CS 550:

Advanced Operating Systems

Remote Procedure Call

Ioan Raicu
Computer Science Department
Illinois Institute of Technology

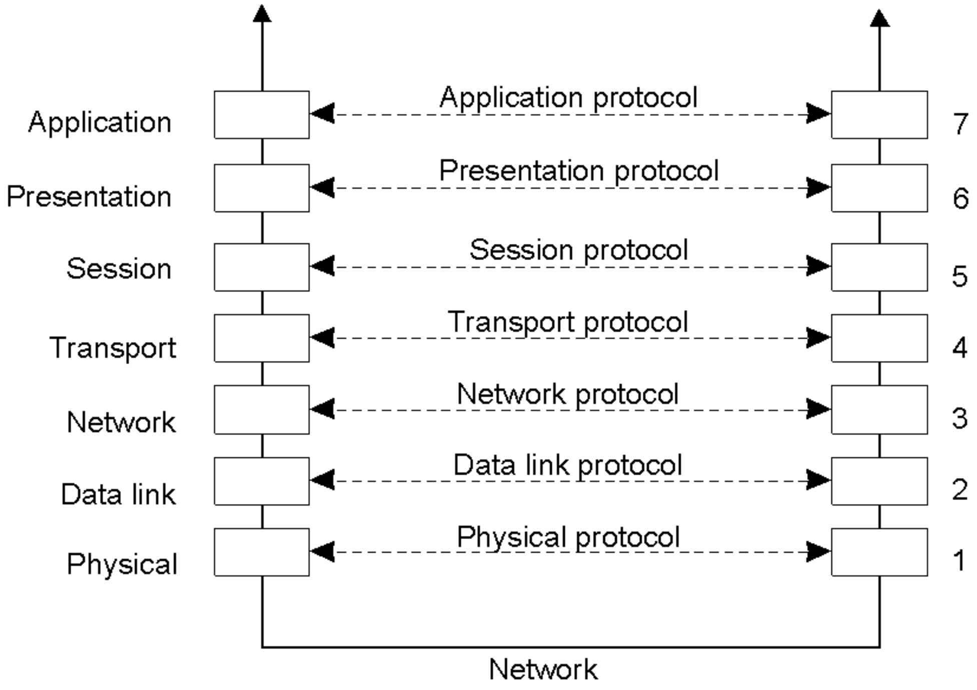
CS 550
Advanced Operating Systems
February 1st, 2011

Outline

- Layered protocols
- Remote Procedure Call (RPC)
- Issues:
  - – Parameter passing
  - – Binding
  - – Failure handling
  - – Performance and implementation issues

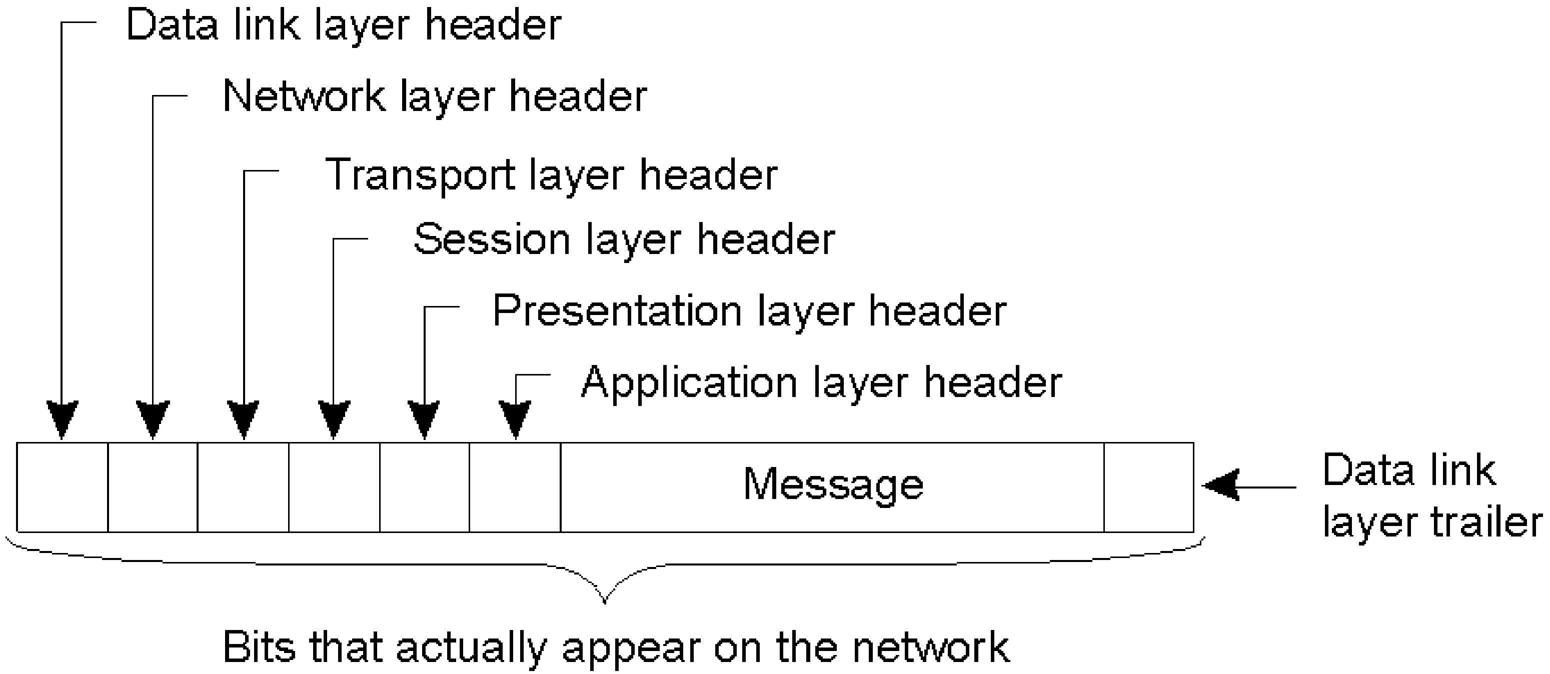
Communication Protocols

- Protocols are agreements/rules on communication
- Protocols could be connection-oriented or connectionless



Layered Protocols

- A typical message as it appears on the network.



Physical Layer

- Goal: ?
- Sample Issues:
  - how to encode a 0 Vs. 1?
  - what voltage should be used?
  - how long does a bit need to be signaled?
  - what does the cable, plug, antenna, etc. look like?
- Examples:
  - modems
  - “knock once for yes, twice for no”
  - X.21

Data Link Layer

- Goal: ?
- Sample Issues:
  - how big is a frame? (framing)
  - can I detect an error in sending the frame? (error control)
  - what demarks the end of the frame?
  - how to control access to a shared channel? (flow control)
- Examples:
  - Ethernet framing, Serial line IP (SLIP), Point-to-point protocol (PPP)

Network Layer

- Goal: ?
- Sample Issues:
  - how to route packets that have to travel several hops?
  - Congestion control algorithm: traffic shaping, flow specifications, and bandwidth reservation
  - accounting - charge for use of the network
  - fragment or combine packets depending on rules of link layer
- Examples:
  - IP

Transport Layer

- Goal: ?
- Sample Issues:
  - how to order messages and detect duplicates
  - error detection (corrupt packets) and retransmission
  - connectionless or connection-oriented
- Examples:
  - TCP (transmission control protocol)
  - UDP (universal datagram protocol)

Session and Presentation Layer

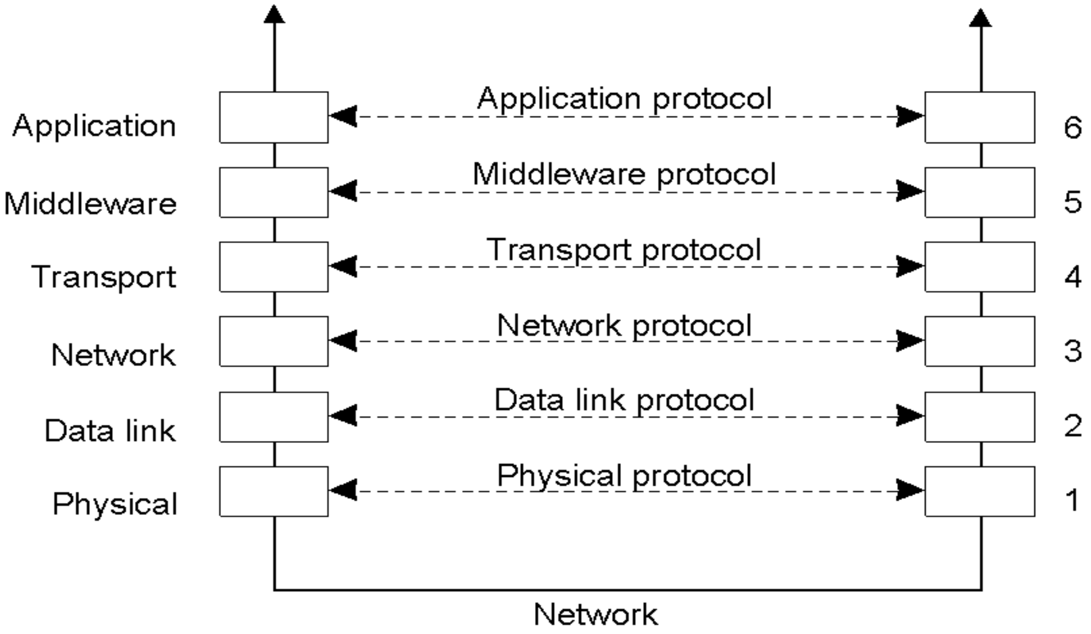
- Goal: ?
- Sample Issues:
  - Session layer: Allows users on different machines to establish sessions between them; Provides dialog control, to keep track of which party is currently talking, and synchronize
  - Presentation layer: Encodes data in a standard agreed upon way; Allows users to insert checkpoints into long transfer
- Examples:
  - eXternal Data Representation (XDR)

Application Layer

- Goal: ?
- Sample Issues:
  - when sending email, what demarks the subject field
  - how to represent cursor movement in a terminal
- Examples:
  - Simple Mail Transport Protocol (SMTP), File Transfer Protocol (FTP), Hyper-Text Transport Protocol (HTTP), Simple Network Management Protocol (SNMP), Network File System (NFS), Network Time Protocol (NTP), Net News Transport Protocol (NNTP), X (X Window Protocol)

Middleware Protocols

- Middleware:
  - An application that logically lives in the application layer
  - Contains many general-purpose protocols that warrant their own layers



Remote Procedure Call (RPC)

- Client-Server provides a mechanism for services in distributed systems BUT
  - requires explicit communication (send-receive)
- Q: How do me make “distributed computing look like traditional (centralized) computing”?
- Can we use procedure calls?

RPC

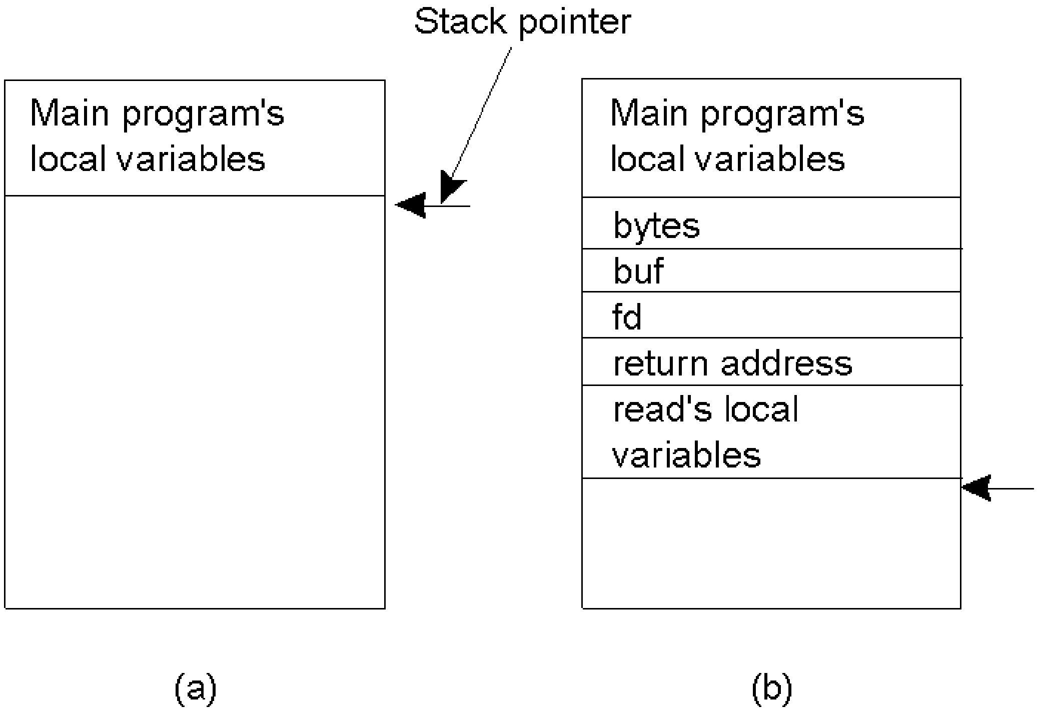
- In Distributed systems: the callee may be on a different system
  - Remote Procedure Call (RPC)
  - NO EXPLICIT MESSAGE PASSING
- Goal: Make RPC look like local procedure call

Design Issues

- Parameter passing
- Binding
- Reliability/How to handle failures
  - messages losses
  - client crash
  - server crash
- Performance and implementation issues
- Exception handling
- Interface definition

Conventional Procedure Call

count=read(fd,buf,nbytes);



a) Parameter passing in a local procedure call: the stack before the call to read

b) The stack while the called procedure is active

Observations

- Parameters (in C):
  - call-by-reference OR call-by-value
- Value parameter (e.g., fd, nbytes)
- Reference parameter (array buf)
- Many options are language dependent

Parameter Passing

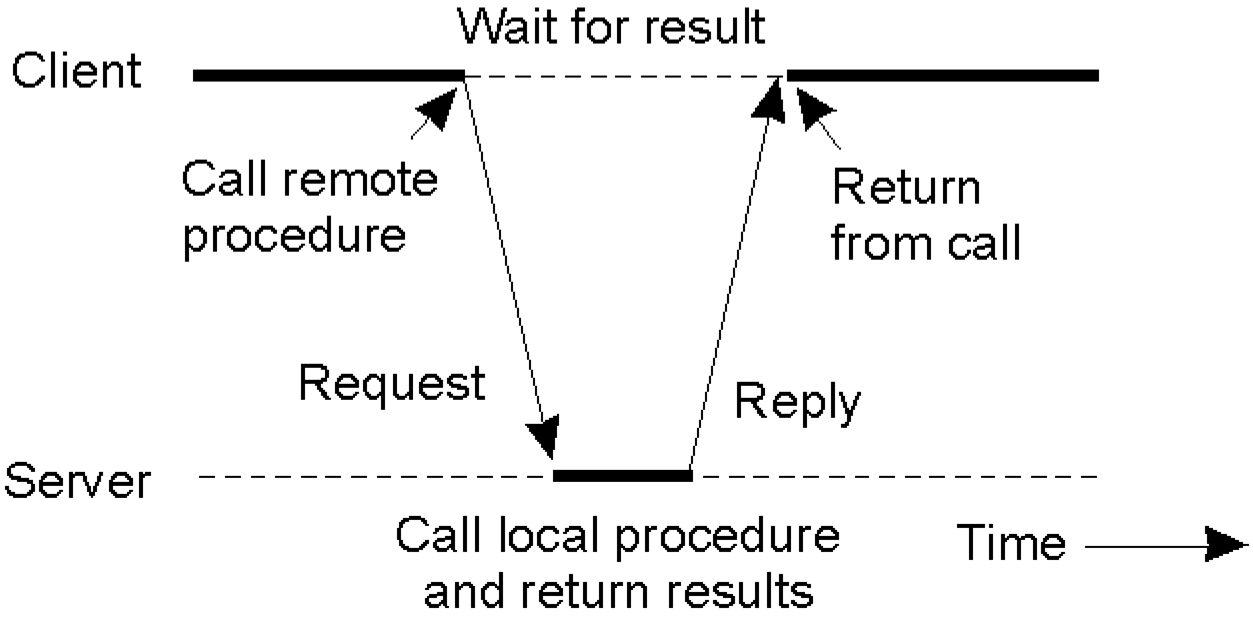
- Local procedure parameter passing
  - Call-by-value
  - Call-by-reference
- Remote procedure calls simulate this through:
  - Stubs – proxies
  - Marshaling
- How about global variables?

Stubs

- Client makes procedure call (just like a local procedure call) to the client stub
- Server is written as a standard procedure
- Stubs take care of packaging arguments and sending messages
- Packaging is called *marshaling*
- Stub compiler generates stub automatically from specs in an Interface Definition Language (IDL)

Client and Server Stubs

- Principle of RPC between a client and server program.

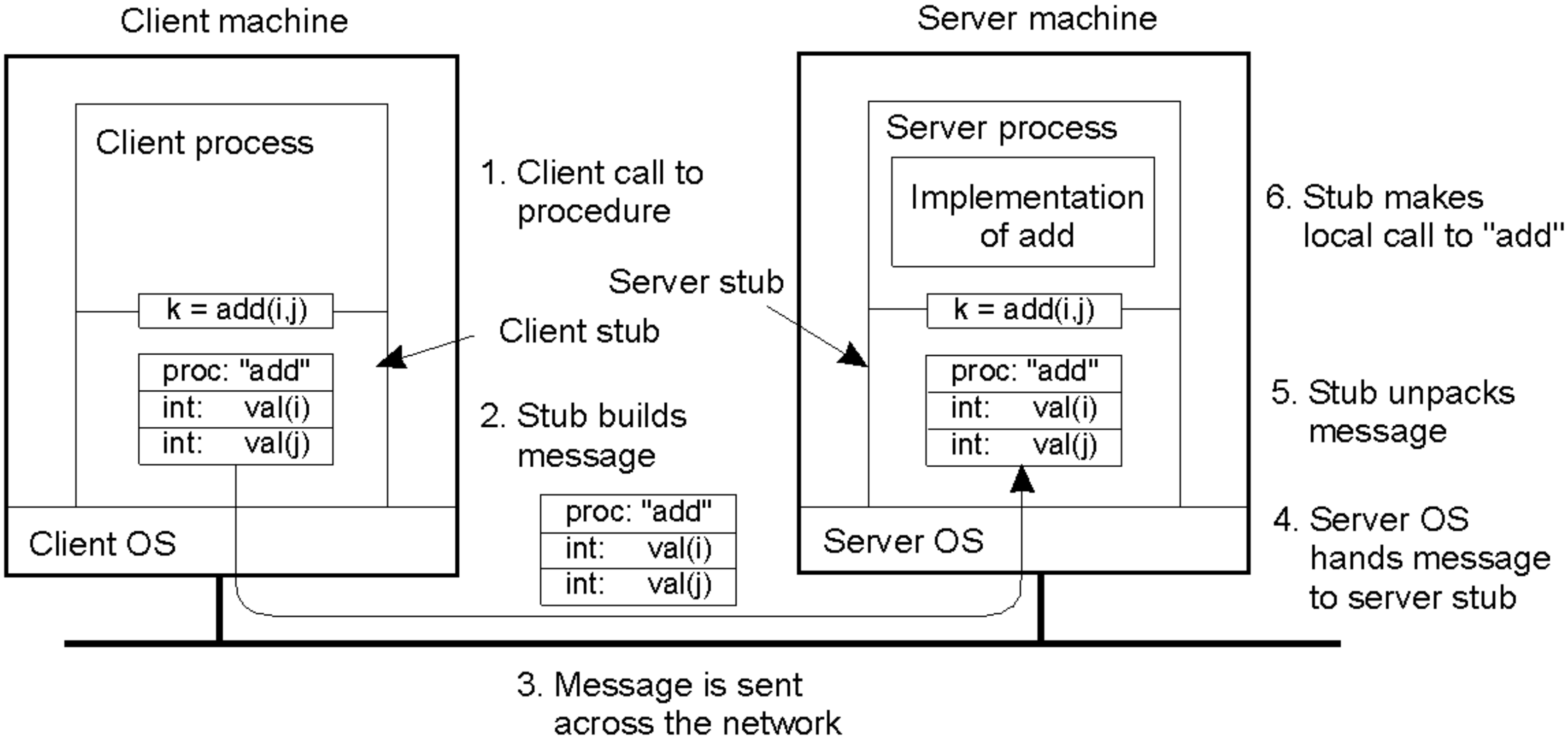


Steps of a Remote Procedure Call

1. Client procedure calls client stub in normal way
2. Client stub builds message, calls local OS
3. Client's OS sends message to remote OS
4. Remote OS gives message to server stub
5. Server stub unpacks parameters, calls server
6. Server does work, returns result to the stub
7. Server stub packs it in message, calls local OS
8. Server's OS sends message to client's OS
9. Client's OS gives message to client stub
10. Stub unpacks result, returns to client

Marshalling: Value Parameters

- Steps involved in doing remote computation through RPC



Marshalling: Value Parameters

- Problem: different machines have different data formats
  - Intel: little endian, SPARC: big endian
- Solution: ?

Marshalling: Reference Parameters

- Problem: how do we pass pointers?
  - If it points to a well-defined data structure, ?
  - What about data structures containing pointers?

- Marshalling: transform parameters/results into a byte stream

Binding

- Problem: how does a client locate a server?
- Binder can be a bottleneck
- Binder can do load balancing

Failure Semantics

- Client unable to locate server: return error
- Lost request messages: simple timeout mechanisms
- Lost replies: timeout mechanisms
  - Make operation idempotent
  - Use sequence numbers, mark retransmissions
- Server failures: did failure occur before or after operation?
  - At least once semantics (SUNRPC)
  - At most once
  - No guarantee
  - Exactly once: desirable but difficult to achieve
- Client failures

Client Cannot Locate Server

- Reasons:
  - –server may be down
  - –new version of server but older client
- Solutions
  - –respond with error type “cannot locate server”
  - –raise exception

Lost Request Message

- Time Out
  - –Kernel starts timer when request sent
  - –If timer expires, resend message
  - –If message was lost - server cannot tell the difference
  - –If message lost too many times ==> “cannot locate server”

Lost Reply Message

- More difficult to handle
- Rely on timer again?
- Problem: Client’s kernel doesn't know why no answer!
- Must distinguish between
  - –request/reply got lost?
  - –server slow
- Why?

Server Crashes

- When server crashes?
  - – After execution
  - – After receiving message but BEFORE execution
- Solutions:
  - – Wait until server reboots (or rebind)
  - – Give up immediately and report failure
  - – Guarantee nothing
  - – “exactly once semantics”

Client Crashes

- Client sends a request and crashes
  - Computation active - but no parent active
  - unwanted computation called “orphan”
- Orphan’s can create problems:?
- Solutions:
  - *Extermination*:
  - *Reincarnation*:
  - *Gentle reincarnation*:
  - *Expiration*:

Questions

?