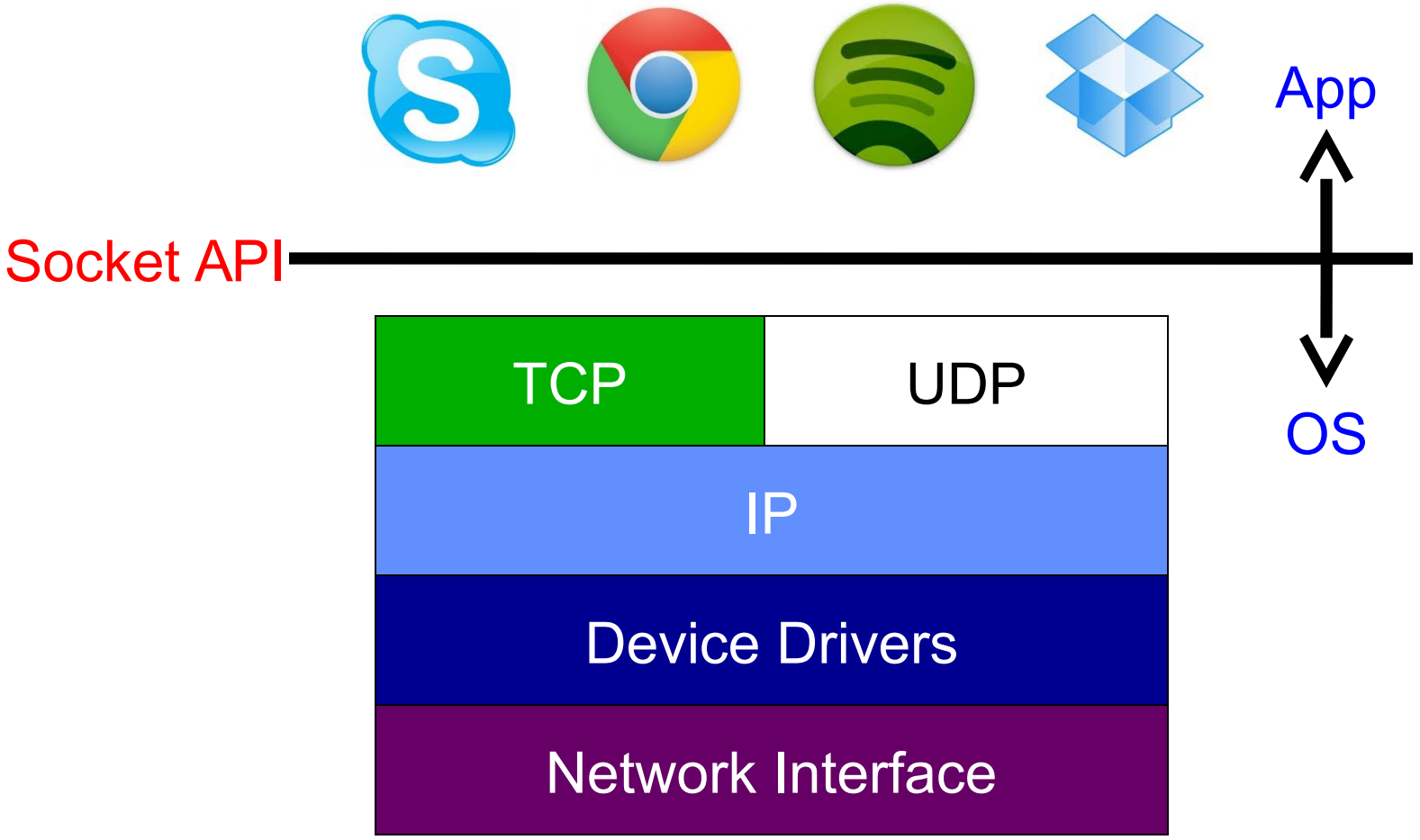
CSE 486/586 Distributed Systems

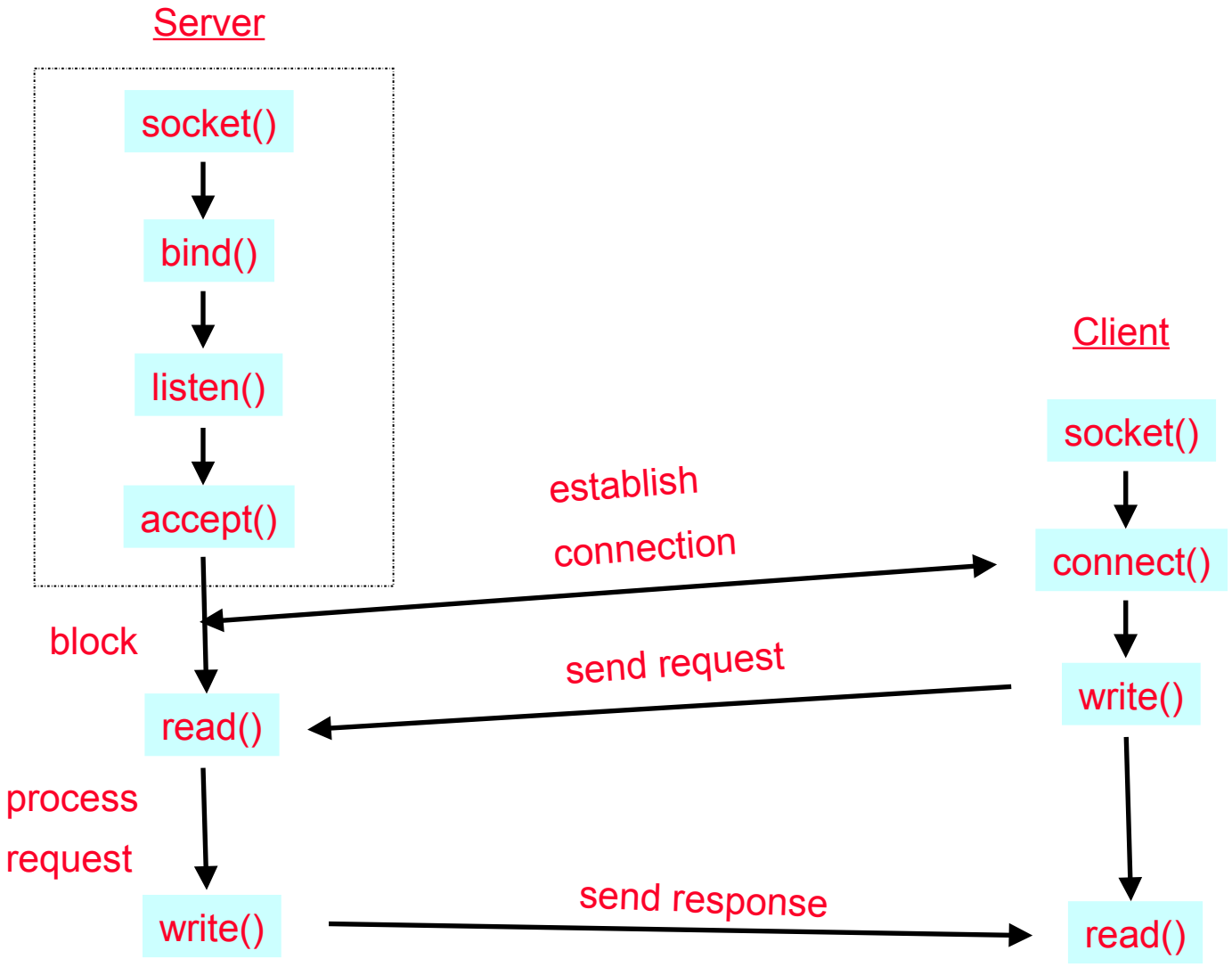
Remote Procedure Calls

Slides by Steve Ko
Computer Sciences and Engineering
University at Buffalo

Recall the Socket Model



Socket API


Server
socket()
bind()
listen()
accept()
block
read()
process
request
write()
Client
socket()
connect()
write()
read()
establish
connection
send request
send response

What's Wrong with the Socket API?

- Low-level read/write – looks like file I/O
- Communication oriented
- Same sequence of calls, repeated many times
- *Etc.*, *etc*…
- Not very programmer-friendly

Another Abstraction

- RPC (Remote Procedure Call)
  - Goal: to appear that the programmer is calling a local function
  - Mechanism to enable function calls between different processes
  - First proposed in the 80's
- Examples
  - Sun RPC (which we saw used for NFS)
  - Java RMI
  - CORBA
- Other examples that borrow the idea
  - XML-RPC
  - Android Bound Services with AIDL
  - Google Protocol Buffers

RPC

- Client

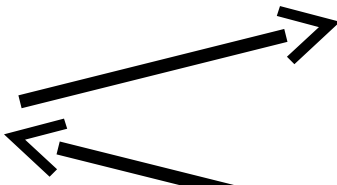
```
int main (…)
{
   …
   rpc_call(…);
   …
}
```

- Server

```
…

void rpc_call(…) {
   …
}

…
```



Local Procedure Call

```
x = local_call("str");
```

- The compiler generates code to transfer necessary information to `local_call`:
  - Push parameters onto the stack
  - Call `local_call`
- The compiler generates code to execute the procedure.
  - Assigns registers
  - Adjust stack pointers
  - Saves the return value
  - Calls the return instruction or jumps back to the calling location

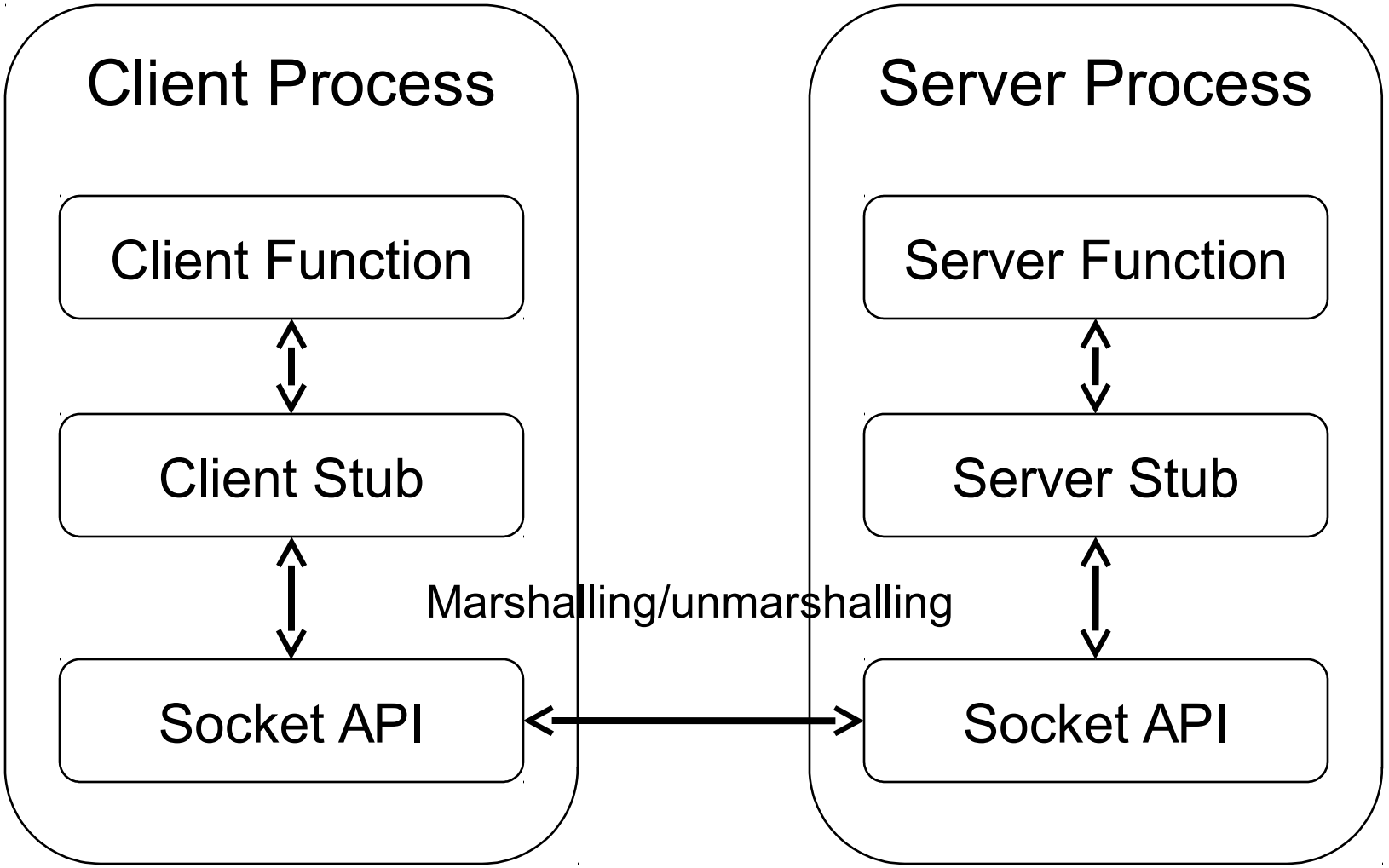
Remote Procedure Call

- Give the illusion of doing a local call by using whatever facilities the OS provides
- Closer to the programmer's preferred abstraction
  - Language-level construct, not OS-level support
- What are some of the challenges?
  - How do you know that there are remote calls available?
  - How do you pass parameters?
  - How do you find the correct server process?
  - How do you retrieve return values?

Stubs, Marshalling, & Unmarshalling

- Stub functions: local procedures to make it appear that the call is local.
- Marshalling: the act of taking a collection of platform-dependent data items (*e.g.*, arguments) and assembling them into a platform-independent external data representation.
- Unmarshalling: the process of returning data that is in an external data representation into a locally interpretable form.

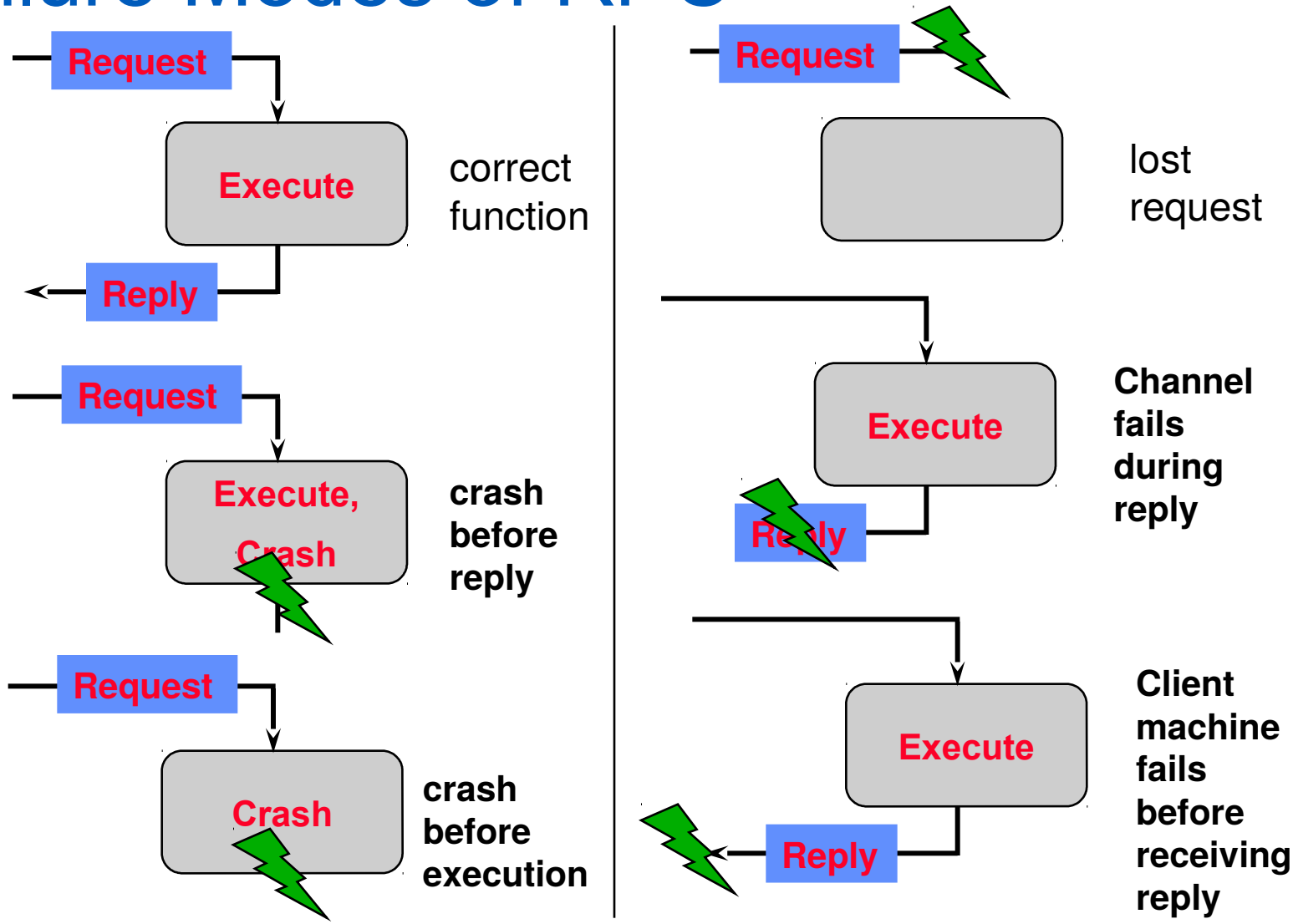
RPC Process


Client Process
Client Function
Client Stub
Socket API
Marshalling/unmarshalling
Server Process
Server Function
Server Stub
Socket API

Invocation Semantics with Failure

- Local calls may be called unexceptionally.
- Remote calls might fail.
- Programmers must deal with this.
  - No transparency here

Failure Modes of RPC



Invocation Semantics

- Local procedures execute exactly once per invocation
- Remote procedure call:
  - 0 times: server crashed or server process died before executing server code
  - 1 time: everything worked as expected
  - 1 or more: excess latency or lost server reply with client retransmission
- When do these make sense?
  - Idempotent functions: OK to run any number of times
  - Non-idempotent functions: cannot allow this
- What we can offer
  - At least once
  - At most once

Invocation Semantics

Fault tolerance measures			*Invocation semantics*
Retransmit request message	*Duplicate filtering*	*Re-execute procedure or retransmit reply*	
No	Not applicable	Not applicable	*Maybe*
Yes	No	Re-execute procedure	*At-least-once*
Yes	Yes	Retransmit old reply	*At-most-once*

How Do You Generate Stubs?

- Ever heard of C/C++, Java, Python syntax for RPC?
  - There is none!
- Language compilers don't generate procedure stubs.
- Common solution
  - Use a separate language
  - Compile into stubs with a pre-compiler

Interface Definition Language (IDL)

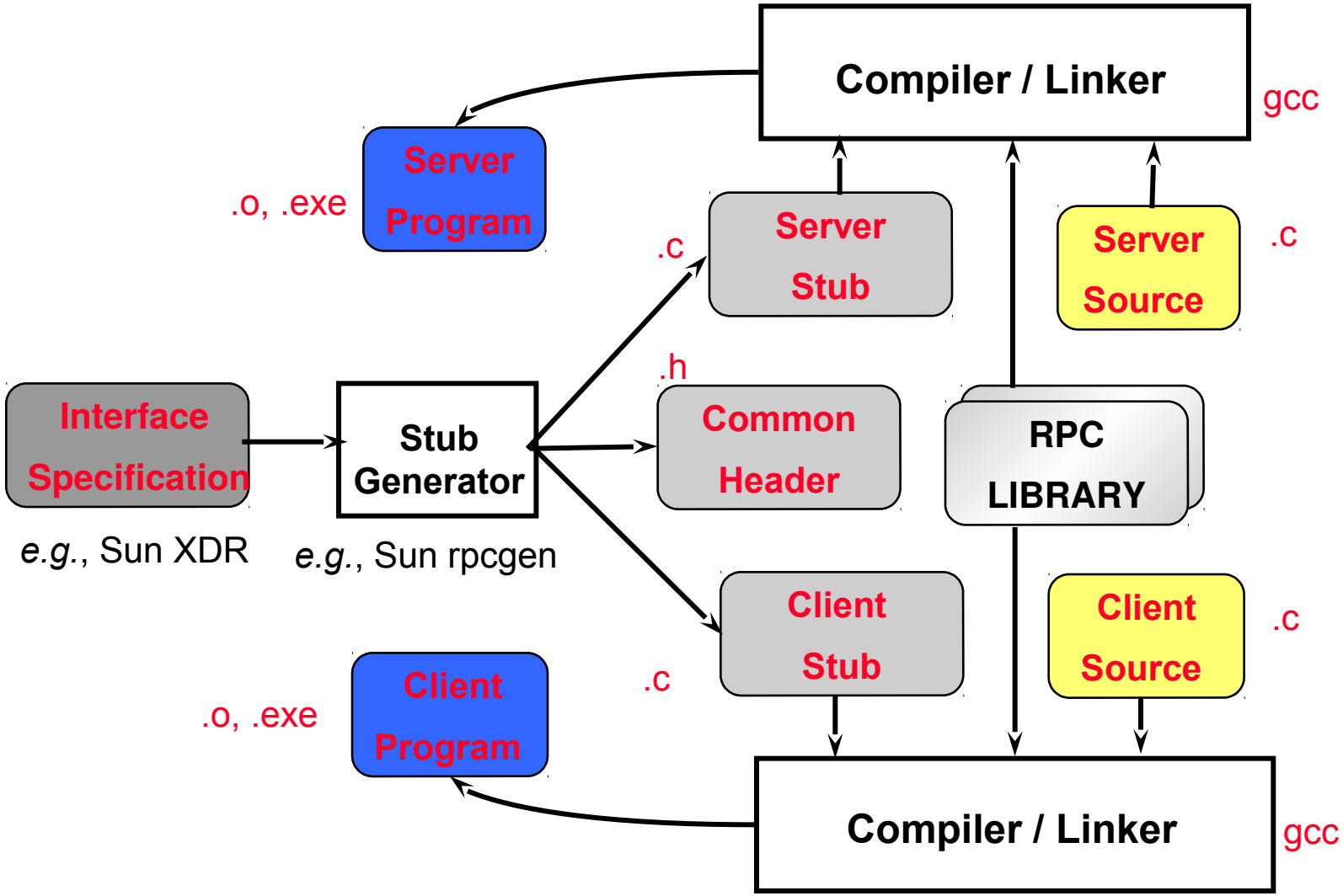
- The stub description language is called an interface definition language (IDL)
- Programmers use the IDL to describe procedures
  - *e.g.*, names, parameters, and return values
- Pre-compilers take this description and generate stubs, marshalling/unmarshalling mechanisms.
- Similar to writing function definitions

Example: OMG IDL

OMG IDL is used for CORBA and other RPC technologies.

```
interface account {
    boolean authenticate(in string username,
                         in string password);
    void deposit(in float amount);
    void withdraw(in float amount);
    float balance();
};
```

Stub Generation

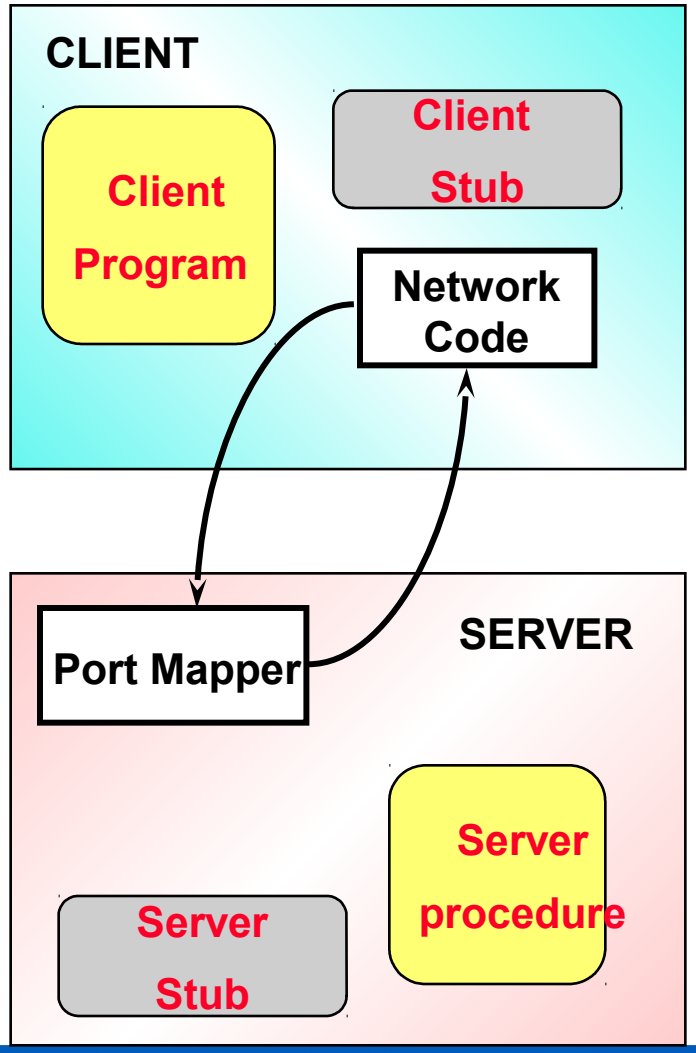


Finding the Server

The client process must somehow locate the server.

- Solution 1
  - Central DB (the first solution proposed)
- Solution 2
  - Local DB with a well-known port (SUN RPC)

Local DB with Well-Known Port



Finding An RPC:

RPCs live on specific hosts at specific ports.

Port mapper on the host maps from RPC name to port #

When a server process is initialized, it registers its RPCs (handle) with the port mapper on the server

A client first connects to port mapper (daemon on standard port) to get this handle

The call to the RPC is then made by connecting to the corresponding port

Parameter Passing

- Pass by value: no problem
  - Just copy the value
- What about pointers/references?
  - Need to copy the actual data as well
  - Marshall them at the client and unmarshall them at the server
  - Pass the local pointers/references
- What about complex data structures? struct, class, *etc.*
  - Need to have a platform independent way of representing data

External Data Representation

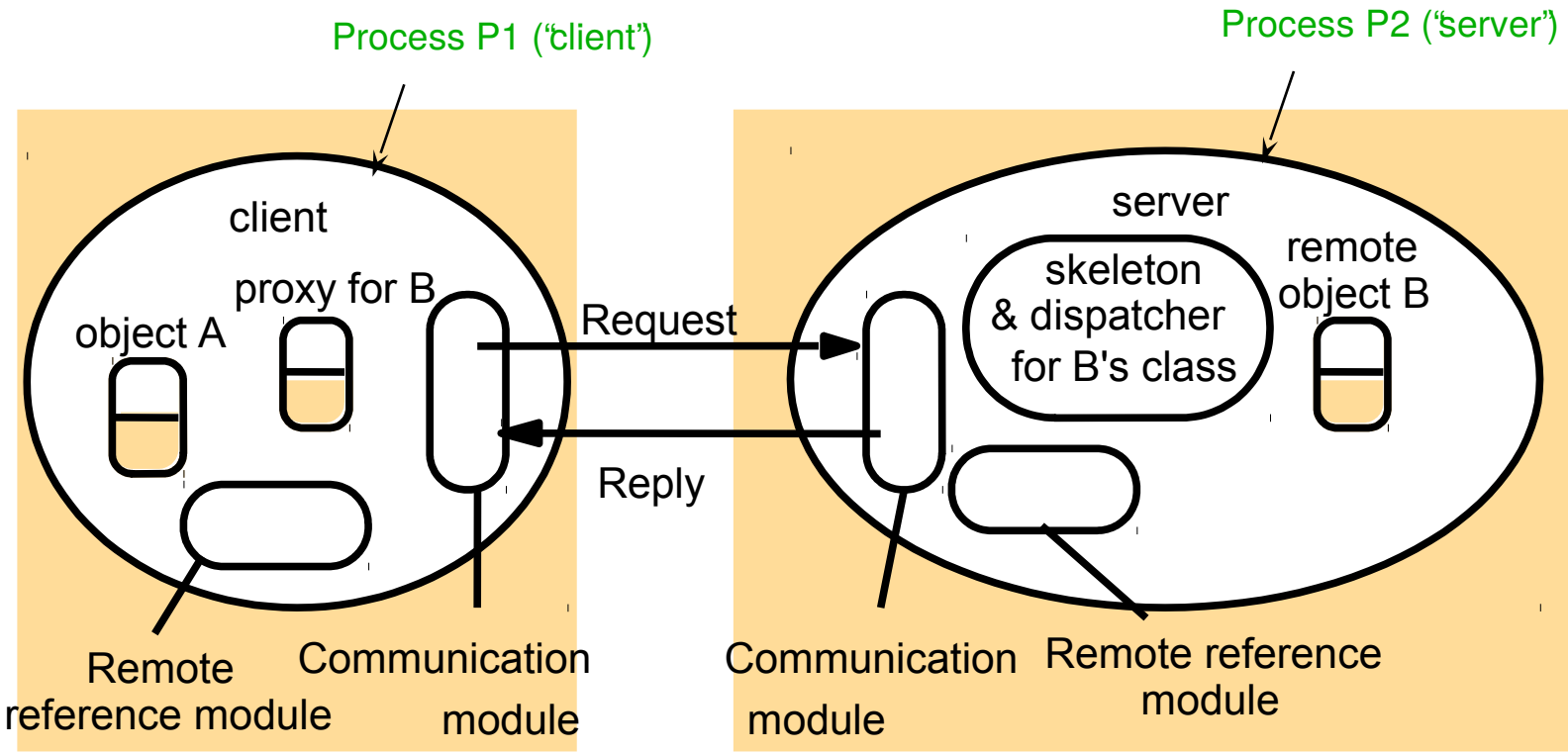
- Communication between two heterogeneous machines
  - Different byte ordering (big-endian *vs.* little-endian)
  - Different sizes of integers and other types
  - Different floating point representations
  - Different character sets
  - Alignment requirements
- Used in general contexts, not just in RPCs
  - (This is exactly what Google Protocol Buffers are)
  - JSON is widely used

Example: Google Protocol Buffers

- Provide a language- and platform-neutral way to specify and serialize data
- Provides syntax & pre-compiler (open-source)
  - Pre-compiler generates code to manipulate objects for a specific language, e.g, C++, Java, Python.
  - Runtime support applies a fast & sloppy compression algorithm.

```
message Book {
    required string title = 1;
    repeated string author = 2;
    optional BookStats statistics = 3;
    message BookStats {
        required int32 sales =1;
    }
}
```

Remote Method Invocation (RMI)



Summary

- RPC enables programmers to call functions in remote processes.
- IDL (Interface Definition Language) allows programmers to define remote procedure calls.
- Stubs are used to make it appear that the call is local.
- Semantics
  - Cannot provide exactly once
  - At least once
  - At most once
  - Depends on the application requirements

References

- Textbook Sections 5.1-5.3, 21.4. **Required Reading.**
- Textbook Sections 5.5, 8.1-8.3. (Optional)

Acknowledgements

- These slides originally by Steve Ko, lightly modified and used with permission by Ethan Blanton.
- These slides contain material developed and copyrighted by Indranil Gupta (UIUC).