Computer Networks

Chapter 2. Network Architecture

Autumn 2017

GUO Xunhua

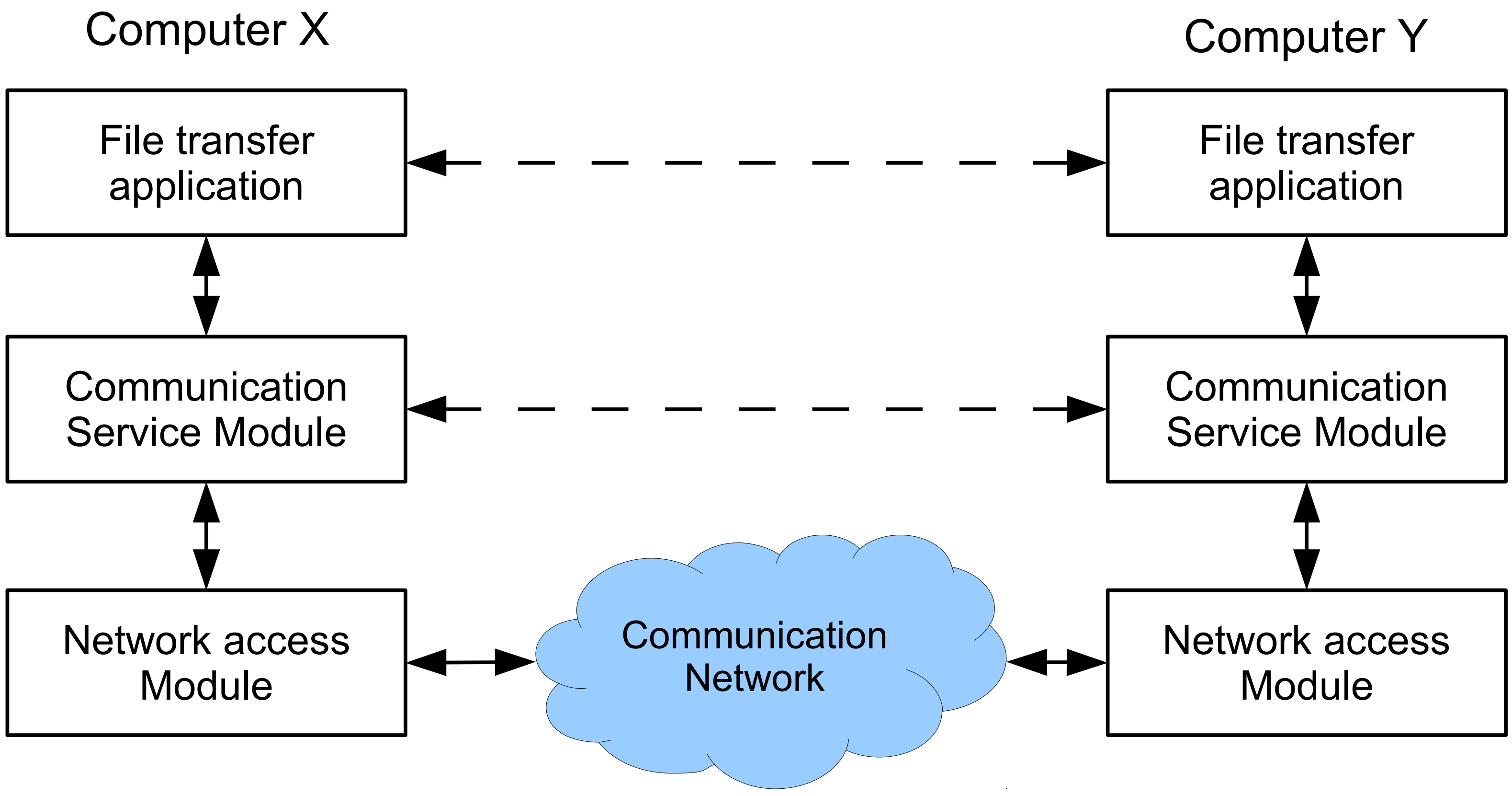
清华经管学院
Tsinghua SEM

Network architecture

- Layering and Protocols
- Architecture
- Application Programming Interface (Sockets)

Layering and Protocols

Layering

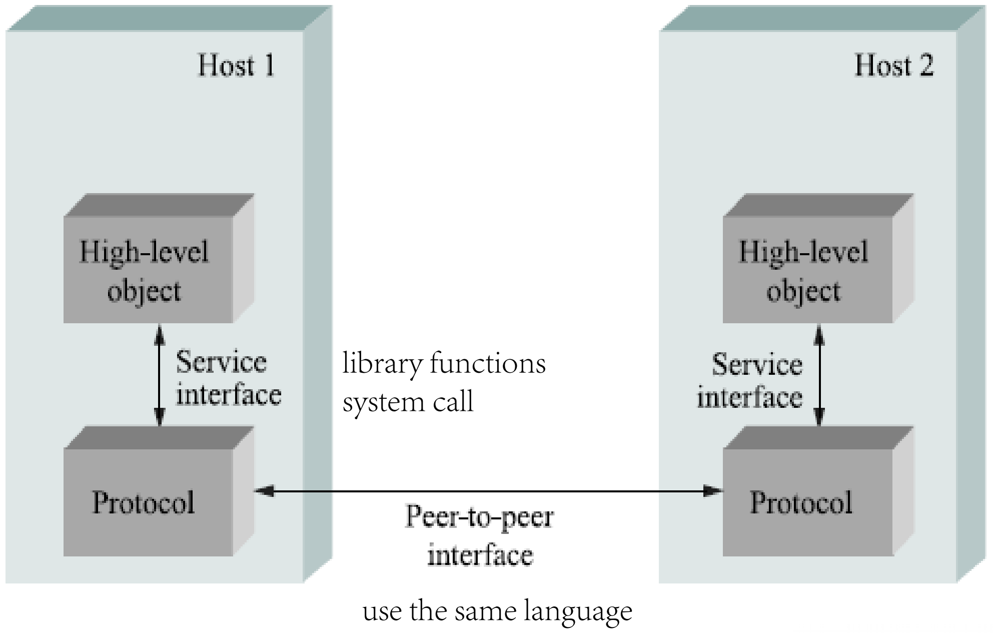

Computer X
Computer Y
File transfer application
File transfer application
Communication Service Module
Communication Service Module
Network access Module
Communication Network
Network access Module

Protocols (协议)

- Standards at a layer
- Term "protocol" is overloaded
  - specification of peer-to-peer interface (rules)
  - module that implements this interface
- Protocol stack: set of consecutive layers

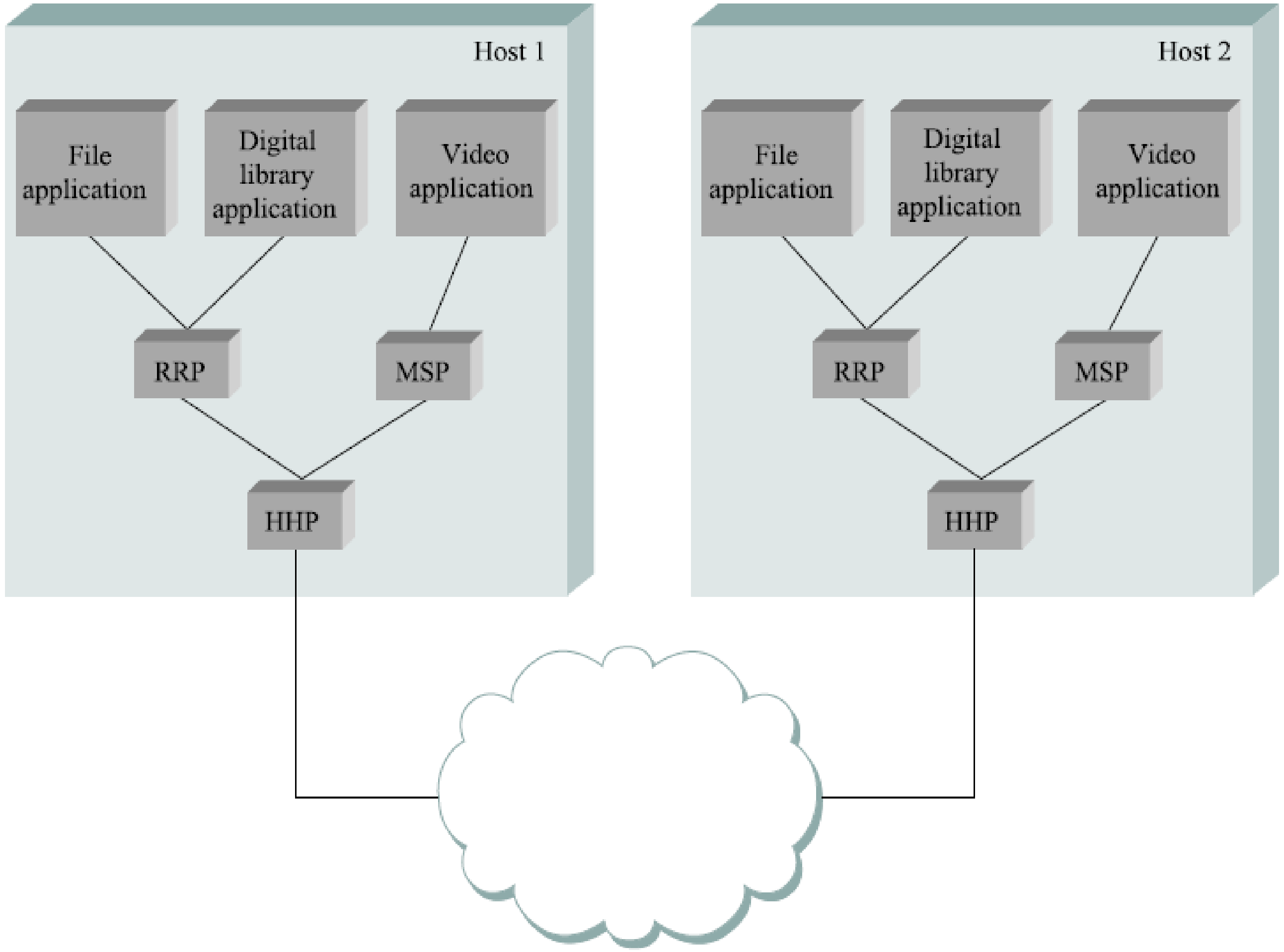
Interfaces(接口 / 界面)

- Each protocol object has two different interfaces
  - service interface: operations on this protocol
  - peer-to-peer interface: messages exchanged with peer



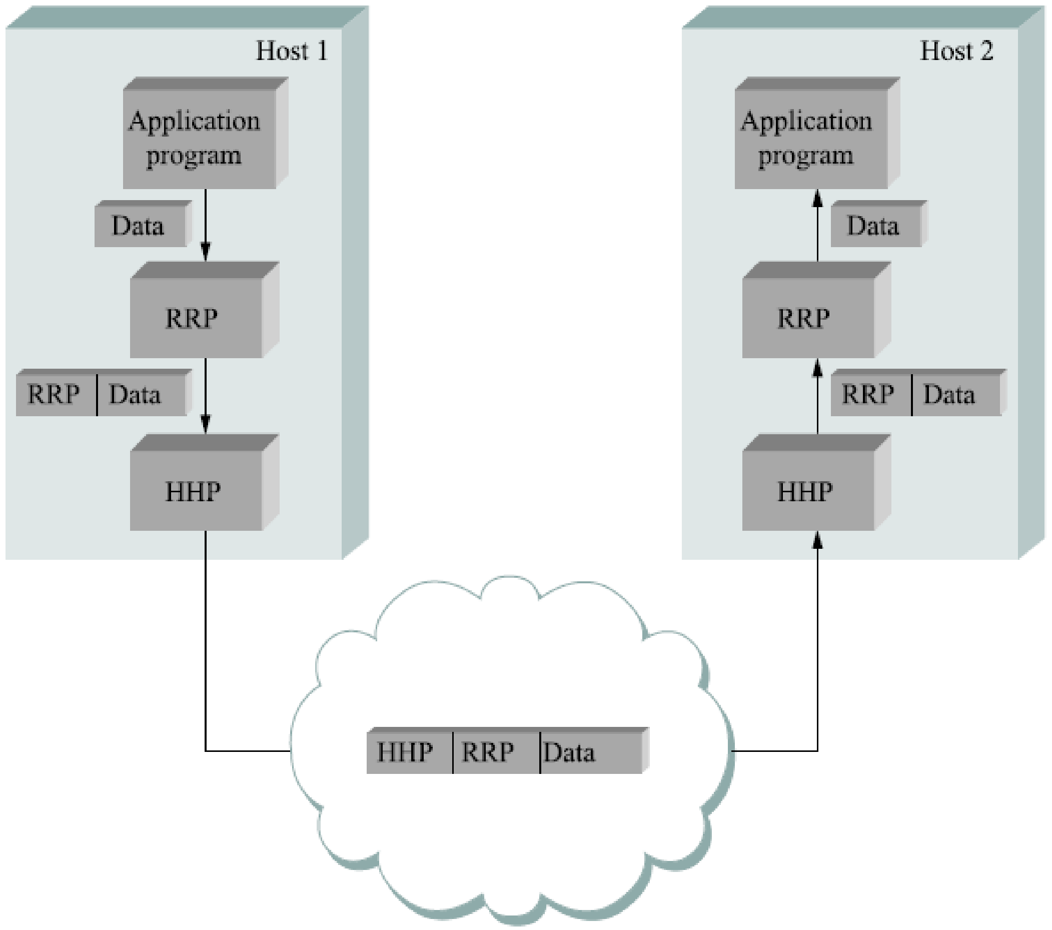
Protocol Graph

- most peer-to-peer communication is indirect
- peer-to-peer is direct only at hardware level



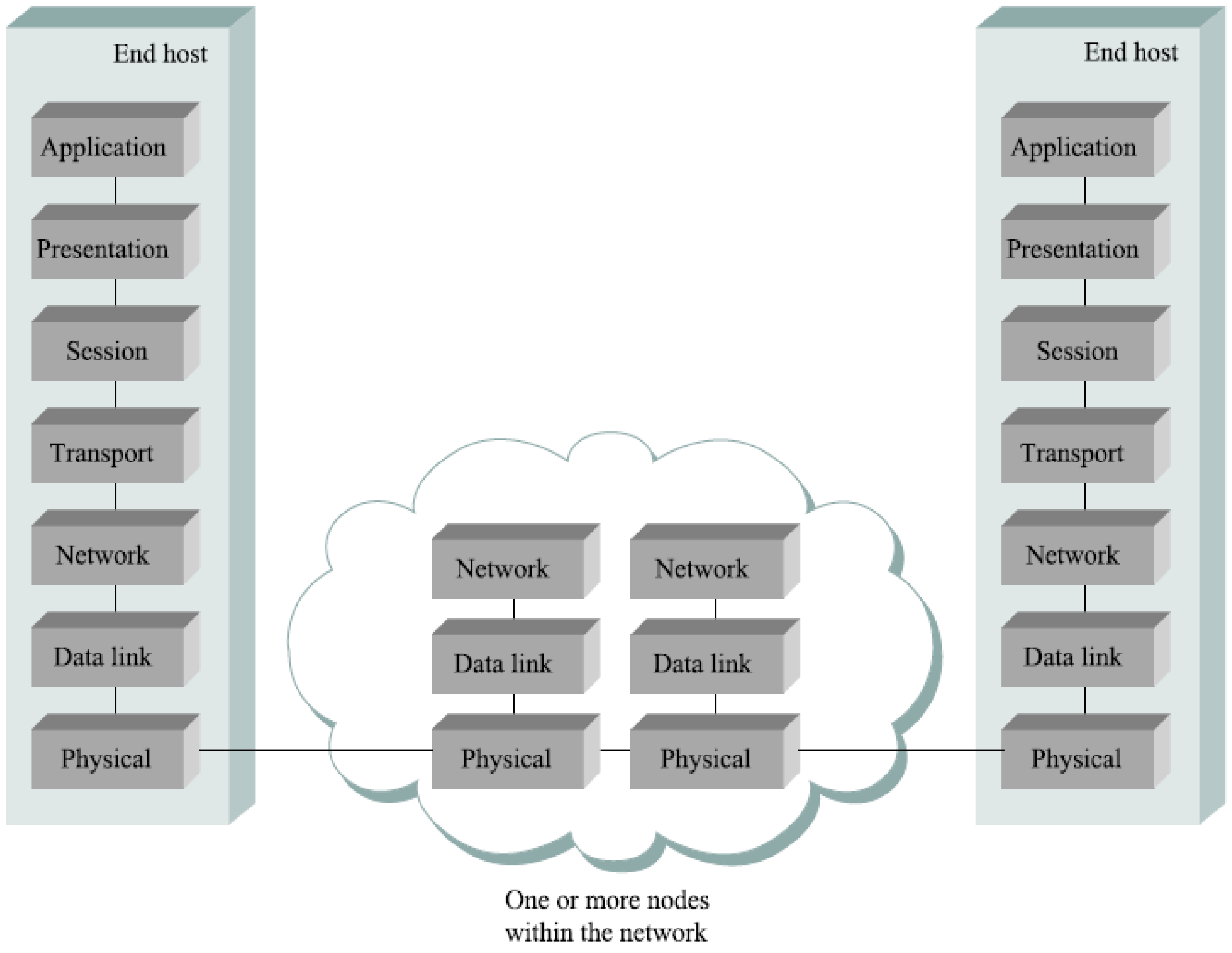
Encapsulation (封装)

- High-level messages are encapsulated inside of low-level messages



Architecture

ISO/OSI architecture



Lower three layers

- Physical layer
  - Handles the transmission of raw bits over a communication link.
- Data link layer
  - Collects a stream of bits into a larger aggregate called frame
  - Network adapter
- Network layer
  - Handles routing among nodes within a packet-switch network

Transport layer

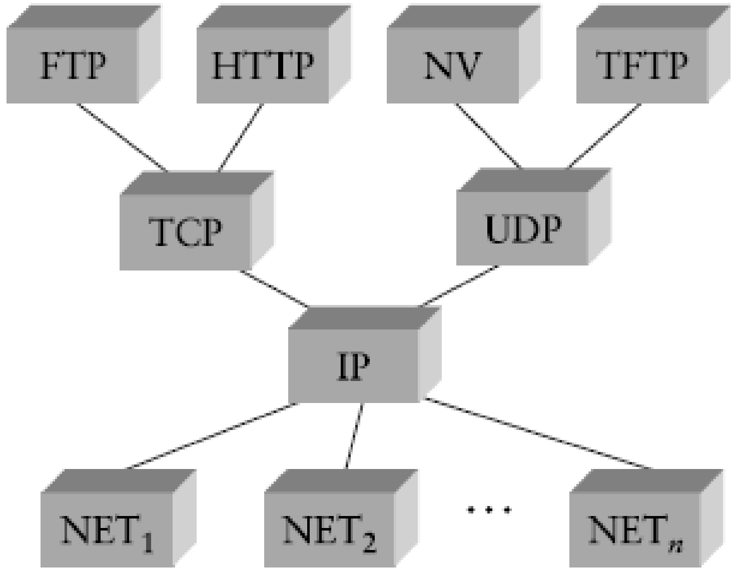
- Provides a reliable mechanism to transmit messages between two end-nodes through:
  - Message fragmentation into packets
  - Packets reassembly in original order
  - Retransmission of lost packets
  - End-to-end flow control
  - Congestion control

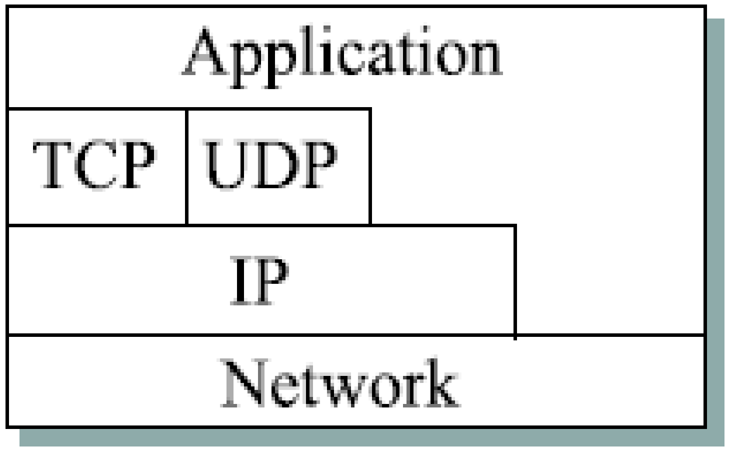
Top three layers

- Session layer
  - Handles the interaction between two end points in setting up a session
  - For example, managing an audio and video stream in a teleconferencing application
- Presentation layer
  - The format of the data exchanged
  - Data encryption, compression, code conversion
- Application layer
  - Examples: WWW, Email, Telnet

Internet (TCP/IP) architecture

- Defined by Internet Engineering Task Force (IETF)



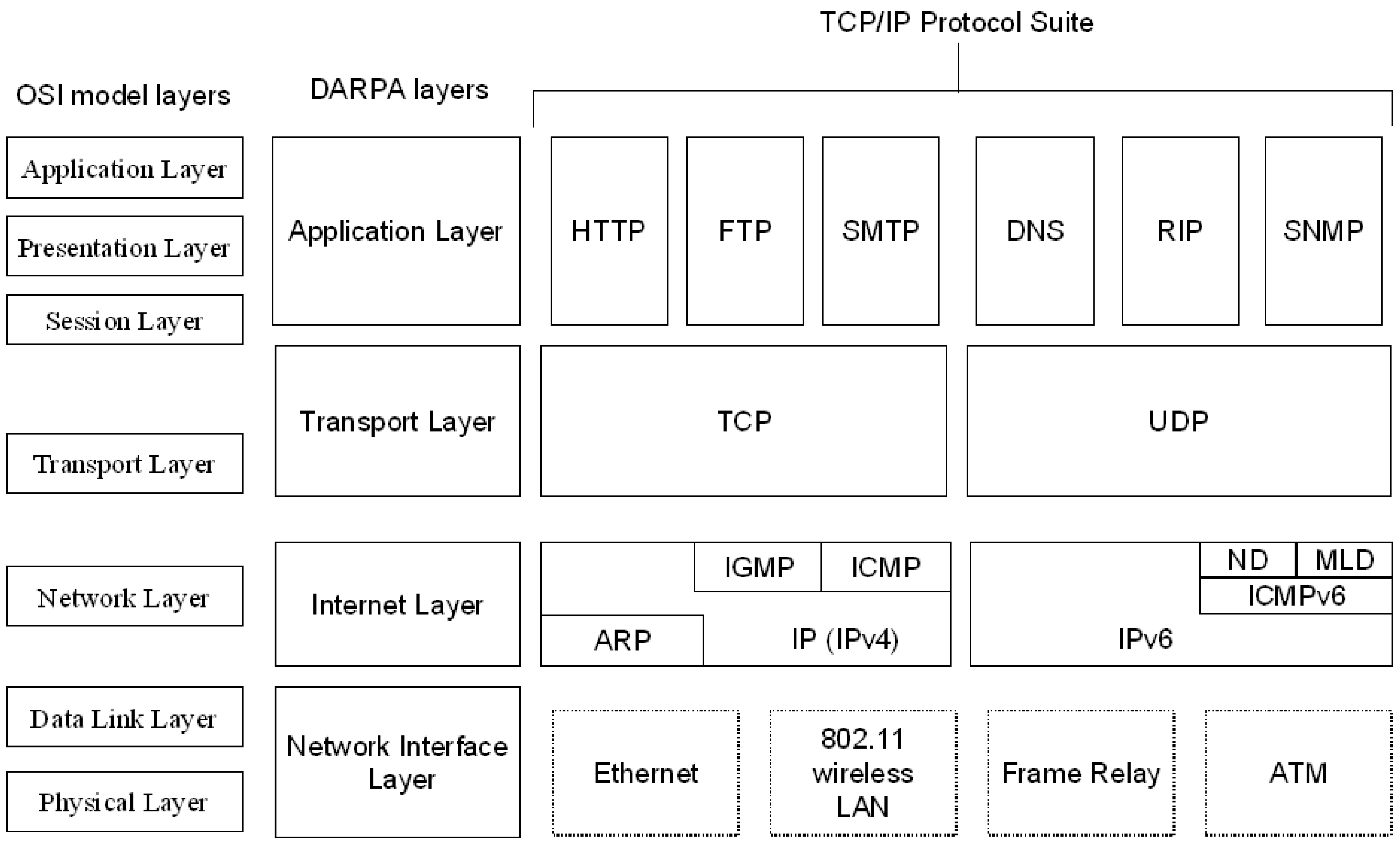


沙漏型

Features of the Internet architecture

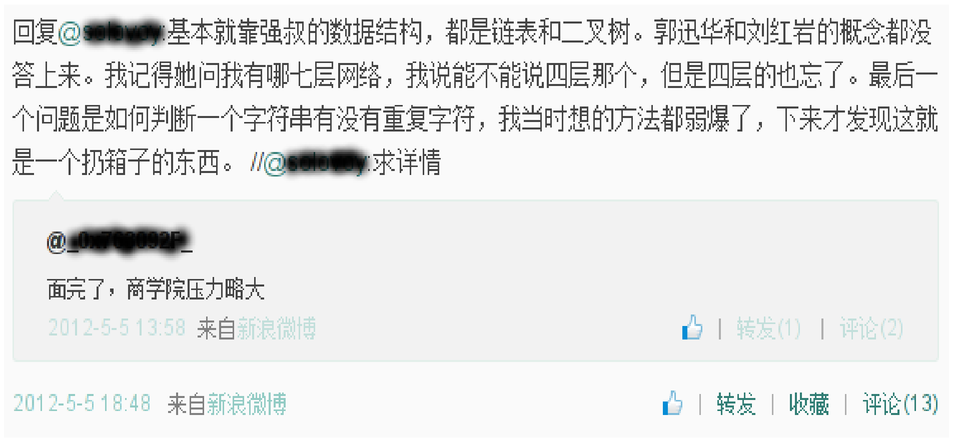
- The Internet architecture does not imply strict layering
- An hourglass shape—wide at the top, narrow in the middle, and wide at the bottom.
- in order for a new protocol to be officially included in the architecture, there needs to be both a protocol specification and at least one representative implementations of the specification.
  - *We reject kings, presidents, and voting. We believe in rough consensus and running code.* (Dave Clark)

OSI vs. TCP/IP



Source: http://www.microsoft.com

A Weibo



回复@[redacted]:基本就靠强叔的数据结构，都是链表和二叉树。郭迅华和刘红岩的概念都没答上来。我记得她问我有哪七层网络，我说能不能说四层那个，但是四层的也忘了。最后一个问题是如何判断一个字符串有没有重复字符，我当时想的方法都弱爆了，下来才发现这就是一个扔箱子的东西。 //@[redacted]:求详情

@[redacted]

面完了，商学院压力略大

2012-5-5 13:58 来自新浪微博 | 转发(1) | 评论(2)

2012-5-5 18:48 来自新浪微博 | 转发 | 收藏 | 评论(13)

- A student's job interview experience
  - *I was unable to answer the concepts about Guo Xunhua's course.*
  - *She asked me what are the seven layers in networks.*
  - *I said “can I just talk about the four-layer one?”*
  - *But eventually I also failed to recall what the four layers are.*

清华经管学院 Tsinghua SEM

Extended reading

- Wikipedia page
  - http://en.wikipedia.org/wiki/TCP/IP
- W3School TCP/IP Tutorial
  - http://www.w3schools.com/tcpip/
- “Wireshark Lab: Getting Started”
  - Can be downloaded from our discussion board.

Application Programming Interface (Sockets)

Common services

- A computer network provides more than packet delivery between nodes
  - We don’t want application developers to write each application from the packet level
- A network should also provide Common Services (公用服务)
  - High level networking services
  - Channel/pipe connecting two applications

How can we build network applications?

- Applications: inter-process communication
- Common services
  - Considering delivery guarantees, packet length, delay, security, reliability.
  - Provided in a layered structure.
- A network architecture is
  - A layered arrangement …
  - … of common services, which support …
  - … inter-process communication.

Application Programming Interface (API)

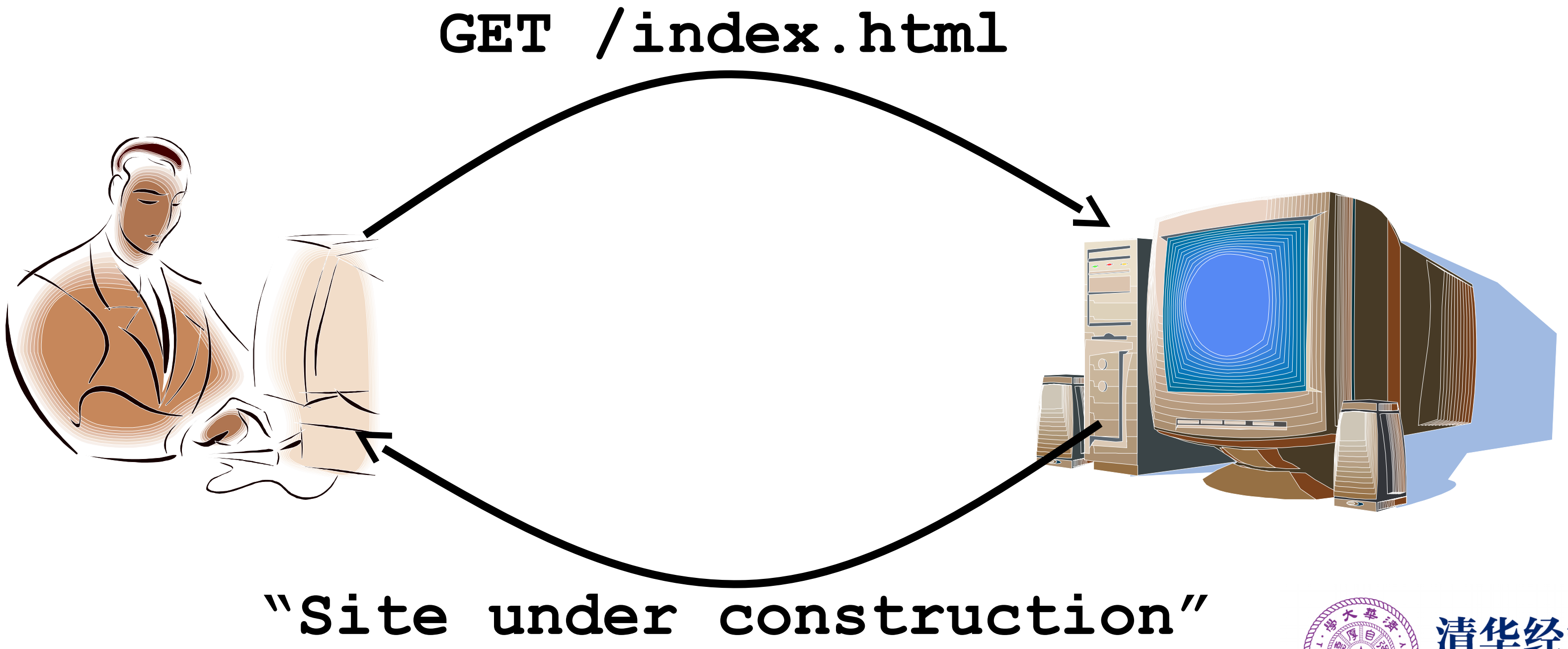
- Each OS can have a special interface exported by the network to the applications developer
- Most widely used network API: socket interface
  - Initially developed by the Berkeley Distribution of Unix and today ported to almost all OS

Sockets

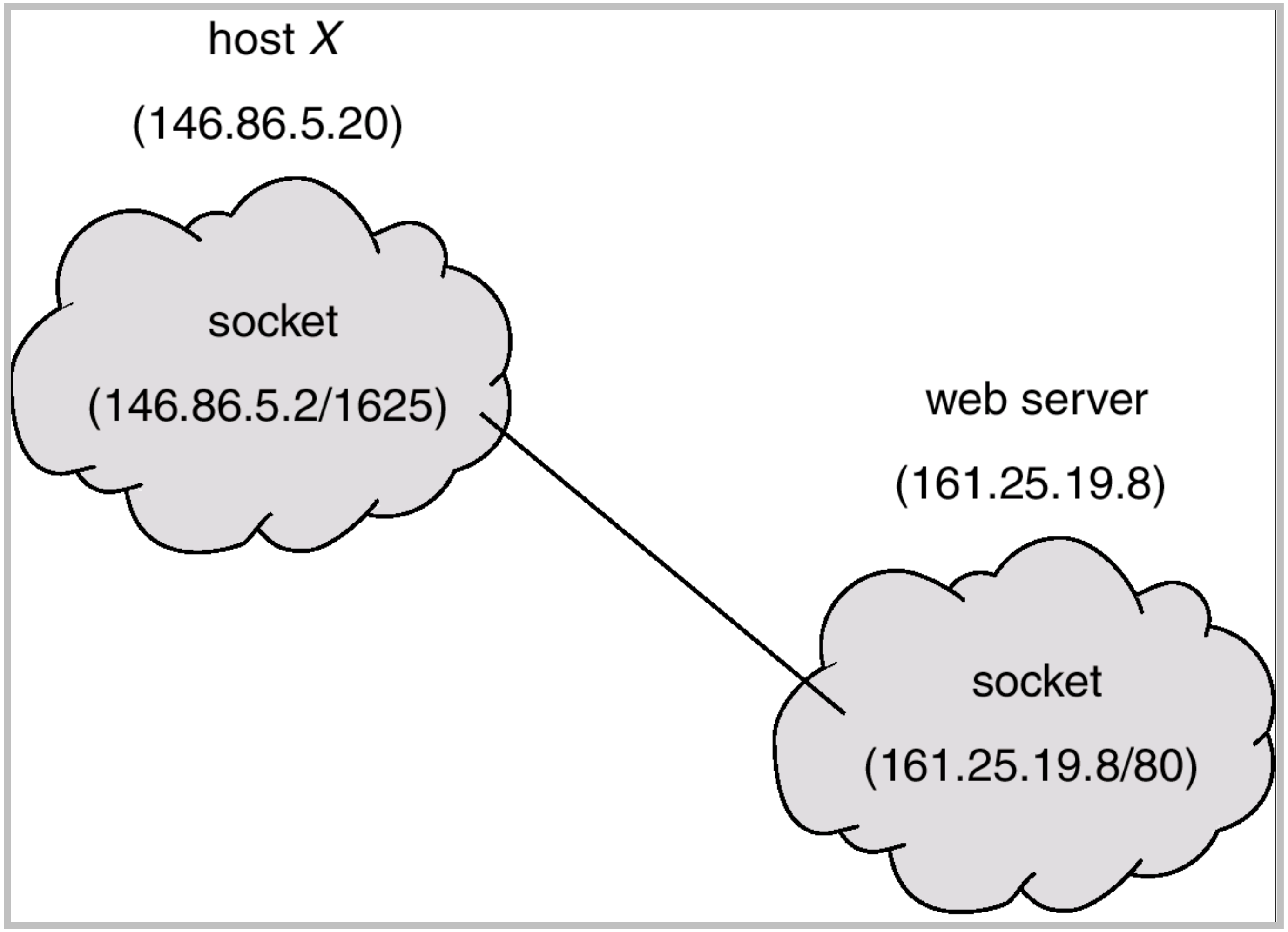
- A socket is defined as an endpoint for communication.
- Concatenation of IP address and port
- The socket `161.25.19.8:1625` refers to
  - Port `1625`
  - on host `161.25.19.8`
- Communication consists between a pair of sockets

The Client/Server Paradigm

- Client program
  - Running on end host
  - Requests service
  - E.g., Web browser
  - initiates communication
- Server program
  - Running on end host
  - Provides service
  - E.g., Web server
  - waits to be contacted



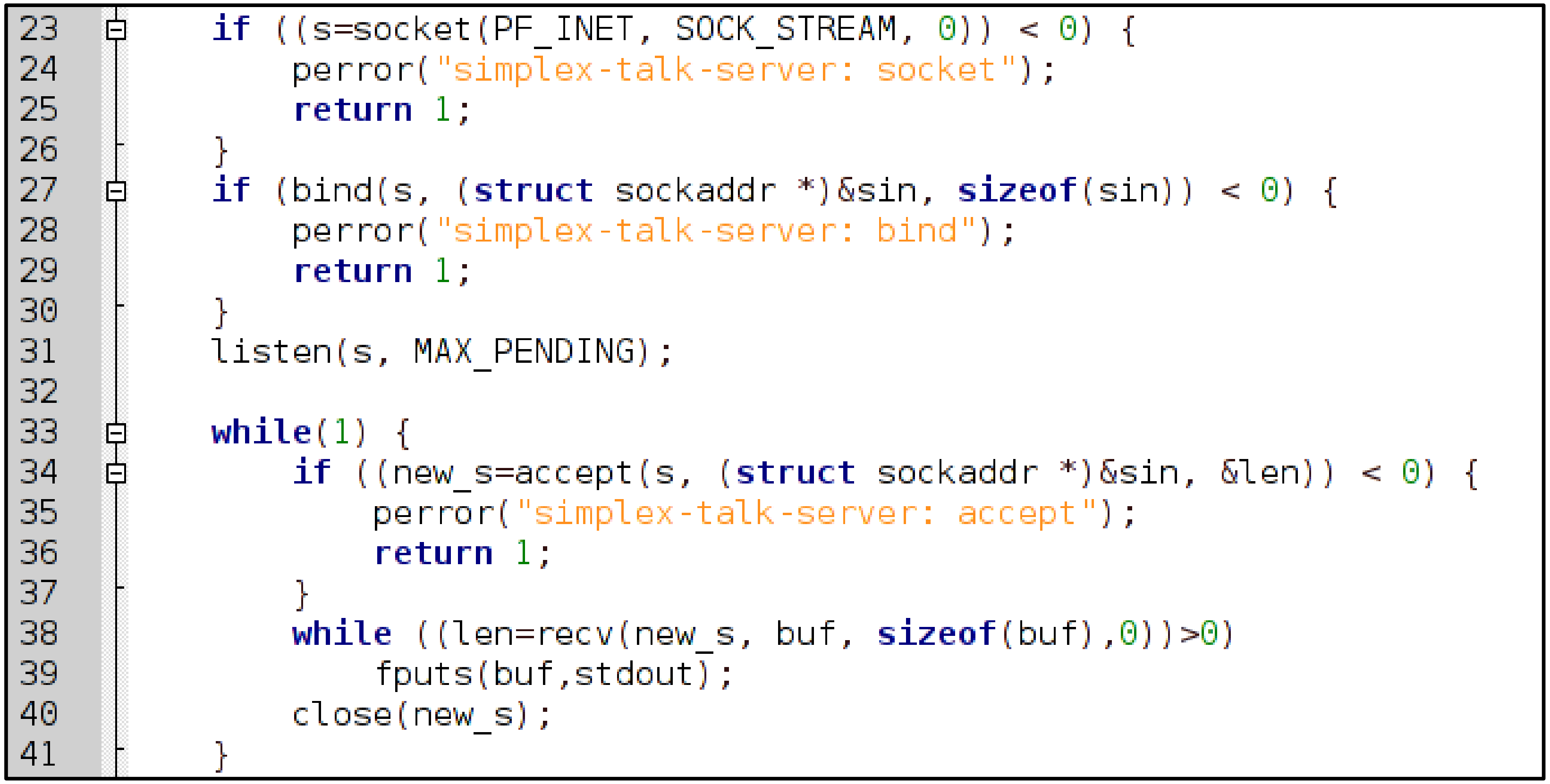
Socket communication



Socket API

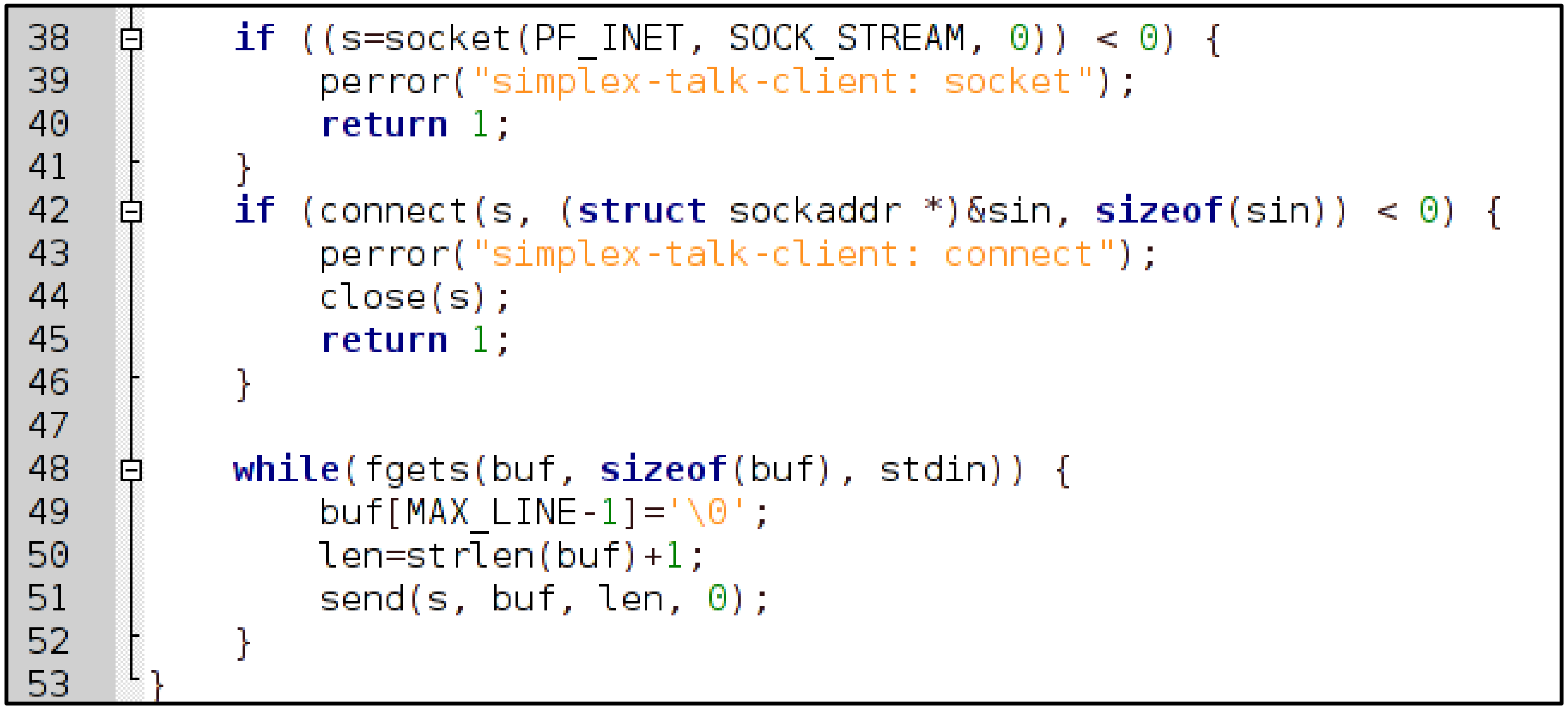
- Creating a socket
  - int socket(int domain, int type, int protocol)
    - domain = PF_INET, PF_UNIX
    - type = SOCK_STREAM, SOCK_DGRAM, SOCK_RAW
- Passive Open (on server)
  - int bind(int socket, struct sockaddr *addr, int addr_len)
  - int listen(int socket, int backlog)
  - int accept(int socket, struct sockaddr *addr, int addr_len)
- Active Open (on client)
  - int connect(int socket, struct sockaddr *addr, int addr_len)
- Sending/Receiving Messages
  - int send(int socket, char *msg, int mlen, int flags)
  - int recv(int socket, char *buf, int blen, int flags)

Simplex-talk example: server



```
    if ((s=socket(PF_INET, SOCK_STREAM, 0)) < 0) {
        perror("simplex-talk-server: socket");
        return 1;
    }
    if (bind(s, (struct sockaddr *)&sin, sizeof(sin)) < 0) {
        perror("simplex-talk-server: bind");
        return 1;
    }
    listen(s, MAX_PENDING);

    while(1) {
        if ((new_s=accept(s, (struct sockaddr *)&sin, &len)) < 0) {
            perror("simplex-talk-server: accept");
            return 1;
        }
        while ((len=recv(new_s, buf, sizeof(buf),0))>0)
            fputs(buf,stdout);
        close(new_s);
    }
```

Simplex-talk example: client



```c
    if ((s=socket(PF_INET, SOCK_STREAM, 0)) < 0) {
        perror("simplex-talk-client: socket");
        return 1;
    }
    if (connect(s, (struct sockaddr *)&sin, sizeof(sin)) < 0) {
        perror("simplex-talk-client: connect");
        close(s);
        return 1;
    }

    while(fgets(buf, sizeof(buf), stdin)) {
        buf[MAX_LINE-1]='\0';
        len=strlen(buf)+1;
        send(s, buf, len, 0);
    }
}
```

清华经管学院
Tsinghua SEM

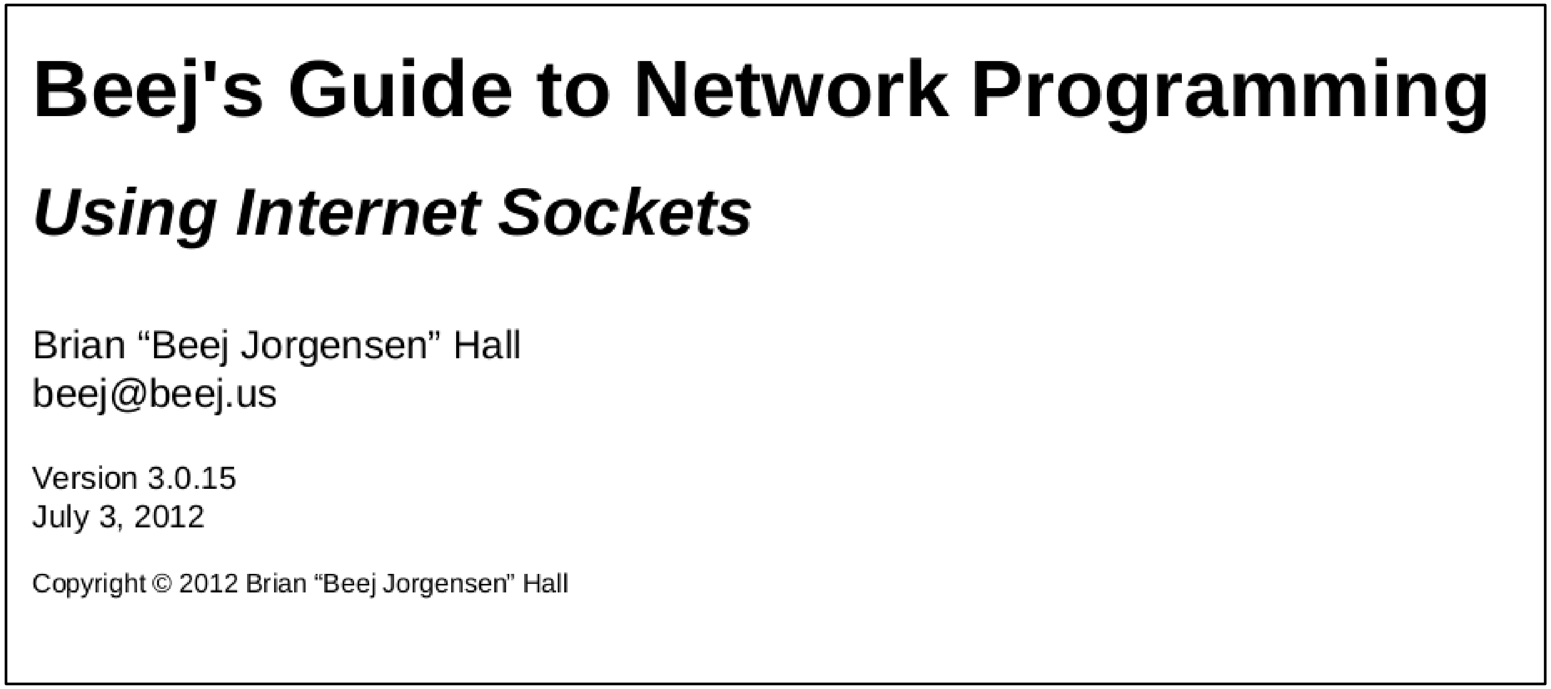
Socket server in Java

```
public class Server
{
  public static void main(String[] args) throws IOException {
    Socket client = null;
    PrintWriter pout = null;
    ServerSocket sock = null;

    try {
      sock = new ServerSocket(5515);
      while (true) {
        client = sock.accept();
        pout=new PrintWriter(client.getOutputStream(),true);
        pout.println("Hello!");
        pout.close();
        client.close();
      }
    }
    catch (IOException e) {
      System.err.println(e);
    }
    finally {
      if (client != null) client.close();
      if (sock != null) sock.close();
    }
  }
}
```

Socket client in Java

```
public class Client
{
  public static void main(String[] args) throws IOException {
    InputStream in = null;
    BufferedReader bin = null;
    Socket sock = null;

    try {
      sock = new Socket("127.0.0.1", 5515);
      in = sock.getInputStream();
      bin = new BufferedReader(new InputStreamReader(in));
      String line;
      while ((line = bin.readLine()) != null)
        System.out.println(line);
    }
    catch (IOException e) {
      System.err.println(e);
    }
    finally {
      if (sock != null) sock.close();
    }
  }
}
```

For More: A Reference Book


Beej's Guide to Network Programming

Using Internet Sockets

Brian “Beej Jorgensen” Hall
beej@beej.us

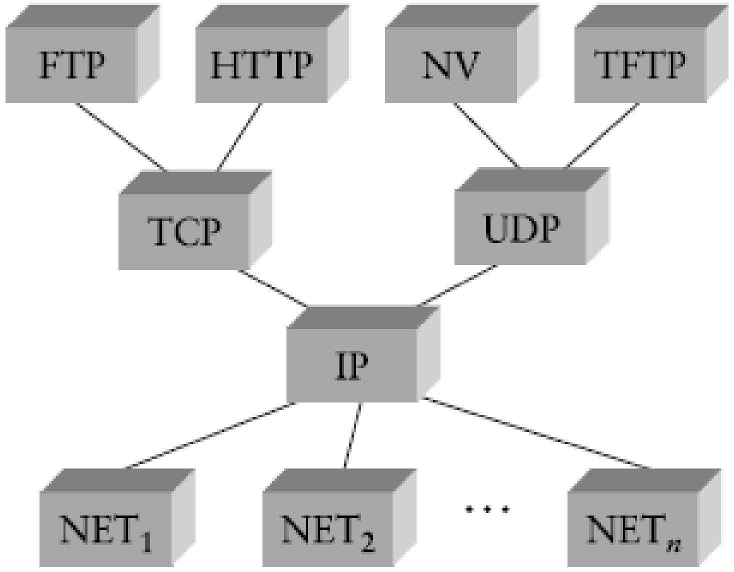
Version 3.0.15
July 3, 2012

Copyright © 2012 Brian “Beej Jorgensen” Hall

http://beej.us/guide/bgnet/

Further notes about socket API

- The socket API
  - a service provided on top of TCP/UDP
  - Supports the exchange of numbers and characters
  - Does not deal with data formats and structures
  - Powerful, but not a convenient choice in often cases
- Higher level APIs are provided by various languages, e.g.:
  - RPC (Remote Procedure Call)
  - RMI (Remote Method Invocation)
  - Web services (based on HTTP)



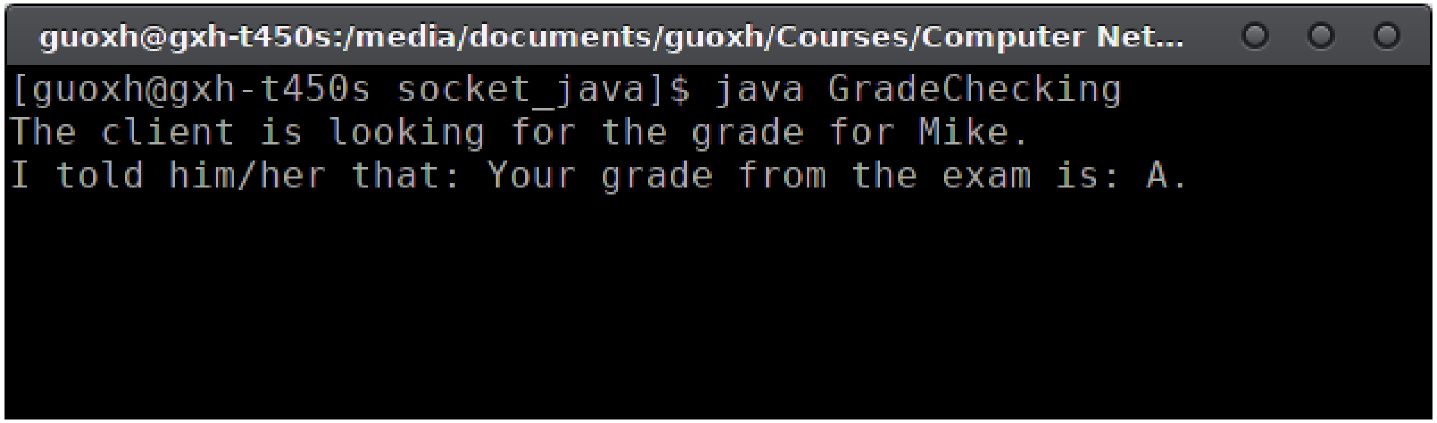
Exercise

- Write the simplex-talk client and server programs given in Section 1.4.2 of the textbook
  - Given in C
  - You may also write it with Java (or other languages)
- Run the client and the server on same machine and test
- Run the client and the server on different machines

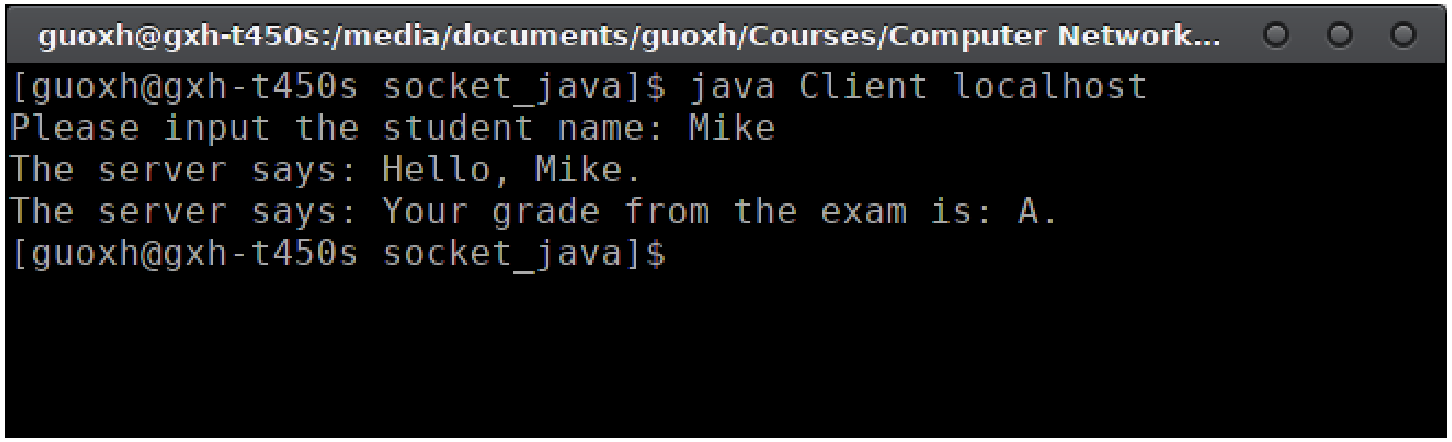
Assignment 1

- Based on the “simplex-talk” programs, write client and server programs that provide a “grade checking” service.
  - The server opens a server-side socket and wait for connection.
  - The client connects to the server's socket, and sends the name of a student to the server.
  - The server returns the exam grade of the student to the client.
  - The client then prints out the grade information that it receives.
  - Note: the grade information can be generated randomly.

The “Weather Predictor”


guoxh@gxh-t450s:/media/documents/guoxh/Courses/Computer Net...

```
[guoxh@gxh-t450s socket_java]$ java GradeChecking
The client is looking for the grade for Mike.
I told him/her that: Your grade from the exam is: A.
```


guoxh@gxh-t450s:/media/documents/guoxh/Courses/Computer Network...

```
[guoxh@gxh-t450s socket_java]$ java Client localhost
Please input the student name: Mike
The server says: Hello, Mike.
The server says: Your grade from the exam is: A.
[guoxh@gxh-t450s socket_java]$
```

Assignment 1

- Answer the following questions
  1. Can a client programmed in C communicate with a server programmed in Java? Why?
  2. The current “simplex-talk” server can only serve one client at one time. If we want the server to serve multiple client at the same time, what shall we do? Explain your solution (no need to implement).
- Requirements
  - Please submit your source codes (with necessary in-line comments) as well as your answers to the questions.
  - Please submit through our discussion board or email to TA no later than September 28.
  - Should there be any questions, feel free to ask on our discussion board or WeChat group.