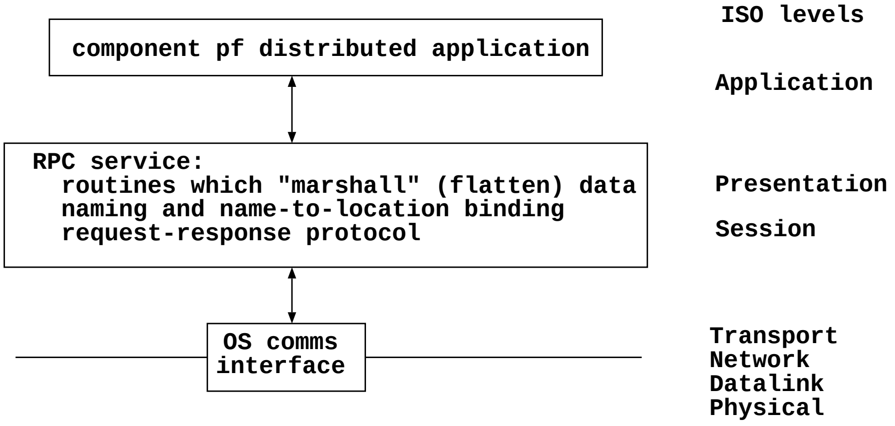
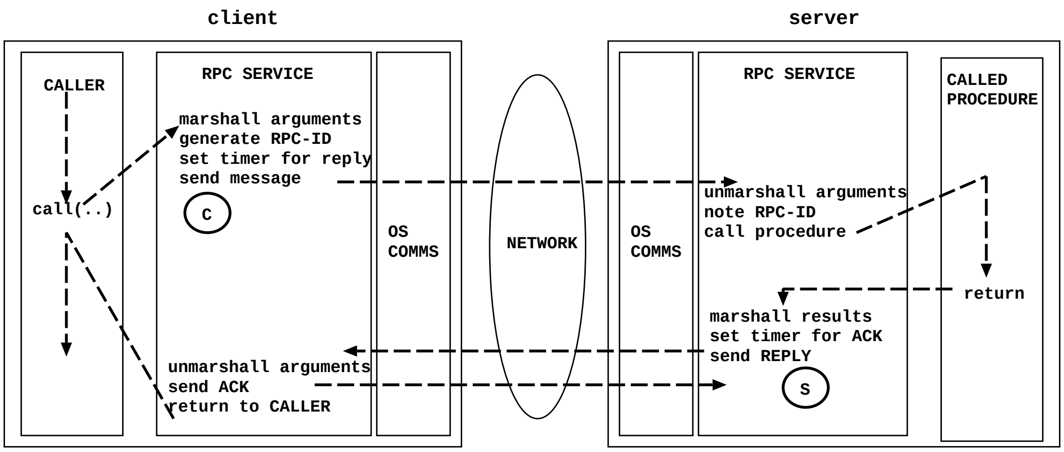
Remote Procedure Call (RPC)



examples: Mayflower/CCLU RPC, SUN RPC, ANSA RPC, MSRPC
Xerox Courier over XNS (SPP, Ethernet)
ISO-ODP, OSF DCE

RPC Request-Reply Acknowledge (RRA) protocol



C client timer

S server times

RPC semantics

recall that client, server and network may be congested or may fail independently of each other

RPC systems may offer AT MOST ONCE or EXACTLY ONCE semantics

C if the client timer expires:

AT MOST ONCE semantics:

exception return to the application
it is likely to repeat the call but this is not detectable
i.e. it will have a new RPC-ID

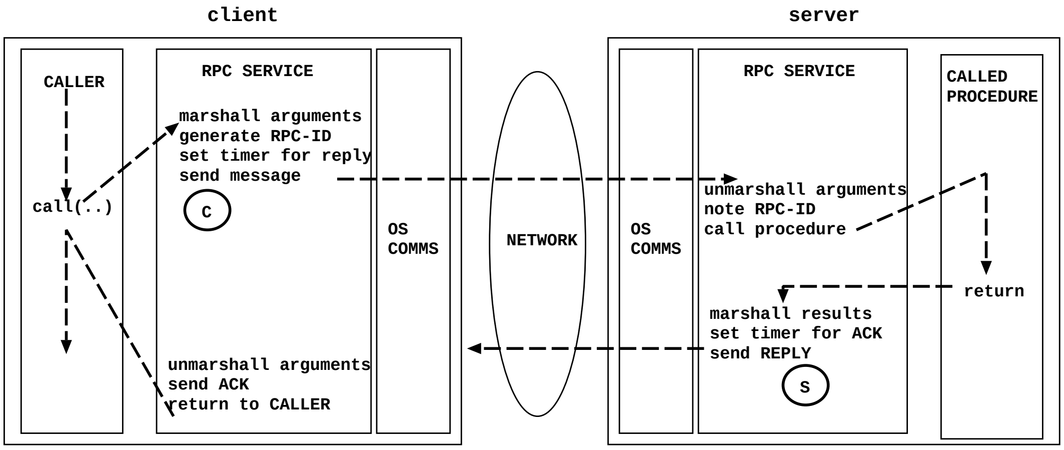
EXACTLY ONCE semantics:

retry a few times
RPC-ID means that the server can detect repeats
if no reply, exeption return to client

S if the server timer expires:

resend reults
RPC-ID means that the client can detect repeats

RPC client crash

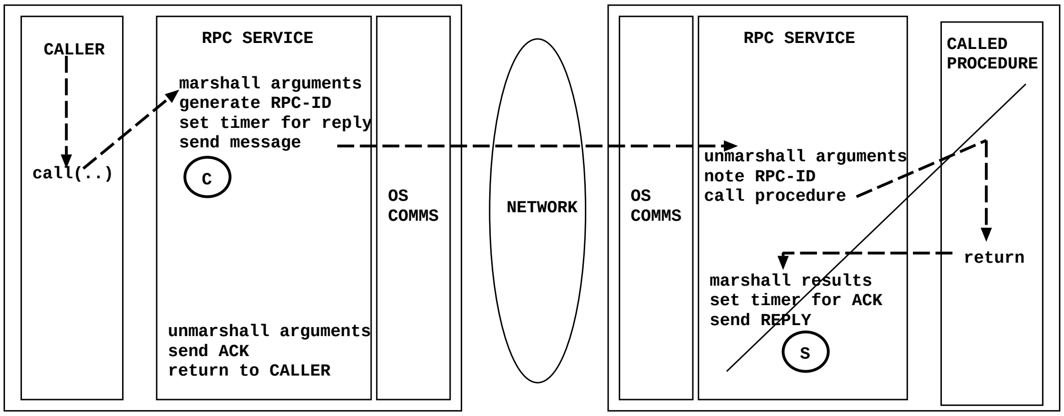


results are sent to crashed machine, are not acknowledged, S
and server timer expires repeatedly on resend

persistent state may have been changed by the procedure call
- should this be handled by RPC service?

NO - application-level transaction semantics (commit/abort)
should be used.

RPC server crash



the server fails at some stage during the call
results are not sent and the client timer expires repeatedly C

persistent state may or may not have been changed by the procedure call
- should this be handled by RPC service?

NO - application-level transaction semantics (commit/abort)
should be used.

Integration of Programming Languages and RPC

* some early RPC systems aimed for complete distribution transparency
  e.g. Xeroc PARC, Mesa language, Courier RPC
  a preprocessor detects which calls are not to local procedures
  and replaces them by calls to RPC support

  problem of incorrect procedure names that don’t exist anywhere
  problem of call semantics for some arguments

* Cambridge Mayflower system, CCLU RPC - made distribution explicit
  the compiler was changed
  different syntax for definition and call of procedures
  that can be called remotely

  BUT - this was still for a single language, CCLU

* ANSA RPC, initially developed for C but later for e.g. C++, Modula3
  defined a Distributed Programming Language (DPL)
  DPL statement are embedded in the programming language, and tagged
  a preprocessor detects these statement, replaces them with calls to
  RPC service

All RPC systems automatically generate STUBS (marshalling routines)
to flatten the arguments into packet format, suitable for transmission

Integration of Programming Languages and Middleware

Assume that we wish to support a number of different programming languages, i.e. components written in different languages can interoperate

* standard approach (ANSA, ISO-ODP, OSF-DCE), O-O platforms
  - define an Interface Definition Language (IDL)
  - provide mappings for programming language's type systems onto IDL
  - (internally) define the transfer syntax for IDL types

* how do platforms that support objects and object invocation differ from the RPC scheme described above?

  when objects are registered with the platform, an object-ID is returned and the object becomes known globally and may be invoked remotely

  Object-IDs are first-class values which may be passed as arguments

  some RPC systems restricted the argument types
  e.g. SUN RPC: C base-types only
  CCLU RPC: most types including procedure names defined as remote

  user supplies marshalling and unmarshalling routines for constructed types (recursive descent)