CS 5204 Operating Systems
Godmar Back
Fall 2004 - Lecture 4

Announcements
• Presentation schedule
• No office hours today

Recap
• Goals for distributed systems
  – Transparency, Consistency, Robustness, Scalability, Openness, Flexibility
• Scalability challenges and scaling techniques

Flexibility
• As a design goal for distributed systems
• Separate Policy from Mechanism
  – avoid embedding policy ideas into implementation
• Examples?

Layered Protocols
• OSI model vs E2E argument
  – Application vs Presentation
  – Session vs Transport
  – Transport vs Network
  – Network vs Data link
  – Data link vs Physical
Traditional TCP

Client

SYN
SYN, ACK(SYN)
request
ACK(req+FIN)
reply
FIN
ACK(FIN)

Server

SYN, ACK(SYN)

Transaction TCP

Client

SYN, ACK(SYN)
request
ACK(req+FIN)
reply
FIN
ACK(FIN)

Server

SYN, request, FIN
SYN, ACK(FIN), reply, FIN
ACK(FIN)

Outline for Today

• Communication Paradigms in Distributed Systems
  – Remote Procedure Call
    • Steps; failure modes
  – Remote Object Invocation
    • Runtime aspects
    • IDL Language Binding

RPC (Birrell/Nelson 1984)

Caller

arguments

results

Callee

Client Stub

arguments

results

Server Stub

arguments

results

Transport

request/reply msgs

RPC: Failure modes

• What can go wrong?

RPC: Failure modes

• Request msg is lost
• Server crashes after receiving request
  – Before executing callee
  – After executing callee
• Reply msg is lost
• Client crashes after outstanding request
Strategies

- Server:
  - Either execute call, then reply
  - Or reply with ACK that call will be executed next (if possible)
- Client (after server crash):
  - Never repeat
  - Always repeat
  - Repeat when ACKed
  - Repeat when not ACKed

Outcomes

<table>
<thead>
<tr>
<th>Client Reissue Strategy</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reply before Exec (RE)</td>
</tr>
<tr>
<td></td>
<td>REC</td>
</tr>
<tr>
<td>Always</td>
<td>2</td>
</tr>
<tr>
<td>Never</td>
<td>1</td>
</tr>
<tr>
<td>Only when ACKed</td>
<td>2</td>
</tr>
<tr>
<td>When not ACKed</td>
<td>1</td>
</tr>
</tbody>
</table>

RPC: Failure Semantics

- No strategy will guarantee “exactly-once” semantics
- At-most-once
- At-least-once
- Best-effort
- Idempotence

Remote Object Invocation

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);
    }
}
```
Using Gnome’s Orbit2

• Running
  # orbit-idl-2 phone.idl
  # orbit-idl-2 –skeleton-impl phone.idl
• Creates
  – phone-common.c (shared)
  – phone-skels.c (for server)
  – phone-stubs.c (for client)
  – phone.h (shared)
  – phone-skelimpl.c (for server)

phone.h (1)

```c
typedef CORBA_string data_name;
typedef CORBA_string data_phone;
typedef struct data_entry_type data_entry;
struct data_entry_type
{   data_name n;
    data_phone p;
};
```

```c
typedef struct data_duplicate_type data_duplicate;
struct data_duplicate_type
{   data_phone p;
};
```

typedef CORBA_Object data_phonebook;

phone.h (2)

```c
typedef struct {
    void * _private;
    void (*add) (PortableServer_Servant _servant, const data_entry * e, CORBA_Environment * ev);
    void (*remove) (PortableServer_Servant _servant, const CORBA_char * n, CORBA_Environment * ev);
    data_phone(*find) (PortableServer_Servant _servant, const CORBA_char * n, CORBA_Environment * ev);
} POA_data_phonebook__epv;
```

phone-stubs.c

```c
void data_phonebook_add(data_phonebook _obj, const data_entry * e, CORBA_Environment * ev);
```

phone-skelimpl.c

```c
typedef struct {
    POA_data_phonebook servant;
    PortableServer_POA poa;
    /* ------ add private attributes here ------ */
} impl_POA_data_phonebook;

/* create method, init, fini, ftables, … */
static void impl_data_phonebook_add(impl_POA_data_phonebook * servant, const data_entry * e, CORBA_Environment * ev);
```

Summary

• RPC
  – Steps; failure modes
• Use of IDLs to provide openness
• Optimizations
  • Thursday: software vs. hardware protection