Data Communication EDA344, DIT420
Assignment 1

Bapi Chatterjee

Overview

- Multi-threaded Web Server
  - What to do and how to do it
  - HTTP messages
  - Processes and threads
- Wireshark lab
  - General Description

Important

- You have to read **Instructions for assignment process and submission** in pingpong
- Choose one option: either HTTP server programming or Wireshark lab
- Http programming: you can do it at home, but you have to demonstrate your program in the lab session
- Wireshark lab should be done in the lab room in Lindholmen

Important date

- Jan 22: Invitations for the HTTP lab are sent (to everybody, if you want to attend this lab, then accept(book) before Jan 26, otherwise ignore).
- Feb 4: Submission for the preparation report of Wireshark lab (if you want to attend this lab)
- Feb 6: Invitations for the wireshark lab are sent (only to the students who submit wireshark preparation report)

After the invitation

- (Accept 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

Multi-threaded Web Server

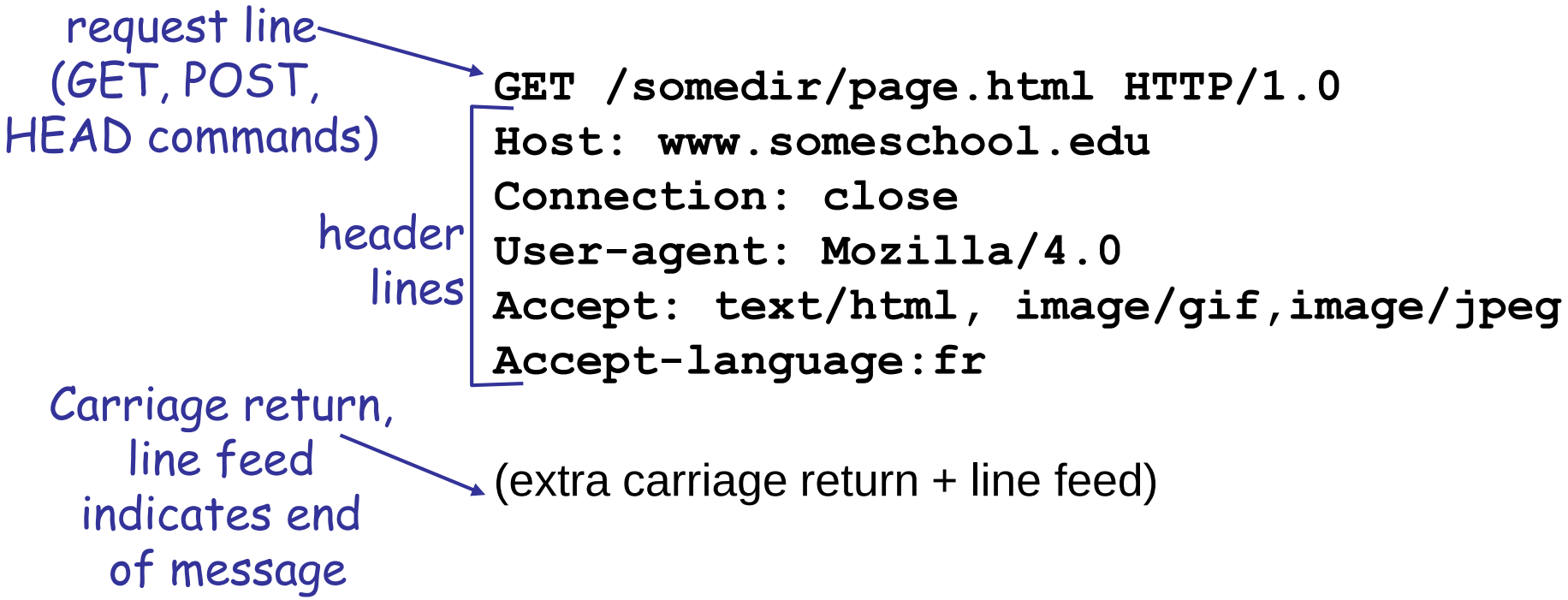
- The task:
  - Write a small Web server that supports a subset of the HTTP 1.0 specifications
  - The server should
    - be able to handle simultaneous requests
    - implement the HTTP methods GET and HEAD
    - handle and respond to invalid requests
    - Include Date:, Server:, Content-Type: and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

Multi-threaded Web Server

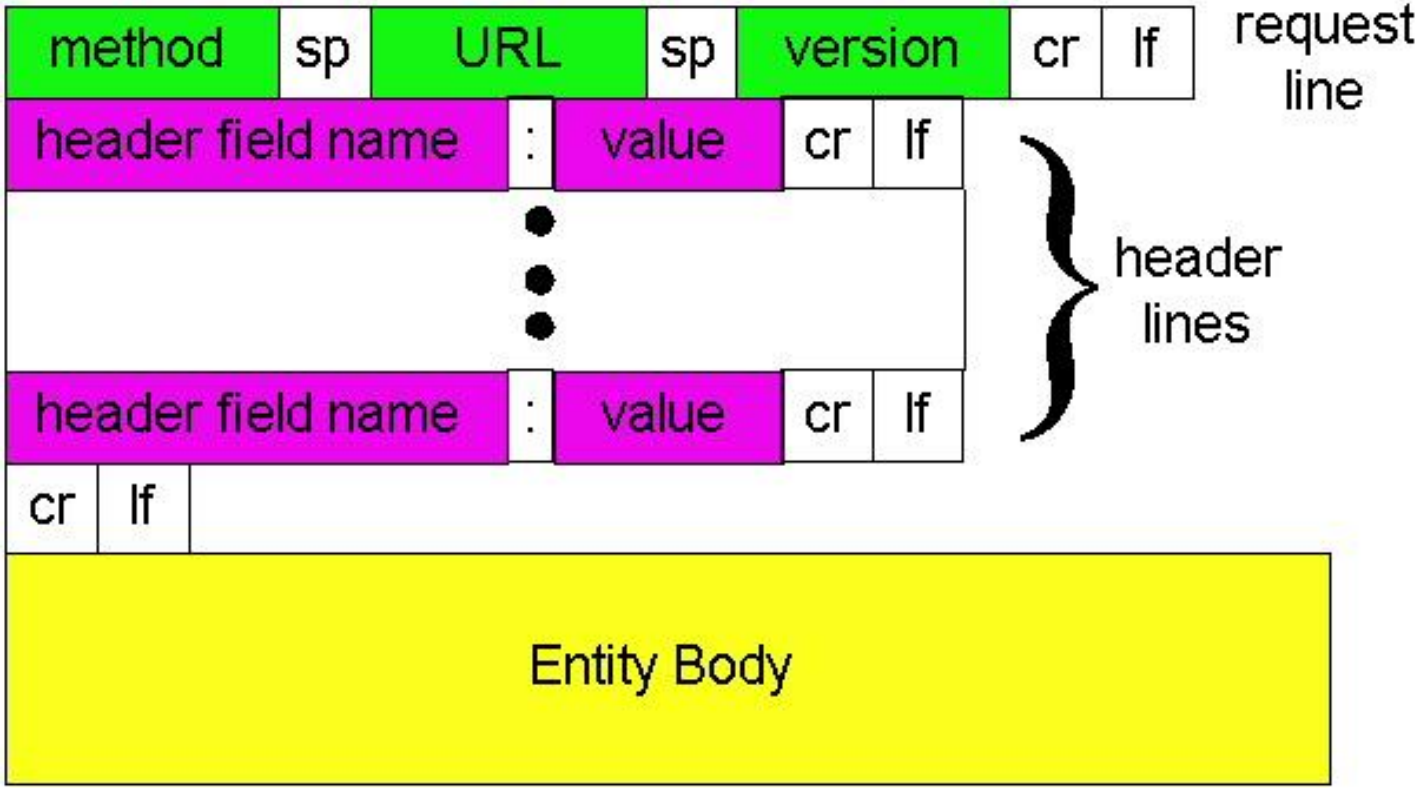
- Hints
  - Read the textbook
    - an example: simple Web server that does not handle simultaneous requests (Section 2.7, 2.9, 5th edition)
  - To handle concurrent requests
    - One way is to create a thread for each request
      - Java tutorial *Writing a Client/Server pair*
  - Check course assignments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

http message format: request

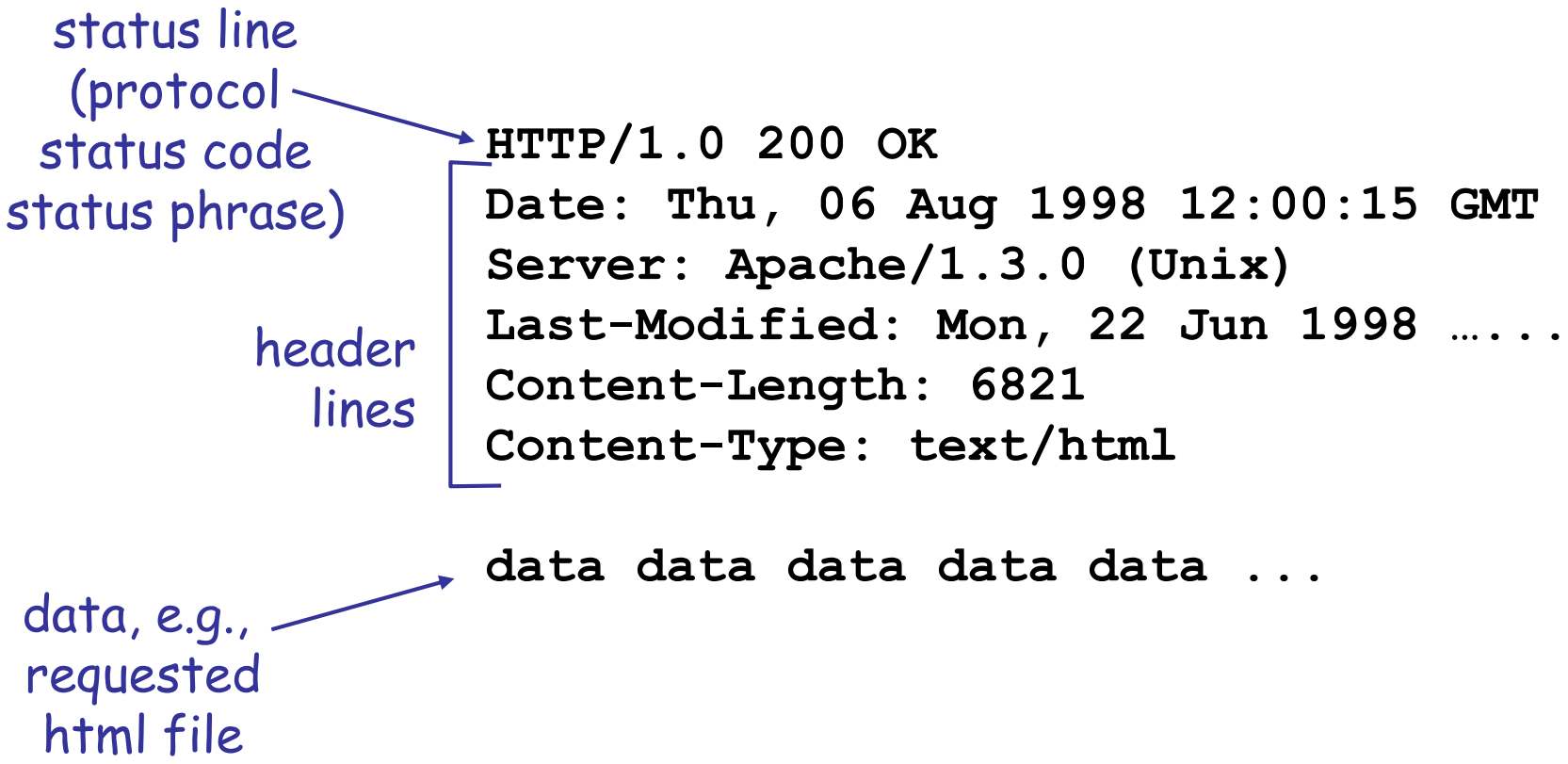
ASCII (human-readable format;
try telnet to www server, port 80)



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

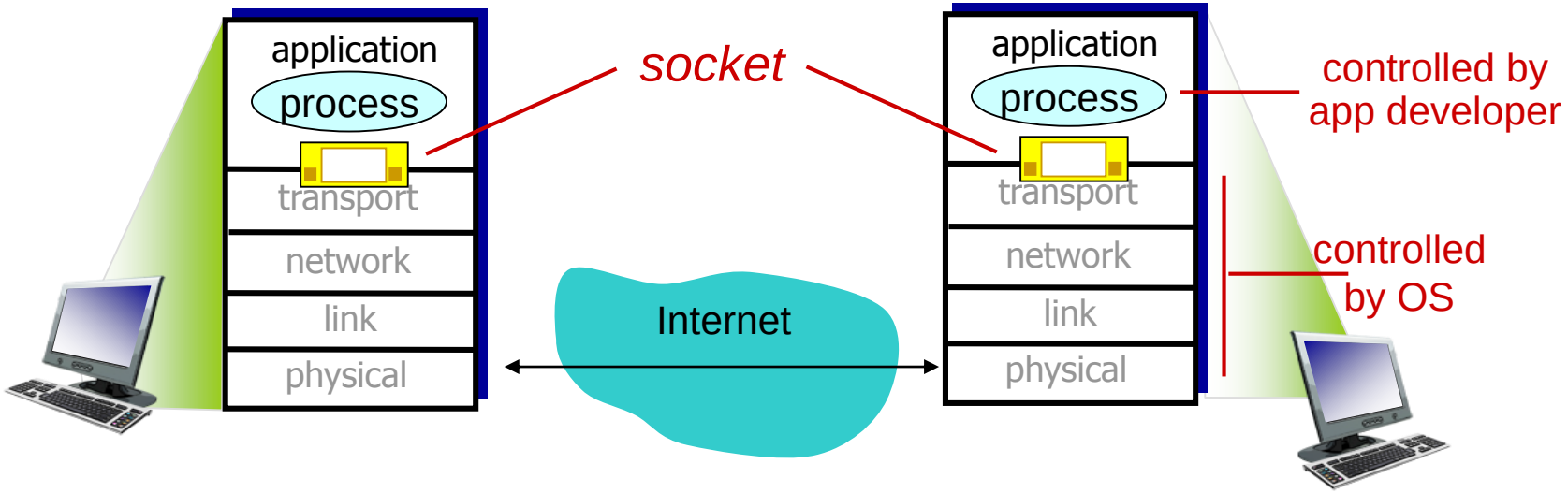
```
do forever
   accept client connection
   process all client requests
   close connection
```

- 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

Wireshark Lab

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