Recap: Finger Table

- Finding a <key, value> using fingers

Recall?

Socket API

Socket API

Server
socket()
bind()
listen()
accept()
block
read()
write()
process request
handle response

Client
socket()
connect()
send request
send response
write()
read()
What's Wrong with Socket API?
• Low-level read/write
• Communication oriented
• Same sequence of calls, repeated many times
• Etc, etc...
• Not programmer friendly

Another Abstraction
• RPC (Remote Procedure Call)
  – Goal: it should appear that the programmer is calling a local function
  – Mechanism to enable function calls between different processes
  – First proposed in the 80's
• Examples
  – Sun RPC
  – Java RMI
  – CORBA
• Other examples that borrow the idea
  – XML-RPC
  – Android Bound Services with AIDL
  – Google Protocol Buffers

RPC
• Client
int main (…)
{
...
rpc_call(…);
...
}
• Server
void rpc_call(…) {
...}

Local Procedure Call
• E.g., x = local_call("str");
• The compiler generates code to transfer necessary things to local_call
  – Push the parameters to the stack
  – Call local_call
• The compiler also generates code to execute the local call:
  – Assigns registers
  – Adjust stack pointers
  – Saves the return value
  – Calls the return instruction

Remote Procedure Call
• Give an illusion of doing a local call by using whatever the OS gives
• Closer to the programmers
  – Language-level construct, not OS-level support
• What are some of the challenges?
  – How do you know that there are remote calls available?
  – How do you pass the parameters?
  – How do you find the correct server process?
  – How do you get the return value?

Stub, Marshalling, & Unmarshalling
• Stub functions: local interface to make it appear that the call is local.
• Marshalling: the act of taking a collection of data items (platform dependent) and assembling them into the external data representation (platform independent).
• Unmarshalling: the process of disassembling data that is in external data representation form, into a locally interpretable form.
RPC Process

Client Process
- Client Function
- Client Stub

Server Process
- Server Function
- Server Stub

Marshalling/unmarshalling

Socket API

How Do You Generate Stubs?
- Ever heard of C/C++, Java, Python syntax for RPC? None!
- Language compilers don’t generate client and server stubs.
- Common solution: use a separate language and a pre-compiler

Interface Definition Language (IDL)
- Allow programmers to express remote procedures, e.g., names, parameters, and return values.
- Pre-compilers take this and generate stubs, marshalling/unmarshalling mechanisms.
- Similar to writing function definitions

Example: SUN XDR

```plaintext
const MAX = 1000;
typedef int FileIdentifier;
typedef int FilePointer;
typedef int Length;
struct Data {
    int length;
    char buffer[MAX];
};
struct writeargs {
    FileIdentifier f;
    FilePointer position;
    Data data;
} = 9999;
```

Stub Generation
How Do You Find the Server Process?

- **Solution 1**
  - Central DB (the first solution proposed)
- **Solution 2**
  - Local DB with a well-known port (SUN RPC)

### Finding An RPC

RPCs live on specific hosts at specific ports. Port mapper on the host maps from RPC name to port.

When a server process is initialized, it registers its RPCs (handle) with the port mapper on the server.

A client first connects to port mapper (daemon on standard port) to get this handle.

The call to RPC is then made by connecting to the corresponding port.

---

How to Pass Parameters?

- **Pass by value:** no problem
  - Just copy the value
- **What about pointers/references?**
  - Need to copy the actual data as well
  - Marshall them at the client and unmarshall them at the server
  - Pass the local pointers/references
- **What about complex data structures?** struct, class, etc.
  - Need to have a platform independent way of representing data

---

External Data Representation

- Communication between two heterogeneous machines
  - Different byte ordering (big-endian & little-endian)
  - Different sizes of integers and other types
  - Different floating point representations
  - Different character sets
  - Alignment requirements

- Used in general contexts, not just in RPCs

---

Example: Google Protocol Buffers

- **Goal:** language- and platform-neutral way to specify and serialize data
- **Provides syntax & pre-compiler (open-source)**
  - Pre-compiler generates code to manipulate objects for a specific language, e.g., C++, Java, Python.
  - The runtime support applies a fast & sloppy compression algorithm.

```plaintext
message Book {
  required string title = 1;
  repeated string author = 2;
  optional BookStats statistics = 3;
  message BookStats {
    required int32 sales = 1;
  }
}
```

---

What About Failures?

- **Local calls do not fail.**
- **Remote calls might fail.**
- **Programmers should deal with this.**
  - No transparency here
**Failure Modes of RPC**

- Execute, crash before reply
- Crash before execution
- Channel fails during reply
- Client machine fails before receiving reply
- Lost request
- Request
- Execute
- Correct function
- Reply!

**Invocation Semantics**

- Local procedure call: exactly-once
- Remote procedure call:
  - 0 times: server crashed or server process died before executing server code
  - 1 time: everything worked well, as expected
  - 1 or more: excess latency or lost reply from server and client retransmission

- When do these make sense?
  - Idempotent functions: OK to run any number of times
  - Non-idempotent functions: cannot do it

- What we can offer
  - At least once
  - At most once

<table>
<thead>
<tr>
<th>Fault tolerance measures</th>
<th>Invocation semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retransmit request message</td>
<td>Duplicate filtering</td>
</tr>
<tr>
<td>No</td>
<td>No applicable</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Remote Method Invocation (RMI)**

- Process P1 ("client")
- Process P2 ("server")
- Request
- Reply
- Remote method execution

**Summary**

- RPC enables programmers to call functions in remote processes.
- IDL (Interface Definition Language) allows programmers to define remote procedure calls.
- Stubs are used to make it appear that the call is local.
- Semantics
  - Cannot provide exactly once
  - At least once
  - At most once
  - Depends on the application requirements

**Acknowledgements**

- These slides contain material developed and copyrighted by Indranil Gupta (UIUC).