Computer Networks and Security Lab

18B15CS212

(ODD, 2021)

Assignment - 6

Socket Programming

Patil Amit Gurusidhappa

19104004

B11

Initially run this file

import socket

import time

import threading

import numpy

import pickle

#Local Dictionary:

n=0

m=0

local\_dict={'A':n,'B':m}

#Local LOG:

local\_log={'data':[]}

#Local 2DTT:

local\_2dtt = numpy.array([[0,0,0],[0,0,0],[0,0,0]])

with open('state\_severA.p', 'wb') as pfile:

    pickle.dump(local\_dict, pfile)

    pickle.dump(local\_log, pfile)

    pickle.dump(local\_2dtt, pfile)

with open('state\_severB.p', 'wb') as pfile:

    pickle.dump(local\_dict, pfile)

    pickle.dump(local\_log, pfile)

    pickle.dump(local\_2dtt, pfile)

with open('state\_severC.p', 'wb') as pfile:

    pickle.dump(local\_dict, pfile)

    pickle.dump(local\_log, pfile)

    pickle.dump(local\_2dtt, pfile)

Server A

# Server A

import socket

import time

import threading

import numpy

import pickle

# Declaring ips and ports

A\_ip = '127.0.0.1'

A\_port = 10011

B\_ip = '127.0.0.1'

B\_port = 20022

C\_ip = '127.0.0.1'

C\_port = 30033

def update(A\_socket\_speaks,local\_2dtt,local\_log):

    threading.Timer(0.5,update,args=(A\_socket\_speaks,local\_2dtt,local\_log,)).start()

    if networkWorks==True:

        table\_sent=numpy.array([local\_2dtt[0][0],local\_2dtt[0][1],local\_2dtt[0][2],local\_2dtt[2][0],local\_2dtt[2][1],local\_2dtt[2][2]])

        table\_sent\_str=str(table\_sent[0])+str(table\_sent[1])+str(table\_sent[2])+str(table\_sent[3])+str(table\_sent[4])+str(table\_sent[5])

        # Log send

        log\_array = pass\_log(local\_log)

        for i in log\_array:

            table\_sent\_str += str(i)

        A\_socket\_speaks.send(table\_sent\_str.encode())

        print('Message sent from Server A to B')

        with open('state\_severA.p', 'wb') as pfile:

            pickle.dump(local\_dict, pfile)

            pickle.dump(local\_log, pfile)

            pickle.dump(local\_2dtt, pfile)

def update\_table\_A(array,local\_2dtt):

    a=local\_2dtt[1][0]

    b=local\_2dtt[1][1]

    c=local\_2dtt[1][2]

    d=local\_2dtt[2][0]

    e=local\_2dtt[2][1]

    f=local\_2dtt[2][2]

    local\_2dtt[2][0]=max(array[3],d)

    local\_2dtt[2][1]=max(array[4],e)

    local\_2dtt[2][2]=max(array[5],f)

    local\_2dtt[1][0]=max(array[0],a)

    local\_2dtt[1][1]=max(array[1],b)

    local\_2dtt[1][2]=max(array[2],c)

    local\_2dtt[0][0]=max(max(local\_2dtt[1][0],local\_2dtt[2][0]),local\_2dtt[0][0])

    local\_2dtt[0][1]=max(max(local\_2dtt[1][1],local\_2dtt[2][1]),local\_2dtt[0][1])

    local\_2dtt[0][2]=max(max(local\_2dtt[1][2],local\_2dtt[2][2]),local\_2dtt[0][2])

def update\_log(array,local\_log,local\_2dtt):

    length\_array = len(array)

    length\_log = len(local\_log['data'])

    if length\_array < 6 or length\_array < length\_log+6:

        return

    supposed\_length\_log = sum(local\_2dtt[1][:])

    if supposed\_length\_log > length\_log:

        logs2add = supposed\_length\_log-length\_log

    else:

        return

    i=1

    a=0

    start = 6+length\_log

    while i:

        #print('First a\n',start,'\n')

        start = start+a

        if array[start]==0:

            local\_dict['A']+=1

            new\_log={}

            new\_log['Vote']='A'

            add\_log(new\_log)

        elif array[start]==1:

            local\_dict['B']+=1

            new\_log={}

            new\_log['Vote']='B'

            add\_log(new\_log)

        a+=1

        #print('Second a\n',start,'\n')

        if a>=logs2add:

            i=0

def update\_receive(connA,local\_2dtt,local\_log):

    threading.Timer(0.5,update\_receive,args=(connA,local\_2dtt,local\_log,)).start()

    time.sleep(2)

    data=connA.recv(512).decode()

    length\_data=len(data)

    i=1

    a=0

    array\_2dtt=[]

    while i:

        array\_2dtt.append(int(data[a]))

        a+=1

        if a>=length\_data:

            i=0

    if array\_2dtt:

        print('Message received from Server C\n')

    update\_log(array\_2dtt,local\_log,local\_2dtt)

    update\_table\_A(array\_2dtt,local\_2dtt)

    garbageCollection(local\_2dtt)

def garbageCollection(local\_2dtt):

    if local\_2dtt[0][0] == local\_2dtt[1][0] and local\_2dtt[1][0] == local\_2dtt[2][0]:

        local\_log={'data':[]}

def add\_log(new\_log):

    local\_log['data'].append(new\_log)

def voteA():

    local\_dict['A']+=1

    new\_log={}

    new\_log['Vote']='A'

    add\_log(new\_log)

    local\_2dtt[0][0]+=1

def voteB():

    local\_dict['B']+=1

    new\_log={}

    new\_log['Vote']='B'

    add\_log(new\_log)

    local\_2dtt[0][0]+=1

def pass\_log(local\_log):

    array = []

    for i in local\_log['data']:

        if i == {'Vote': 'A'}:

            array.append(0)

        elif i == {'Vote': 'B'}:

            array.append(1)

    return array

def printDict():

    print(local\_dict)

def printLog():

    print(local\_log['data'])

def printTable():

    print (local\_2dtt)

def command\_response(command):

    if command=='Vote,A':

        voteA()

    elif command=='Vote,B':

        voteB()

    elif command=='printDict':

        printDict()

    elif command=='printLog':

        printLog()

    elif command=='printTable':

        printTable()

def Main():

    #Local Dictionary:

    n=0

    m=0

    global local\_dict

    local\_dict={'A':n,'B':m}

    #Local LOG:

    global local\_log

    local\_log={'data':[]}

    #Local 2DTT:

    global local\_2dtt

    local\_2dtt = numpy.array([[0,0,0],[0,0,0],[0,0,0]])

    #Network Failure Emulator:

    global networkWorks

    networkWorks=True

    # Create sockets:

    # Hearing socket:

    A\_socket\_hears = socket.socket(socket.AF\_INET,socket.SOCK\_STREAM)  #Listening socket (listens to server C).

    A\_socket\_hears.bind((A\_ip,A\_port))

    A\_socket\_hears.listen(1)

    print("\nServer A ready and listening...\n")

    connA, addr = A\_socket\_hears.accept()

    # Speaking socket:

    A\_socket\_speaks = socket.socket()   #Destination socket: from server A to server B:

    A\_socket\_speaks.connect((B\_ip,B\_port))

    with open('state\_severA.p', 'rb') as pfile:

        local\_dict = pickle.load(pfile)

        local\_log = pickle.load(pfile)

        local\_2dtt = pickle.load(pfile)

    update(A\_socket\_speaks,local\_2dtt,local\_log)

    update\_receive(connA,local\_2dtt,local\_log)

    while True:

        print "\n----------------------------------------\n"

        command=raw\_input("ENTER COMMAND/ACTION FOR SERVER: \n----------------------------------------\n")

        if command=='networkFail':

            networkWorks=False

        elif command=='networkWorks':

            networkWorks=True

        command\_response(command)

    A\_socket\_hears.close()

if \_\_name\_\_ == '\_\_main\_\_':

        Main()

Sever B

# Server B

import socket

import time

import threading

import numpy

import pickle

# Declaring ips and ports

A\_ip = '127.0.0.1'

A\_port = 10011

B\_ip = '127.0.0.1'

B\_port = 20022

C\_ip = '127.0.0.1'

C\_port = 30033

def update(B\_socket\_speaks,local\_2dtt,local\_log):

    threading.Timer(0.5,update,args=(B\_socket\_speaks,local\_2dtt,local\_log,)).start()

    if networkWorks==True:

        table\_sent=numpy.array([local\_2dtt[0][0],local\_2dtt[0][1],local\_2dtt[0][2],local\_2dtt[1][0],local\_2dtt[1][1],local\_2dtt[1][2]])

        table\_sent\_str=str(table\_sent[0])+str(table\_sent[1])+str(table\_sent[2])+str(table\_sent[3])+str(table\_sent[4])+str(table\_sent[5])

        # Log send

        log\_array = pass\_log(local\_log)

        for i in log\_array:

            table\_sent\_str += str(i)

        B\_socket\_speaks.send(table\_sent\_str.encode())

        print('\nMessage sent from Server B to C')

        with open('state\_severB.p', 'wb') as pfile:

            pickle.dump(local\_dict, pfile)

            pickle.dump(local\_log, pfile)

            pickle.dump(local\_2dtt, pfile)

def update\_table\_B(array,local\_2dtt):

    a=local\_2dtt[0][0]

    b=local\_2dtt[0][1]

    c=local\_2dtt[0][2]

    d=local\_2dtt[2][0]

    e=local\_2dtt[2][1]

    f=local\_2dtt[2][2]

    local\_2dtt[2][0]=max(array[3],d)

    local\_2dtt[2][1]=max(array[4],e)

    local\_2dtt[2][2]=max(array[5],f)

    local\_2dtt[0][0]=max(array[0],a)

    local\_2dtt[0][1]=max(array[1],b)

    local\_2dtt[0][2]=max(array[2],c)

    local\_2dtt[1][0]=max(max(local\_2dtt[0][0],local\_2dtt[2][0]),local\_2dtt[1][0])

    local\_2dtt[1][1]=max(max(local\_2dtt[0][1],local\_2dtt[2][1]),local\_2dtt[1][1])

    local\_2dtt[1][2]=max(max(local\_2dtt[0][2],local\_2dtt[2][2]),local\_2dtt[1][2])

def update\_log(array,local\_log,local\_2dtt):

    length\_array = len(array)

    length\_log = len(local\_log['data'])

    if length\_array < 6 or length\_array < length\_log+6:

        return

    supposed\_length\_log = sum(local\_2dtt[1][:])

    if supposed\_length\_log > length\_log:

        logs2add = supposed\_length\_log-length\_log

    else:

        return

    i=1

    a=0

    start = 6+length\_log

    while i:

        start = start+a

        if array[start]==0:

            local\_dict['A']+=1

            new\_log={}

            new\_log['Vote']='A'

            add\_log(new\_log)

        elif array[start]==1:

            local\_dict['B']+=1

            new\_log={}

            new\_log['Vote']='B'

            add\_log(new\_log)

        a+=1

        if a>=logs2add:

            i=0

def update\_receive(connB,local\_2dtt,local\_log):

    threading.Timer(0.5,update\_receive,args=(connB,local\_2dtt,local\_log,)).start()

    time.sleep(2)

    data=connB.recv(512).decode()

    length\_data=len(data)

    i=1

    a=0

    array\_2dtt=[]

    while i:

        array\_2dtt.append(int(data[a]))

        a+=1

        if a>=length\_data:

            i=0

    if array\_2dtt:

        print('Message received from Server A\n')

    update\_log(array\_2dtt,local\_log,local\_2dtt)

    update\_table\_B(array\_2dtt,local\_2dtt)

    garbageCollection(local\_2dtt)

def garbageCollection(local\_2dtt):

    if local\_2dtt[0][1] == local\_2dtt[1][1] and local\_2dtt[1][1] == local\_2dtt[2][1]:

        local\_log={'data':[]}

def add\_log(new\_log):

    local\_log['data'].append(new\_log)

def voteA():

    local\_dict['A']+=1

    new\_log={}

    new\_log['Vote']='A'

    add\_log(new\_log)

    local\_2dtt[1][1]+=1

def voteB():

    local\_dict['B']+=1

    new\_log={}

    new\_log['Vote']='B'

    add\_log(new\_log)

    local\_2dtt[1][1]+=1

def pass\_log(local\_log):

    array = []

    for i in local\_log['data']:

        if i == {'Vote': 'A'}:

            array.append(0)

        elif i == {'Vote': 'B'}:

            array.append(1)

    return array

def printDict():

    print(local\_dict)

def printLog():

    print(local\_log['data'])

def printTable():

    print (local\_2dtt)

def command\_response(command):

    if command=='Vote,A':

        voteA()

    elif command=='Vote,B':

        voteB()

    elif command=='printDict':

        printDict()

    elif command=='printLog':

        printLog()

    elif command=='printTable':

        printTable()

def Main():

    #Local Dictionary:

    n=0

    m=0

    global local\_dict

    local\_dict={'A':n,'B':m}

    #Local LOG:

    global local\_log

    local\_log={'data':[]}

    #Local 2DTT:

    global local\_2dtt

    local\_2dtt = numpy.array([[0,0,0],[0,0,0],[0,0,0]])

    #Network Failure Emulator:

    global networkWorks

    networkWorks=True

    # Create sockets:

    # Hearing socket:

    B\_socket\_hears = socket.socket(socket.AF\_INET,socket.SOCK\_STREAM)  #Listening socket (listens to server A).

    B\_socket\_hears.bind((B\_ip,B\_port))

    B\_socket\_hears.listen(1)

    print("\nServer B ready and listening...\n")

    connB, addr = B\_socket\_hears.accept()

    # Speaking socket:

    B\_socket\_speaks = socket.socket()   #Destination socket: from server B to server C:

    B\_socket\_speaks.connect((C\_ip,C\_port))

    with open('state\_severB.p', 'rb') as pfile:

        local\_dict = pickle.load(pfile)

        local\_log = pickle.load(pfile)

        local\_2dtt = pickle.load(pfile)

    update(B\_socket\_speaks,local\_2dtt,local\_log)

    update\_receive(connB,local\_2dtt,local\_log)

    while True:

        print "\n----------------------------------------\n"

        command=raw\_input("ENTER COMMAND/ACTION FOR SERVER: \n----------------------------------------\n")

        if command=='networkFail':

            networkWorks=False

        elif command=='networkWorks':

            networkWorks=True

        command\_response(command)

    B\_socket\_hears.close()

if \_\_name\_\_ == '\_\_main\_\_':

        Main()

Server C

# Server C

import socket

import time

import threading

import numpy

import pickle

# Declaring ips and ports

A\_ip = '127.0.0.1'

A\_port = 10011

B\_ip = '127.0.0.1'

B\_port = 20022

C\_ip = '127.0.0.1'

C\_port = 30033

def update(C\_socket\_speaks,local\_2dtt,local\_log):

    threading.Timer(0.5,update,args=(C\_socket\_speaks,local\_2dtt,local\_log,)).start()

    if networkWorks==True:

        table\_sent=numpy.array([local\_2dtt[1][0],local\_2dtt[1][1],local\_2dtt[1][2],local\_2dtt[2][0],local\_2dtt[2][1],local\_2dtt[2][2]])

        table\_sent\_str=str(table\_sent[0])+str(table\_sent[1])+str(table\_sent[2])+str(table\_sent[3])+str(table\_sent[4])+str(table\_sent[5])

        # Log send

        log\_array = pass\_log(local\_log)

        for i in log\_array:

            table\_sent\_str += str(i)

        C\_socket\_speaks.send(table\_sent\_str.encode())

        print('\nMessage sent from Server C to A')

        with open('state\_severC.p', 'wb') as pfile:

            pickle.dump(local\_dict, pfile)

            pickle.dump(local\_log, pfile)

            pickle.dump(local\_2dtt, pfile)

def update\_table\_C(array,local\_2dtt):

    a=local\_2dtt[0][0]

    b=local\_2dtt[0][1]

    c=local\_2dtt[0][2]

    d=local\_2dtt[1][0]

    e=local\_2dtt[1][1]

    f=local\_2dtt[1][2]

    local\_2dtt[0][0]=max(array[0],a)

    local\_2dtt[0][1]=max(array[1],b)

    local\_2dtt[0][2]=max(array[2],c)

    local\_2dtt[1][0]=max(array[3],d)

    local\_2dtt[1][1]=max(array[4],e)

    local\_2dtt[1][2]=max(array[5],f)

    local\_2dtt[2][0]=max(max(local\_2dtt[0][0],local\_2dtt[1][0]),local\_2dtt[2][0])

    local\_2dtt[2][1]=max(max(local\_2dtt[0][1],local\_2dtt[1][1]),local\_2dtt[2][1])

    local\_2dtt[2][2]=max(max(local\_2dtt[0][2],local\_2dtt[1][2]),local\_2dtt[2][2])

def update\_log(array,local\_log,local\_2dtt):

    length\_array = len(array)

    length\_log = len(local\_log['data'])

    if length\_array < 6 or length\_array < length\_log+6:

        return

    supposed\_length\_log = sum(local\_2dtt[1][:])

    if supposed\_length\_log > length\_log:

        logs2add = supposed\_length\_log-length\_log

    else:

        return

    i=1

    a=0

    start = 6+length\_log

    while i:

        start = start+a

        if array[start]==0:

            local\_dict['A']+=1

            new\_log={}

            new\_log['Vote']='A'

            add\_log(new\_log)

        elif array[start]==1:

            local\_dict['B']+=1

            new\_log={}

            new\_log['Vote']='B'

            add\_log(new\_log)

        a+=1

        if a>=logs2add:

            i=0

def update\_receive(connC,local\_2dtt,local\_log):

    threading.Timer(0.5,update\_receive,args=(connC,local\_2dtt,local\_log,)).start()

    time.sleep(2)

    data=connC.recv(512).decode()

    length\_data=len(data)

    i=1

    a=0

    array\_2dtt=[]

    while i:

        array\_2dtt.append(int(data[a]))

        a+=1

        if a>=length\_data:

            i=0

    if array\_2dtt:

        print('Message received from Server B\n')

    update\_log(array\_2dtt,local\_log,local\_2dtt)

    update\_table\_C(array\_2dtt,local\_2dtt)

    garbageCollection(local\_2dtt)

def garbageCollection(local\_2dtt):

    if local\_2dtt[0][2] == local\_2dtt[1][2] and local\_2dtt[1][2] == local\_2dtt[2][2]:

        local\_log={'data':[]}

def add\_log(new\_log):

    local\_log['data'].append(new\_log)

def voteA():

    local\_dict['A']+=1

    new\_log={}

    new\_log['Vote']='A'

    add\_log(new\_log)

    local\_2dtt[2][2]+=1

def voteB():

    local\_dict['B']+=1

    new\_log={}

    new\_log['Vote']='B'

    add\_log(new\_log)

    local\_2dtt[2][2]+=1

def pass\_log(local\_log):

    array = []

    for i in local\_log['data']:

        if i == {'Vote': 'A'}:

            array.append(0)

        elif i == {'Vote': 'B'}:

            array.append(1)

    return array

def printDict():

    print(local\_dict)

def printLog():

    print(local\_log['data'])

def printTable():

    print (local\_2dtt)

def command\_response(command):

    if command=='Vote,A':

        voteA()

    elif command=='Vote,B':

        voteB()

    elif command=='printDict':

        printDict()

    elif command=='printLog':

        printLog()

    elif command=='printTable':

        printTable()

def Main():

    #Local Dictionary:

    n=0

    m=0

    global local\_dict

    local\_dict={'A':n,'B':m}

    #Local LOG:

    global local\_log

    local\_log={'data':[]}

    #Local 2DTT:

    global local\_2dtt

    local\_2dtt = numpy.array([[0,0,0],[0,0,0],[0,0,0]])

    #Network Failure Emulator:

    global networkWorks

    networkWorks=True

    # Create sockets:

    # Speaking socket:

    C\_socket\_speaks = socket.socket()   #Destination socket: from server C to server A:

    C\_socket\_speaks.connect((A\_ip,A\_port))

    # Hearing socket:

    C\_socket\_hears = socket.socket(socket.AF\_INET,socket.SOCK\_STREAM)  #Listening socket (listens to server B).

    C\_socket\_hears.bind((C\_ip,C\_port))

    C\_socket\_hears.listen(1)

    print("\nServer C ready and listening...\n")

    connC, addr = C\_socket\_hears.accept()

    with open('state\_severC.p', 'rb') as pfile:

        local\_dict = pickle.load(pfile)

        local\_log = pickle.load(pfile)

        local\_2dtt = pickle.load(pfile)

    update(C\_socket\_speaks,local\_2dtt,local\_log)

    update\_receive(connC,local\_2dtt,local\_log)

    while True:

        print "\n----------------------------------------\n"

        command=raw\_input("ENTER COMMAND/ACTION FOR SERVER: \n----------------------------------------\n")

        if command=='networkFail':

            networkWorks=False

        elif command=='networkWorks':

            networkWorks=True

        command\_response(command)

    C\_socket\_hears.close()

if \_\_name\_\_ == '\_\_main\_\_':

        Main()