## Implementing Remote Procedure Calls

by A.D. Birrell and B.J. Nelson

- Paradigm for communication across network between programs written in a high-level language
- Look like local procedure calls
- Across network (*caller, callee*)
- Alternatives: message passing, P and V operations, monitors, remote forks
- Environment
  - Xerox Park and University of Illinois (Cedar)
  - Hardware: Dorados, connected through Ethernet
  - Programming language: Mesa, Smalltalk, Lisp



## Goals/Features

- Simplicity
- User-friendly and general
- Efficiency
- Natural failure handling
  - no time-outs
  - ACKs and probe mechanisms
  - retransmissions
- Security at an intermediate level (encryption)



## Design Issues

- Precise semantics in the presence of machine and communication failures
- Semantics of address-containing arguments
- Integration of RPCs into existing programming systems
- Binding (i.e., how a caller determines the identity and location of the callee)
- Protocols for transfer of data and control
- Data integrity and security
- Performance considerations/Aims
  - communication efficient
  - secure end-to-end communication



## Structure

- Five components
  - *user*
  - *user-stub*
  - *RPCruntime*
  - *server-stub*
  - *server*



## Structure (cont.)

- RPCruntime: retransmissions, acks, packet routing, encryption
- *Lupine*: generates the user-stab and server-stub and dispatches to the correct procedure an incoming call
- *Interface modules*: list of procedure names with arguments and results
- Programmer writes interface modules (both *export* and *import*)
- No need for the programmer to build detailed communication-related code!



## Mental Model of a Simple Call



## Binding

- Naming: how does a client specifies what he wants to be bound to?
  OR how to bind an importer of an interface to an exporter of an interface
  - Two parts of an interface:
  - *Name*: which interface (abstract)
  - *Instance*: which particular implementor of an abstract interface
- Locating: what is the machine address of the callee and how to specify to the callee the procedure to be invoked?
  - *grapevine* database
  - highly reliable, replicated info



## Grapevine Distributed Database

- *RName* entry
  - Two types:
  - *individuals*: connect side (i.e., network address)
  - *groups*: member list (i.e., list of RNames)
  - two entries for each interface name: one for each name and one for each instance $\rightarrow$ both Grapevine RNames
  - Grapevine entry for instance: network address where the instance was last exported
  - Grapevine entry for type: group whose members are the instances of that type which have been exported
  - Example
- ExportInterface (procedure call)

## Binding and a Subsequent Call



## Packet-Level Transport Protocol

- Transport protocol just for RPC: efficiency
- Aim: minimize the elapsed time between initiating a call and getting results
- Must be cheap and fast (especially, if high performance of small messages is important)
- Main idea:
  - if the call returns, all went well
  - otherwise, exception is reported (did the server crash? are there network problems?)
  - no upper bound on how long the server waits: like regular procedures...



## Simple Calls



## Complicated Calls



## Final Points

- Implicit unbinding: it's the job of the caller to keep checking if the callee does not fail
- No time-out (like regular procedure calls)
- Binding can happen at three different times
  - dynamic ($\Rightarrow$ every time the call goes to grapevine to get RName)
  - first call goes to grapevine, but all subsequent calls use and "old" RName
  - static (at compile time) – if you know it, use it – can be the most efficient...



## Final Points (cont.)

- Each packet has its own increasing sequence numbers
  (time of day, name number, process number)
  this sequence number is used to eliminate duplicates and is also used in the returned result packet
- Increasing probe packet intervals, have implicit ACKs
- Handle multiple packet messages
  (not best for large data transfers)
- Callee can return "exception packets"



## Final Points (cont.)

- Have idle processes hanging around waiting for calls
  (saves on process creation)
- Use data encryption standard to encode messages
  - Grapevine provides the keys
  - RPCRuntime uses the keys