Implementing Remote Procedure Calls*
Birrell, A. D. and Nelson, B. J.

Presented by Emil Constantinescu


Overview
- Brief introduction
- RPC issues
- Implementation
- Examples
- Current RPC implementations
- Review
- Conclusions

What are Remote Procedure Calls (RPCs) . . . . in a nutshell (1)
RPCs represent a set of communication paradigms that allow one procedure to call another procedure on a different machine.

1. one procedure (caller) calls another procedure (callee)
2. the caller waits for the result from the callee
3. the callee receives the request, computes the results, and then send them to the caller
4. the caller resumes upon receiving the results from the callee

What are Remote Procedure Calls (RPCs) . . . . in a nutshell (2)

RPC Issues (1)
- transparency, mimics the local procedure call
- precise semantics
  - machine failure
  - communication failure
- address based arguments
  - the address space is not shared
- programming integration
  - integration into programming environment
- data integrity

Attractive aspects:
- simple semantics
- efficiency
- generality
RPC Issues (2)

- data transfer protocols
  - network protocols
- binding
  - caller determines
    - the location
    - the identity
    - ... of the callee
- security
  - open communication network

RPC Implementation (1) - Basic Concepts

**interface**

**NAME**          | **type** | **type** | **type**
---                | ---      | ---      | ---
procedure_X       | arg1     | arg2     | ret1
procedure_Y       | arg1     | ret1     | ret2

**local program**

- call procedure_X(arg1,arg2,ret1)

**import procedure_X**

**remote program**

Implement:

```cpp
procedure_X(arg1,arg2,ret1) {
...
}
```

RPC Implementation (2) - Overview

- Caller machine
  - User
    - local call
    - local return
  - User stub
- Network
  - RPC runtime
  - Server stub
- Callee machine
  - Server
    - call
    - return
  - User
    - unpack
    - pack
- Database
  - lookup
  - register

RPC Implementation (3) - Local vs. Remote

```cpp
int main(...) { 
...
func(al, a2, ..., am): 
...
}

void func(pl, p2, ..., pm) {
...
}
```

RPC Implementation (4) - Local vs. Remote

```cpp
(*)

...) push a1 push a2 push ...
...) call func

...
...
...
...
```

RPC Implementation (5) - User/Server Stub

**User/Server Stub:**
- arguments are serialized/marshaled
- handles language binding (IDL, object passing)
- pass by value not by reference
**RPC Implementation (6) – RPCRuntime**

RPCRuntime deals with:
- data (re)transmission
- data acknowledge
- packet routing
- encryption
- exception handling
- binding

**RPC Implementation (7) – Binding**

Binding ...
- is concerned with
  - location
  - identity
  - implements
  - ImportInterface
  - ExportInterface
  - has issues
  - granularity
  - security

**RPC Implementation (8) – RPC flow**

Semantics & Data Transfer issues:
- scalable, connectionless scheme, reduced acknowledgements
- unique ID/incremental packets
- at least/most or exactly once semantics

**RPC Implementation (9) – Semantics**

- DCE RPC (Distributed Computing Environment RPC)
- SUN RPC
- DCOM (Distributed COM)
- CORBA (Common Object Request Broker Arch.)
- XML RPC
- SOAP (Simple Object Access Protocol)
RPC Implementations (2)
- DCE RPC
  - low level
  - handles the binding & transport
- Sun RPC
- DCOM & CORBA
  - object abstractions

RPC Implementations (3)
- XML RPC & SOAP
  - somewhat lightweight
  - use HTTP and XML
  - Port 80

Conclusions & Future Research
- Transparency is imperative, and leads to effectiveness
- Maintain local procedure calling semantics
- Binding strategies influences efficiency
- Emulate shared address space
- Timeout implementation

About the paper (1)
- Paper:
  - Birrell, A. D. and Nelson, B. J.
  - “Implementing Remote Procedure Calls”
  - ACM Transactions on Computer Systems
  - published in 1984
  - first paper to formalize RPC
  - concepts started to appear in 1976

About the paper (2)
- Special treatment for missing remote implementation (late binding)
- What if the client crashes?
- Machine specific binary data representation
- RPC in the HPC field?

References
- Garry Nutt; Operating systems; 2nd Edition, Addison Wesley
- Andrew Tanenbaum and Maarten van Steen; Distributed Systems: Principles and Paradigms, Prentice Hall, 2002
Any Questions?

Paper Implementation (1)

Paper Implementation (2)

Simple RPC

Paper Implementation (3)

"Complicated" RPC

Paper Implementation (4)

Results

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Minimum</th>
<th>Median</th>
<th>Transmission</th>
<th>Load-only</th>
</tr>
</thead>
<tbody>
<tr>
<td>no args/results</td>
<td>1059</td>
<td>1067</td>
<td>121</td>
<td>9</td>
</tr>
<tr>
<td>1 arg/results</td>
<td>1075</td>
<td>1105</td>
<td>142</td>
<td>10</td>
</tr>
<tr>
<td>2 args/results</td>
<td>1113</td>
<td>1157</td>
<td>174</td>
<td>15</td>
</tr>
<tr>
<td>4 args/results</td>
<td>1202</td>
<td>1211</td>
<td>239</td>
<td>17</td>
</tr>
<tr>
<td>10 args/results</td>
<td>1069</td>
<td>1105</td>
<td>176</td>
<td>13</td>
</tr>
<tr>
<td>10 word array</td>
<td>1214</td>
<td>1239</td>
<td>239</td>
<td>16</td>
</tr>
<tr>
<td>40 word array</td>
<td>1461</td>
<td>1605</td>
<td>503</td>
<td>51</td>
</tr>
<tr>
<td>100 word array</td>
<td>2915</td>
<td>2989</td>
<td>949</td>
<td>96</td>
</tr>
<tr>
<td>1000 word array</td>
<td>5555</td>
<td>5607</td>
<td>164</td>
<td>134</td>
</tr>
<tr>
<td>receive exception</td>
<td>1574</td>
<td>1597</td>
<td>204</td>
<td>196</td>
</tr>
<tr>
<td>receive exception</td>
<td>1574</td>
<td>1597</td>
<td>204</td>
<td>196</td>
</tr>
</tbody>
</table>