Introduction to assignment 2 and socket programming

TDTS06: Computer Networks

Carl Magnus Bruhner

September 2020

LINKÖPING UNIVERSITY

Lesson agenda

- General lab assignment information
- Assignment 2
- HTTP & Proxy
- Socket programming
- Questions

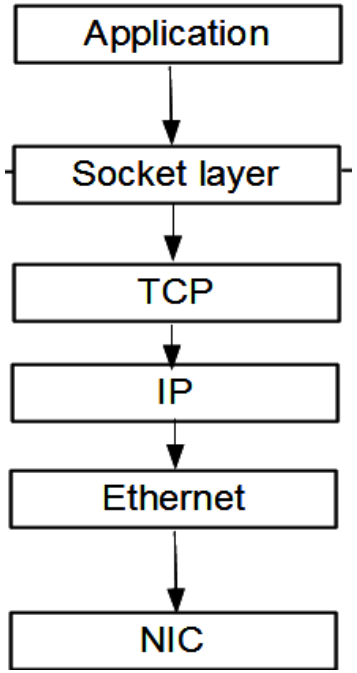
General lab assignment information

- Assignment 1 should be finished as soon as possible
- Assignment 2 takes time, and has a soft deadline September 25. It has changed since last year, but the concepts are very simliar the same.
- Assignment 3 is the same type as the first, and shouldn't take too much time.
- Assignment 4 needs a bit more time than 1 & 3, so don't put it off. A Python version is in the making, but might not be available in time (and might be buggy).

- Semi-hard deadline and last time to demonstrate easily is the day of your last lab session (October 14 or 15, depending on your lab group)
- Check with the TA if you plan to use languages other than those prescribed!

Assignment 2 – what will we do?

- Learn about HTTP, TCP/IP and WWW
- Learn socket programming and web proxies
- Build a simple proxy to alter content



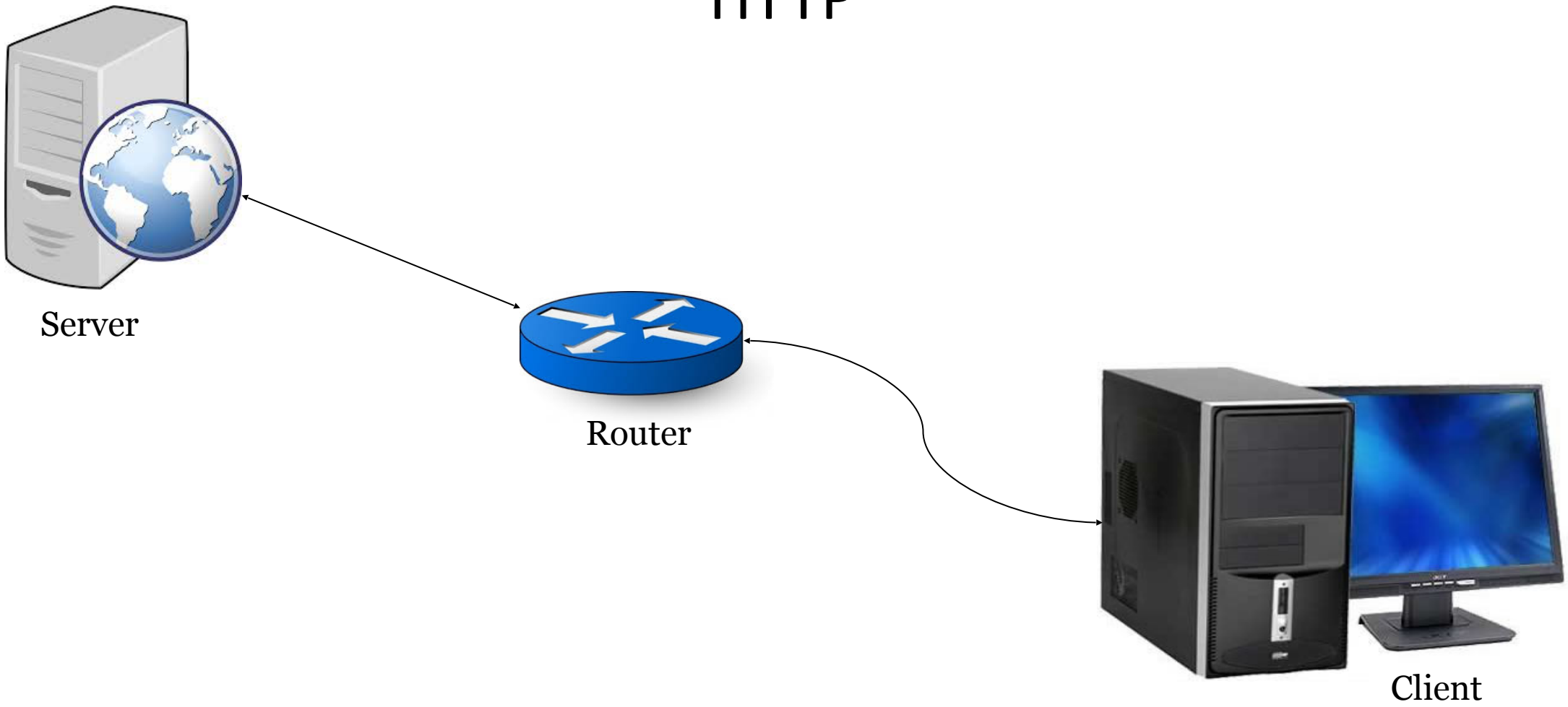
What is WWW?

- It is a world-wide system of interconnected servers which distribute a special type of document.
- Documents are marked-up to indicate formatting (Hypertexts).
- This idea has been extended to embed multimedia and other content within the marked-up page.

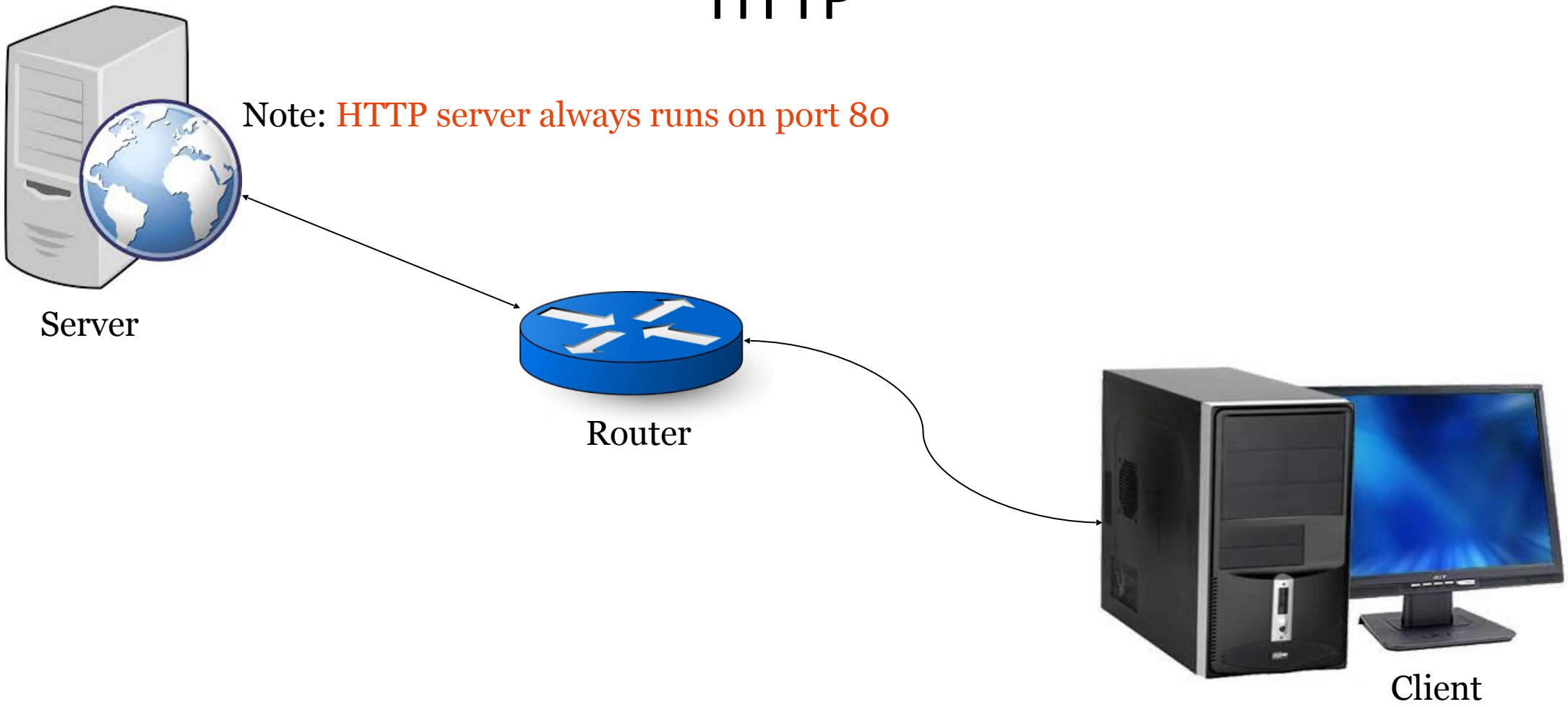
What is HTTP?

- HTTP is WWW's application layer protocol.
- HyperText Transfer Protocol (HTTP) to transfer HyperText Markup Language (HTML) pages and embedded objects.
- Works on a client-server paradigm.
- Needs reliable transport mechanism (TCP).

HTTP



HTTP


Note: HTTP server always runs on port 80
Server
Router
Client

HTTP

Server

Note: HTTP server always runs on port 80

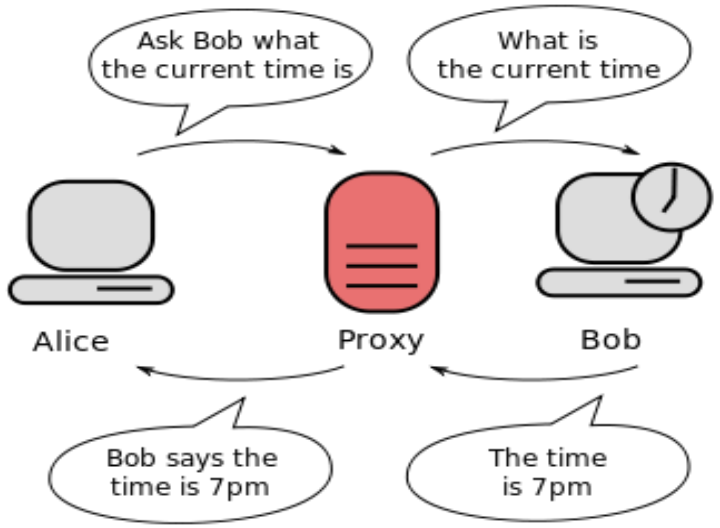
Router

Note: Client can use any unrestricted port
Generally >1024

Client

Proxy

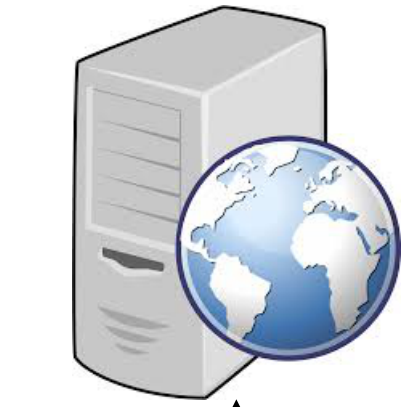
- Acts as intermediary between client and server.



Benefits of a proxy

- Hide your internal network information (such as host names and IP addresses).
- You can set the proxy to require user authentication.
- The proxy provides advanced logging capabilities.
- Proxy helps you control which services users can access.
- Proxy-caches can be used to save bandwidth.

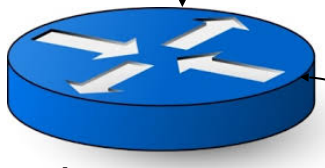
HTTP with proxy



Server

Note: HTTP server always runs on port 80

Proxy listens on a port (>1024) and talks to server on another (>1024)



Router

Proxy

Note: Client can use any unrestricted port Generally >1024

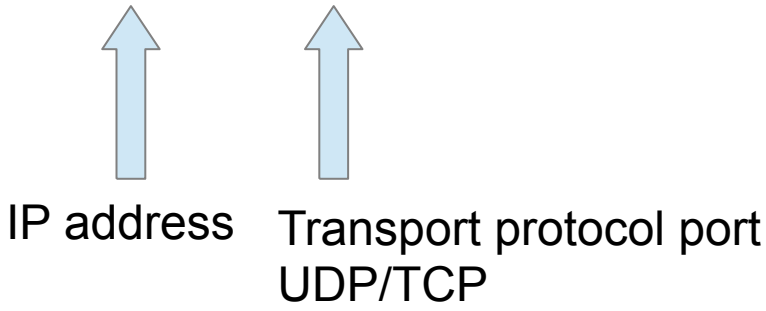
Client

What is a port?

- A port is an application-specific or process-specific software construct serving as a communications endpoint.
- The purpose of ports is to uniquely identify different applications or processes running on a single computer and thereby enable them to share a single physical connection to a packet-switched network like the Internet.

Ports continued

- Port only identifies processes/applications.
- With regard to the Internet, ports are always used together with IP.
- Notation 192.168.1.1:80



Socket programming

- These are software constructs used to create ports and perform operations on them.
- We will talk about these types of sockets:
    - Datagram socket
    - Stream socket
    - SSL sockets

Datagram sockets

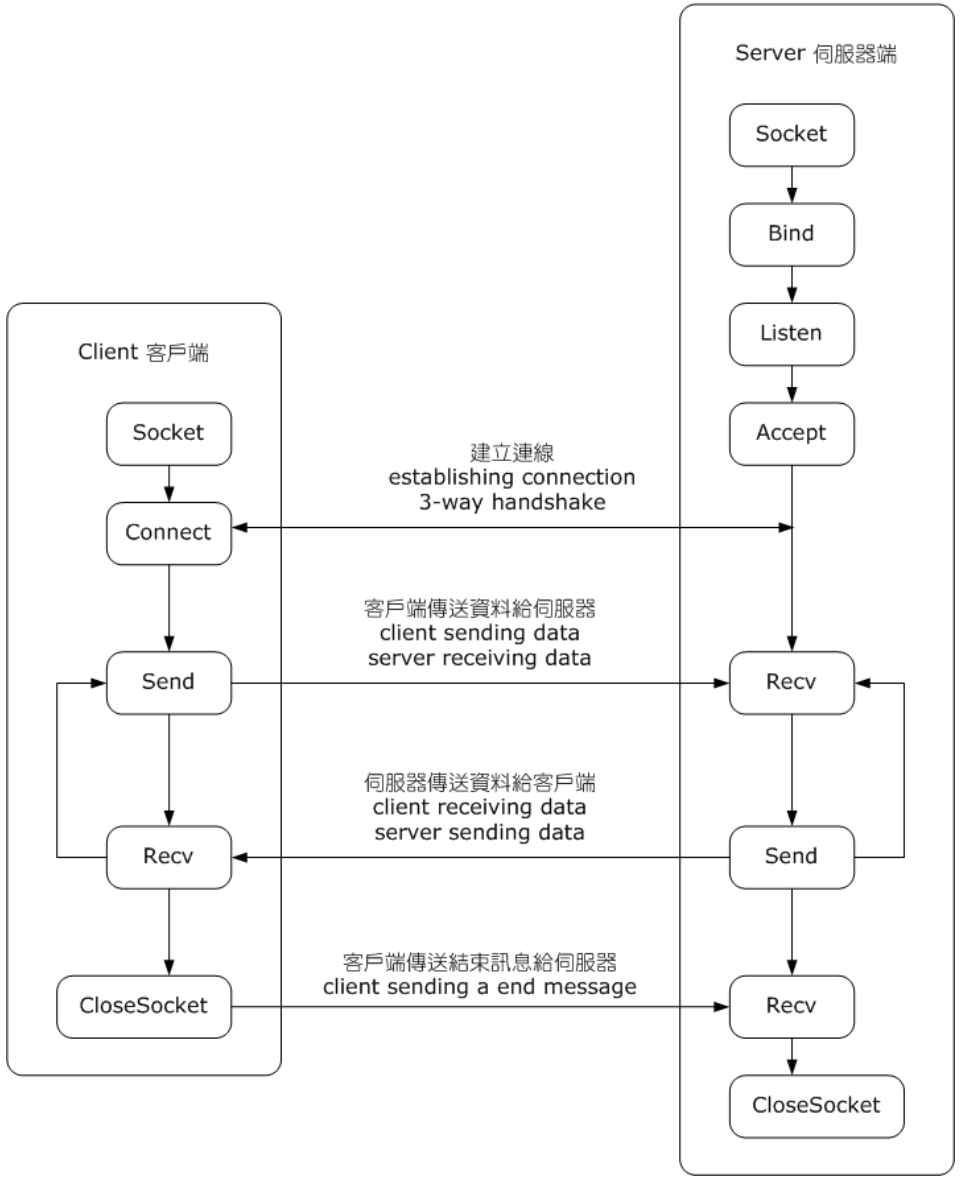
- They are connectionless
- Do not guarantee in order delivery
- No form of loss recovery
- No congestion control
- No flow control
- Datagram sockets use UDP

Stream sockets

- Connection oriented sockets
- In order and guaranteed delivery
- Error identification and recovery
- Congestion control
- Flow control
- Stream sockets use TCP protocol
- SSL sockets are similar to stream sockets, but include functions to handle encryption

Important socket calls

- socket
- bind
- listen
- accept
- connect
- send
- recv



Socket programming calls

- **socket()**
  - Takes as input
    - Address family (=AF_INET)
    - Socket type (=SOCK_STREAM)
  - Returns
    - A socket object

Socket programming calls

- **bind()**
    - Takes as input
        - address/port tuple (for AF_INET)
- What does this do?
    - Associate the socket with an address/port tuple

Socket programming calls

- **listen()**
    - Takes as input
        - Backlog (max queue of incoming connection)
- This must run at the server side to listen to incoming connection

Socket programming calls

- **connect()**
    - Takes as input
        - Address/port tuple
- What does this do?
    - Attempts to setup a connection with the other end

Socket programming calls

- **accept()**
    - Takes as input
        - –
    - Returns
        - conn - a new socket object
        - address - address/port tuple
- Reads through the backlog and picks one from the list to connect to it.
- Runs at the server side

Socket programming calls

- **send()**
    - Takes as input
        - Message
    - Returns
        - Number of bytes sent
- Send is always best effort. If it cant send the whole message, the value returned is smaller.

Socket programming calls

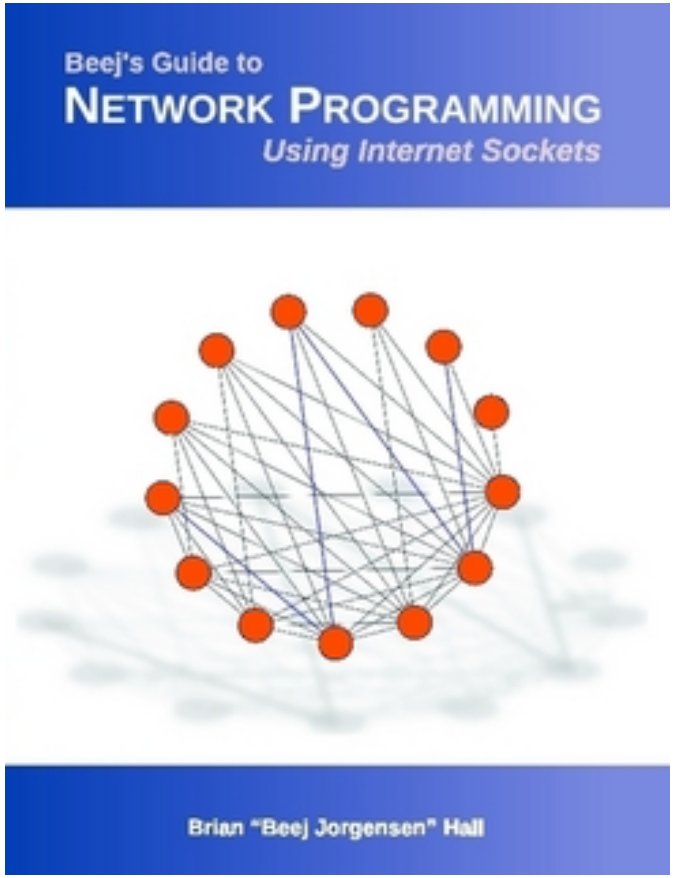
- **recv()**
  - Takes as input
    - Max buffer length
  - Returns
    - bytes object representing the data received

Socket programming calls

- **close()**
    - Takes as input
        - –
- Marks the socket as closed

Socket programming resource

- Helpful guide linked from the assignment text: Beej's Guide to Network Programming
- Based on C, but can be used as a foundation for other languages



Assignment 2 introduction

- You are to develop a “Fake News” proxy that can modify the content sent from the server before returning to the browser.
- “Smiley” from “Stockholm” should be altered to “Trolly” from “Linköping”
- Images of Smiley should be altered to troll images

Assignment 2 description

- Socket programming is the key
- Build a proxy to which a user can connect to
- The proxy connects to the server on the user's behalf (recollect how proxy works)
- Proxy receives the response from the server
- Alters any occurrences of Smiley and Stockholm
- Redirects the (potentially) altered content to the user

Assignment 2 requirements

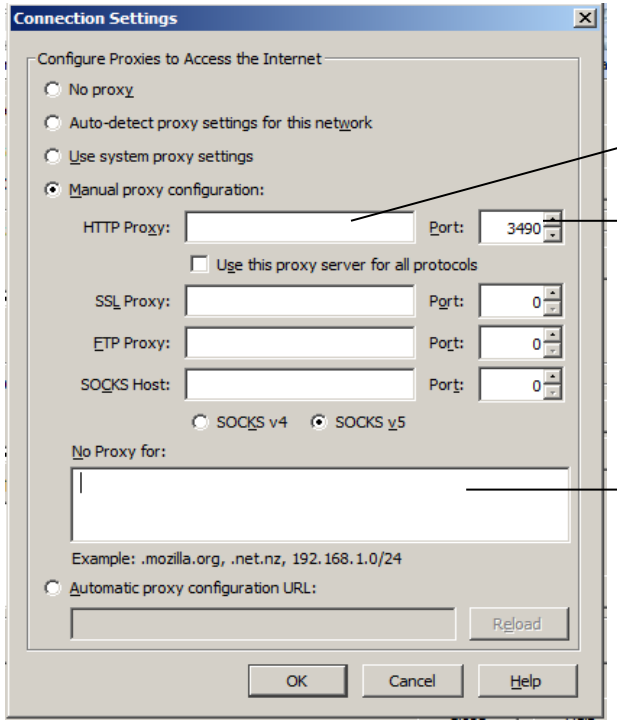
- Handles simple HTTP GET interactions between client and server
- Alters any text occurrences of “Smiley” and “Stockholm”
- Replaces any images of Smiley

- Uses at least one TCP socket
- Imposes no limit on the size of the transferred HTTP data
- Uses only *basic* libraries (e.g. not “HttpURLConnection” Java class)

- Works with all web browser and systems (HTTP only)

Browser configuration

- Proxy listens on a particular port



127.0.0.1

Proxy's port number

Make sure it is blank

HTTP basics

- Recollect lab 1. It contains things that you need in lab 2.
- HTTP request
    - Get
        - Syn, SynAck, Ack

```
⊞ Transmission Control Protocol, Src Port: 50139 (50139), Dst Port: http (80), Seq: 1, Ack: 1, Len: 276
⊟ Hypertext Transfer Protocol
  ⊞ GET /vod/final_1.3.f4m HTTP/1.1\r\n
    Host: 130.236.182.199\r\n
    Connection: keep-alive\r\n
    User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/37.0.2062.103 Safari/537.36\r\n
    Accept-Encoding: gzip,deflate,sdch\r\n
    Accept-Language: en-US,en;q=0.8,ms;q=0.6\r\n
    \r\n
    [Full request URI: http://130.236.182.199/vod/final_1.3.f4m]
```

HTTP basics

- HTTP response
  - OK

```
⊞ Transmission Control Protocol, Src Port: http (80), Dst Port: 50139 (50139), Seq: 4381, Ack: 277, Len: 1215
⊞ [4 Reassembled TCP Segments (5595 bytes): #248(1460), #249(1460), #251(1460), #252(1215)]
⊟ Hypertext Transfer Protocol
  ⊞ HTTP/1.1 200 OK\r\n
    Date: Sun, 07 Sep 2014 10:06:36 GMT\r\n
    Server: Apache/2.2.17 (Unix) DAV/2\r\n
  ⊞ Content-Length: 5354\r\n
    Last-Modified: Tue, 04 Feb 2014 12:25:40 GMT\r\n
    Keep-Alive: timeout=15, max=100\r\n
    Connection: Keep-Alive\r\n
    Content-Type: text/xml\r\n
    \r\n
```

HTTP basics

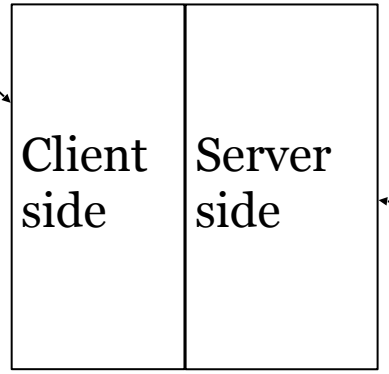
- HTTP 1.0 vs HTTP 1.1
    - Many differences (compare RFC:s)
    - For this assignment:
        - Connection: close
            - Handshake-Get-response-OK-Teardown
        - Connection: keep-alive
            - Handshake-Get-response-OK-wait-Get-response
- What should you use for the proxy?

How to handle connections

- With connection: keep-alive, the connection is kept open. You are responsible to figure out when the response is completed.
- With connection: close, the server closes the connection after the response is sent.
- How can you enforce connection: close on HTTP 1.1?

General overlay

Server



Proxy

Client

General overlay

Server

Server side: listens on a port, accepts, receives, forwards to client side

Client side | Server side

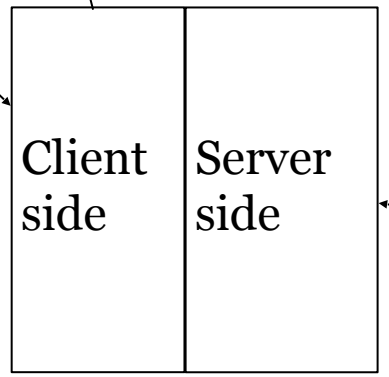
Proxy

Client

General overlay

Server

Client side: connects to the server, send request, receive response,
Forwards to server side



Proxy

Client

Content altering

- Need to be able to filter both based on URL and content.
- In which of the two halves of the proxy will you implement altering based on URL?
- In which of the two halves of the proxy will you implement content altering?
- How to actually do content altering?

Content altering

- Response from the server comes in segments
- Remember TCP segmentation?
- Reconstruct the message in a temporary buffer
- No dynamic sizing of buffer, chose a value and stick with it
- Do not type-cast non-text data!
- Then run content altering only on the text message

Text vs other binary data

- What is the requirement for filtering with regard to binary data?
    - Only that you have to be smart in handling any data type
- What will happen if you attempt to reconstruct an image or video and filter it?
- Solutions?

Text vs binary data

- Content-type header
- Differentiate content type
    - Alter/don't alter
    - Send request directly or alter before

```
⊞ Transmission Control Protocol, Src Port: http (80), Dst Port: 50139 (50139), Seq: 4381, Ack: 277, Len: 1215
⊞ [4 Reassembled TCP Segments (5595 bytes): #248(1460), #249(1460), #251(1460), #252(1215)]
⊟ Hypertext Transfer Protocol
  ⊞ HTTP/1.1 200 OK\r\n
    Date: Sun, 07 Sep 2014 10:06:36 GMT\r\n
    Server: Apache/2.2.17 (Unix) DAV/2\r\n
  ⊞ Content-Length: 5354\r\n
    Last-Modified: Tue, 04 Feb 2014 12:25:40 GMT\r\n
    Keep-Alive: timeout=15, max=100\r\n
    Connection: Keep-Alive\r\n
    Content-Type: text/xml\r\n
    \r\n
```

Debugging advice

- Stick to simple web pages initially
- Debug incrementally
- Check and double check request string for formatting and completeness
- Source of many errors like 'server closed connection unexpectedly'
- If developing on own computers, use Wireshark to debug. Can save a lot of time!

Debugging advice

- HTTP vs HTTPS
  - Requirements do not ask for a proxy which works with HTTPS
  - If the browser allows selective proxy, enable for HTTP only
  - Restrict yourselves to simple sites and basic test cases

Debugging advice

- Header manipulation
    - First thing to check at a proxy is the URL that it sends out to the server
    - It might require different manipulations based on the site. Be sure that you test for all sites mentioned in the test scenario
    - If you change some fields in the header, the packet length has to be changed or brought back to the original length

Debugging advice

- Read all the instructions, even though its lengthy
- Develop incrementally
- Look at the debugging checklist
- Experiment with Wireshark
- Document the limitations of your proxy

Questions?

carbr307@student.liu.se (subject “TDTS06 […]”)

www.liu.se