Data Communication EDA344, DIT420
Description of Lab 1 and Optional Programming HTTP Assignment

Aras Atalar

Prajith R G

24/01/2018

Overview

- Lab 1 (Compulsory): WIRE**SHARK** lab
  - General Description

- Programming Assignment (Optional): Multi-threaded Web Server
  - What to do and how to do it
  - HTTP messages
  - Processes and threads

Lab 1 (Compulsory): WIRE**SHARK** lab

- Get insight into Internet traffic through the WIRE**SHARK** tool

- *Beware of ethical considerations! E.g. it is inappropriate to use this and similar tools outside the lab environment.*

WIRE**SHARK** Lab: Important Deadlines

- Jan 29: Study and submit in ping-pong the preparatory assignment.

** ONLY after this submission, you can get an invitation to the lab and book a timeslot. Accept the invitation & book lab-slot in ping-pong **

- Feb 5, 6: Labs @ Lindholmen. Study and get a printed copy of the manual for this lab and get the printed copy along when coming to lab
- Feb 12: Submit Lab Report in ping-pong

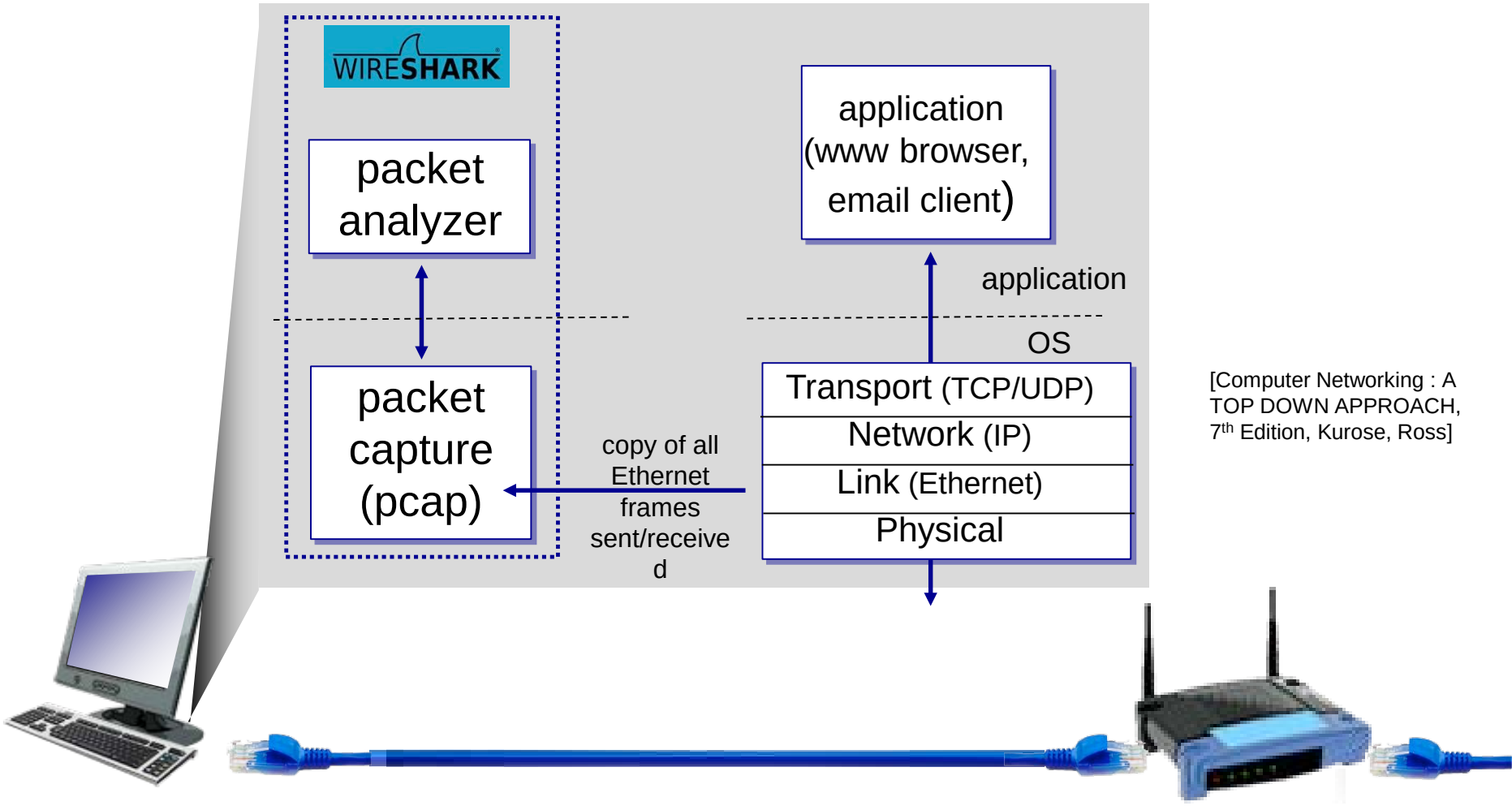
Accepting the invitation

- When an invitation is sent to you, you will also be notified by mail to your mail-id.
- You should log-in to your ping-pong account and go to the invitations (On top menu : Tools -> Invitations).
- Check the current invitations and click on the link 'Book me on event' to book yourself for the event.

Focus on the labs now

WIRE**SHARK** Lab

- Download WIRE**SHARK** and install it in your machine
- Follow the preparation notes for the lab to get familiar with WIRE**SHARK** .
- Try the lab instruction manual yourself and there will be help during the lab session.



[Computer Networking : A TOP DOWN APPROACH, 7th Edition, Kurose, Ross]

Programming Assignment (Optional): Multi-threaded Web Server

- Implement a multi-threaded webserver following the HTTP RFC specifications

- Successful completion gives extra credit if you pass the March 2018 exam

Programming Assignment: Important Deadlines

- Study the assignment description. Work on the assignment. For support with questions consult the Freuently-Asked-Questions Document
- Jan 31, Feb 14: Join the Q&A sessions
- Feb 21: Present your solution at the demo session
- Feb 28: Submit your code in ping-pong system

Multi-threaded Web Server

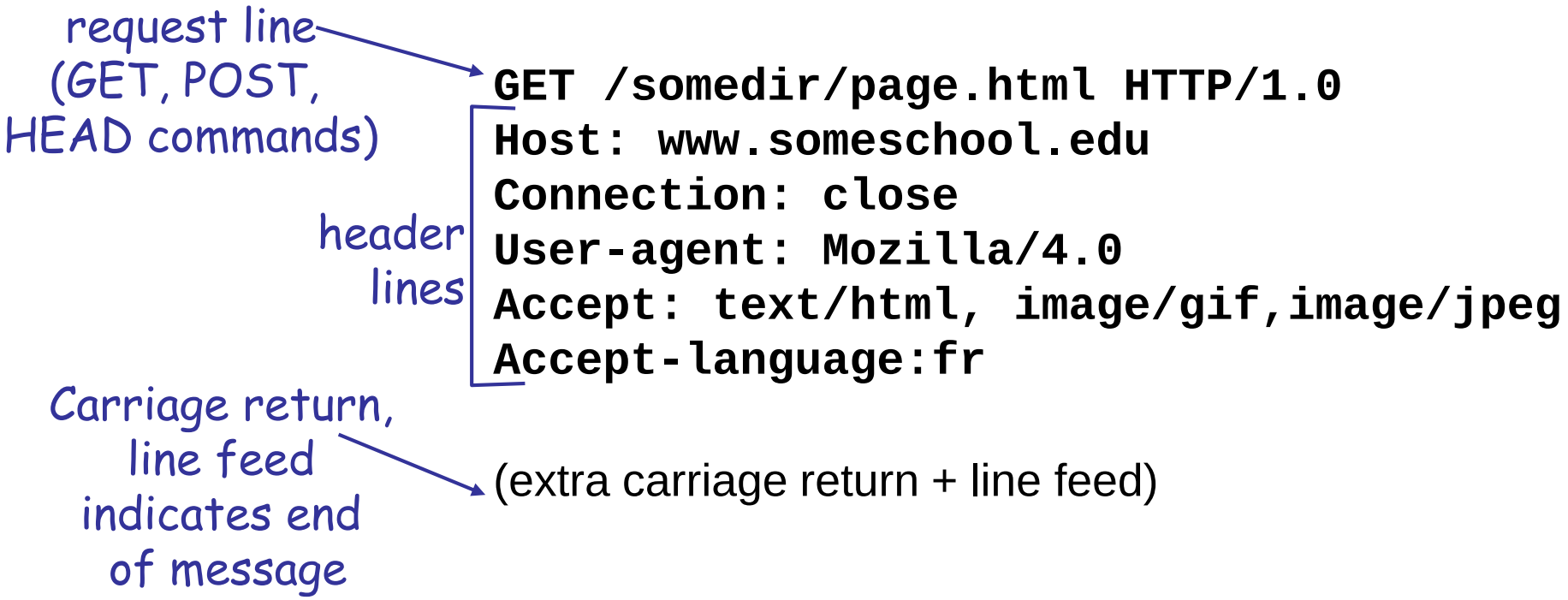
- The task:
  - Write a small Web server that supports a subset of the HTTP 1.0 specifications
  - The server should
    - be able to handle simultaneous requests
    - implement the HTTP methods GET and HEAD
    - handle and respond to invalid requests
    - Include Date:, Server:, Content-Type: and Content-Length: headers in all responses. Last-Modified: should be included where appropriate.

Multi-threaded Web Server

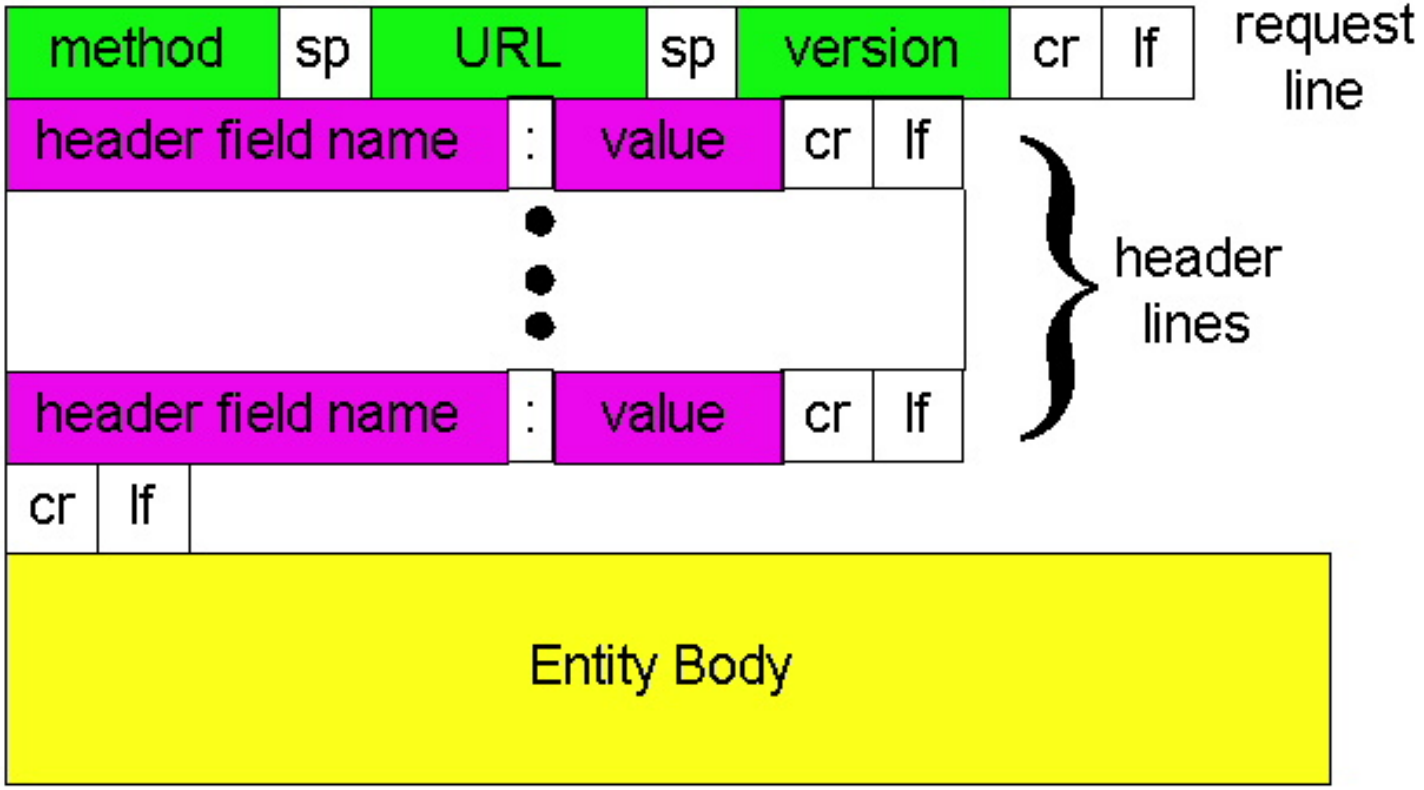
- Hints
  - Read the textbook
    - an example: simple Web server that does not handle simultaneous requests (Section 2.7, 2.9, 5th edition)
  - To handle concurrent requests
    - One way is to create a thread for each request
      - Java tutorial *Writing a Client/Server pair*
  - Check course assignments page for hints

http message format: request

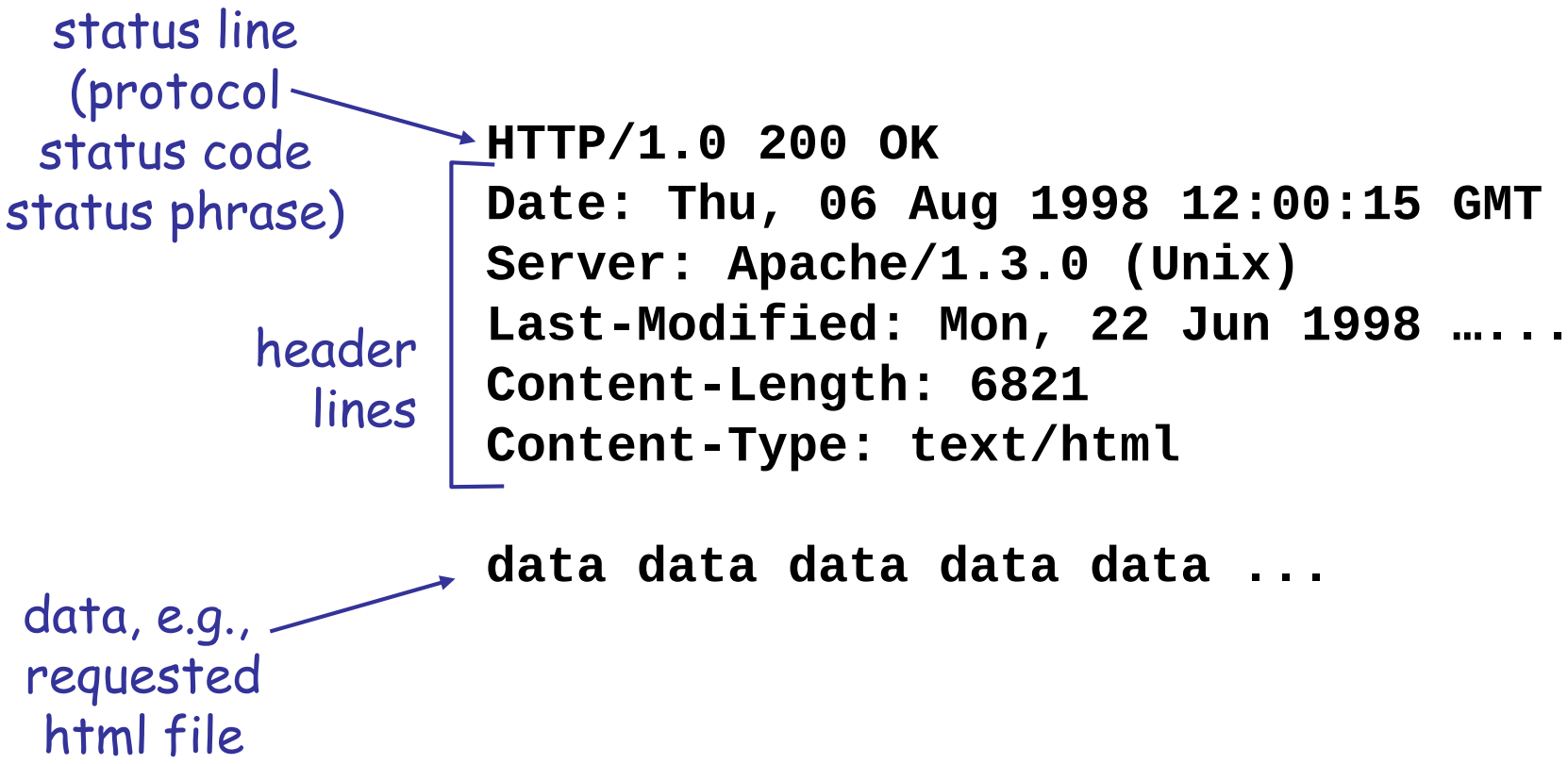
ASCII (human-readable format;
try telnet to www server, port 80)



http request message: general format



http message format: response



http response status codes

In first line in server->client response message.

A few sample codes:

200 OK

- request succeeded, requested object later in this message

301 Moved Permanently

- requested object moved, new location specified later in this message (Location:)

400 Bad Request

- request message not understood by server

404 Not Found

- requested document not found on this server

505 HTTP Version Not Supported

Java Concurrency Support

```
class MessagePrinter implements Runnable {
  protected String msg_;  The message to print
  protected PrintStream out_;  The place to print it

  MessagePrinter(String msg, PrintStream out)
  {
    out_ = out;
    msg_ = msg;
  }
  public void run() {
    out_.print(msg_); // display the message
  }
}
```

Sequential Version

```
class SequentialPrinter {

  public static void main(String[] args) {

    MessagePrinter mpHello = new
   MessagePrinter("Hello\n", System.out);
    MessagePrinter mpGoodbye = new
   MessagePrinter("Goodbye\n", System.out);

    mpHello.run();
    mpGoodbye.run();
  }
}
```

MultiThreaded Version

```
class ConcurrentPrinter {

   public static void main(String[] args) {

     MessagePrinter mpHello = new
  MessagePrinter("Hello\n", System.out);
     MessagePrinter mpGoodbye = new
  MessagePrinter("Goodbye\n", System.out);
     Thread tHello = new Thread(mpHello);
     Thread tGoodbye = new Thread(mpGoodbye);
     tHello.start();
     tGoodbye.start();
   }
}
```

Different types of servers

- Single process/thread

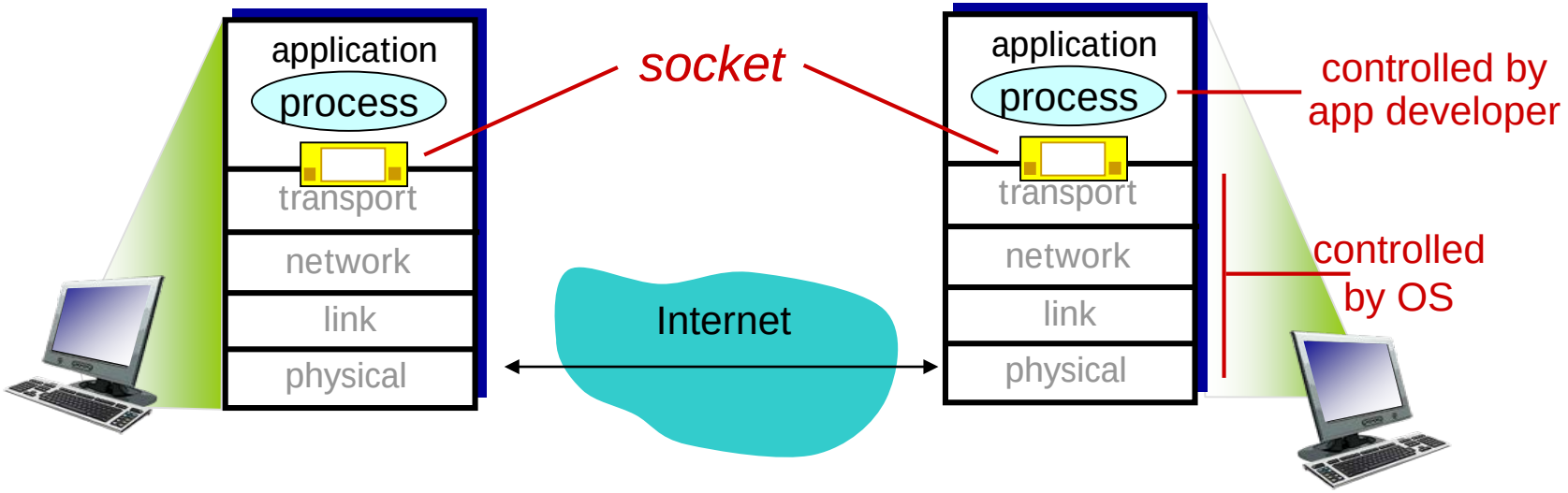
```
do forever
    accept client connection
    process all client requests
    close connection
```

- One thread per connection

```
do forever
    accept client connection
    create a new thread to process requests
```

Socket programming

- goal: learn how to build client/server applications that communicate using sockets
- socket: door between application process and end-end-transport protocol



Socket programming

Two socket types for two transport services:

- *UDP:* unreliable datagram
- *TCP:* reliable, byte stream-oriented

TCP Client Socket: Socket
TCP Server Socket: ServerSocket

We will see examples in our skeleton code