SYSTEM AND NETWORK PROGRAMMING LAB

MINOR PROJECT
(VIth Semester)

Network Collaboration Toolkit

Submitted by:-
ANKIT KUMAR-13103605(B6)
PRANAV SHUKLA-13103610(B6)
MADHUR MITTAL-13103614(B6)
PARITOSH DUBEY-13103620(B6)

ABSTRACT

In most enterprises, we have a setup where there is one main computer which is handled by the supervisor and all the other computers are connected to it having limited capabilities for the employees. Such systems require a NETWORK COLLABORATION TOOLKIT for the supervisor or admin to remotely access, control, monitor, and share with and communicate with other devices.
The whole PC-based systems, which are available presently, are all running through the Network type communication. This type of system is mainly used in net centers, offices, mainly in industries. It is more time consuming to control all the systems connected in a network manually. Network Administrator Tool provides remote service to its entire client over the network. It acts as a network administrator to its client to provide remote service like remote shutdown, remote logoff, remote file transfer, remote desktop sharing and remote messaging.

OBJECTIVES:

1. Remote File transfer

Remote file transfer module provides file transfer operation from server to the requested client. First, it receives the request and recognizes the file name, if the file exists in server, transfers it otherwise sends a file not found error message. *Remote File Transfer* contains information for transferring files to the remote location. Remote file transfers are used to transfer files from the development environment to a server that can be installed locally or remotely. We can transfer files between two connected PCs and chat with the remote user. Transfer files and folders between local and remotely accessed computers.

2. Remote Desktop Sharing

Remote Desktop Sharing module provides the administrators to gain access to remote Windows desktops in the network. It is a tool enabling access from anywhere in the network without requiring any native client. Administrator can directly access the remote system by sharing the requested system desktop. Desktop Sharing is a server application that allows sharing current session with a user on another machine, who can use a client to view or even control the desktop.

Desktop Sharing lets users call a remote computer to access its shared desktop and applications. With the desktop sharing we can operate our office computer from our home or vice versa.

3. Remote Messaging

Remote Messaging is a small application that facilitates communication between different hosts on the same local area network. It does not require a central server and uses very little bandwidth by taking advantage of a lightweight protocol and UDP packets. Administrator can communicate with the remote systems that are connected with in the local network administrator can communicate publicly or privately. Messaging is nothing but passing data to and from applications over the network which makes the synchronization of data simple. Messaging allows users across the network to exchange data in real time.

WORK DIVISION:

We shared different modules to work on with each other, giving each one of module to a group member. We independently worked on our projects and once done, the other two refactored and tested the module of the other. The work division was as follows:

1. Remote File Sharing: Pranav Shukla
2. Remote Messaging: Paritosh Dubey & Ankit Kumar
3. Remote Desktop Sharing & Controlling: Pranav Shukla, Paritosh Dubey, Ankit Kumar and Madhur Mittal
4. Documentation: Ankit Kumar & Paritosh Dubey

BACKGROUND STUDY AND FINDING:

There are many such applications in market such as VNC or RDPY, but each of these offer separate modules. What our application offers is an all in one approach to network administration, where one could remotely see (as in VNC and TeamViewer), send files (as in FileZilla), chat (as in WIFI Messenger) and

remotely control as well. Though these tools are much more complex, but their basic functionalities remain same.
Findings from these other projects were that all these applications essentially use TCP for making connections, Python is a preferred language for network programming, and there will be a limit to number of connections we can make from server to client.

DESIGNING:

We used the following languages and libraries for our project (as of now):
1. Pyhton (2.7 and up): For programming the logic, structure and design

2. Win32Api: For GUI Automation on Windows

3. Tkinter: For GUI Implementation

4. Os: For system functions

5. Socket: For Network Programming

6. Pillow(PIL): For Image Processing and graphic capabilities

7. pyftpdlib: For sharing file in FTP

8. Thread: For multithreading of multiple clients on server

9. Logging: for event logging of libraries and chats

10. Time, Sys: Others

11. More as required by the application

The project design consists of two basic tiers:
1. User Interface: Created using Python's Tkinter Library, which will provide the starting menu and subsequent options for choosing if server or client. Next would be the applications programmed for each specific task of remote chatting, desktop sharing or file transfer.

2. Application Logic: To handle user's request, three independent programs concerning each module is applied at back end. Authentication is also verified.

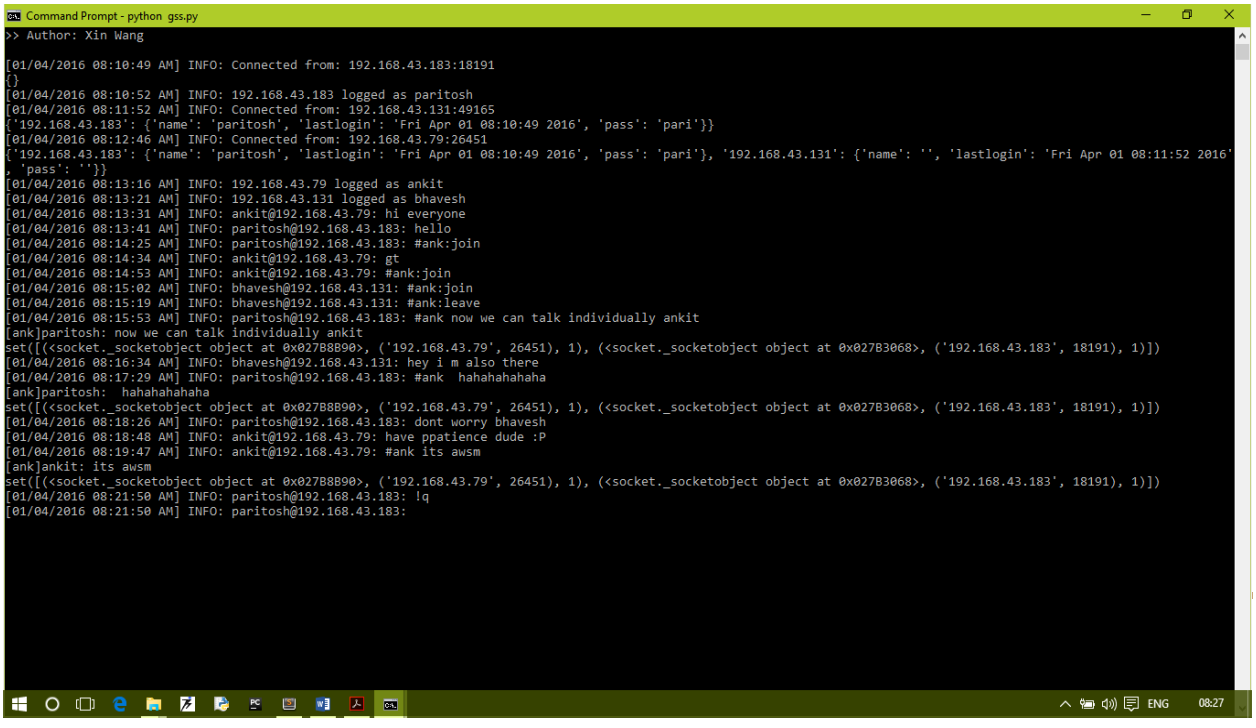
TESTING:

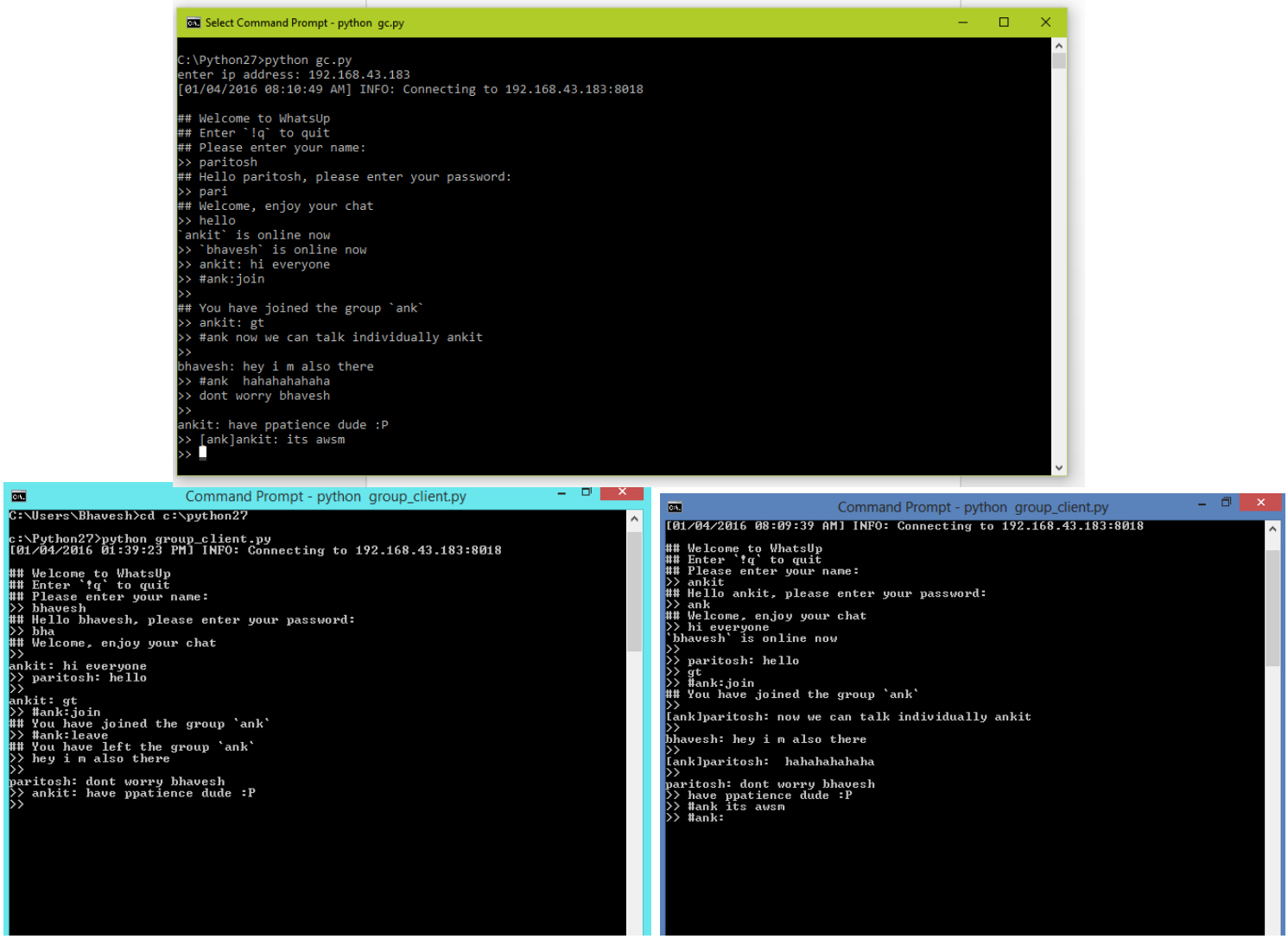
We tested all the three out of four modules and all ran successfully.
The File Transfer module was successfully able to send .txt file,.jpg file, .mp3 file, .avi/.mp4 file during the tests. It was also able to search correctly for the file in the computer to be shared.
The Remote Desktop Sharing module was able to successfully transmit desktop view after every 3 to 4 seconds effectively and seamlessly. Mouse pointer reporting could be improved.
Remote Messaging service allowed as many as three clients to chat on the test run through Wifi hotspot of our mobile phones. It also shows Last Logged In.
Messages sent to a particular user are being stored and displayed each time the user logs back on and as long as the server continues to run.

CODE SNAPSHOTS:

1. REMOTE MESSAGING:

1 Server and 3 Clients





Gss.py

```
import socket
import threading
import time
import logging

HOST = ''
PORT = 8018
TIMEOUT = 5
BUF_SIZE = 1024

class WhatsUpServer(threading.Thread):
    global clients
    global messages
```

```
        global accounts
        global onlines
        global groups
        global group_name

        def __init__(self, conn, addr):
            threading.Thread.__init__(self)
            self.conn = conn
            self.addr = addr
            self.ip = self.addr[0]
            self.name = ''
            self.flag = 0

        def print_indicator(self, prompt):
            self.conn.send('%s\n>> ' % (prompt,))

        def login(self):
            logging.info('Connected from: %s:%s' %(self.addr[0], self.addr[1]))
            msg = '\n## Welcome to WhatsUp\n## Enter `!q` to quit\n'

            # new user
            print accounts
            if self.ip not in accounts:
                msg += '## Please enter your name:'
                self.print_indicator(msg)
                accounts[self.ip] = {
                    'name': '',
                    'pass': '',
                    'lastlogin': time.ctime()
                    }

                while 1:
                    name = self.conn.recv(BUF_SIZE).strip()
                    if name in messages:
                        self.print_indicator(
                            '## This name already exists, please try another')
                    else:
                        break
                accounts[self.ip]['name'] = name
                self.name = name
                logging.info('%s logged as %s' % (self.addr[0], self.name))
                messages[name] = []
                self.print_indicator('## Hello %s, please enter your password:' %
(self.name,))
                password = self.conn.recv(BUF_SIZE)
```

```
                accounts[self.ip]['pass'] = password.strip()
                self.flag=1
                self.print_indicator('## Welcome, enjoy your chat')
                clients.add((self.conn, self.addr,self.flag))
            else:
                self.name = accounts[self.ip]['name']
                msg += '## Hello %s, please enter your password:' % (self.name,)
                # print accounts
                self.print_indicator(msg)
                while 1:
                    password = self.conn.recv(BUF_SIZE).strip()
                    if password != accounts[self.ip]['pass']:
                        self.print_indicator('## Incorrect password, please enter again')
                    else:
                        self.print_indicator('## Welcome back, last login: %s'
%(accounts[self.ip]['lastlogin'],))
                        accounts[self.ip]['lastlogin'] = time.ctime()
                        break
                self.conn.send(self.show_mentions(self.name))
            self.broadcast('`%s` is online now' % (self.name,), clients, False)
            onlines[self.name] = self.conn

        def logoff(self):
            self.conn.send('## Bye!\n')
            print online
            msg=self.name+" is offline now!"
            print msg
            print "1"
#           del onlines[self.name]
            print "2"
            clients.remove((self.conn, self.addr, self.flag))
            #self.broadcast('## `%s` is offline now' %(self.name,), clients)
            for conn, addr, flag in clients:
                print "22222"
                conn.send(msg)
            #self.broadcast(msg, clients,False)
            self.conn.close()
            exit()

        def check_keyword(self, buf):
            global onlines
            if buf.find('!q') == 0:
                self.logoff()

            if buf.find('#') == 0:
```

```
            group_keyword = buf.split(' ')[0][1:]
            group_component = group_keyword.split(':')

            # to post in a group
            if len(group_component) == 1:
                group_name = group_component[0] #ankit
                try:
                    msg = '[%s]%s: %s' % (
                        group_name, self.name, buf.split(' ', 1)[1])
                    print msg
                    self.group_post(group_name, msg)
                except IndexError:
                    self.print_indicator('## What do you want to do with `#%s`?' %
(group_name))

            # to join / leave a group
            elif len(group_component) == 2: #joining
                group_name = group_component[0] #group_name=ankit
                if group_component[1] == 'join':
                    self.group_join(group_name) #function called group_join to add
member
                elif group_component[1] == 'leave':
                    self.group_leave(group_name)
            return True

        if buf.find('@') == 0:
            to_user = buf.split(' ')[0][1:]
            from_user = self.name
            msg = buf.split(' ', 1)[1]

            # if user is online
            if to_user in onlines:
                onlines[to_user].send('@%s: %s\n>> ' % (from_user, msg))
                self.mention(from_user, to_user, msg, 1)
            # offline
            else:
                self.mention(from_user, to_user, msg)
            return True

    def group_post(self, group_name, msg):
        # if the group does not exist, create it
        groups.setdefault(group_name, set())

        # if current user is a member of the group
        if (self.conn, self.addr,self.flag) in groups[group_name]:
```

```
            print groups[group_name]
            self.broadcast(msg, groups[group_name])
        else:
            self.print_indicator(
                '## You are currently not a member of group `%s`' % (group_name,))

    def group_join(self, group_name):
        groups.setdefault(group_name, set())
        groups[group_name].add((self.conn, self.addr,self.flag))
        self.print_indicator('## You have joined the group `%s`' %
                             (group_name,))

    def group_leave(self, group_name):
        try:
            groups[group_name].remove((self.conn, self.addr,self.flag))
            self.print_indicator('## You have left the group `%s`' %
                                 (group_name,))
        except Exception, e:
            pass

    def mention(self, from_user, to_user, msg, read=0):
        if to_user in messages:
            messages[to_user].append([from_user, msg, read])
            self.print_indicator('## Message has sent to %s' % (to_user,))
        else:
            self.print_indicator('## No such user named `%s`' % (to_user,))

    def show_mentions(self, name):
        res = '## Here are your messages:\n'
        if not messages[name]:
            res += '   No messages available\n>> '
            return res
        for msg in messages[name]:
            if msg[2] == 0:
                res += '(NEW) %s: %s\n' % (msg[0], msg[1])
                msg[2] = 1
            else:
                res += '      %s: %s\n' % (msg[0], msg[1])
        res += '>> '
        return res

    def broadcast(self, msg, receivers, to_self=True):
        for conn, addr, flag in receivers:
            # if the client is not the current user
            if addr[0] != self.ip and flag==1:
```

```
                conn.send(msg + '\n>> ')
            # if current user
            else:
                self.conn.send('>> ') if to_self else self.conn.send('')

    def run(self):
        self.login()

        while 1:
            try:
                self.conn.settimeout(TIMEOUT)
                buf = self.conn.recv(BUF_SIZE).strip()
                logging.info('%s@%s: %s' % (self.name, self.addr[0], buf))
                # check features
                if not self.check_keyword(buf):
                    # client broadcasts message to all
                    self.broadcast('%s: %s' % (self.name, buf), clients)

            except Exception, e:
                # timed out
                pass

def main():
    global clients
    global messages
    global accounts
    global onlines
    global groups

    # logging setup
    logging.basicConfig(level=logging.INFO,
                        format='[%(asctime)s] %(levelname)s: %(message)s',
                        datefmt='%d/%m/%Y %I:%M:%S %p')

    # initialize global vars
    clients = set()
    messages = {}
    accounts = {}
    onlines = {}
    groups = {}

    # set up socket
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    sock.bind((HOST, PORT))
```

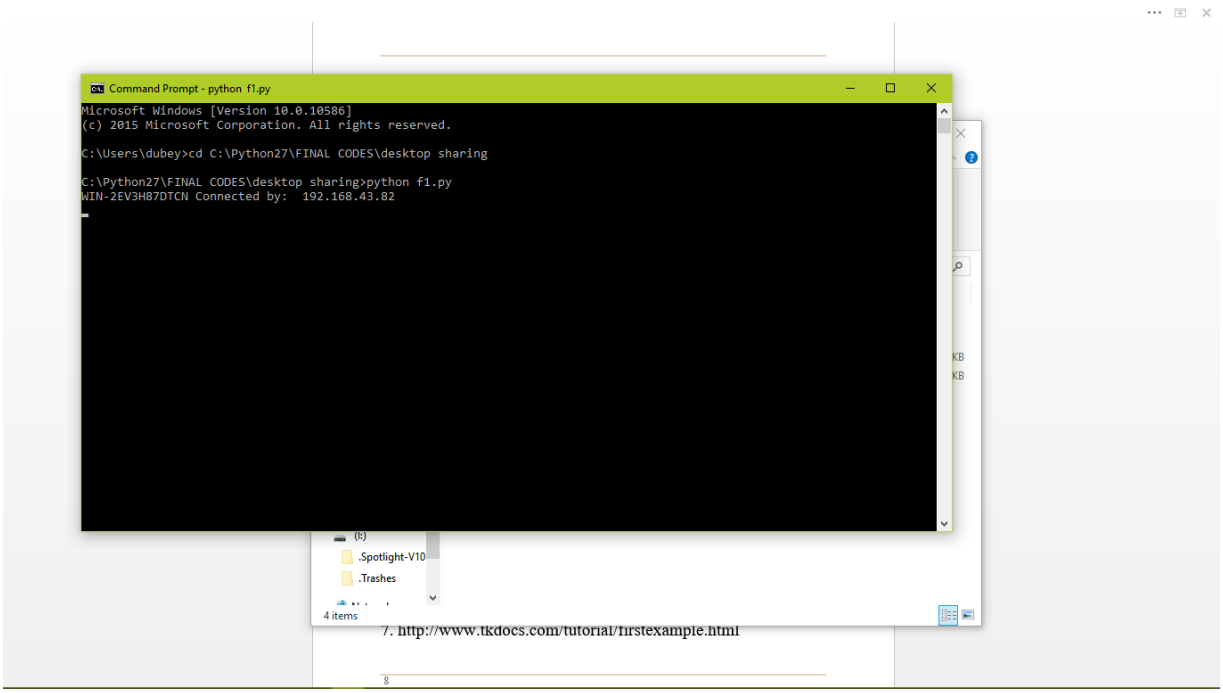
```
    sock.listen(5)
    print '-= WhatsUp Server =-'
    print '>> Listening on:', PORT
    print '>> Author: Xin Wang'
    print ''

    while 1:
        try:
            conn, addr = sock.accept()
            server = WhatsUpServer(conn, addr)
            server.start()
        except Exception, e:
            print e

if __name__ == '__main__':
    try:
        main()
    except KeyboardInterrupt:
        print 'Quited'
```

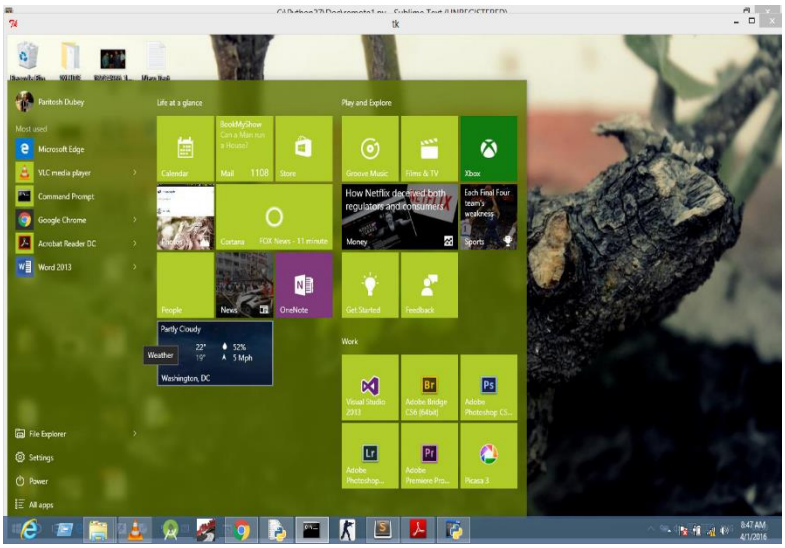
gc.py

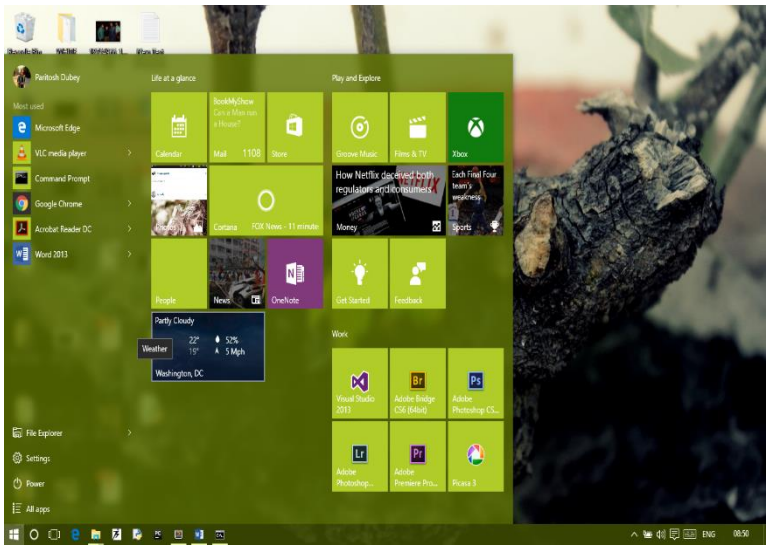
```
import socket
import time
import logging
import sys

HOST = raw_input('enter ip address: ')
PORT = 8018
TIMEOUT = 5
BUF_SIZE = 1024


class WhatsUpClient():

    def __init__(self, host=HOST, port=PORT):
        self.sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.sock.connect((host, port))
        logging.info('Connecting to %s:%s' % (host, port))
        while 1:
            try:
                buf = self.sock.recv(BUF_SIZE)
                sys.stdout.write(buf)
                cmd = raw_input()
                if str(cmd.strip()) == '!q':
                    self.sock.send(cmd.strip())
```

```
                buf = self.sock.recv(BUF_SIZE)
                sys.stdout.write(buf)
                self.sock.close()
            self.sock.send(cmd)
        except:
            self.sock.close()

    def run(self):
        pass


def main():
    logging.basicConfig(level=logging.INFO,
                        format='[%(asctime)s] %(levelname)s: %(message)s',
                        datefmt='%d/%m/%Y %I:%M:%S %p')
    client = WhatsUpClient()

if __name__ == '__main__':
    main()
```

2. Remote Desktop Sharing







F1.py

```
from PIL import ImageGrab
from subprocess import *
from Tkinter import *
from socket import *
import win32api, win32con
import Image, ImageTk
import thread
import os

PORT = 9000
LPORT = PORT + 10
liveuser = []
cflag = 0
sock = socket(AF_INET, SOCK_STREAM)
sock.bind(("", PORT))
sock.setsockopt(SOL_SOCKET, SO_REUSEADDR, 1)
sock.listen(1)

#socket for checking live nodes
slive = socket(AF_INET, SOCK_DGRAM)
slive.bind(("", LPORT))

def main():
    global cflag
    #starting thread for accepting msg from live nodes
    #thread.start_new_thread(recieve,())
```

```
    #sending msg to all nodes connected
    #connected()
    #main code
    conn,addr = sock.accept()
    print gethostname() + ' Connected by: ',gethostbyname(addr[0])
    #print '1'
    while True:
        try:
            #taking screenshot
            ImageGrab.grab().save("images\\img1.jpg", "JPEG")
            #sending image to client
            fp = open("images\\img1.jpg","rb")
            data = fp.read()
            fp.close()
            conn.sendall(data)
            #print '2'
            #recieving mouse coordinates or keypressed
            rec = conn.recv(1024)
            #print rec
            while rec != "start":
                if '~' in rec:
                    lr = rec[0]
                    rec = rec[1:]
                #    print '3'
                    x,y = map(int, rec.split('~'))
                    #mouse pos. set nd single click done
                    win32api.SetCursorPos((x,y))
                    if lr == 'l':

win32api.mouse_event(win32con.MOUSEEVENTF_LEFTDOWN,x,y,0,0)
                        win32api.mouse_event(win32con.MOUSEEVENTF_LEFTUP,x,y,0,0)
                    elif lr == 'r':

win32api.mouse_event(win32con.MOUSEEVENTF_RIGHTDOWN,x,y,0,0)
                        win32api.mouse_event(win32con.MOUSEEVENTF_RIGHTUP,x,y,0,0)
                elif rec == 'close':
                    cflag = 1
                    break
                elif rec:
                    keypress = int(rec)
                  #  print '4'
                    #particular key pressed
                    win32api.keybd_event(keypress,0,0,0)
                rec = conn.recv(1024)
        except:
```

```
            continue
        if cflag == 1:
            break

main()
for i in liveuser:
    slive.sendto("going",(i,LPORT))
#    print '5'
```

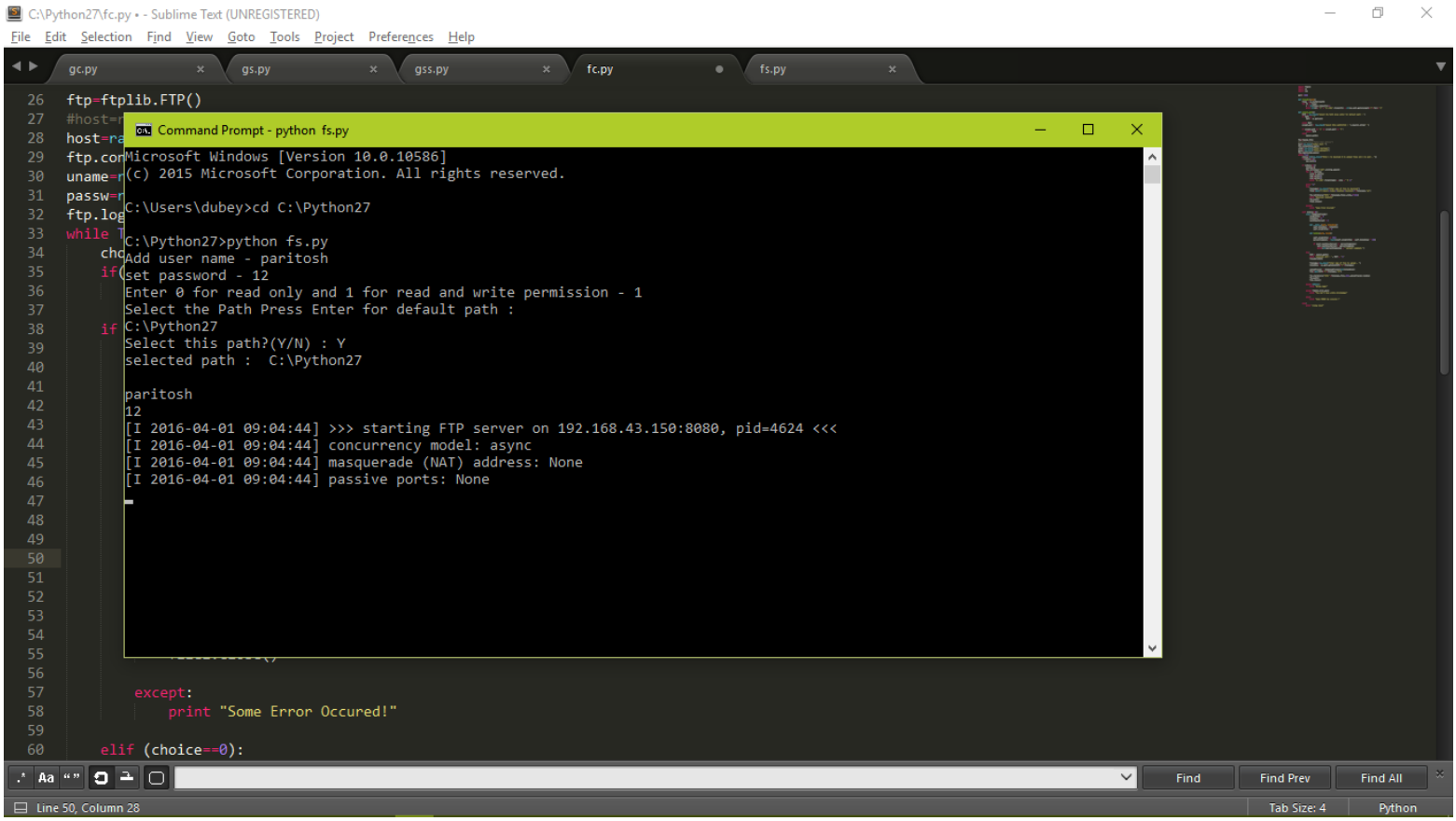
Remote1.py

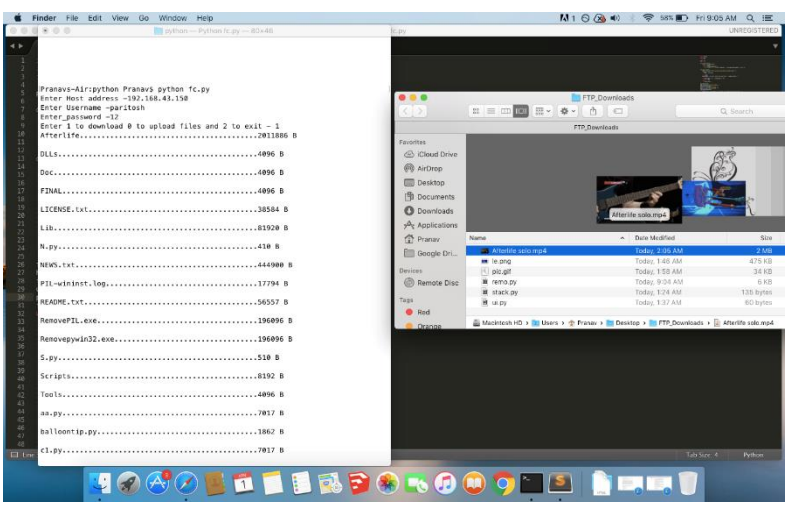
```
from PIL import ImageGrab
from subprocess import *
from Tkinter import *
from socket import *
from PIL import Image, ImageTk
import thread
import os
import cStringIO
PORT = 9000
LPORT = PORT + 10
liveuser = []
cflag = 0

sock = socket(AF_INET,SOCK_STREAM)
sock.setsockopt(SOL_SOCKET,SO_REUSEADDR,1)
#socket for checking live nodes
slive = socket(AF_INET, SOCK_DGRAM)
#connecting with server
sock.connect(("192.168.43.79",PORT))
print "1"

def start(image2):
    #mouse is clicked
    def leftclick(event):
        #outputting x and y coords to console
        x = event.x
        y = event.y
        sock.send('l' + str(x) + '~' + str(y))
        #root.quit()

    def rightclick(event):
        #outputting x and y coords to console
        x = event.x
        y = event.y
        sock.send('r' + str(x) + '~' + str(y))
```

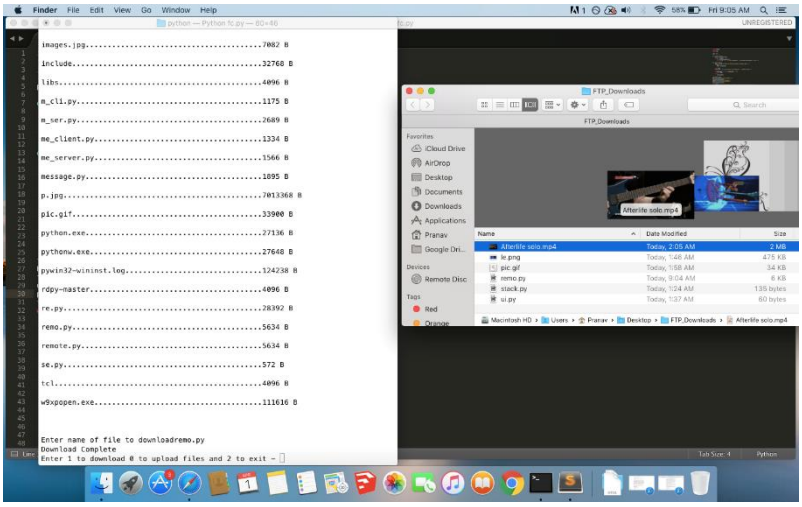
```
        #root.quit()

    #key is pressed
    def key(event):
        keypress = event.keycode
        sock.send(str(keypress))
        #root.quit()

    def image():
        root.quit()

    try:
        #adding the image
        print "an"
        img = Image.open(cStringIO.StringIO(image2))
        print "bn"
        img = img.resize((root.winfo_screenwidth(),root.winfo_screenheight()-
50),Image.ANTIALIAS)
        print "cn"
        img = ImageTk.PhotoImage(img)
        label.config(image = img)

        #mouseclick and keyboard event
        label.bind("<Button-1>",leftclick)
        label.bind("<Button-3>",rightclick)
        label.bind("<Key>", key)
        label.pack()
        root.focus_set()
        label.focus_set()

        #updating img after every 3 sec.
        root.after(2000,image)
        root.mainloop()
        return 0
    except:
        #print 'po'
        return 0

#image display gui
root = Tk()
root.geometry("%dx%d"%(root.winfo_screenwidth(), root.winfo_screenheight()-50))
label = Label(root)
print "2"

while True:
```

```
        print "3"
        msg = sock.recv(256456)
        print "4"
        image1 = msg
        #img = Image.open(cStringIO.StringIO(image1))
        #img.show()
        print "5"
        if start(image1) == 0:
            sock.send("start")
        else:
            sock.send("close")
            break
    print 3

    sock.close()
```

3.Remote File Transfer







Fs.py

```
from pyftpdlib.authorizers import DummyAuthorizer
from pyftpdlib.handlers import FTPHandler
from pyftpdlib.servers import FTPServer
import os

def select_path():
        PATH = raw_input('Select the Path Press Enter for default path : ')
        if PATH == '':
                PATH = os.getcwd()

        print PATH
        accept_path = raw_input('Select this path?(Y/N) : ').lower().strip(' ')

        if accept_path == 'y' or accept_path == 'Y':
                return PATH
        else:
                select_path()

authorizer = DummyAuthorizer()
uname=raw_input("Add user name - ")
passw=raw_input("set password - ")
flag=int(raw_input("Enter 0 for read only and 1 for read and write permission - "))
PATH = select_path()
print 'selected path : ', PATH + '\n'
print uname
print passw
if (flag==1):
```

```
        authorizer.add_user(uname,passw,PATH, perm='elradfmwM')
else:
        authorizer.add_user(uname,passw,PATH)

authorizer.add_anonymous(PATH)

handler=FTPHandler
handler.authorizer=authorizer

address=("192.168.43.150",8080)
server=FTPServer(address,handler)

server.max_cons = 5
server.max_cons_per_ip = 2

server.serve_forever()
```

Fc.py

```
import ftplib
import os
import sys

port= 8080

def filesDir(path):
        files = os.listdir(path)
        for fl in files:
                i = int(files.index(fl))+1
                print str(i)+ ')' + '{:.<50}'.format(fl) + str(os.path.getsize(path+"/"+fl))+" B"

def select_path():
        PATH = raw_input('Select the Path press enter for default path : ')
        if PATH == '':
                PATH = os.getcwd()

        print PATH
        accept_path = raw_input('Select this path?(Y/N) : ').lower().strip(' ')

        if accept_path == 'y' or accept_path == 'Y':
                return PATH
        else:
                select_path()

ftp=ftplib.FTP()
#host=raw_input("Enter Host address")
```

```
host=raw_input("enter host: ")
ftp.connect(host,port)
uname=raw_input("Enter Username")
passw=raw_input("Enter_password")
ftp.login(uname,passw)
while True:
	choice=int(raw_input("Enter 1 to download 0 to upload files and 2 to exit - "))
	if(choice==2):
		sys.exit(0)

	if (choice==1):
		listing=[]
		ftp.retrlines('LIST',listing.append)
		for i in listing:
			words=i.split()
			name=words[8]
			size=words[4]
			print '{:.<50}'.format(name) + size + " B \n"

		print "\n"
		try:
			filenamed=raw_input("Enter name of file to download")
			file1=open("C:\Users\\dubey\\Desktop\\download\\"+filenamed,"wb")

			ftp.retrbinary("RETR "+filenamed,file1.write,8*1024)
			print "Download Complete"
			ftp.quit()
			file1.close()

		except:
			print "Some Error Occured!"

	elif (choice==0):
		class FtpUploadTracker:
			sizeWritten = 0
			totalSize = 0
			lastShownPercent = 0

			def __init__(self, totalSize):
				self.totalSize = totalSize
				self.sizeWritten = 0

			def handle(self, block):

				self.sizeWritten += 1024
```

```
                percentComplete = round((self.sizeWritten / self.totalSize) * 100)

                if (self.lastShownPercent != percentComplete):
                        self.lastShownPercent = percentComplete
                        print(str(percentComplete) + " percent complete ")

        try:
                PATH = select_path()
                print 'selected path : ', PATH + '\n'
                filesDir(PATH)

                filenameu=raw_input("Enter name of file to upload - ")
                totalSize = os.path.getsize(PATH+"/"+filenameu)

                uploadTracker = FtpUploadTracker(int(totalSize))
                file=open(PATH+"/"+filenameu,"rb")

                ftp.storbinary("STOR "+filenameu,file,1024,uploadTracker.handle)
                ftp.quit()
                file.close()

        except OSError:
                print "Wrong name!"

        except ftplib.error_perm:
                print "You don't have write priviledges"

        else:
                print "Some ERROR has occured !"

else:
        print "wrong input"
```

REFERENCES:

1. https://docs.python.org/2/library/socket.html
2. Stack Overflow Questions
3. http://timgolden.me.uk/python/win32_how_do_i.html
4. Foundation of Python Network Programming- Brandon Rhodes

5. Python Docs
6. www.learnxinyminutes.com/python
7. http://www.tkdocs.com/tutorial/firstexample.html
8. Python Wiki