

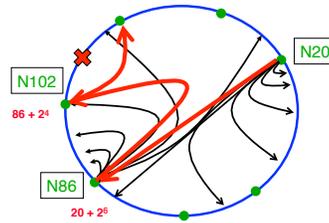
CSE 486/586 Distributed Systems Remote Procedure Call

Steve Ko
Computer Sciences and Engineering
University at Buffalo

CSE 486/586, Spring 2013

Recap: Finger Table

- Finding a <key, value> using fingers



CSE 486/586, Spring 2013

2

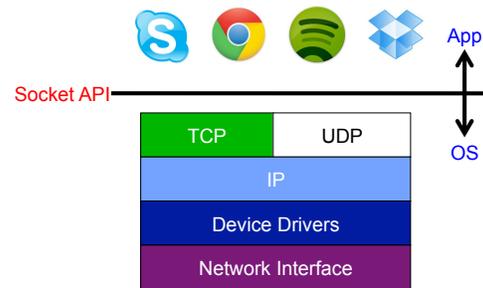
Chord: Node Joins and Leaves

- When a node joins
 - Node does a lookup on its own id
 - And learns the node responsible for that id
 - This node becomes the new node's successor
 - And the node can learn that node's predecessor (which will become the new node's predecessor)
- Monitor
 - If doesn't respond for some time, find new
- Leave
 - Clean (planned) leave: notify the neighbors
 - Unclean leave (failure): need an extra mechanism to handle lost (key, value) pairs

CSE 486/586, Spring 2013

3

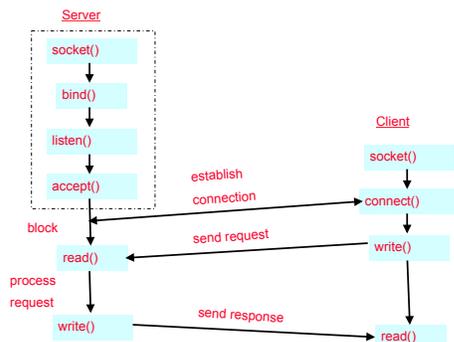
Recall?



CSE 486/586, Spring 2013

4

Socket API



CSE 486/586, Spring 2013

5

What's Wrong with Socket API?

- Low-level read/write
- Communication oriented
- Same sequence of calls, repeated many times
- Etc, etc...
- Not programmer friendly

CSE 486/586, Spring 2013

6

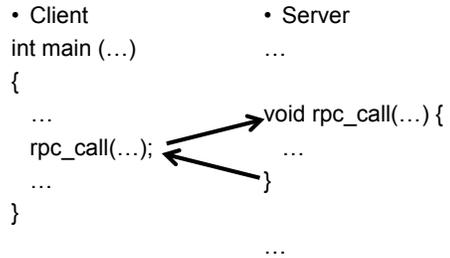
Another Abstraction

- RPC (Remote Procedure Call)
 - Goal: it should 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
 - Java RMI
 - CORBA
- Other examples that borrow the idea
 - XML-RPC
 - Android Bound Services with AIDL
 - Google Protocol Buffers

CSE 486/586, Spring 2013

7

RPC



CSE 486/586, Spring 2013

8

Local Procedure Call

- E.g., `x = local_call("str");`
- The compiler generates code to *transfer necessary things* to `local_call`
 - Push the parameters to the stack
 - Call `local_call`
- The compiler also generates code to *execute the local call*.
 - Assigns registers
 - Adjust stack pointers
 - Saves the return value
 - Calls the return instruction

CSE 486/586, Spring 2013

9

Remote Procedure Call

- Give an illusion of doing a local call by using whatever the OS gives
- Closer to the programmers
 - 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 the parameters?
 - How do you find the correct server process?
 - How do you get the return value?

CSE 486/586, Spring 2013

10

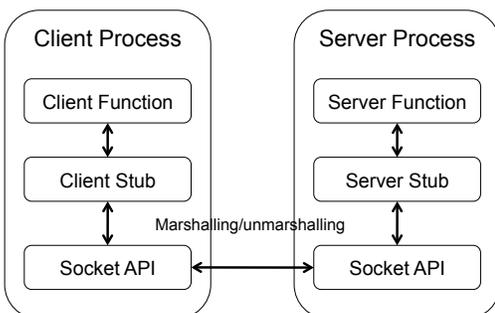
Stub, Marshalling, & Unmarshalling

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

CSE 486/586, Spring 2013

11

RPC Process



CSE 486/586, Spring 2013

12

CSE 486/586 Administrivia

- PA2 due in ~2 weeks
- PA1 grades are out.
- AWS codes are on UBLearns.
 - Setup instructions have been posted as well.
- Practice problem set 1 & midterm example posted on the course website.
- Midterm on Wednesday (3/6) @ 3pm
 - Not Friday (3/8)
- Come talk to me!

CSE 486/586, Spring 2013

13

How Do You Generate Stubs?

- Ever heard of C/C++, Java, Python syntax for RPC?
 - None!
- Language compilers don't generate client and server stubs.
- **Common solution:** use a separate language and a pre-compiler

CSE 486/586, Spring 2013

14

Interface Definition Language (IDL)

- Allow programmers to express remote procedures, e.g., names, parameters, and return values.
- Pre-compilers take this and generate stubs, marshalling/unmarshalling mechanisms.
- Similar to writing function definitions

CSE 486/586, Spring 2013

15

Example: SUN XDR

```

const MAX = 1000;
typedef int FileIdentifier;
typedef int FilePointer;
typedef int Length;
struct Data {
    int length;
    char buffer[MAX];
};

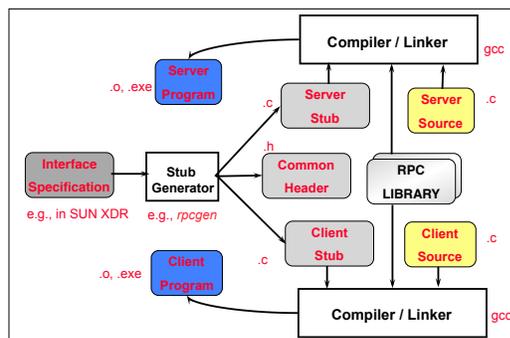
struct readargs {
    FileIdentifier f;
    FilePointer position;
    Length length;
};

program FILEREADWRITE {
    version VERSION {
        void WRITE(writeargs)=1;
        Data READ(readargs)=2;
    }=2;
} = 9999;
    
```

CSE 486/586, Spring 2013

16

Stub Generation



CSE 486/586, Spring 2013

17

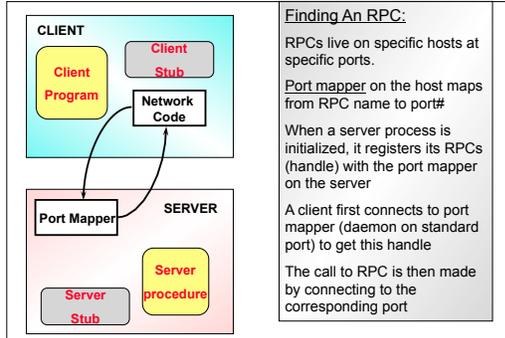
How Do You Find the Server Process?

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

CSE 486/586, Spring 2013

18

Local DB with Well-Known Port



CSE 486/586, Spring 2013

19

How to Pass Parameters?

- 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

CSE 486/586, Spring 2013

20

External Data Representation

- Communication between two heterogeneous machines
 - Different byte ordering (big-endian & 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

CSE 486/586, Spring 2013

21

Example: Google Protocol Buffers

- Goal: 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.
 - The 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;
  }
}
```

CSE 486/586, Spring 2013

22

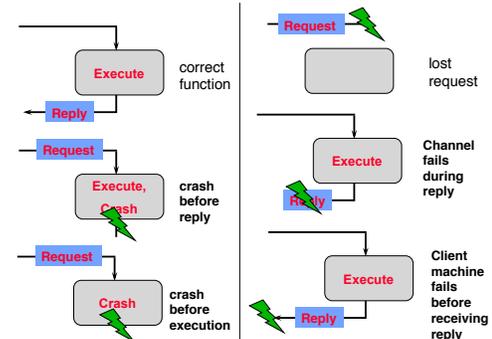
What About Failures?

- Local calls do not fail.
- Remote calls might fail.
- Programmers should deal with this.
 - No transparency here

CSE 486/586, Spring 2013

23

Failure Modes of RPC



CSE 486/586, Spring 2013

24

Invocation Semantics

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

CSE 486/586, Spring 2013

25

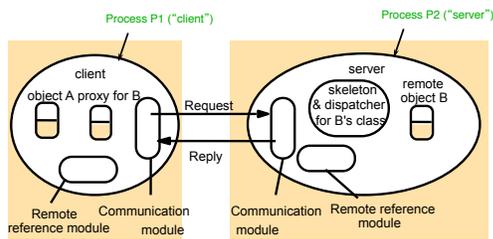
Invocation Semantics

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

CSE 486/586, Spring 2013

26

Remote Method Invocation (RMI)



CSE 486/586, Spring 2013

27

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

CSE 486/586, Spring 2013

28

Acknowledgements

- These slides contain material developed and copyrighted by Indranil Gupta (UIUC).

CSE 486/586, Spring 2013

29