**DCN Homework 3**

**Programming**

**Name: Shen, Yulin**

**Date: 03/26/2019**

**NetID: ys2542**

**Section: 001**

1. Web Server

I use the skeleton code from the companion website ([www.pearsonhighered.com/cs-resouces](http://www.pearsonhighered.com/cs-resouces)). My server will create a connection socket, then receive the HTTP request from this connection, then parse the request, then get the requested file, then create a HTTP response message of the file with the header line, finally send the response to the browser. If the requested file is not found, my server will return a “404 Not Found” error message. My server mainly implements two features: show the file found and show the not found message.

Source code:

# Name: Shen, Yulin

# NetID: ys2542

# WebServer.py

# The code is based on the skeleton code taken from companion website for the textbook in www.pearsonhighered.com/cs-resources

# We will need the following module to make a web server

import sys

from socket import \*

# Check terminal command and print the right format if wrong

if (len(sys.argv) != 2):

print('Format: python WebServer.py <Server\_Port\_Number>')

sys.exit()

# Create a UDP socket

# Notice the use of SOCK\_DGRAM for UDP packets

serverSocket = socket(AF\_INET, SOCK\_STREAM)

# Assign IP address and port number to socket

serverSocket.bind(('', int(sys.argv[1])))

# Wait for at most one request at a time

serverSocket.listen(1)

while True:

# Establish a new connection

connectionSocket, addr = serverSocket.accept()

try:

# Receive the message from the client

message = connectionSocket.recv(1024).decode()

# Extract the path from the message

filename = message.split()[1]

# Read the path

f = open(filename[1:], 'r')

outputdata = f.read()

# Send the response message

connectionSocket.send('HTTP/1.1 200 OK\r\n\r\n'.encode())

# Send the content

for i in range(0, len(outputdata)):

connectionSocket.send(outputdata[i].encode())

connectionSocket.send('\r\n'.encode())

# Close the client connection

connectionSocket.close()

except IOError:

# send the response menssage

connectionSocket.send('HTTP/1.1 404 Not Found\r\n\r\n'.encode())

connectionSocket.send('<html><head></head><body><h1>404 Not Found</h1></body></html>\r\n'.encode())

# Close the client connection

connectionSocket.close()

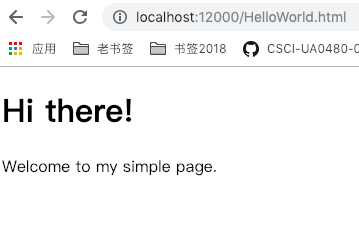
# Close the UDP socket

serverSocket.close()

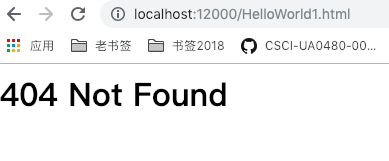
Demo:



I make a HelloWorld.html file in the same directory. The browser shows it below.



But I do not have a file named HelloWorld1.html, the browser shows it below.



1. UDP Pinger

The server code is completely got from the companion website ([www.pearsonhighered.com/cs-resouces](http://www.pearsonhighered.com/cs-resouces)). I imitate the server code to implement my own client code. My UDP Pinger program client part will send a simple ping message to the server part to receive a corresponding pong message back. Then, my program will determine the delay between when the client sent the ping message and received the pong message. The delay time is named the Round Trip Time (RTT). And I assume the packet lost rate is 40% in the server, because UDP is an unreliable protocol. The client will wait up to one second for a reply, and print a lost message if it does not get a reply in one second. My program mainly implements two features: record the RTT with current time and show the packet lost.

Source code:

# Name: Shen, Yulin

# NetID: ys2542

# UDPPingerServer.py

# The code is completely taken from companion website for the textbook in www.pearsonhighered.com/cs-resources

# I add an option in terminal command line to manually set server port number

# I add two print methods to see messages and cases of packet lost

# We will need the following module to generate randomized lost packets

import sys

import random

from socket import \*

# Check terminal command and print the right format if wrong

if (len(sys.argv) != 2):

print('Format: python UDPPingerServer.py <Server\_Port\_Number>')

sys.exit()

# Create a UDP socket

# Notice the use of SOCK\_DGRAM for UDP packets

serverSocket = socket(AF\_INET, SOCK\_DGRAM)

# Assign IP address and port number to socket

port = int(sys.argv[1])

serverSocket.bind(('', port))

while True:

# Generate random number in the range of 0 to 10

rand = random.randint(0, 10)

# Receive the client packet along with the address it is coming from

message, address = serverSocket.recvfrom(1024)

# Capitalize the message from the client

message = message.upper()

# Print message for demo

print(message)

# If rand is less is than 4, we consider the packet lost and do not respond

if rand < 4:

# Print message for packet lost

print('Packet lost')

continue

# Otherwise, the server responds

serverSocket.sendto(message, address)

# Name: Shen, Yulin

# NetID: ys2542

# UDPPingerClient.py

# The code is imitated from UDPPingerServer.py

# We will need the following module to record Round Trip Time (RTT)

import sys

from socket import \*

from time import time, ctime

# Check terminal command and print the right format if wrong

if (len(sys.argv) != 2):

print('Format: python UDPPingerClient.py <Server\_Port\_Number>')

sys.exit()

# Create a UDP socket

# Notice the use of SOCK\_DGRAM for UDP packets

clientSocket = socket(AF\_INET, SOCK\_DGRAM)

# Assume client waits up to one second for a reply from the server

clientSocket.settimeout(1)

# Assign IP address and port number to socket

port = int(sys.argv[1])

serverSocket = ('', port)

# Send 10 ping messages to the target server over UDP

for i in range(10):

# Record the start time

sendTime = time()

# Create a message with the current time

message = 'Ping message ' + str(i+1) + ' ' + ctime(sendTime)[11:19]

# Send the message to the server

clientSocket.sendto(message.encode(), serverSocket)

# Successfully receive pong messages in one second

try:

# Wait pong messages from the server

pong, serverAddress = clientSocket.recvfrom(1024)

# Record the finish time

receiveTime = time()

# Calculate RTT

rtt = receiveTime - sendTime

# Print the pong message and Round Trip Time

print('Pong message:', pong.decode())

print('Round Trip Time:', rtt,'\n')

# Not receive pong messages due to request time out

except timeout:

# Print the time out message and the ping message number

print("Ping message %i request time out\n" %(i+1))

# Close the UDP socket

clientSocket.close()

Demo:

The left terminal window shows that I run the server code, and it constantly run for waiting the ping message from the client. The right terminal window shows the information in the client code. We can see sometimes packet lost happens.

