Based on slides by Harsha V. Madhyastha and Manos Kapritsos

EECS 482 Introduction to Operating Systems

Spring/Summer 2020

Lecture 22: Remote procedure calls

Nicole Hamilton
https://web.eecs.umich.edu/~nham/
nham@umich.edu

Agenda

1. Bonus lecture on sockets Tue 3:00 pm.
2. Remote procedure calls.

Agenda

1. Bonus lecture on sockets Tue 3:00 pm.
2. Remote procedure calls.

Bonus lecture

Tues 3:00 pm, streamed and recorded as usual.

This is material from my search engine class.

Will not appear on the final.

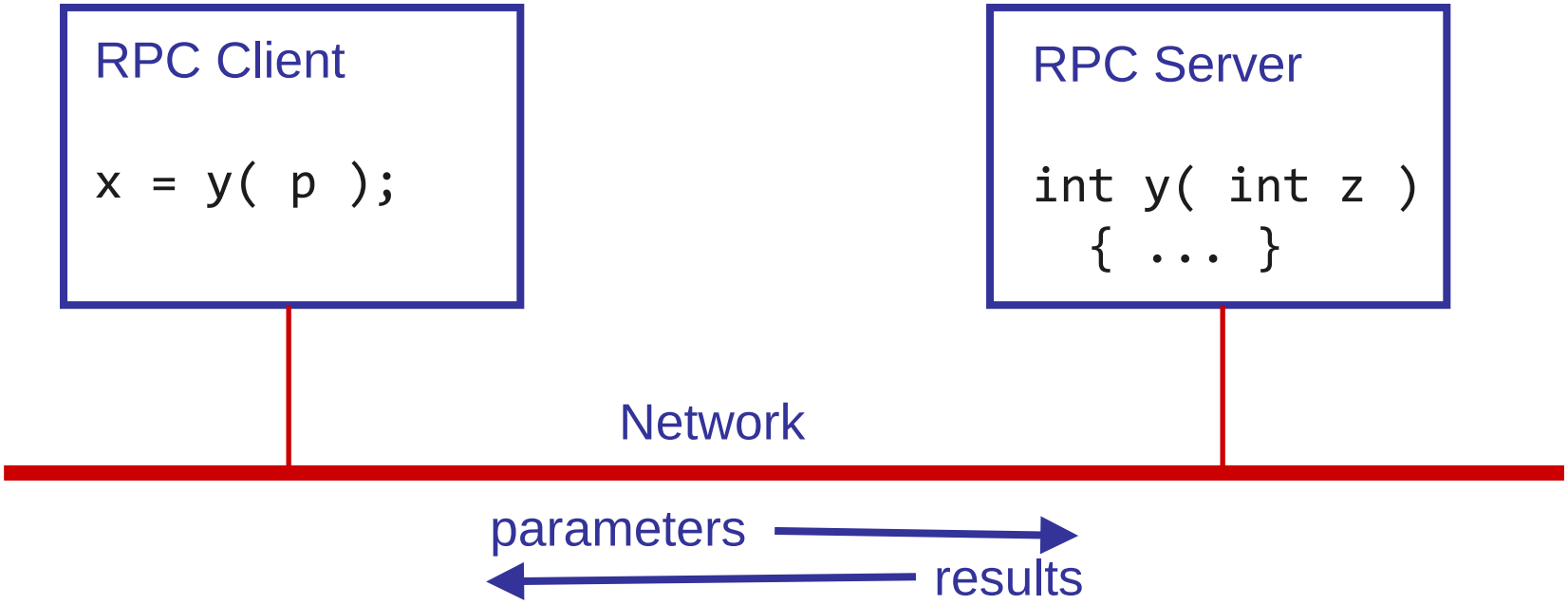
Intended only to help you understand how sockets and servers work to help you with P4.

Agenda

1. Bonus lecture on sockets Tue 3:00 pm.
2. Remote procedure calls.

Remote Procedure Calls

Mechanism for allowing an application on one machine to call a procedure on another machine:



Remote Procedure Call

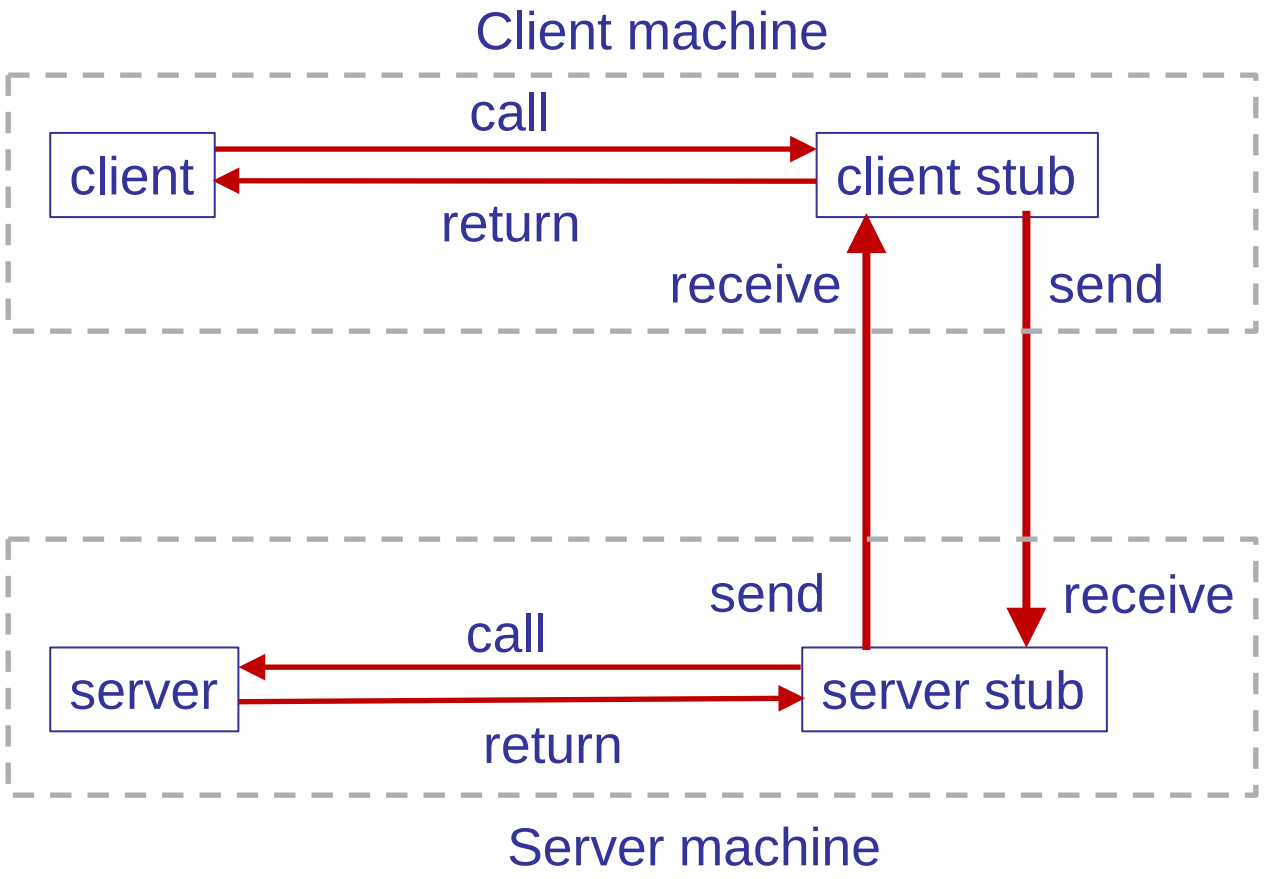
Hide complexity of message-based communication from developers.

Procedure calls more natural for inter-process communication.

Goals of RPC:

- Client sending request → function call
- Client receiving response → returning from function
- Server receiving request → function invocation
- Server sending response → returning to caller

RPC abstraction via stub functions on client and server



RPC stubs

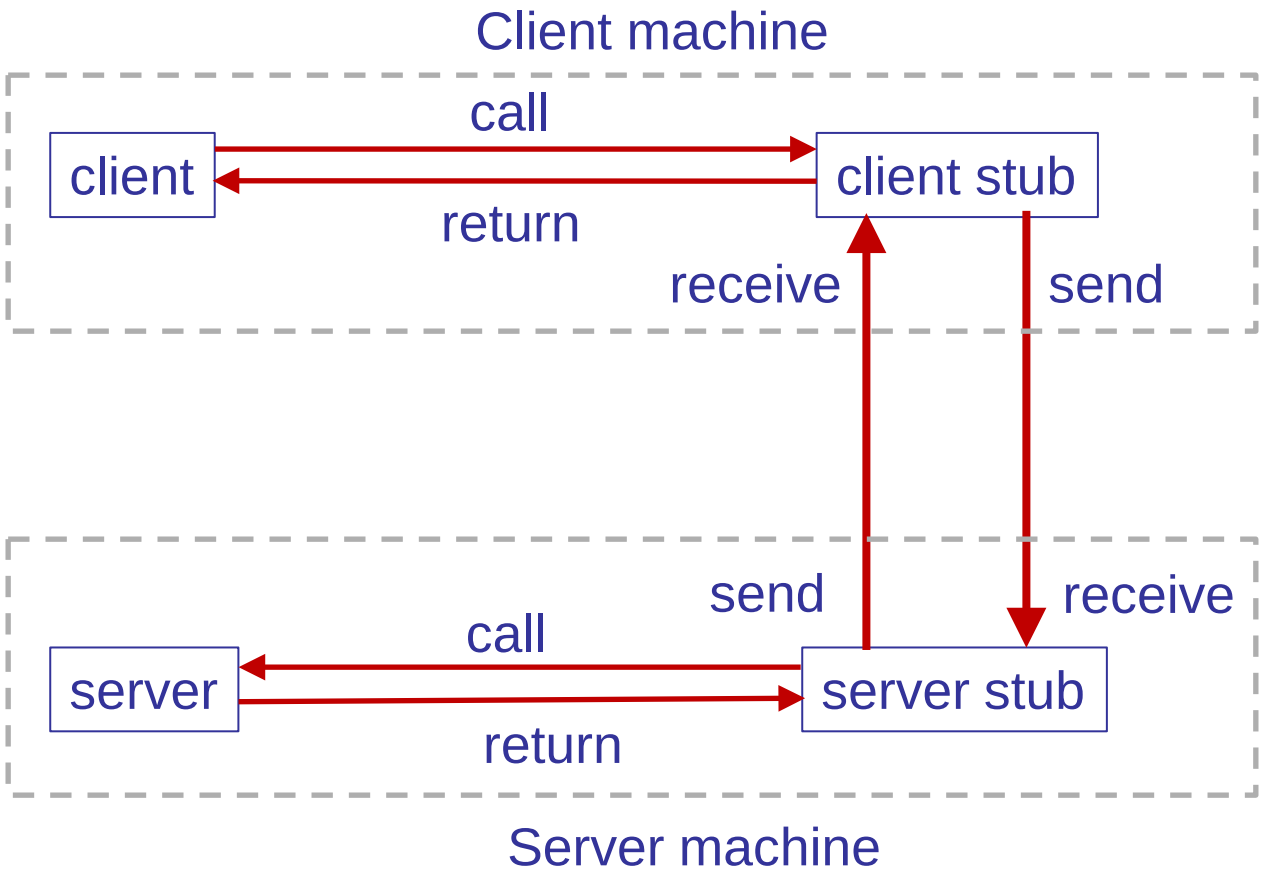
Client stub

- Constructs message with function name and parameters
- Sends request message to server
- Receives response from server
- Returns response to client

Server stub

- Receives request message
- Invokes correct function with specified parameters
- Constructs response message with return value
- Sends response to client stub

RPC abstraction via stub functions on client and server



Producer-consumer using RPC

Client side

```
int Remote( int n )
   {
   int status;
   send( sock, &n,
      sizeof( n ) );
   recv( sock, &status,
      sizeof( status ) );
   return( status );
   }
```

Server side

```
void RemoteStub( )
   {
   int n, result;
   recv (sock, &n,
      sizeof( n ) );
   status = Local( n );
   send (sock, &status,
      sizeof( status ));
   }
```

Generation of stubs

Stubs can be generated automatically

What do we need to know to do this?

Interface description:

Types of arguments and return value

e.g., rpcgen on Linux

RPC Transparency

RPC makes remote communication look like local procedure calls

Basis of CORBA, Thrift, Microsoft SOAP, Java RMI, …

What factors break illusion?

Failures	Remote nodes/networks can fail.
Performance	Remote communication is inherently slower.
Service discovery	Client stub needs to bind to server stub on an appropriate machine.

RPC serialization

Basic strategy is to transfer information between possibly dissimilar machines by *serializing* it over a network using a *protocol*.

Often called *marshalling*.

Generally support all the usual C++ types, including structs and classes.

Lots of possible choices for serializing.

Binary

json

XML

Compressed

HTTP MIME types

RPC Arguments

Can I have pointers as arguments?

How to pass a pointer as argument?

- Client stub transfers data at the pointer
- Server stub stores received data and passes pointer

Challenge:

- Data representation should be same on either end
- Example: I want to send a 4-byte integer:
  - 0xDE AD BE EF
  - Send byte 0, then byte 1, byte 2, byte 3
  - What is byte 0?

Endianness

Machines differ in how they store integers and we need to account for this in transferring data.