Proj1 – Http Proxy

Danyang Zhuo
Proj1 – Http Proxy

• Due: Monday May 11 by 5pm

• Programming Language: Anything supported by attu1.cs.washington.edu

• Please add a script
  – .\run <port_num>
Page Load

• HTTP GET (url)/index.html
• Page is a HTML file pointing to resources:
  – Other HTML files
  – CSS files
  – Pictures
• Subsequent HTTP GET for each resource
• Construct the view
• Demo
Http Proxy for HTTP Traffic

Browser → Proxy → Website

GET → Proxy

GET → Website
Http Proxy for HTTP Traffic

Browser → Proxy ← Response → Website
GET google.com HTTP/1.1
Host: google.com
...
...
Connection: keep-alive
Proxy-Connection: keep-alive
Http Proxy for HTTP Traffic

GET google.com HTTP/1.1
Host: google.com
...
...
...
Connection: keep-alive
Proxy-Connection: keep-alive

GET google.com HTTP/1.0
Host: google.com
...
...
...
Connection: close
Proxy-Connection: close
Use Wireshark to figure out your request format

- GET google.com HTTP/1.0  
- Host: google.com  
- ...
- ...
- ...
- Connection: close  
- Proxy-Connection: close  
-  
-  
-  
-
Please DO NOT do anything fancier than forwarding raw bits

- (Java programmer) Do not use Scanner, BufferedReader, CharBuffer, ... because those modules assume you are forwarding text
  - Does not work for gzip and picture!!
  - Use BufferedInputStream/BufferedOutputStream
Http Tunnel

Browser ➔ TCP CONNECT ➔ Proxy ➔ Website
Http Tunnel

Browser → Proxy (TCP CONNECT) → Website

Browser → Proxy (HTTP CONNECT) → Proxy (TCP Connect) → Website
Http Tunnel

Browser ➔ HTTP CONNECT ➔ Proxy ➔ Website

Browser ➔ HTTP CONNECT ➔ Proxy ➔ Website

Browser ➔ HTTP 200 ➔ Proxy ➔ Website

Proxy ➔ TCP Connect ➔ Website

Proxy ➔ TCP Conn ➔ Website
Http Tunnel

Browser -> Proxy: TCP CONNECT
Proxy -> Website

Browser -> Proxy: HTTP CONNECT
Proxy -> Website: TCP Connect

Browser -> Proxy: HTTP 502
Proxy -> Website
Buffer-less Proxy

• Only Buffer Header

• (Async Proxy Programmer) Only forward when the writing socket and reading socket are both ready

• Do not try to read everything then write
TCP

Server:
Socket = new Socket()
Socket.bind(‘localhost’, 9999)
Socket.listen()
While 1:
    conn = Socket.accept()
    data = conn.recv()
    conn.close()
Socket.close()

Client:
Socket = new Socket()
Socket.connect(‘localhost’, 9999)
Socket.send(‘hahaha’)
Design and Implementation

Main Thread

While 1:
    conn = socket.accept()
    start new thread for conn

Per-ConnectionThread

If HTTP CONNECT:
    Connect to Server
    if not succeed:
        send 502 to client
        return
    send 200 to client
    start forwarding

ELSE:
    Connect to Server
    Modify request
    Send Request to server
    Forward response to client