Implementing Remote Procedure Calls

Andrew Birrell and Bruce Nelson

Presented by
Kai Cong
Conventional Procedure Calls

Within a program on a single computer!!!
Extension of the Procedural Model

Across a network between programs on different computers!!!
Remote Procedure Call (RPC)
RPC – How it works

**client process**

- client
  - procedure call
- client stub
  - locate
  - (un)marshal
  - (de)serialize
  - send (receive)

**server process**

- server
  - procedure
- server stub
  - (un)marshal
  - (de)serialize
  - receive (send)

**RPC Runtime module**
Issues

- Binding
- Passing data
- Implementation
- Exceptions
- RPC systems
Binding

• The binding operation is to bind an importer of an interface to an exporter of an interface.

• After binding, calls made by the importer invoke procedures implemented by the remote exporter.

• static binding vs dynamic binding
Static Binding

- static binding (not presented by the paper)
  - hard coded stub
  - simple
  - efficient
  - not flexible
  - stub recompilation is necessary if the location of the server changes
  - use of redundant servers is not possible
Dynamic Binding

- dynamic binding
  - Interface: type and instance
  - Database
    - load balancing
  - flexible
  - redundant servers is possible
Dynamic Binding

client process

client

procedure call

client stub

7 bind
(un)marshal
(de)serialize

4 find
send
receive

RPC Runtime module

server process

server

procedure

server stub

register
(un)marshal
(de)serialize
receive
send

RPC Runtime module

Database
Issues

- Binding
- **Passing data**
- Implementation
- Exceptions
- RPC systems
Passing Data

• Can't use the stack!
• Can't use shared memory!
• Generally use message passing
Passing data

Build a message that includes:

- Who and what's being called
- Identity of the caller
- Data values in known byte order
  - Using an intermediate data representation
Issues

• Binding
• Passing data
• **Implementation**
• Exceptions
• RPC systems
Implementation
Implementation

• Function prototype is (almost) all that's needed to build the client stub
  – Also need binding information

• Function prototype is (almost) all that's needed to build server stub
  – Also need method to wait for message
Implementation

• Stub compiler
  – Generates stubs for client and server
  – Language dependent
  – Compile into machine-independent format

• Transport protocol
  – PUP, XML, SOAP, DCOM, CORBA, …
Implementation

Clients
Threaded

Servers
Event Driven
Issues

- Binding
- Passing data
- Implementation
- Exceptions
- RPC systems
Exceptions

• What can happen in "normal" procedures?
  – Procedure generates an exception
  – Procedure infinite loops
  – Procedure generates wrong results
Exceptions

• What can happen in "remote" procedures?
  – Client stub generates an exception
  – Transmission failure
    ▪ knowable failure
    ▪ unknowable failure
  – Remote procedure generates an exception
  – Remote procedure infinite loops
  – Remote procedure generates wrong results
Issues

• Binding
• Passing data
• Implementation
• Exceptions
• RPC systems
RPC Systems

- Sun RPC
- DCE RPC
- DCOM
- CORBA
- Java RMI
- XML RPC, SOAP/.NET, AJAX, REST
- Protocol Buffers (Google)
- Thrift (Facebook)
Conclusion

• Remote Procedure Call should look and feel like local call
• Remote Procedure Call should be independent of where it executes
• Remote Procedure Call should be "efficient"