Concurrence

Luc Courtrai

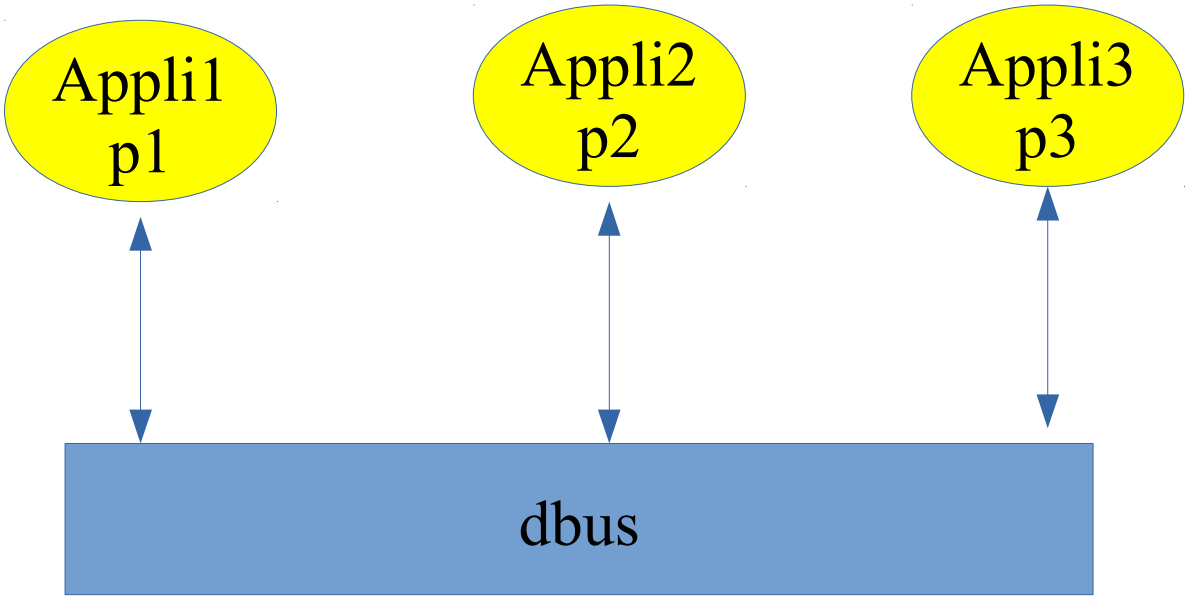
Introduction à **dbus**

Système de communication Inter Processus

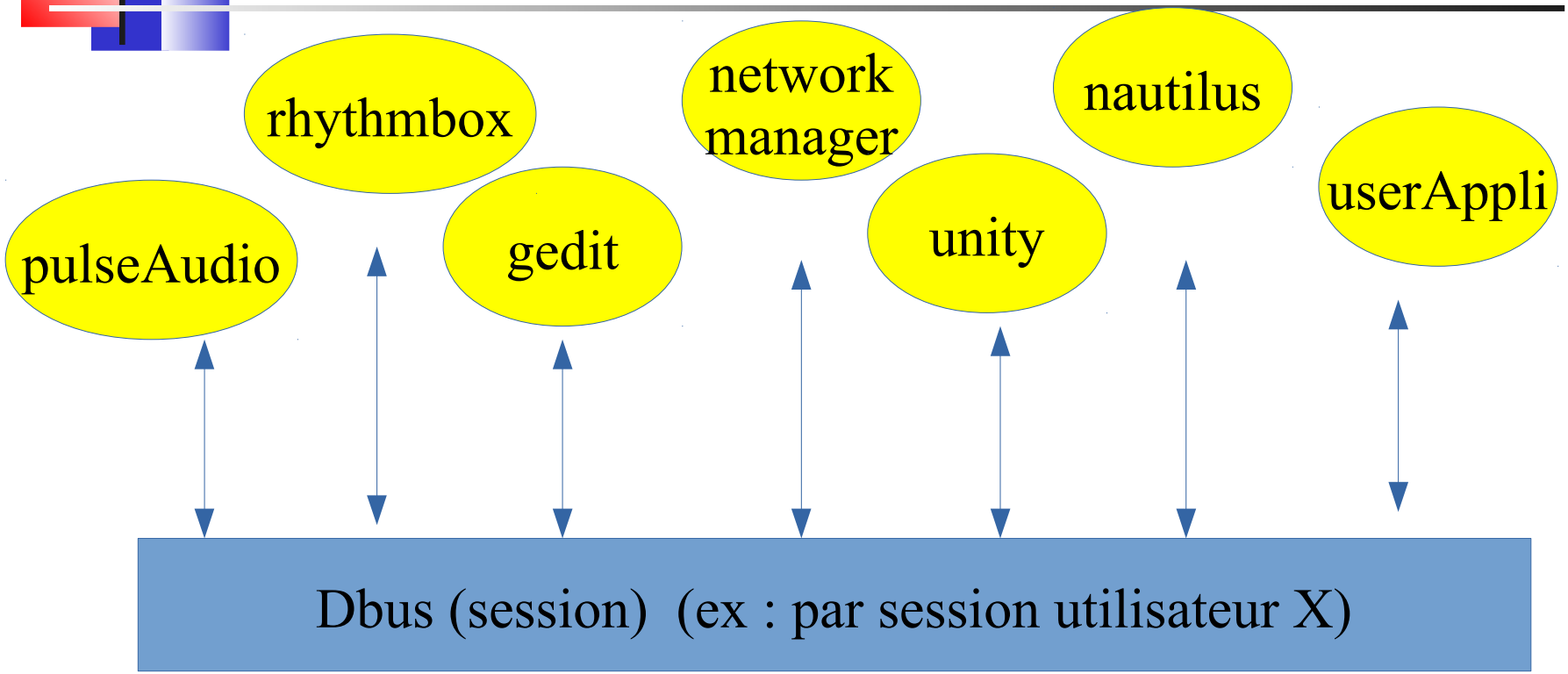
IPC 2002

Communications entre applications via le bus
(ie CORBA)

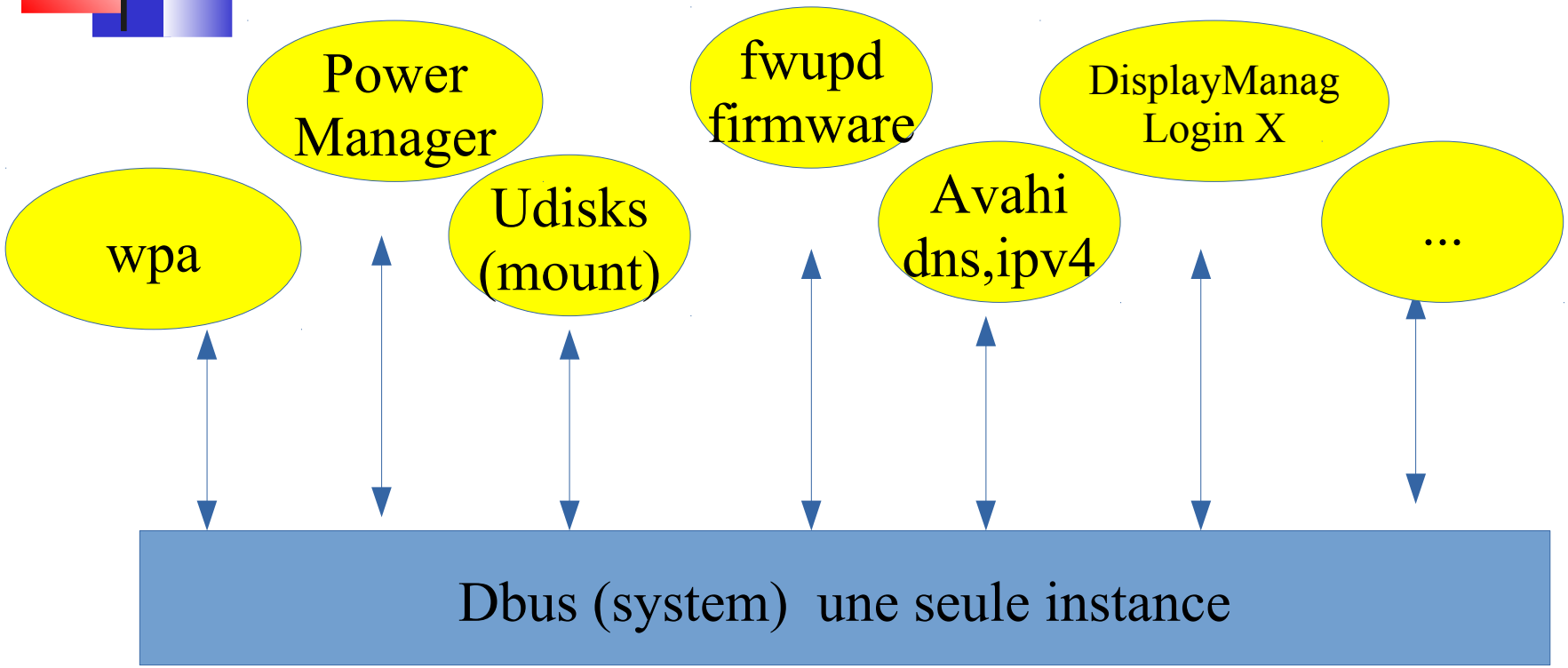
Concurrence



Concurrence



Concurrence



Concurrence

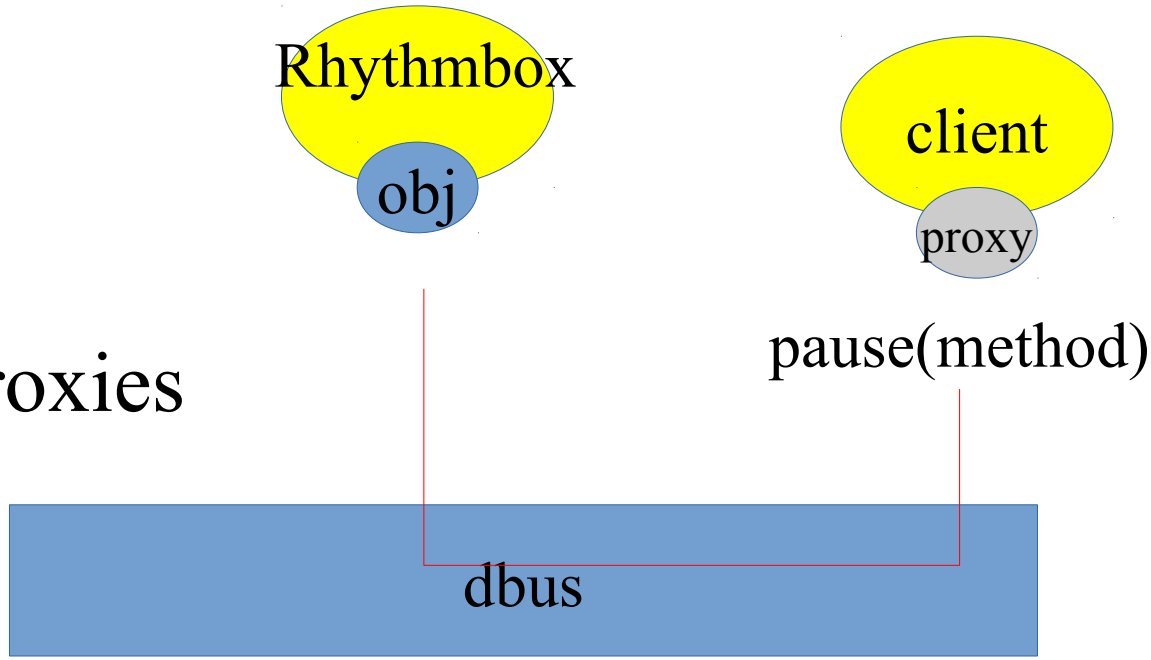
DBUS

- appel de mèthodes (sur les objets services)
- multilanguage
- activation des services
- message d’erreur (exception)
- signaux

Concurrence

DBus

- Services et proxies



Activation possible du service

dbus-launch : permet de démarrer un bus session depuis un script shell

dbus-cleanup-sockets : fait le ménage dans les sockets ouverts par des bus et qui ne sont plus utilisés.

dbus-send : permet d'envoyer un message sur le bus depuis un script shell

dbus-daemon : Le plus important, le daemon D-Bus

dbus-monitor : Permet d'observer ce qui transite sur un ou plusieurs bus

dbus-uuidgen : génère des uuids pour les sessions de D-Bus

Concurrence

Utilisation

Programmation : bindings : C, C++, Python, Java, Perl, Php, Ruby,...)

Ligne de commande : *dbus-send, dbus-monitor*

Ligne de commande

```
> dbus-send --type=method_call  --print-reply  \
--dest=org.mpris.MediaPlayer2.rhythmbox     \
/org/mpris/MediaPlayer2 \
org.mpris.MediaPlayer2.Player.Pause
```

Client Python (proxy et appel de method)

```
import dbus
session_bus = dbus.SessionBus()
player = session_bus.get_object(
        'org.mpris.MediaPlayer2.rhythmbox',
         '/org/mpris/MediaPlayer2')
iface = dbus.Interface(
         player,
        dbus_interface='org.mpris.MediaPlayer2.Player')
iface.Pause()
```

Concurrence

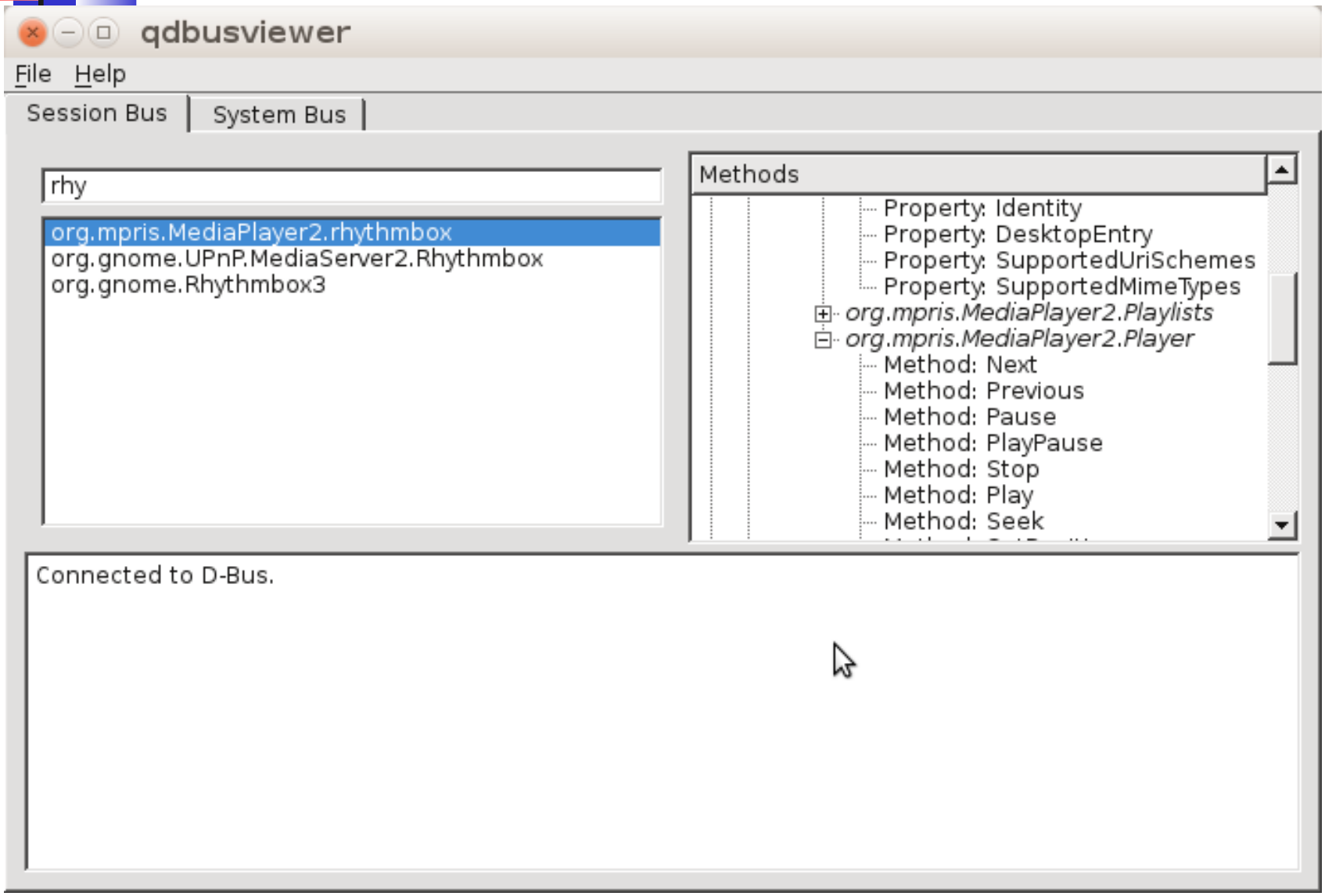
Introspection

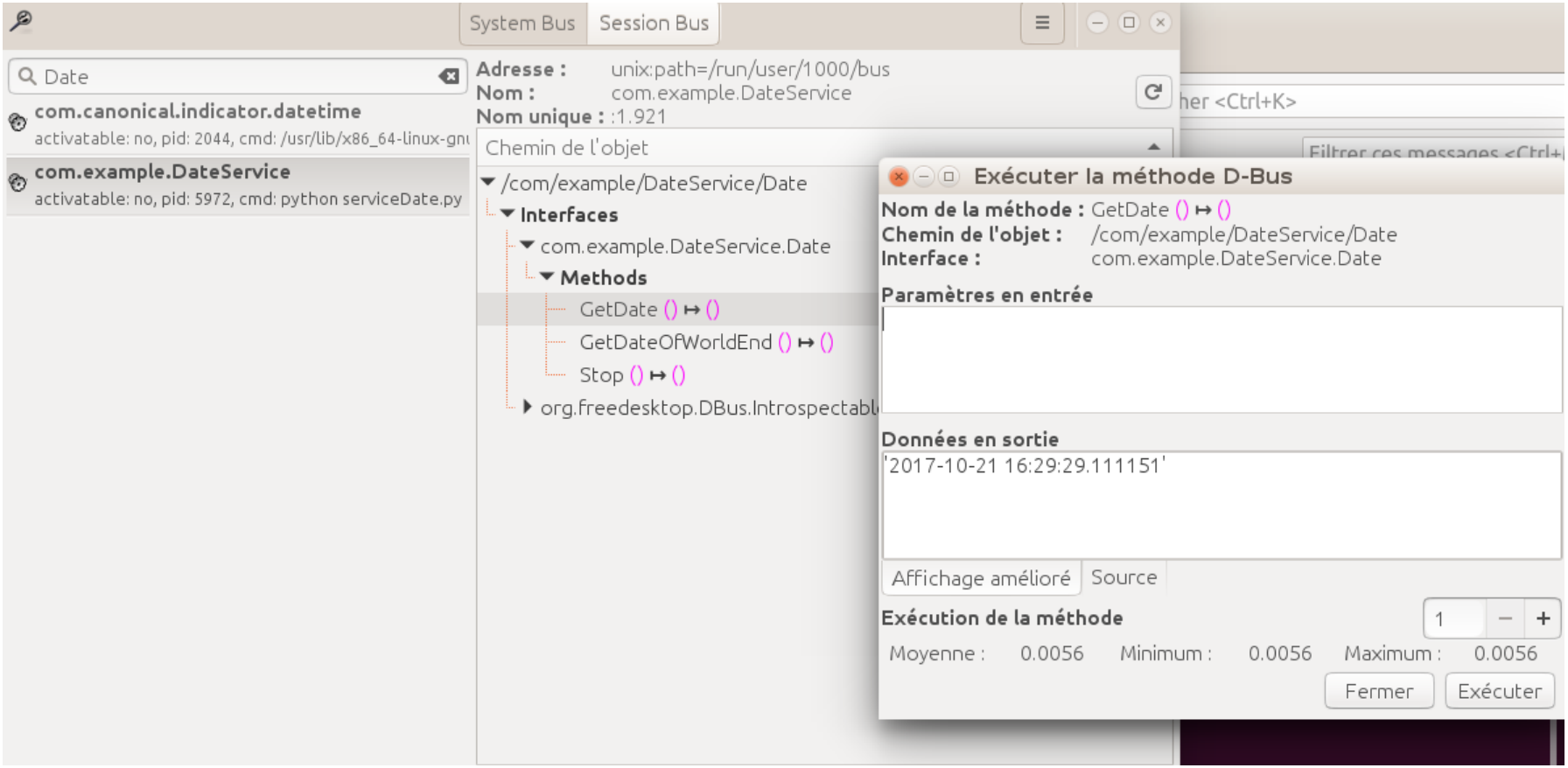
```
dbus-send --type=method_call  --print-reply
   --dest=org.mpris.MediaPlayer2.rhythmbox
   /org/mpris/MediaPlayer2
   org.freedesktop.DBus.Introspectable.Introspect
```

Concurrence

Monitoring

```
dbus-monitor --session
   "path=/org/mpris/MediaPlayer2,member=Properties
   Changed" --monitor
```


qdbusviewer
File
Help
Session Bus
System Bus
rhy
org.mpris.MediaPlayer2.rhythmbox
org.gnome.UPnP.MediaServer2.Rhythmbox
org.gnome.Rhythmbox3
Methods
Property: Identity
Property: DesktopEntry
Property: SupportedUriSchemes
Property: SupportedMimeTypes
org.mpris.MediaPlayer2.Playlists
org.mpris.MediaPlayer2.Player
Method: Next
Method: Previous
Method: Pause
Method: PlayPause
Method: Stop
Method: Play
Method: Seek
Connected to D-Bus.


System Bus
Session Bus
Date
com.canonical.indicator.datetime
activatable: no, pid: 2044, cmd: /usr/lib/x86_64-linux-gnu
com.example.DateService
activatable: no, pid: 5972, cmd: python serviceDate.py
Adresse : unix:path=/run/user/1000/bus
Nom : com.example.DateService
Nom unique : :1.921
Chemin de l'objet
/com/example/DateService/Date
Interfaces
com.example.DateService.Date
Methods
GetDate () ↦ ()
GetDateOfWorldEnd () ↦ ()
Stop () ↦ ()
org.freedesktop.DBus.Introspectabl
her <Ctrl+K>
Exécuter la méthode D-Bus
Nom de la méthode : GetDate () ↦ ()
Chemin de l'objet : /com/example/DateService/Date
Interface : com.example.DateService.Date
Paramètres en entrée
Données en sortie
'2017-10-21 16:29:29.111151'
Affichage amélioré
Source
Exécution de la méthode
1
Moyenne : 0.0056
Minimum : 0.0056
Maximum : 0.0056
Fermer
Exécuter

d-feet

Concurrence

Ecriture d’un service helloWorld Python

```
import gobject
import dbus
import dbus.service

from dbus.mainloop.glib import DBusGMainLoop
DBusGMainLoop(set_as_default=True)


OPATH = "/com/example/HelloWorld"
IFACE = "com.example.HelloWorld"
BUS_NAME = "com.example.HelloWorld"
```

Concurrence

```
class Example(dbus.service.Object):
  def __init__(self):
      bus = dbus.SessionBus()
  •   bus.request_name(BUS_NAME)
      bus_name= dbus.service.BusName(BUS_NAME, bus=bus)
      dbus.service.Object.__init__(self, bus_name, OPATH)

    @dbus.service.method(dbus_interface=IFACE + ".SayHello",
                         in_signature="", out_signature="")
  def SayHello(self):
      print "hello, world"
if __name__ == "__main__":
  a = Example()
  loop = gobject.MainLoop()
  loop.run()
```

Concurrence

Ecriture d’un service helloWorld Python

. > python helloService.py

client

>dbus-send --type=method_call --print-reply --dest=com.example.HelloWorld /com/example/HelloWorld com.example.HelloWorld.SayHello.SayHello

Concurrence

Ecriture d’un service Date (avec signature interface)

```
class Date(dbus.service.Object):

    loop = None
    def __init__(self, lo):
        print "Lancement de DateService Date"
        bus = dbus.SessionBus()
        bus.request_name("com.example.DateService")
        bus_name = dbus.service.BusName("com.example.DateService",
                                        bus=bus)
        dbus.service.Object.__init__(self, bus_name,
                                     "/com/example/DateService/Date")
```

Concurrence

Ecriture d’un service Date (avec signature interface)

```
@dbus.service.method(dbus_interface="com.example.DateService.Date
                     in_signature="", out_signature="s")
    def GetDate(self):
        print "Appel GetDate"
        return str(datetime.datetime.now())
```

.

```
    @dbus.service.method("com.example.DateService.Date")
    def Stop(self):
        print "Stop DateService"
        self.loop.quit()
        return 'Quit '
```

Python type	converted to D-Bus type	notes
D-Bus proxy object	ObjectPath (signature 'o')	(+)
dbus.Interface	ObjectPath (signature 'o')	(+)
dbus.service.Object	ObjectPath (signature 'o')	(+)
`dbus.Boolean`	Boolean (signature 'b')	a subclass of `int`
`dbus.Byte`	byte (signature 'y')	a subclass of `int`
`dbus.Int16`	16-bit signed integer ('n')	a subclass of `int`
`dbus.Int32`	32-bit signed integer ('i')	a subclass of `int`
`dbus.Int64`	64-bit signed integer ('x')	(*)
`dbus.UInt16`	16-bit unsigned integer ('q')	a subclass of `int`
`dbus.UInt32`	32-bit unsigned integer ('u')	*(*)_*
`dbus.UInt64`	64-bit unsigned integer ('t')	*(*)_*
`dbus.Double`	double-precision float ('d')	a subclass of `float`
`dbus.ObjectPath`	object path ('o')	a subclass of `str`
`dbus.Signature`	signature ('g')	a subclass of `str`
`dbus.String`	string ('s')	a subclass of `unicode`
`dbus.UTF8String`	string ('s')	a subclass of `str`
`bool`	Boolean ('b')	
`int` or subclass	32-bit signed integer ('i')	
`long` or subclass	64-bit signed integer ('x')	
`float` or subclass	double-precision float ('d')	
`str` or subclass	string ('s')	must be valid UTF-8

Array "ax" structure (xxs) dicothonaire 'a{xy}' x key

Concurrence

Client en ligne de commande

```
> dbus-send --type=method_call
    --print-reply    # réponse
    --dest=com.example.DateService       # service
    "/com/example/DateService/Date"   # objet
     com.example.DateService.Date.GetDate  # method

method return time=1508735716.142016 sender=:1.943 ->
destination=:1.992 serial=4 reply_serial=2
   string "2017-10-23 07:15:16.141586"

> dbus-send --type=method_call   --dest=com.example.DateService
"/com/example/DateService/Date"  com.example.DateService.Date.Stop
```

Concurrence

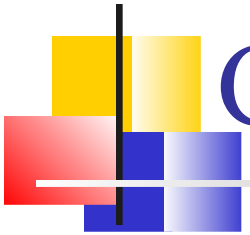
Client Python

```
import dbus
session_bus = dbus.SessionBus()
dateService = session_bus.get_object(
        'com.example.DateService',
        '/com/example/DateService/Date')
iface = dbus.Interface(
        dateService,
        dbus_interface='com.example.DateService.Date')

print iface.GetDate()

> python getDate.py
  2017-10-23 07:19:55.003694
```

Concurrence

Client C (morceaux de code :manque la gestion des erreurs)

```
#include <dbus/dbus.h>
int main(){
// connect to the system bus and check for errors
DBusConnection*conn = dbus_bus_get(
                DBUS_BUS_SESSION, &err);
  // request our name on the bus
int rc = dbus_bus_request_name(conn, "com.example.DateService",
                  DBUS_NAME_FLAG_REPLACE_EXISTING , &err);
msg = dbus_message_new_method_call("com.example.DateService", // target for the method call
                                   "/com/example/DateService/Date", // object to call on
                                   "com.example.DateService.Date", // interface to call on
                                   "GetDate"); // method name
// send message and get a handle for a reply
 if (!dbus_connection_send_with_reply (conn, msg, &pending, -1))
 ….
// block until we recieve a reply
 dbus_pending_call_block(pending);
// read the parameters
 if (dbus_message_iter_init(msg, &args))  printf("Date Reply:%s\n",str);
```



Concurrence

Ecriture d’un service Date (Exception)

Exception ..

```
@dbus.service.method("com.example.DateService.Date")
    def GetDateOfWorldEnd(self):
        print "Appel GetDateOfWorldEnd"
        raise Exception('Unknow Date Error !!!!!')
```

Concurrence

Les signaux (coté réception) en python

```
# fonction
def quit_handler():
        """Signal handler for quitting the receiver."""
        print 'Quitting....'
        loop.quit()

bus.add_signal_receiver(
                quit_handler,
                dbus_interface='com.example.DateService.Date',
                signal_name='quit_signal'
                )
```

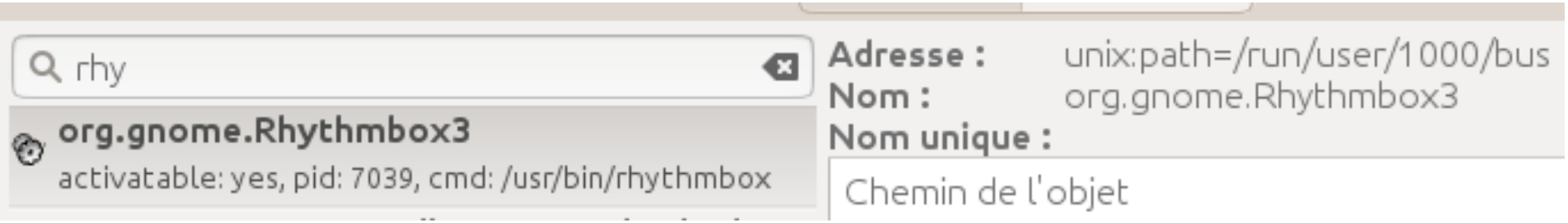
Concurrence

```
dbus-send
   --type=signal
  --dest=com.example.DateService
    "/com/example/DateService/Date"
    com.example.DateService.Date.quit_signal
```

Concurrence

Activation

/usr/share/dbus-1/services/org.gnome.Rhythmbox3.service
[D-BUS Service]
Name=org.gnome.Rhythmbox3
Exec=/usr/bin/rhythmbox



/org/gnome/Rhythmbox3

- **Interfaces**
  - org.freedesktop.Application
  - org.freedesktop.DBus.Introspectable
  - org.freedesktop.DBus.Peer
  - org.freedesktop.DBus.Properties
  - org.gtk.Actions
    - **Methods**
      - Activate (String action_name, Array of [Variant] parameter, Dict of {String
      - Describe (String action_name) ↦ (Struct of (Boolean, Signature, Array of
      - DescribeAll () ↦ (Dict of {String, Struct of (Boolean, Signature, Array of [V
      - List () ↦ (Array of [String] list)
      - SetState (String action_name, Variant value, Dict of {String, Variant} plat
    - **Signals**
      - Changed (Array of [String], Dict of {String, Boolean}, Dict of {String, Variar
  - org.gtk.Application