

Telepathy
One Laptop Per Child
Just Released: Tubes

- Telepathy channel for exchanging data between contact's applications
- Do the NAT punching and provide TCP, UDP or D-Bus link-up
- Hook up e.g. Inkscape, Abiword or Jokosher without any fragile networking code
- Implemented on XMPP, but more coming soon
import dbus
import time

# connect to the bus
bus = dbus.SessionBus()

# get a connection manager object
gabble = bus.get_object('org.freedesktop.Telepathy.ConnectionManager.gabble',
                        '/org/freedesktop/Telepathy/ConnectionManager/gabble')

# request a connection from it
(bus_name, object_path) = gabble.RequestConnection('jabber',
                                                     {'account':'test1@thubuntu', 'password':'badger'},
                                                     dbus_interface='org.freedesktop.Telepathy.ConnectionManager')

# get the connection object
conn = bus.get_object(bus_name, object_path)
Trivial Example

# tell it to connect and wait a bit
conn.Connect(
    dbus_interface='org.freedesktop.Telepathy.Connection')
time.sleep(3)

# request a handle for our contact
handles = conn.RequestHandles(dbus.UInt32(1),
    ['test2@thubuntu'],
    dbus_interface='org.freedesktop.Telepathy.Connection')

# request a text channel with that handle
object_path = conn.RequestChannel('org.freedesktop.Telepathy.Channel.Type.Text',
    dbus.UInt32(1), handles[0], False,
    dbus_interface='org.freedesktop.Telepathy.Connection')

# get a channel object
channel = bus.get_object(bus_name, object_path)
Trivial Example

# send a message
channel.Send(0, 'Hello, Santa Clara.',
    dbus_interface=
        'org.freedesktop.Telepathy.Channel.Type.Text')

# disconnect
conn.Disconnect(
    dbus_interface='org.freedesktop.Telepathy.Connection')
Exciting Demo

• What could possibly go wrong...
Big up the Telepathy massive!

- Wiki: http://telepathy.freedesktop.org/
- IRC channel: irc.freenode.net #telepathy
- Mailing list: telepathy@lists.freedesktop.org
- Development supported by Collabora Ltd: http://www.collabora.co.uk/