



Naming and TCP Applications

Lecture 37

Klara Nahrstedt

CS241 Administrative

- Read Stallings Chapter 13, 14.4. R&R 18.8 and 19.1-3
- LMP3 (Part II) Due April 30
- Last Regular Quiz will be on Friday, April 27 on Networking
- Homework 2 posted – Due Wednesday, May 2, 4pm
- Monday, April 30
 - We will have LMP3 Quiz
 - We will have guest lecture – speaker from Mathematica talking about challenges of Mathematica Development on Different OS Platforms (UNIX, MAC, Linux, Windows).
- Wednesday, May 2 – In Class Review Session for Final Exam

Host Names and IP Addresses

Internet Users use Host Names

`csil-linux1.cs.uiuc.edu`

`csil-linux3.cs.uiuc.edu`

Host names must be mapped to numeric network addresses for most of the network library calls

System admins define mechanisms by which names are translated into network addresses

Host names are stored in ASCII strings

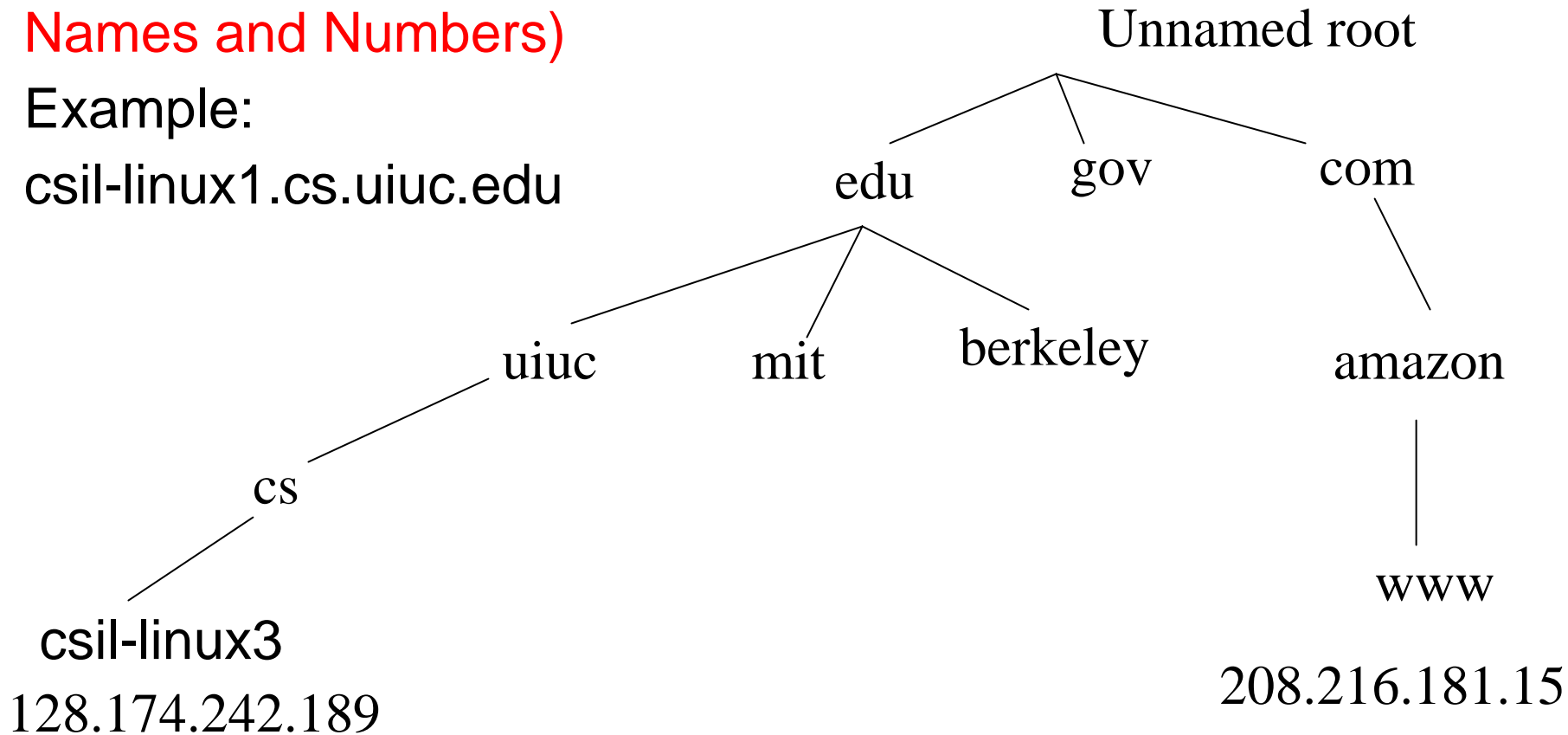
Internet Domain Names

Set of Domain names forms a hierarchy - tree

ICANN (Internet Corporation for Assigned Names and Numbers)

Example:

csil-linux1.cs.uiuc.edu



Domain Naming System (DNS)

Until 1988 Internet mapping between host names and IP addresses was maintained manually in a single text file – HOSTS.TXT

Currently, the mapping has been maintained in a distributed world-wide database known as DNS

Conceptually, DNS database consists of millions of host entry structures (of type 'hostent')

IP Address

IP addresses are specified

- **binary** in network byte order in **s_addr** field of **struct in_addr**

- **human readable form** – dotted-decimal notation or Internet address dot notation

129.115.30.129 (IP address of usp.cs.utsa.edu)

IPv4 address – 32 bits (4Bytes long)

IPv6 address – 128 bits (16Bytes long)

in_addr_t inet_addr(const char *cp)

- Function converts a dotted-decimal notation address to binary in network byte order

char *inet_ntoa(const struct in_addr_in)

- Function converts binary form into dotted-decimal notation

Conversion of Host Name to IP Address

Traditional way of converting host name to a binary address is

```
#include <netdb.h>  
struct hostent *gethostbyname(const char *name)  
  
struct hostent {  
    char *h_name; /*canonical name of host */  
    char ** h_aliases; /*alias list */  
    int h_addrtype; /* host address type*/  
    int h_length; /*length of address */  
    char **h_addr_list; /*list of addresses */  
};
```

Example

```
char *hostn = "usp.cs.utsa.edu";
struct hostent *hp;
struct sockaddr_in server;

if ((hp = gethostbyname(hostn)) == NULL
    fprintf(stderr, "Failed to resolve host name \n");
else
    memcpy((char *)&server.sin_addr.s_addr,
           hp->h_addr_list[0],
           hp->h_length);
```

Internet application retrieves host entry from the DNS database associated with the domain name "hostn"

Conversion from Address to name

```
#include <netdb.h>
```

```
struct hostent *gethostbyaddr (const void *addr,  
                               socklen_t len, int type);
```

For IPv4 – type = AF_INET

Internet Application retrieves host entry associated with the IP address “addr”.

Example

```
struct hostent *hp;
```

```
struct sockaddr_in net;
```

```
int sock;
```

```
if ((hp=gethostbyaddr(&net.sin_addr, 4,AF_INET))  
    printf("Host name is %s\n", hp->h_name);
```

Another approach to Conversion

Use new POSIX standard 2001

1. `getnameinfo` (instead of `gethostbyname`)
2. `getaddrinfo` (instead of `gethostbyaddr`)

These routines do not use static data

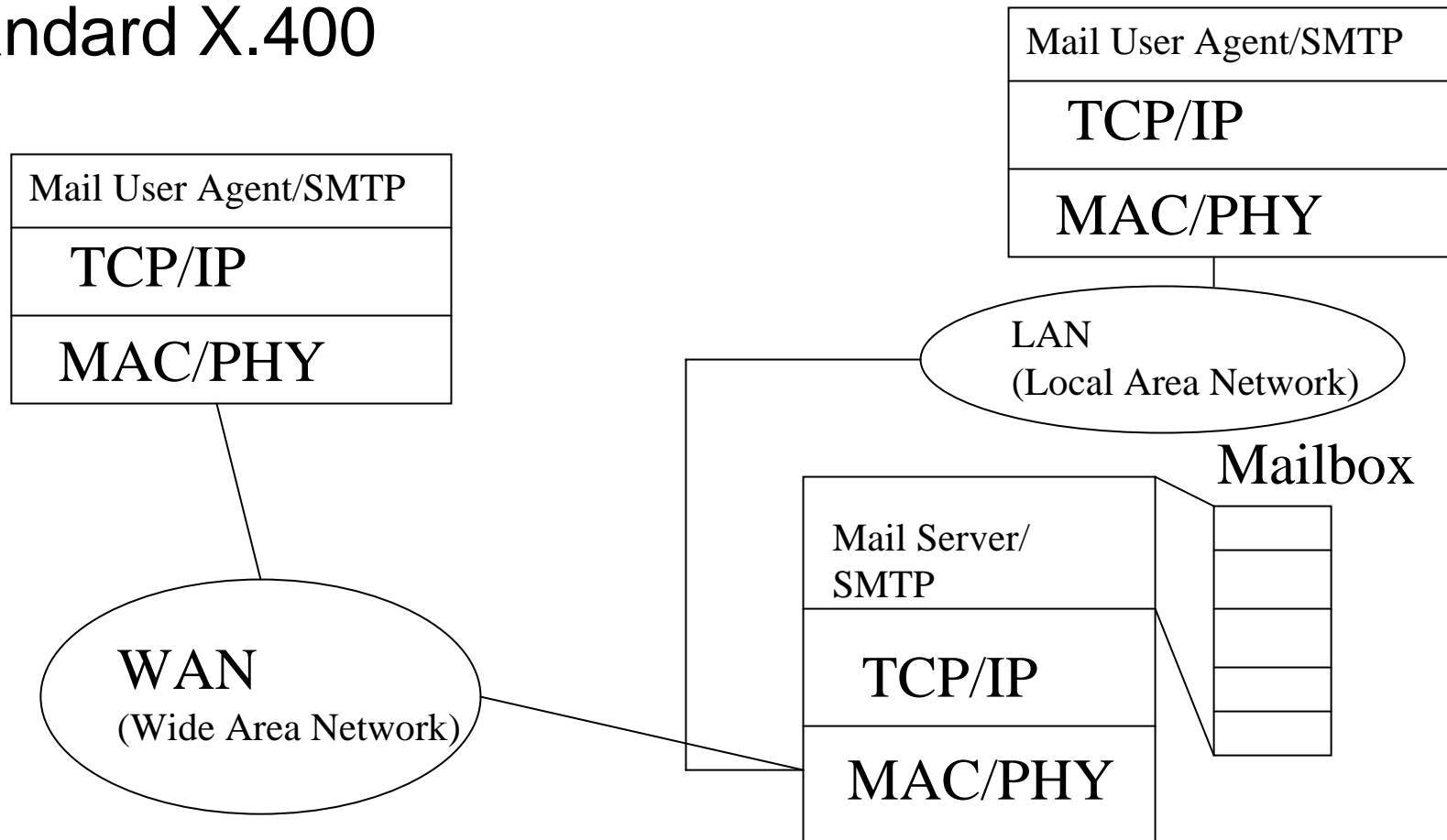
These functions are safe to use in a threaded environment

Problem: not available on many systems

TCP/IP Application – Electronic Mail

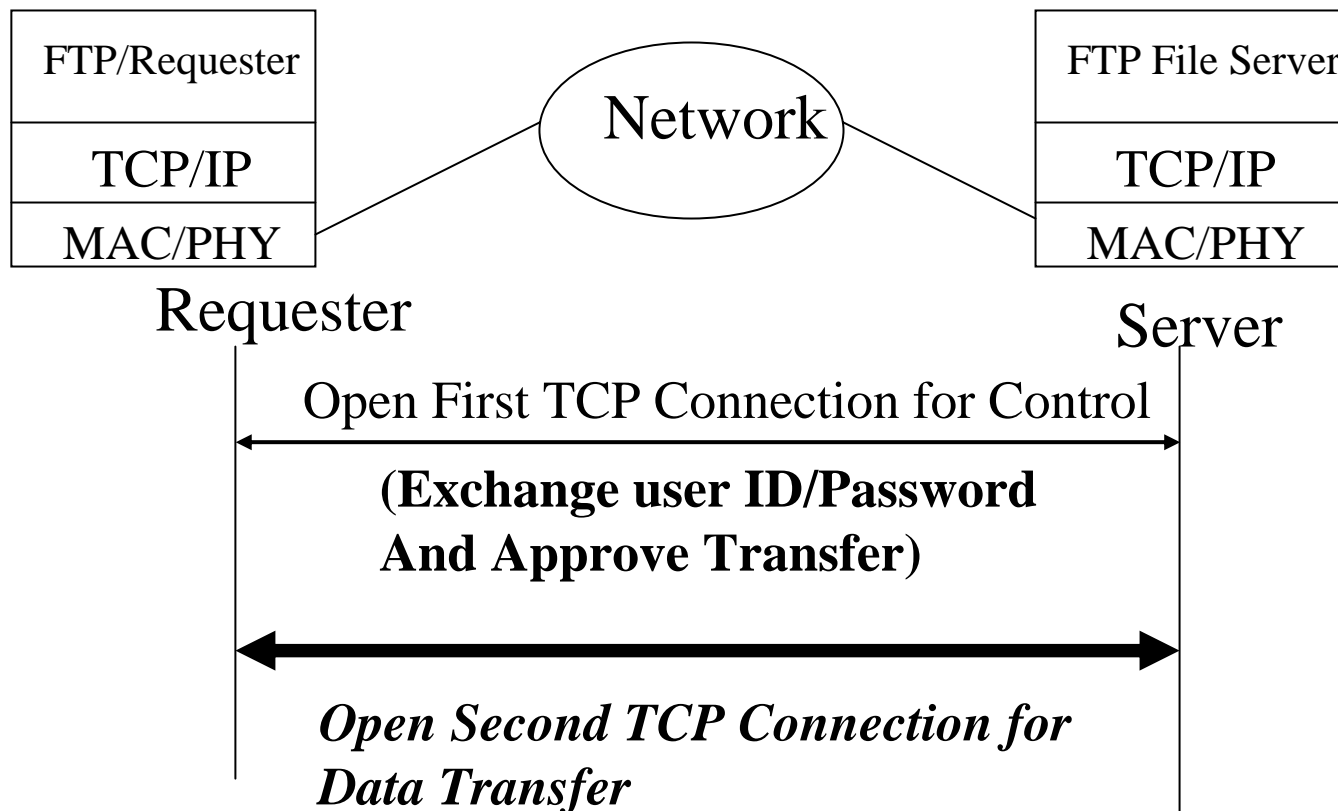
Simple Mail Transfer Protocol (SMTP)

Standard X.400



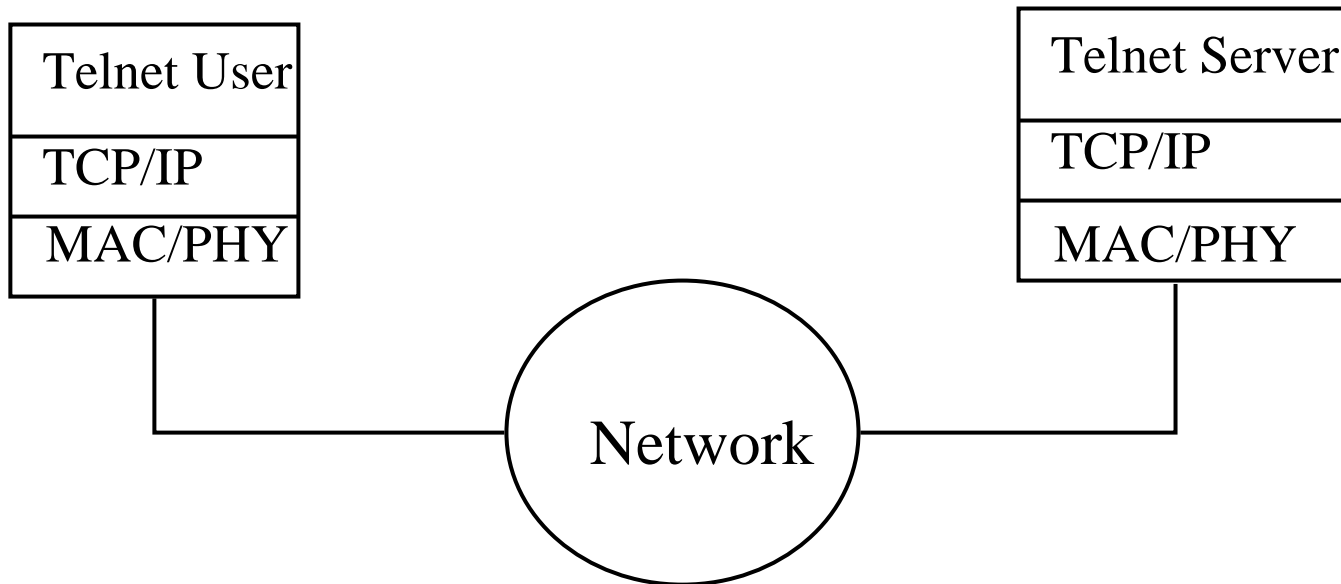
TCP/IP Application – File Transfer

FTP – File Transfer Protocol – sends files from one system to another under user command ('sftp' currently)



TCP Application - Telnet

Telnet – remote login capability which enables a user at a PC to login to a remote computer and function as if directly connected to that computer ('ssh' currently)



TCP Application – World Wide Web

Origin of WWW – 1989 WWW invented by Tim Berners-Lee - software engineer working at CERN – Swiss Physics Laboratory

1993 – Marc Andreessen and his colleagues at NCSA UIUC released a graphical browser, called MOSAIC for all three major platforms: UNIX, Windows, Macintosh

After release of MOSAIC, interest in Web exploded

2007 – 113,658,468 Web sites (according to www.netcraft.com)

Web Markup Language - HTML

What distinguishes Web services from conventional file retrieval such as FTP?

Main difference: Web content can be written in language – HTML (Hypertext Markup Language)

HTML Program – contains tags to tell the browser how to display text and graphic objects

` Make me bold! `

Real Power of HTML – Pointers – Hyperlinks

` CS
Department at UIUC `

Web Content (1)

Content is a sequence of bytes with an associated MIME type (Multipurpose Internet Mail Extensions)

Text/html (HTML page)

Text/plain (unformatted text)

Application/postscript (postscript document)

Image/gif (Binary image encoded in GIF Format)

Image/jpeg (Binary image encoded in JPEG Format)

Web Content (2)

Web servers provide content to clients:

1. Fetch a disk file – static content – serving static content
2. Run executable file and return output to client - dynamic content – serving dynamic output

Each file has a unique name known as URL (Universal Resource Locator)

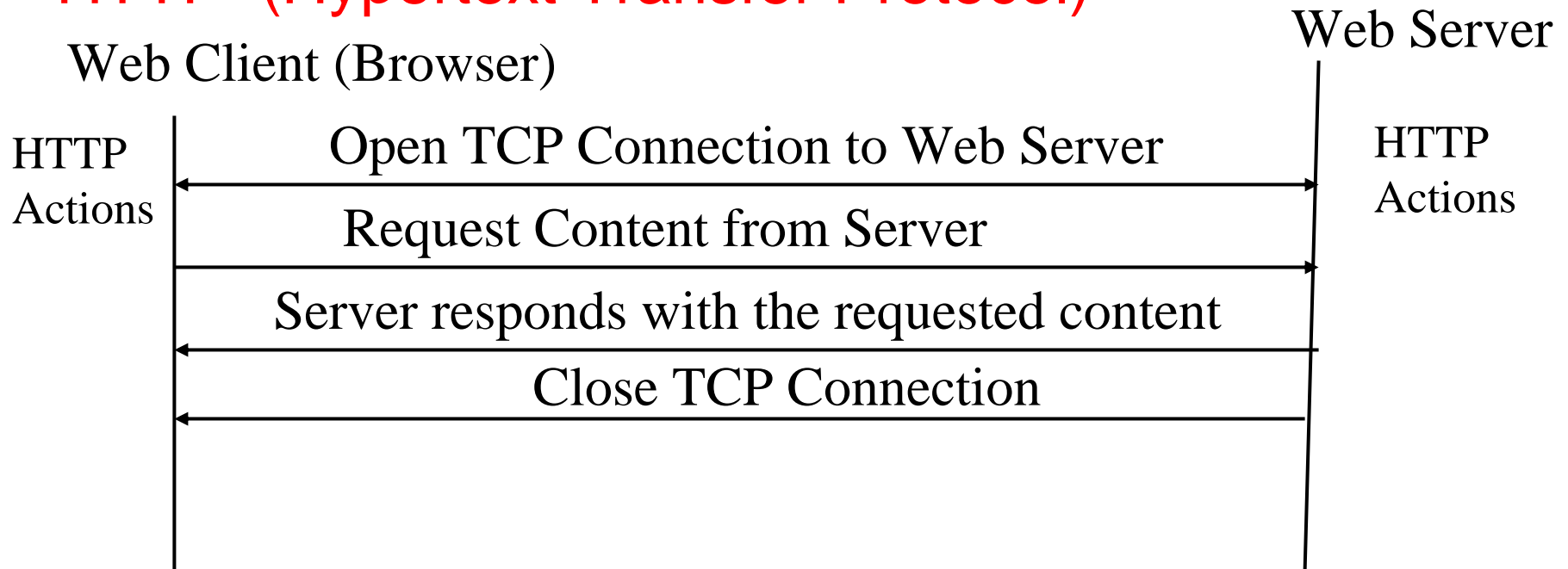
<http://www.aol.com:80/index.html>

Port 80 is a default to well-known HTTP port.

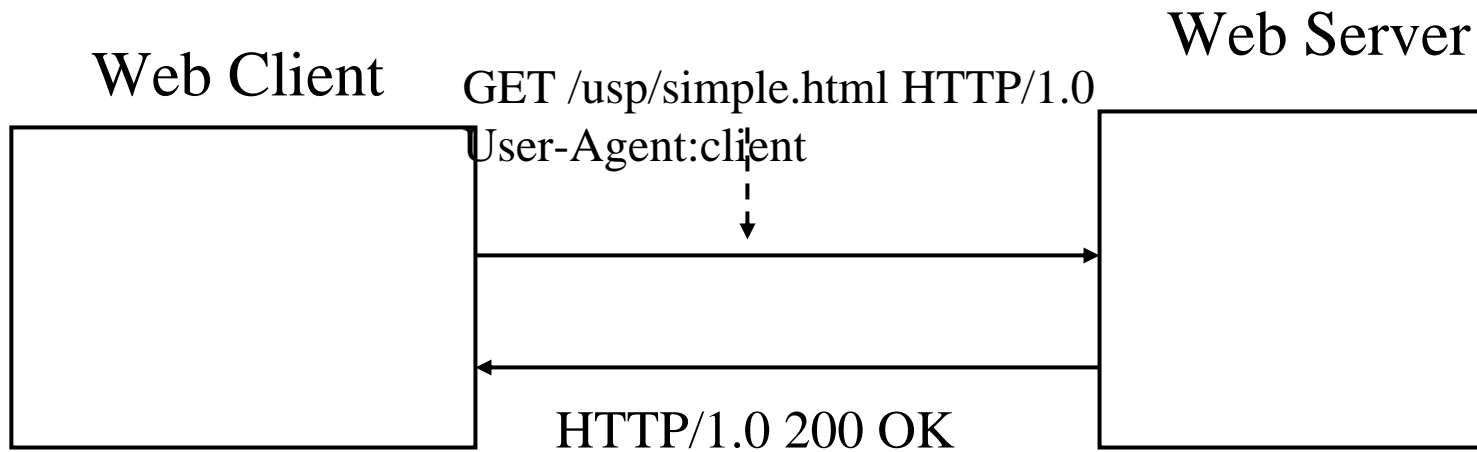
HTTP Web Protocol

Web Client and Web Servers interact using a text-based application-level protocol:

HTTP (Hypertext Transfer Protocol)



Client Requests/Server Response



Client:

Method <SP> Request-URI <SP> HTTP-Version <CRLF>

(Method could be GET, POST,...)

Header Line: Field-Name:Field Value <CRLF>

Server:

HTTP-Version <SP> Status-Code <SP> Reason Phrase <CRLF>²⁰

HTTP Message Exchange

Web Client

Web Server

Create communication
Endpoint bound to
A well-known port

Wait for client
Connection request

Accept connection
Read request

Write response

Close private channel

Three-way handshake)

Initiate TCP
Connection

GET

(status and resource)

Write request

Read response

Close channel

TCP/IP Application: Remote Procedure Call (RPC)

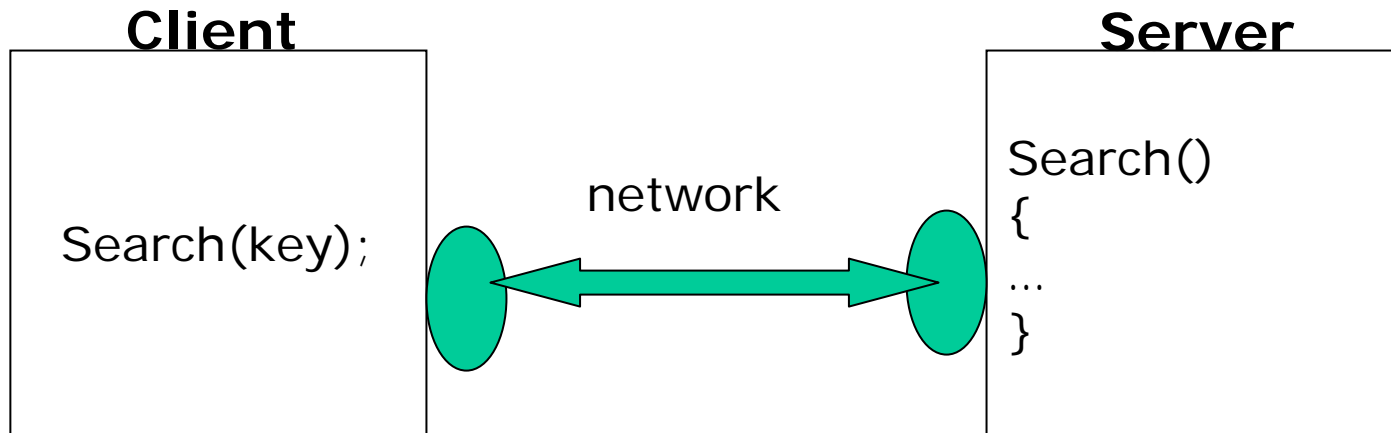
A high level communication protocol built on top of Socket to build higher level services

NFS, SMTP, Telnet

Look like a procedure call

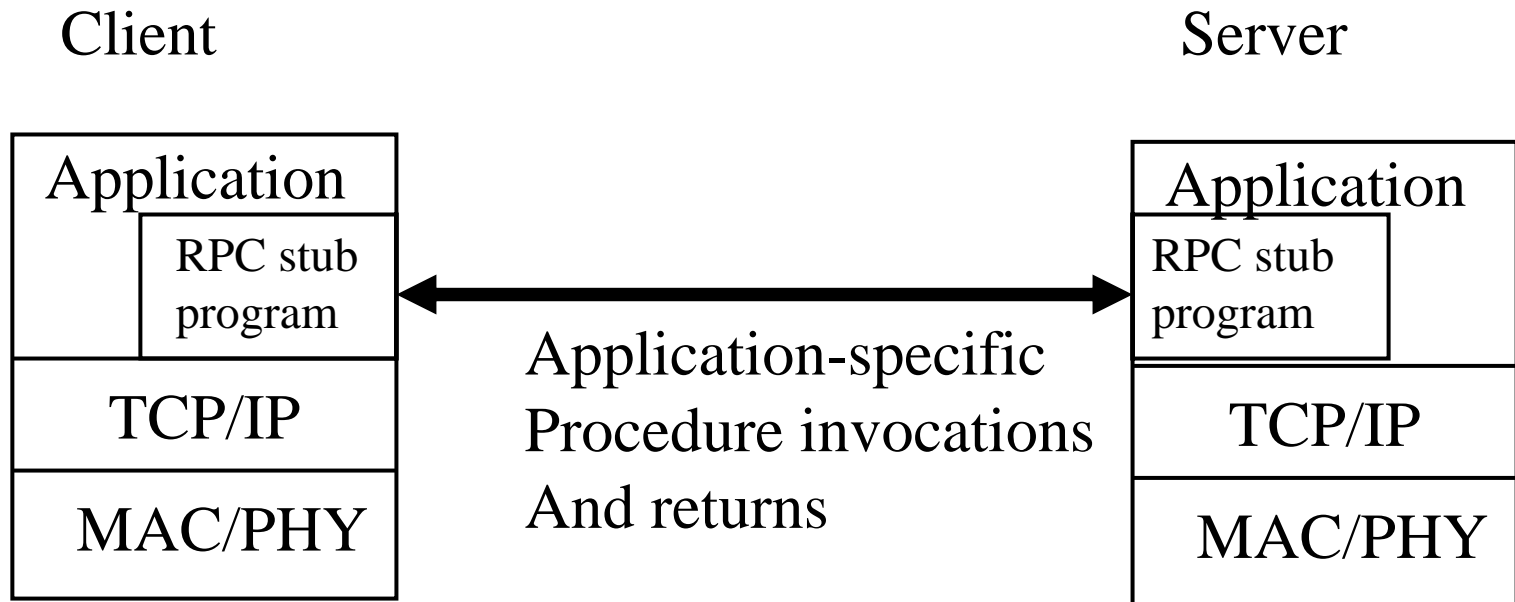
The client just calls the server function directly

The RPC library is responsible to pass parameters and results back and forth



RPC Mechanism

Viewed as a refinement of reliable, blocking message passing



Calling Program makes a normal procedure call $CALL P(X, Y)$ where P – procedure name, X – passed parameters, Y – Returned values;

RPC Implementation

Parameters of RPC are marshaled into message

Message sent and client waits

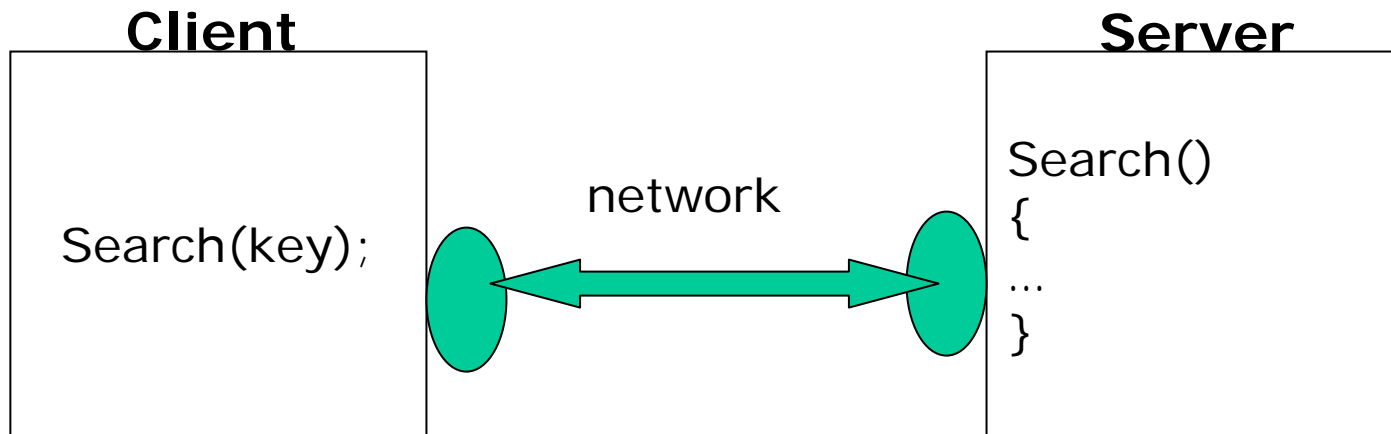
Remote machine receives message,

remote process assembles parameters and makes procedure call

remote process marshals results into message

message sent back to client

Client process resumes and unpacks result



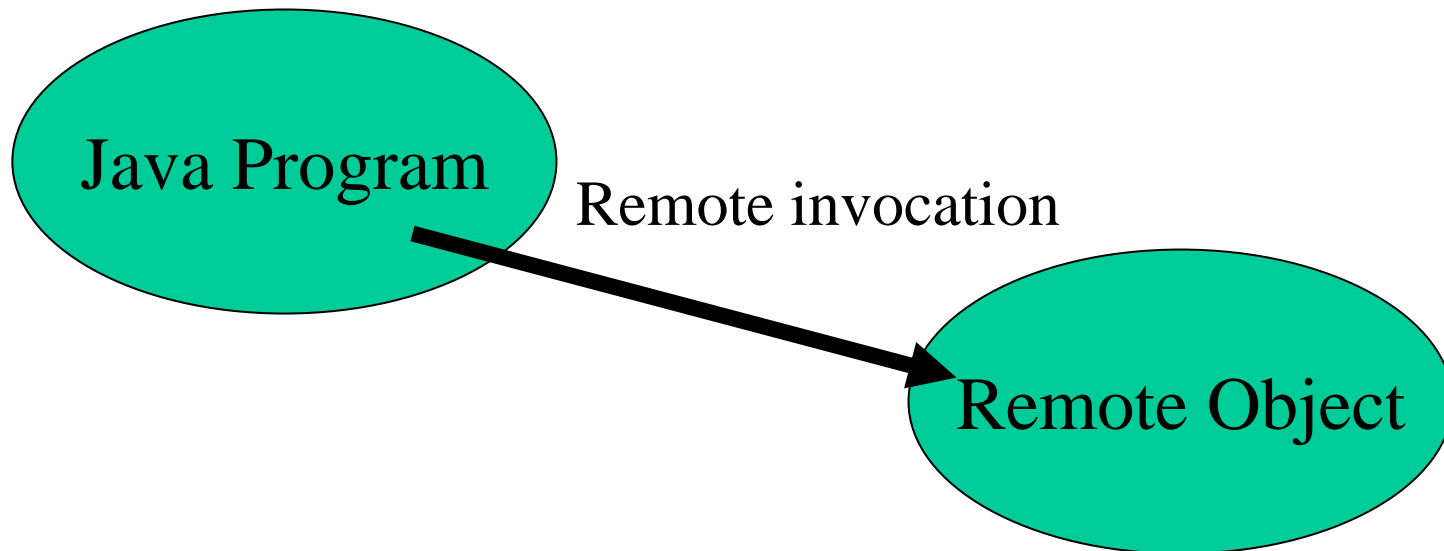
Good Analogy?

IP-phone?

Java Remote Method Invocation

RPC with objects

Includes objects as parameters



RMI

Client has stub for remote object – **proxy** for the remote object

Parcel - marshalling

Server has skeleton

Unmarshals parameters

Invokes method on remote machine

Summary

Host and IP Addressing

TCP Applications

Mail Service

FTP

Telnet

WWW

RPC