COMP750
Distributed Systems
Remote Procedure Calls
Remote Function Execution

• When a program calls a function or method (or procedure or subroutine), the function is usually executed as part of the main program.

\[ x = \text{func}(z); \]

• func is run on the same computer as the main.

• Intermediate software or middleware can be used to execute the procedure on a remote computer.
Remote Procedure Call (RPC)

- A remote procedure call is a paradigm for writing distributed programs or programs that communication between machines.
- An RPC calls a function or procedure that is executed on another computer.
- RPC’s provide a more organized, high level interface to writing distributed software than TCP send and receives.
RPC Advantages

• Appears to the user like a call to a function on the local machine.
• RPC concept is similar to conventional programming.
• Procedure calling has well understood semantics.
• RPC can be extended to Remote Method Invocation (RMI) for OO systems.
• RPC simplifies access to remote systems.
RPC Construction

• Assume the main program is calling
  \[ \text{int} \ \text{myfunc}(\text{float} \ x); \]

• Write a “client stub” called \text{myfunc} that
  will not perform the computation, but will
  send the parameter to a remote system.

• Write a “server stub” that will receive the
  data and call the actual \text{myfunc} function.
  The server stub will send back the results.
RPC Overview

Actual execution flow

Logical
RPC Operation

• The main program calls the client stub.
• The client stub copies or *marshals* the parameters into a communication’s packet.
• The client stub send an RPC request to the server.
• The server calls the desired function passing the parameters from the communication’s packet.
• When the function returns, the server sends the results to the client stub.
• The client stub returns the results to the caller.
Call Semantics

- **Call by Value** works well with RPC
- **Call by Reference** does not fit RPC semantics.
- **Call by Copy** is similar to call by reference (assuming no aliases) and works well with RPC
RPC Input Parameters

- Input is data passed from caller to function.
- **basic datatype** - pass by value semantics, copy data to server
- **arrays or structures** - copy data to server, may be a lot of data
- **pointers to single datatype** - copy data item to server
- **pointers in general** - not allowed
RPC Output Parameters

- Output parameters are passed from the function back to the calling program.
- C programs use pointers to output variables
  
  example: \( x = \text{func}(&z); \)

- Data, not addresses, copied from server
- Return value copied from server
RPC Execution Environment

- No global variables
- No environment variables
- No access to files on the client computer.
Exception Handling

• What does the communication stub do if the first effort to send a message fails?
  – no retry     RPC may not work
  – at-least-once Keep sending until it gets there (idempotent functions)
  – at-most-once retry but server must filter repeats
Synchronization

• Normal RPC are synchronous, the calling program waits until the called function completes.
• Normal RPCs do not provide parallelism.
• The logical thread of execution moves to another computer and then back.
• Asynchronous RPCs execute the called function in parallel with the main program.
Asynchronous RPC

• Call without reply - stub returns to caller after sending message.
• Useful for “void” functions that do not return a result.
• A “print” function might be a good candidate for an asynchronous RPC.
Call Backs

• The RPC returns before the result is complete.
• The server calls a completion function in the client program when the RPC has completed.
• Call backs support event driven programming.
Automatic Stub Generation

• There are several systems that will automatically generate the client and server stubs.

• Distributed Computing Environment (DCE) from the Open Software Foundation (OSF) supports an RPC system.

• The Interface Definition Language (IDL) defines the function interface similar to a function prototype.
IDL Example

int funcabc(
    [in] float x,
    [out] long *y,
    [in out] double *z
);
Distributed Computing Environment

IDL Source

DCE

Client stub

Server stub

main program

function

DCE library

Compiler

Client program

Server program

Compiler
X- Windows Example

- Main computer is the client. User programs call the server to display data.
- The terminal is the display server.
- Uses Asynchronous RPC.