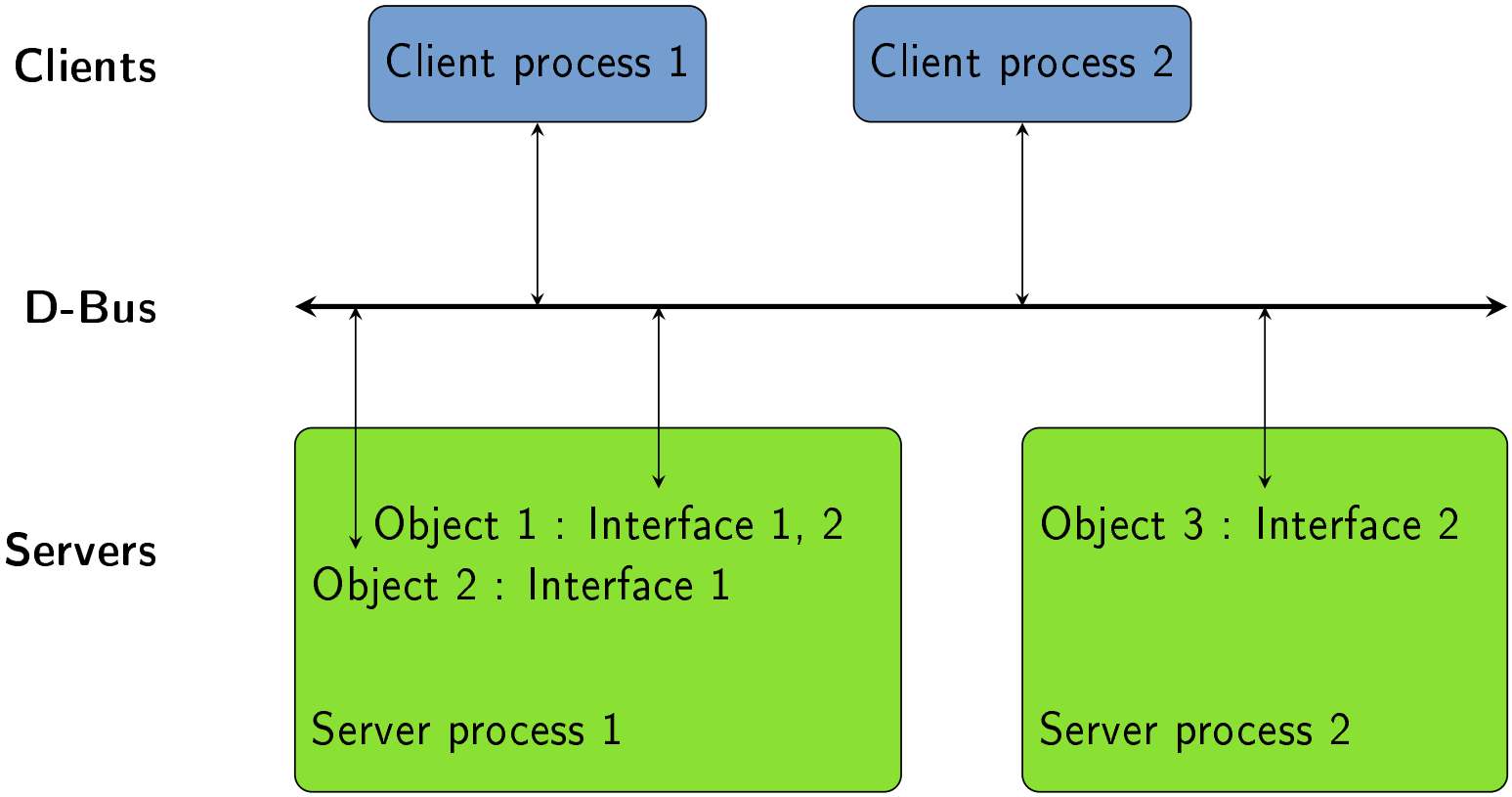
Bendy Bus

Fuzzily impersonating D-Bus services

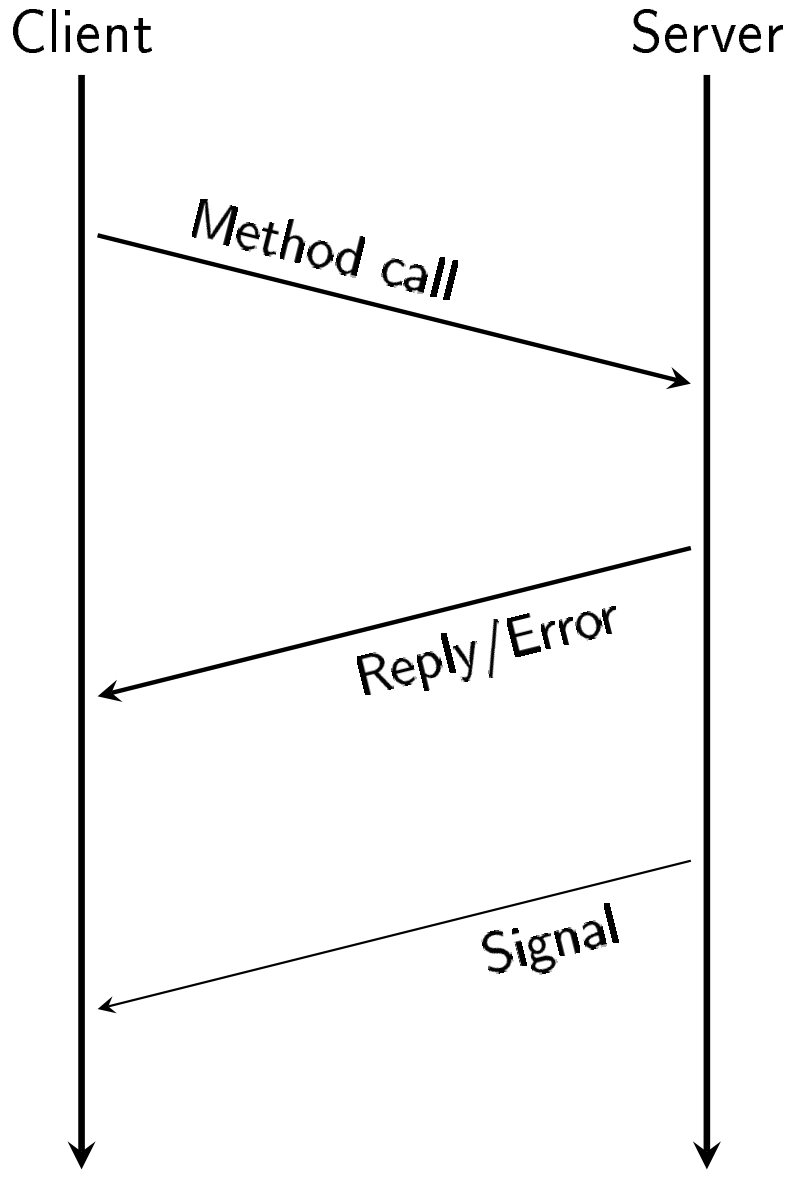
Philip Withnall

July 26, 2012

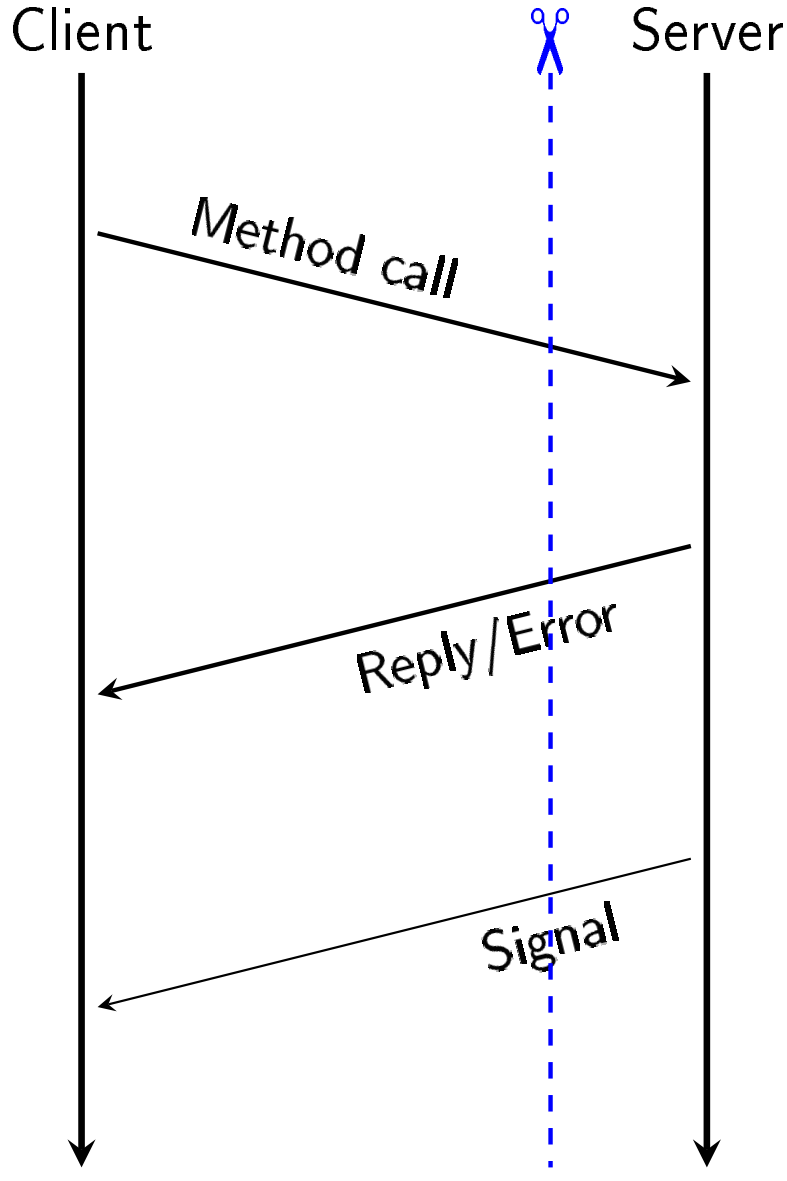
D-Bus recap


Clients
Client process 1
Client process 2
D-Bus
Servers
Object 1 : Interface 1, 2
Object 2 : Interface 1
Server process 1
Object 3 : Interface 2
Server process 2

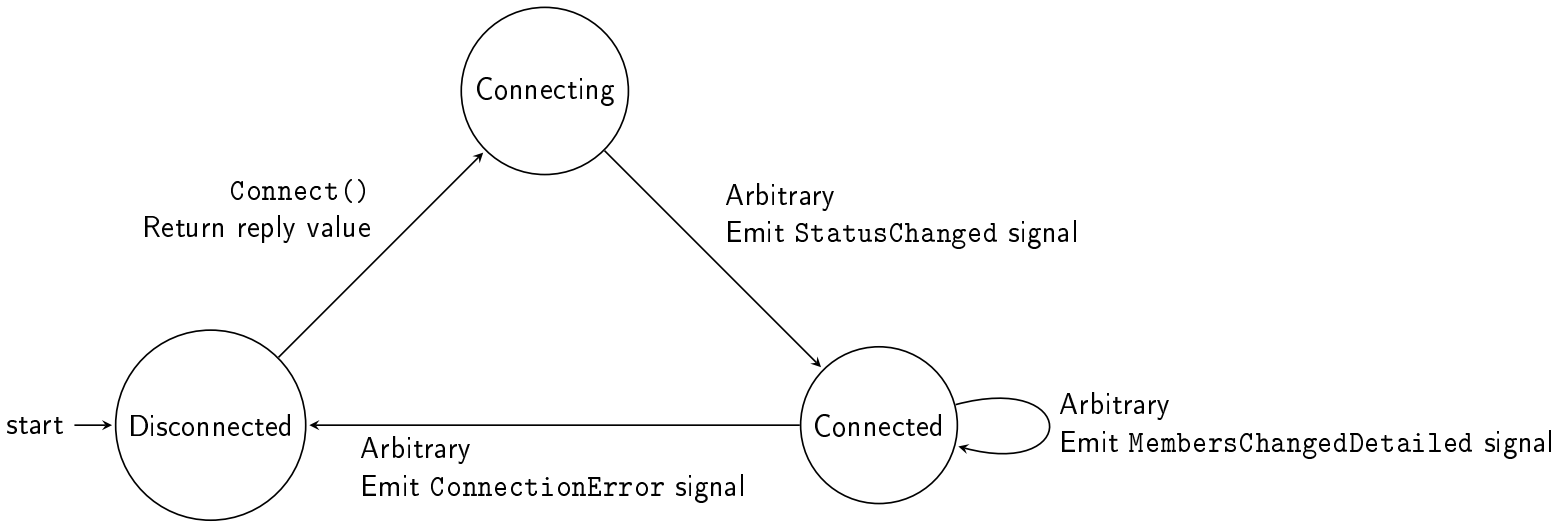
D-Bus conversations



Introducing Bendy Bus



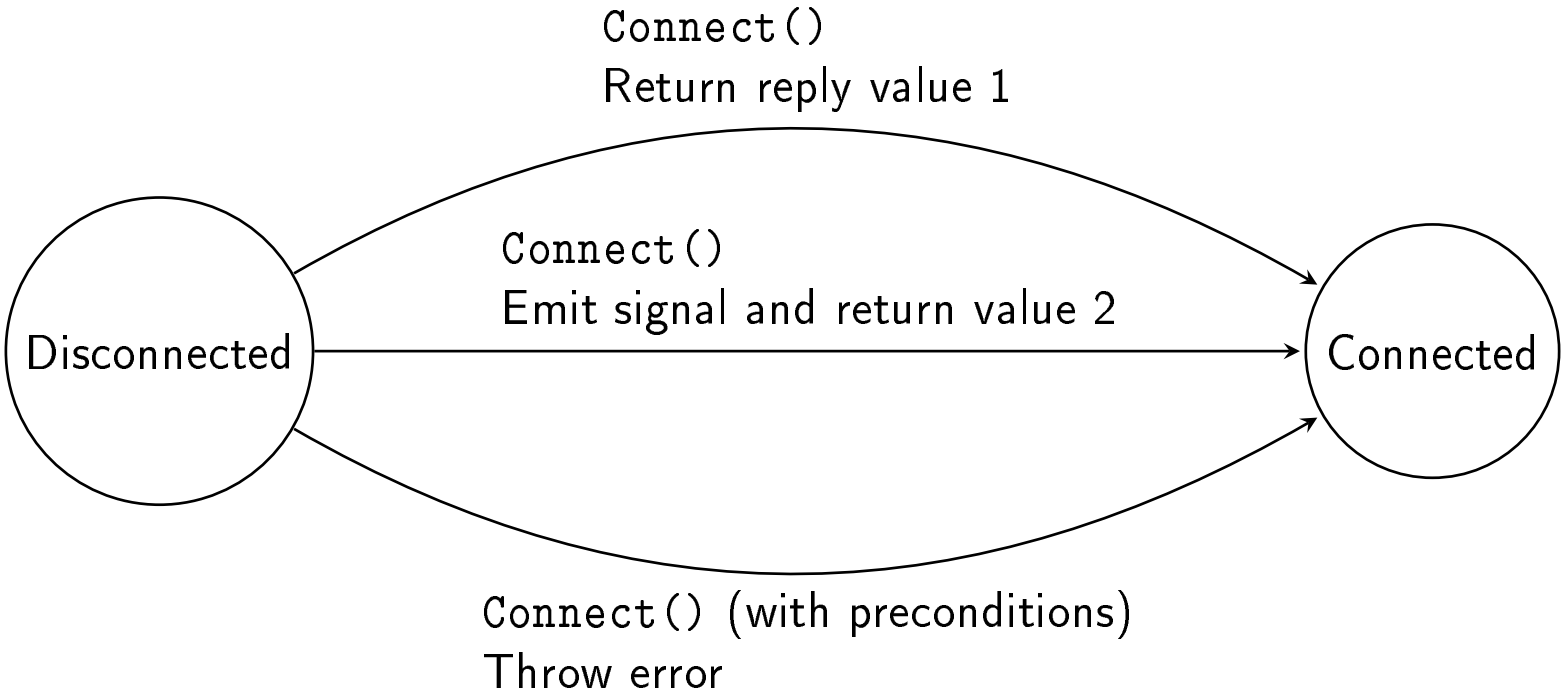
Server state machine



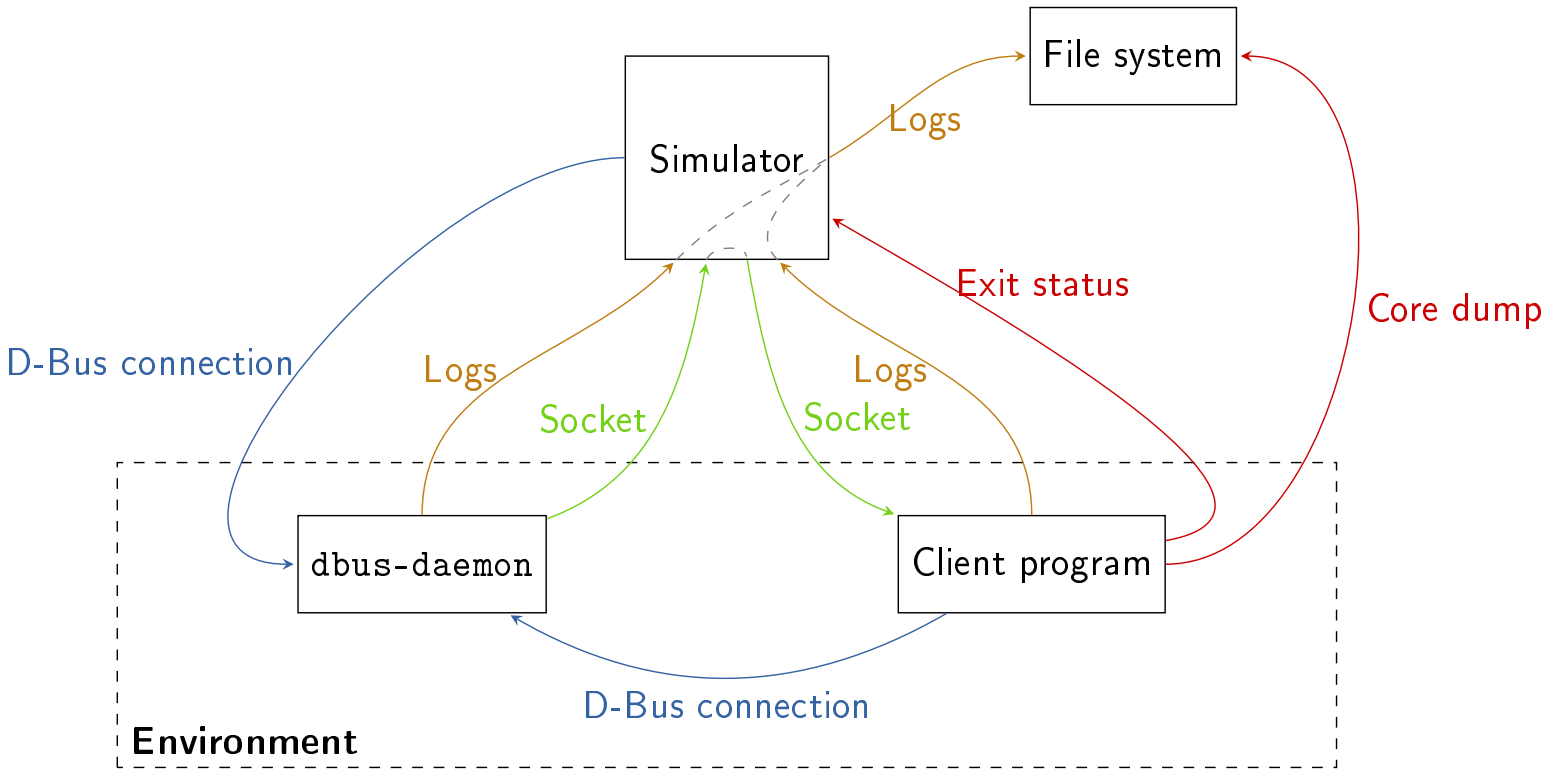
Simulation description

```
1  object at /org/gnome/Example, org.gnome.
       Example implements org.gnome.Example {
2      states { Main, Alternate }
3
4      transition from Main to Alternate,
           Alternate to Main on random {
5          emit SomeSignal ();
6      }
7
8      transition inside Main on method
           GetSomething {
9          reply (["Something"?]?);
10     }
11 }
```

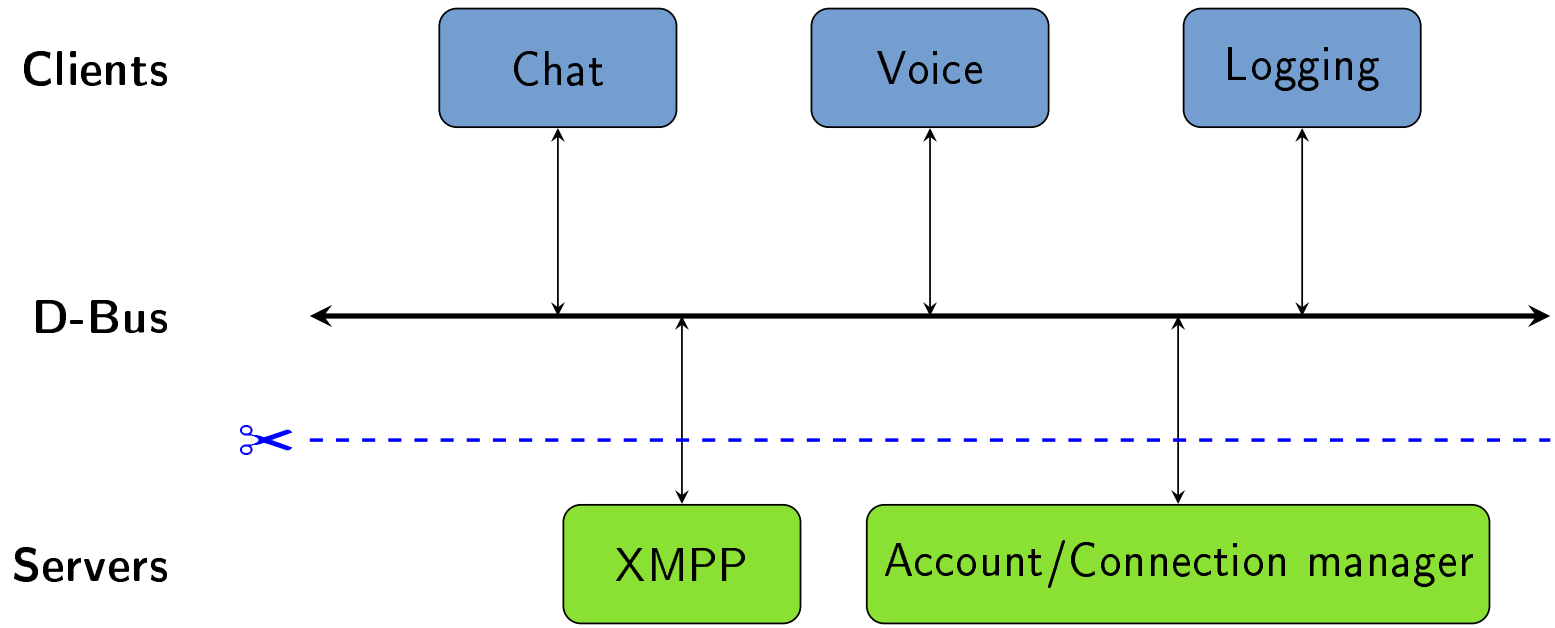
Nondeterministic transitions



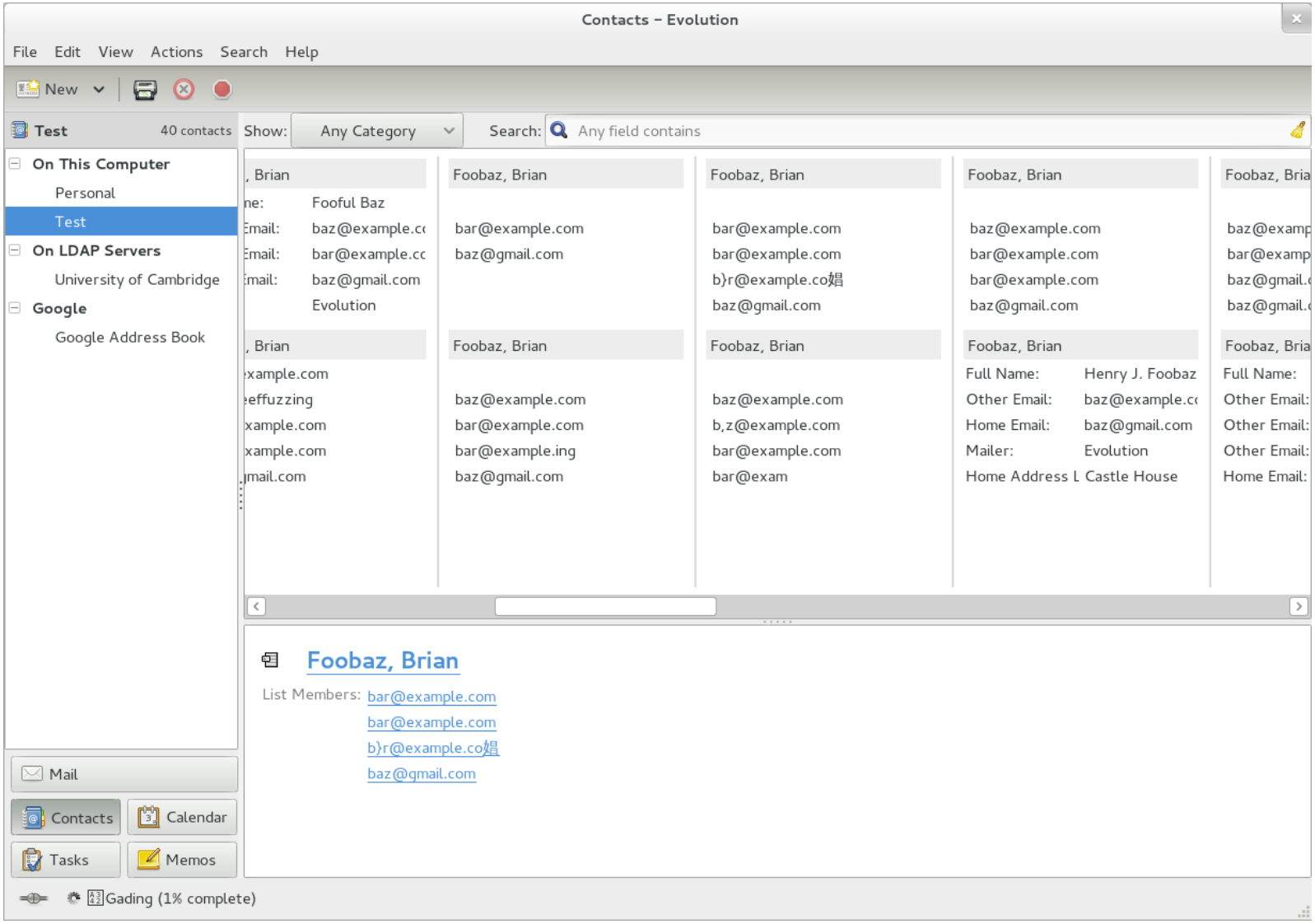
Simulator interactions



Telepathy simulation



Evolution Data Server simulation



Fuzzing a Hamster

- A walkthrough application of Bendy Bus: fuzzing Time Tracker.
- Simple client–server split: server manages the database, client does the UI.

No hamsters were harmed in the making of this walkthrough.

Fuzzing a Hamster

1. Get a D-Bus XML description of the server's interface.

```
<!DOCTYPE node PUBLIC "-//freedesktop//DTD D-BUS Object Introspection 1.0//EN"
"http://www.freedesktop.org/standards/dbus/1.0/introspect.dtd">
<node name="/org/gnome/Hamster">
  <interface name="org.gnome.Hamster">
    <method name="GetTags">
      <arg direction="in"  type="b" name="only_autocomplete" />
      <arg direction="out" type="a(isb)" />
    </method>
    <method name="Quit"></method>
```

Fuzzing a Hamster

2. Decide which objects to implement and stub out the simulation description.

```
1 object at /org/gnome/Hamster, org.gnome.
    Hamster implements org.gnome.Hamster {
2     /* Nothing here yet! */
3 }
```

Fuzzing a Hamster

2. Decide which objects to implement and stub out the simulation description.
3. Work out a set of states for the object.

```
1 object at /org/gnome/Hamster, org.gnome.
    Hamster implements org.gnome.Hamster {
2     states { Main }
3 }
```

Fuzzing a Hamster

4. Get a trace of the startup of the client using dbus-monitor:

```
$ dbus-monitor --session "interface='org.gnome.Hamster'"
method call sender=:1.197 -> dest=:1.84 serial=11
 path=/org/gnome/Hamster; interface=org.gnome.Hamster;
 member= GetActivities
   string ""
method call sender=:1.197 -> dest=:1.84 serial=12
 path=/org/gnome/Hamster; interface=org.gnome.Hamster;
 member= GetCategories
method call sender=:1.197 -> dest=:1.84 serial=13
 path=/org/gnome/Hamster; interface=org.gnome.Hamster;
 member= GetTags
   boolean true
method call sender=:1.197 -> dest=:1.84 serial=15
 path=/org/gnome/Hamster; interface=org.gnome.Hamster;
 member= GetTodaysFacts
```

Fuzzing a Hamster

5. Implement those methods as transitions!

Fuzzing a Hamster

5. Implement those methods as transitions!

```
1 transition inside Main on method
    GetActivities {
2     /* Parameter: search */
3     reply ([
4         ("Something"?, "Activity"?),
5     ]?);
6 }
```

Fuzzing a Hamster

5. Implement those methods as transitions!

```
1  transition inside Main on method
       GetCategories {
2       reply ([
3           (0?, "Category"?),
4       ]?);
5  }
```

Fuzzing a Hamster

5. Implement those methods as transitions!

```
1 transition inside Main on method GetTags {
2     /* Parameter: only_autocomplete */
3     reply ([
4         (0?, "Tag"?, false?),
5     ]?);
6 }
```

Fuzzing a Hamster

5. Implement those methods as transitions!

```
1  transition inside Main on method
       GetTodaysFacts {
2       reply ([
3           (0?, 0?, 0?, "Fact"?, "Factoid?", 0?,
                "Factless"?, ["Array string"?]?,
              0?, 0?),
4       ]?);
5  }
```

Fuzzing a Hamster

5. Implement those methods as transitions!
6. If any data needs storing, add a `data{}` block.

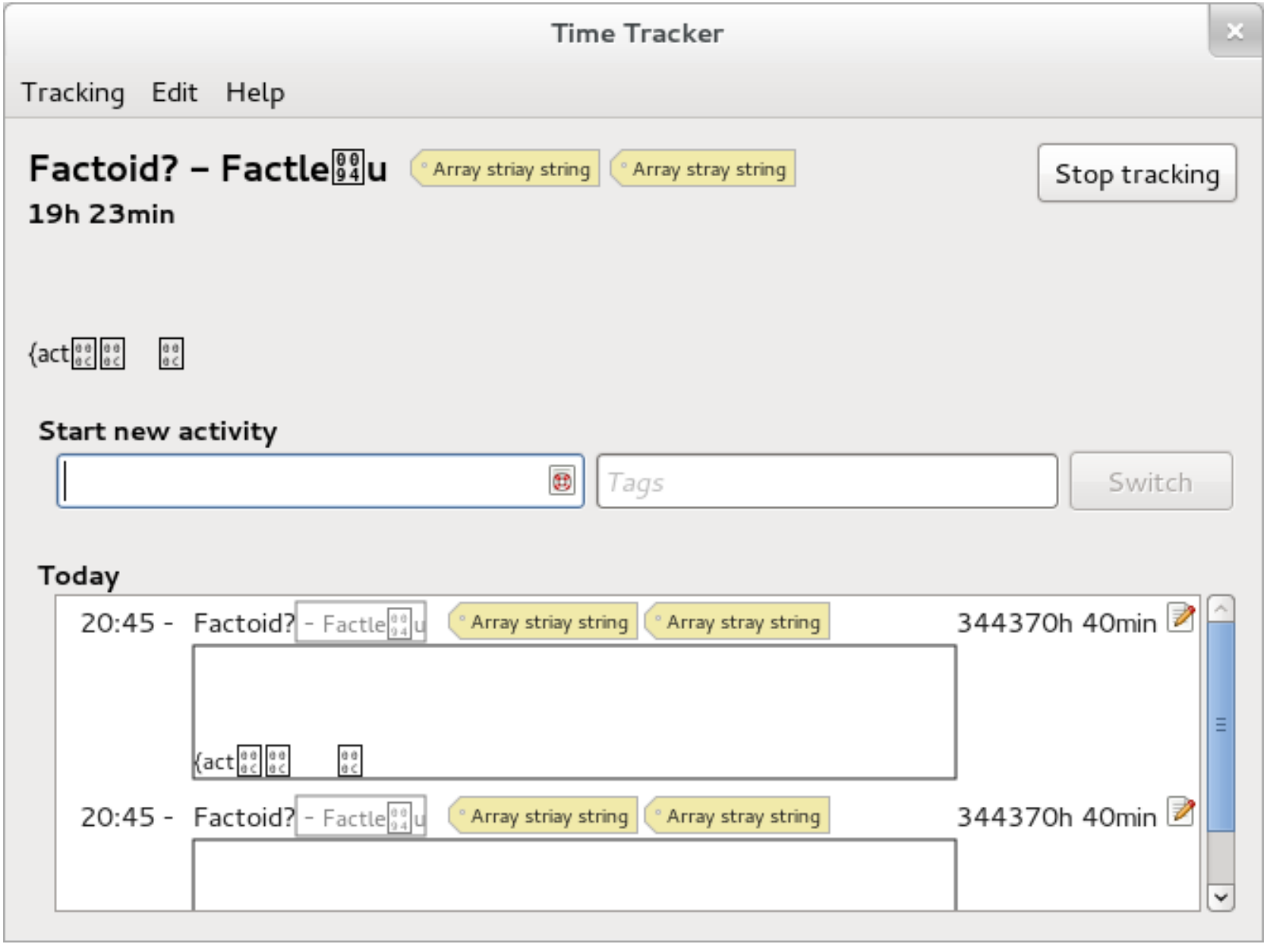
Fuzzing a Hamster

7. Try running it:
   `bendy-bus hamster-server.machine hamster-server.xml -- /usr/bin/hamster-time-tracker`.
8. Add the GConf D-Bus auto-start file to Bendy Bus' working directory and try again:
   `bendy-bus --dbus-daemon-config-file= /tmp/bendy-bus_UFBXGW/dbus-daemon/config.xml hamster-server.machine hamster-server.xml -- /usr/bin/hamster-time-tracker`.

Fuzzing a Hamster

9. Watch things get fuzzed and wait for crashes or misbehaviour. Add the `-t 2 -i` options to have Bendy Bus kill Hamster after 2 s of inactivity then loop.
10. Add some signal emissions to randomise things more.
11. If the client code is supposed to handle application-level errors from the server, add some alternative (random) transitions which throw errors. e.g. To `GetTodaysFacts()` in Hamster.

Fuzzing a Hamster



Future work

- Re-use of code between simulations.

Future work

- Re-use of code between simulations.
- Preferred transition paths.

Future work

- Re-use of code between simulations.
- Preferred transition paths.
- Model checking.

Future work

- Re-use of code between simulations.
- Preferred transition paths.
- Model checking.
- UI fuzzing.

Future work

- Re-use of code between simulations.
- Preferred transition paths.
- Model checking.
- UI fuzzing.
- D-Bus `ObjectManager` support.

Miscellany

Bendy Bus `https://gitorious.org/bendy-bus`

D-Bus `http://dbus.freedesktop.org/`

Creative Commons Attribution-NonCommercial-ShareAlike 3.0 License