Announcements

- Project handout
- Meet w/me to discuss your ideas

Project Schedule

<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sep 28</td>
<td>Initial proposal due</td>
</tr>
<tr>
<td>Sep 29-Oct 12</td>
<td>Get approval for topic</td>
</tr>
<tr>
<td>Oct 21</td>
<td>Milestone 1 due</td>
</tr>
<tr>
<td>Oct 22-26</td>
<td>First formal meeting</td>
</tr>
<tr>
<td>Nov 11</td>
<td>Milestone 2 due</td>
</tr>
<tr>
<td>Nov 12-18</td>
<td>Second formal meeting</td>
</tr>
<tr>
<td>Dec 7</td>
<td>Final project due</td>
</tr>
<tr>
<td>Dec 13-16</td>
<td>Final presentation</td>
</tr>
</tbody>
</table>

Recap

- Communication paradigms for distributed systems
  - Remote Procedure Call
  - Failure Semantics of RPC
  - Remote Object Invocation and IDL

RPC (Birrell/Nelson 1984)

RPC: Failure Semantics

- No strategy will guarantee “exactly-once” semantics
  - Why?
- At-most-once
- At-least-once
  - Idempotence
- Best-effort
Outline for Today

- More on ROI and CORBA
  - Full example for language binding
  - Flick
- Other communication paradigms
  - Content-based communications:
    - Tuples
    - Publish/subscribe systems

Example: OMG IDL (CORBA)

```idl
module data {
  typedef string<200> name;
  typedef string<20> phone;
  struct entry { name n; phone p; };
  exception duplicate { phone p; };
  exception notfound {};
  interface phonebook {
    void add(in entry e) raises (duplicate);
    void remove(in name n) raises (notfound);
    phone find(in name n) raises (notfound);
  };
};
```

Implementation using Orbit2

- Running
  ```bash
  # orbit-idl-2 phone.idl
  # orbit-idl-2 --skeleton-impl phone.idl
  # orbit-idl-2 --l cpp phone.idl
  ```
- Creates
  - phone-common.c; phone-cpp-common.cc/.h (shared)
  - phone-skels.c; phone-cpp-skels.cc/.h (for server)
  - phone-stubs.c; phone-cpp-stubs.cc/.h (for client)
  - phone.h; phone-cpp.cc/.h (shared)
  - phone-skelimpl.c (for server)
- Implemented server in C, client in C++

phone.h (1)

```cpp
#define CORBA_string data_name;
#define CORBA_string data_phone;
#define struct data_entry_type data_entry;
struct data_entry_type {
  data_name n;
  data_phone p;
};
#define struct data_duplicate_type data_duplicate;
struct data_duplicate_type {
  data_phone p;
};
#define CORBA_Object data_phonebook;
```

phone-cpp-common.h (1)

```cpp
namespace data {
  typedef char * name;
  struct entry {
    
  public:
    CORBA::String_mgr n;
    CORBA::String_mgr p;
  };
  class duplicate : public CORBA::UserException {
    
  public:
    CORBA::String_mgr p;
  };
  ...;
```
// C++ binding for client-side proxy
namespace data
{
class phonebook : public virtual CORBA::Object
{
public:
void add (const ::data::entry &e);
void remove (char const *n);
char * find (char const *n);
};
}
phone-client.cpp

```cpp
main() {
  orb = CORBA::ORB_init( ... ); // initialize ORB
  data::phonebook_ptr phonebook_server = orb->string_to_object(ior);
  data::entry e;
  try {
    e.n = CORBA::String_mgr("Godmar");
    e.p = CORBA::String_mgr("231-3046");
    phonebook_server->add(e);
  } catch (data::duplicate &ex) {
    g_print("Godmar already added with %s\n", ex.p.in());
  }
}
```

Flick Benefits

- Efficient Memory Management
  - (buffer size checks)
- Efficient Copying and Presentation
  - (use memcpy)
- Efficient Control Flow
  - (inlining, demultiplexing)
- Increased marshal throughput
- Increased End-to-end throughput

Taxonomy of Coordination

<table>
<thead>
<tr>
<th></th>
<th>Temporal</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Coupled</td>
<td>Uncoupled</td>
</tr>
<tr>
<td>Referential</td>
<td>Direct</td>
<td>Meeting</td>
</tr>
<tr>
<td></td>
<td></td>
<td>oriented</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Generative</td>
</tr>
<tr>
<td>Coupled</td>
<td></td>
<td>communication</td>
</tr>
<tr>
<td>Uncoupled</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Tuple Spaces

- Identity-based vs. Content-based
  - CORBA: Need to know identity (IOR) of receiver
- Tuple Spaces:
  - Message delivery based on content of message: type/structure/values

Tuple Spaces in Linda

$T = \{ (4, "xy\} \}$
Tuple Semantics

- out: out ("result", 42)
  - Write (nonblocking)
- eval: eval ("M", ...)
  - Write + execute
- in: in ("result", x:int ?)
  - Read (blocking)
- rd:
  - Like in, except nonblocking

Programming using Tuples

- Dining Philosophers

Dining Philosophers in Linda

phil(int i) {
  for (;;) {
    think();
    in("room ticket");
    in("fork", i);
    in("fork", (i+1)%5);
    eat();
    out("fork", i);
    out("fork", (i+1)%5);
    out("room ticket");
  }
}

setup() {
  for (int i = 0; i < 5; i++) {
    out ("fork", i);
    eval("phil", i);
    if (i < 4)
      out ("room ticket");
  }
}

Tuple-based communication

- What are advantages/disadvantages?
- How would you implement Tuple Spaces?

Summary

- RPC Language Binding
- Optimizations
- Content-based communication
- Thursday: Two Grid Computing Papers

Scratch