IPC

- A set of methods for exchange information among multiple processes.
IPC Examples

• Example
  o signal
  o synchronization (semaphore, mutex, cond var)
  o pipe (posix pipe, FIFO)
  o message passing (posix message queue, MPI)
  o shared memory (posix shared memory)
  o remote procedure control (RPC)
  o socket
Why IPC matters?

• Performance: speedup by parallelization

• Productivity: modularity

• Design easiness: Decoupling

• Security: privilege separation
OO-IPC

- IPC in an object oriented way.

- Example
  - D-BUS
  - CORBA
  - XML-RPC, SOAP
  - DCE (Distributed Computing Environment)
  - COM, DCOM
  - ICE (Internet Communication Engine)
  - MBUS, ONC/RPC
D-Bus

- Secure message bus in desktop Linux environment
- Standard interface for event delivering and handling
- Replacing Bonobo in Gnome and DCOP in KDE
D-Bus Example Usage

• Notification

• Screen Saver

• Network Manager

• Common Unix Printing System (PrintSpooler)

• Power Management
D-Bus Overview
D-Bus Architecture
D-Bus Internal

- Type system, marshaling and de-marshaling
- Message protocol
- Authentication protocol
- Addressing and transport
- Naming and routing
- Standard interface
- Introspection
- Message bus management
- Well-known bus instances
void signal_sender() {

dbus_bus_get();
sig = dbus_message_new_signal();
my_append_signal_arg(sig);
dbus_connection_send(sig);
dbus_connection_flush();
}

void method_call() {

dbus_bus_get();

mth = dbus_message_new_method_call();
my_append_signal_arg(sig);
dbus_connection_send_with_reply(mth);
dbus_connection_flush();

dbus_pending_call_block();
reply = dbus_pending_call_steal_reply();
my_process_reply();
}

void server() {

dbus_bus_get();
dbus_bus_request_name();
while(1) {

dbus_connection_read_write_dispatch();
msg =
dbus_connection_pop_message();
if(dbus_message_is_method_call()) {
    arg =

my_get_dbus_message_arg();
    ret =

my_call_local_method();
    reply =

    dbus_message_new_method_return();

    my_append_return_arg(reply, ret);

dbus_connection_send(reply);
    dbus_connection_flush();
} else if(dbus_message_is_signal()) {
}
D-Bus Program in C++

```cpp
class MyServer
    : public org::cse438::MyServer_adaptor,
      public DBus::IntrospectableAdaptor,
      public DBus::ObjectAdaptor {
public:
    MyServer(DBus::Connection &connection);
    int32_t Method() { return 0; }
};

class MyClient
    : public org::cse438::MyClient_proxy,
      public DBus::IntrospectableProxy,
      public DBus::ObjectProxy {
public:
    MyClient(DBus::Connection &connection,
             const char *path, const char *name);
    void Signaled(const DBus::Variant &value);
};

class MyServer
    : public org::cse438::MyServer_adaptor,
      public DBus::IntrospectableAdaptor,
      public DBus::ObjectAdaptor {
public:
    MyServer(DBus::Connection &connection);
    int32_t Method() { return 0; }
};

#dbusxx-xml2cpp my.xml
    --proxy=MyClient.h
    --adaptor=MyServer.h
```
D-Bus Program in Python

```python
dbus.SessionBus()

bus = dbus.SessionBus()

helloservice = bus.get_object(
    'org.cse438.busname',
    '/org/cse438/objpath')

hello = helloservice.get_dbus_method(
    'hello', 'org.cse438.interface')

def hello(self):
    return "Hello,World!"

class MyServer(dbus.service.Object):
    def __init__(self):
        bus_name = dbus.service.BusName(
            'org.cse438.busname',
            bus=dbus.SessionBus())

        dbus.service.Object.__init__(self, bus_name, 
        '/org/cse438/objpath')

    @dbus.service.method(
        'org.cse438.interface')
    def hello(self):
        return "Hello,World!"

DBusGMainLoop(set_as_default=True)

myservice = MyServer()
gtk.main()```
Example: AESBA

(a) a typical ASEBA network in a robot

(b) a microcontroller in an ASEBA network
Example: HEAD

Diagram of a robot's control system with various components and layers, including Task Selector, Data Manager, Signal Listener, Signal Emitter, Task Info Updater, Data Manager, Signal Listener, Signal Emitter, Base Application, Signal Listener/Emitter, and D-Bus Session Bus Daemon.
D-Bus evaluation

Pros:
- Clear abstraction, easy to use
- Object-oriented
- Secure
- Multi-language binding

Cons:
- Overhead
- No guaranteed delay, priority and flow control
- Complicated
- Designed for desktop Linux
Improvement: AF_DBUS

• Bypass daemon

• Modified Unix domain socket

• Message routing in kernel
Our alternative: eloombus
• CORBA: Common Object Request Broker Architecture

• An industry standard developed by OMG to help in distributed programming

• It enables separate pieces of software written in different languages and running on different computers to work with each other like a single application or set of services.
CORBA

- Clients invoke methods of objects running on servers
- Clients don’t know what language server is written, how the method is implemented, or even where the server is physically located
CORBA IDL

• CORBA Interface Definition Language (IDL) defines the interface to a CORBA object

• Each specific programming language that support CORBA has its own mapping to IDL.
  o Ada, C, C++, C++11, Lisp, Ruby, Smalltalk, Java, COBOL, PL/I and Python
  o some mappings are easy (Java and Python), but some are not (C++ and C).

• IDL interfaces contain no implementations for their operations
CORBA Overview

Diagram showing the CORBA architecture with components such as Interface Repository, IDL Compiler, Implementation Repository, CLIENT, OBJ, IDL STUBS, ORB INTERFACE, OBJECT (SERVANT), IDL SKELETON, DSI, OBJECT ADAPTER, GIOP/IIOP, ORB CORE, and standard interfaces.
CORBA IDL for Java

- The IDL-to-Java compiler, idlj, maps the IDL-defined interfaces to Java classes and interfaces
Example: Hello.idl

// generate a package statement in the Java code
module HelloApp
{
    // generate an interface statement in the Java code
    interface Hello
    {
        // generates methods statement in the Java interface.
        string sayHello();
        oneway void shutdown();
    }
};

$ idlj -fall Hello.idl
HelloServer.java

• Creates and initializes an ORB instance

• Creates a servant instance (the implementation of one CORBA Hello object) and tells the ORB about it

• Registers the new ORB object under the name "Hello"

• Waits for invocations from the client
CORBA example (Con't)

HelloClient.java

• Creates and initializes an ORB object

• Obtains a reference to the ORB object with the name "Hello"

• Invokes the object's sayHello() and shutdown() operations and prints the result
CORBA example (Con't)

- To start orbd:
  
  ```
  start orbd -ORBInitialPort 1050 -ORBInitialHost localhost
  ```

- To Start the Hello server:

  ```
  start java HelloServer -ORBInitialPort 1050
  ```

- Run the client application:

  ```
  java HelloClient -ORBInitialPort 1050
  ```
Most common usage

• CORBA is used more frequently in servers that must handle large number of clients, at high hit rates, with high reliability.
Future Work

• Experiment more complicated examples

• Compare D-bus and CORBA performance
Thank you!

Object-oriented IPC: D-Bus and CORBA

Duo Lu, Yu Zhang