File Transfer System

Computer Networks Project

GROUP 12

Team Members

- Bavesh Balaji - COE18B007
- Balajee D - COE18B013
- Pratyush V Moorthy - COE18B042
- Sreedhar Arumugam - COE18B051
- Viknesh Rajaramon - COE18B060
- B Vignesh - CED18I007

Overview

The problem statement that we were assigned was to build a file transfer system to send and receive files across machines in different networks. The sender is mandated to convert the files into a bitstream and the receiver is expected to detect the filetype of the file that it receives.

We have implemented a File Transfer system with Python, using PyQt5 for the UI generation and using the in-built threading library for multithreading.. The user can send as many files to as many users, and view the real-time progress of the file being sent in transit. The receiver can also view the progress in how much data is sent so far and how much is left to be received.

Implementation

We have 3 .py files in total - namely Initial.py, receiver.py and fileWindow.py. We run Initial.py in the sender terminal and Receiver.py in the receiver terminal. We get UI windows rendered with appropriate prompts for both sender and receiver processes immediately. The sender provides options to choose the file to be transmitted, and asks for the IP address of the receiver. Both sender and receiver then display progress bars that inform the user on the status of the file transfer.

Initial.py

- This file is used to create the initial UI for our file transfer system.
- Our User interface consists of a main window, in which there are various widgets, buttons and labels appropriately.
- We have defined a function setupUi, which initializes the UI construct of our file transfer system.
- This function defines one central widget in the main window and 2 buttons,one for the sender and another for the receiver.

- In order to render the appropriate text prompts for the main window, we make use of an auxiliary function called retranslateUi.
- And then, we have 2 functions send_pop() and clickRecv() which define the operations that the program needs to perform on clicking the respective buttons.
- On clicking the sender button, the send_pop() function is called which creates a sender window and initializes the user interface for that window.
- Similarly, on clicking the receiver button, the clickRecv() function is called which creates a receiver window and sets up the UI for that window.
- In the main, we have created the main window, which is essentially an object of class Ui_MainWindow. We have then initialized the UI construct for this window.

FileWindow.py

- We have defined a class File, with auxiliary functions that initialize the UI for the sender, open the File explorer as a dialog box for the user to choose the file to be sent, and save the file choice of the user.
- We have also defined a class UiSenderWindow that contains auxiliary functions to initialize the sender window, to initiate the sender function that sends the file, and to set up the UI that renders the progress bar and status to the user.
- In the function send_file(), we get the filename and the absolute path and open a socket.
- The user can send the selected file to upto 3 other receivers at the same time. To enable this, we implement a multithreading solution. We create 3 threads, which then sends the files to the 3 receivers in parallel.
- We then connect the socket to the appropriate PORT. After updating the user that the connection is established, we then send the filename along with the filetype first to the receiver, to intimate the type of file sent.
- This is because we send the file in a bit stream and therefore the receiver requires to be informed of the filetype prior to sending the file.
- We then open the file in binary mode (rb) and send the file data byte by byte in a bitstream, updating the progress bar simultaneously as we send data.
- We then close the socket after the file is sent successfully.
- We also have functions that set up the UI, registers file clicks, and gets the IP of the receiver within a dialog box in the UI.

Receiver.py:-

- This file is used for creating the receiver UI completely and for receiving the data sent by the client.
- Here,we again have a class for the UI which consists of setupUi() function, retranslateUi() and server() function.
- The setupUi() function is used for initializing the UI construct for the receiver window as explained above in initial.py .
- The retranslateUi() function is used to render the appropriate text prompts for the receiver window that we set up.
- The server() function is the actual function which receives the data sent by the client in bytes.
- In this function,a socket is first created for the server.
- The socket then listens for any available client, and gets connected with the client on detecting a client.
- Then multiple threads are spawned for multiple receivers, and in each thread :
- The filename and the file size is then received from the sender and a file with the same name is created in the current directory.
- The created file is then opened in write-binary or “wb”mode since we read and write the data in bitstreams as per the question.
- Then we receive the data sent by the sender byte by byte,and write it in the new file created in this directory while updating the progress bar simultaneously.
- We then close the socket created for the server after all the data sent by the sender is received in the server end.
- In this way, we use multithreading to send to multiple receivers at the same time.

- In the main, we create a receiver window which is basically an object of class Ui_RecvWindow and set up the UI for the receiver window.

Code

Initial.py

```
# -*- coding: utf-8 -*-


# Form implementation generated from reading ui file 'Initial.ui'

#

# Created by: PyQt5 UI code generator 5.15.1

#

# WARNING: Any manual changes made to this file will be lost when pyuic5
is

# run again.  Do not edit this file unless you know what you are doing.




from PyQt5 import QtCore, QtGui, QtWidgets

from PyQt5.QtWidgets import QMessageBox

from fileWindow import Ui_SenderWindow

from receiver import Ui_RecvWindow

import time

import threading
```

```
class Ui_MainWindow(object):

    def setupUi(self, MainWindow):

        MainWindow.setObjectName("MainWindow")

        MainWindow.resize(799, 428)

        self.centralwidget = QtWidgets.QWidget(MainWindow)

        self.centralwidget.setObjectName("centralwidget")

        self.label = QtWidgets.QLabel(self.centralwidget)

        self.label.setGeometry(QtCore.QRect(130, 0, 711, 131))

        font = QtGui.QFont()

        font.setFamily("Noto Serif CJK SC")

        font.setPointSize(28)

        font.setBold(True)

        font.setUnderline(True)

        font.setWeight(75)

        self.label.setFont(font)

        self.label.setObjectName("label")

        self.label_2 = QtWidgets.QLabel(self.centralwidget)

        self.label_2.setGeometry(QtCore.QRect(170, 90, 491, 81))

        font = QtGui.QFont()

        font.setFamily("Noto Serif CJK SC")

        font.setPointSize(28)

        font.setBold(True)

        font.setWeight(75)

        self.label_2.setFont(font)
```

```
        self.label_2.setObjectName("label_2")
        self.pushButton = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton.setGeometry(QtCore.QRect(60, 220, 281, 81))
        font = QtGui.QFont()
        font.setFamily("Noto Sans CJK SC")
        font.setPointSize(20)
        font.setBold(False)
        font.setWeight(50)
        self.pushButton.setFont(font)
        self.pushButton.setObjectName("pushButton")
        self.pushButton_2 = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_2.setGeometry(QtCore.QRect(450, 220, 281, 81))
        font = QtGui.QFont()
        font.setFamily("Noto Sans CJK SC")
        font.setPointSize(20)
        self.pushButton_2.setFont(font)
        self.pushButton_2.setObjectName("pushButton_2")
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 799, 20))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
```

```
        MainWindow.setStatusBar(self.statusbar)


        self.retranslateUi(MainWindow)

        QtCore.QMetaObject.connectSlotsByName(MainWindow)


        self.pushButton.clicked.connect(self.send_pop)

        self.pushButton_2.clicked.connect(self.clickRecv)


    def retranslateUi(self, MainWindow):

        _translate = QtCore.QCoreApplication.translate

        MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))

        self.label.setText(_translate("MainWindow", "Computer Networks
Project"))

        self.label_2.setText(_translate("MainWindow", "File Transfer
Application"))

        self.pushButton.setText(_translate("MainWindow", "Sender"))

        self.pushButton_2.setText(_translate("MainWindow", "Receiver"))


    def clickRecv(self):

        msg = QMessageBox.question(None, "File Transfer", "Are you sure
you want to receive a file?", QMessageBox.Yes | QMessageBox.No)

        if msg == QMessageBox.Yes:

            import sys

            self.RecvWindow = QtWidgets.QMainWindow()

            self.ui = Ui_RecvWindow()
```

```
            self.ui.setupUi(self.RecvWindow)
            self.RecvWindow.show()
            print("Receiver open")


    def send_pop(self):
        msg = QMessageBox.question(None, "File Transfer", "Are you sure
you want to send a file?", QMessageBox.Yes | QMessageBox.No)
        if msg == QMessageBox.Yes:
            import sys
            self.SenderWindow = QtWidgets.QMainWindow()
            self.ui = Ui_SenderWindow()
            self.ui.setupUi(self.SenderWindow)
            self.SenderWindow.show()



if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

FileWindow.py

```
import sys

from PyQt5 import QtCore, QtGui, QtWidgets

from PyQt5.QtWidgets import *

from PyQt5.QtGui import QIcon

import time

import socket

import tqdm

import os

import argparse

import threading




SEPARATOR = "<SEPARATOR>"

BUFFER_SIZE = 1024



class File(QWidget):

    def __init__(self):

        super().__init__()

        self.title = 'Select File'

        self.left = 10

        self.top = 10

        self.width = 640
```

```
        self.height = 480
        self.fileName = ''
        self.host = []
        self.initUI()
        self.close()

    def initUI(self):
        self.setWindowTitle(self.title)
        self.setGeometry(self.left, self.top, self.width, self.height)

        self.openFileNameDialog()

    def openFileNameDialog(self):
        options = QFileDialog.Options()
        options |= QFileDialog.DontUseNativeDialog
        self.fileName, _ =
QFileDialog.getOpenFileName(self,"QFileDialog.getOpenFileName()", "","All
Files (*);;Python Files (*.py)", options=options)
        if self.fileName:
            print(self.fileName)

    def openFileNamesDialog(self):
        options = QFileDialog.Options()
        options |= QFileDialog.DontUseNativeDialog
```

```
        files, _ =
QFileDialog.getOpenFileNames(self,"QFileDialog.getOpenFileNames()",
"","All Files (*);;Python Files (*.py)", options=options)

        if files:

            print(files)



    def saveFileDialog(self):

        options = QFileDialog.Options()

        options |= QFileDialog.DontUseNativeDialog

        fileName, _ =
QFileDialog.getSaveFileName(self,"QFileDialog.getSaveFileName()","","All
Files (*);;Text Files (*.txt)", options=options)

        if fileName:

            print(fileName)



    def client(self, filename):

        port=5001



        #self.send_file(filename, host, port)

        # creating thread

        t1 = threading.Thread(target=self.send_file,
args=(filename,self.host[0],port))

        t2 = threading.Thread(target=self.send_file,
args=(filename,self.host[1],port))

        t3 = threading.Thread(target=self.send_file,
args=(filename,self.host[2],port))
```

```
        # starting thread 1
        t1.start()
        # starting thread 2
        t2.start()
        # starting thread 3
        t3.start()

        # wait until thread 1 is completely executed
        t1.join()
        # wait until thread 2 is completely executed
        t2.join()
        # wait until thread 2 is completely executed
        t3.join()

        # both threads completely executed
        print("Done!")



class Ui_SenderWindow(File):
    def __init__(self):
        super().__init__()
        print("Inside sender window function\n\n\n\n")
        print(self.fileName)
```

```
    def send_file(self, filename, host, port):
        value = 0
        filesize = os.path.getsize(filename)
        s = socket.socket()
        print(f"[+] Connecting to {host}:{port}")
        s.connect((host, port))
        print("[+] Connected.")
        s.send(f"{filename}{SEPARATOR}{filesize}".encode())
        progress = tqdm.tqdm(range(filesize), f"Sending {filename}",
unit="B", unit_scale=True, unit_divisor=1024)
        with open(filename, "rb") as f:
            for _ in progress:
                bytes_read = f.read(BUFFER_SIZE)
                value = value + 1024
                self.progressBar.setProperty("value", value/filesize*100)



                if not bytes_read:
                    break
                s.sendall(bytes_read)
                progress.update(len(bytes_read))
        s.close()
```

```
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(832, 457)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.lineEdit = QtWidgets.QLineEdit(self.centralwidget)
        self.lineEdit.setGeometry(QtCore.QRect(140, 120, 431, 23))
        self.lineEdit.setObjectName("lineEdit")
        self.lineEdit.setText(self.fileName)
        self.fileSelect = QtWidgets.QCommandLinkButton(self.centralwidget)
        self.fileSelect.setGeometry(QtCore.QRect(590, 110, 174, 41))
        self.fileSelect.setObjectName("fileSelect")
        self.fileSelect.clicked.connect(self.fileClick)


        self.lineEdit_2 = QtWidgets.QLineEdit(self.centralwidget)
        self.lineEdit_2.setGeometry(QtCore.QRect(280, 170, 431, 23))
        self.lineEdit_2.setObjectName("lineEdit_2")
        self.progressBar = QtWidgets.QProgressBar(self.centralwidget)
        self.progressBar.setGeometry(QtCore.QRect(80, 370, 671, 41))
        self.progressBar.setProperty("value", 0)
        self.progressBar.setObjectName("progressBar")
        self.label = QtWidgets.QLabel(self.centralwidget)
        self.label.setGeometry(QtCore.QRect(170, 0, 611, 91))
        font = QtGui.QFont()
```

```
        font.setFamily("Noto Serif CJK SC")
        font.setPointSize(36)
        font.setBold(False)
        font.setUnderline(True)
        font.setWeight(50)
        self.label.setFont(font)
        self.label.setObjectName("label")
        self.lineEdit_3 = QtWidgets.QLineEdit(self.centralwidget)
        self.lineEdit_3.setGeometry(QtCore.QRect(280, 210, 431, 23))
        self.lineEdit_3.setObjectName("lineEdit_3")
        self.lineEdit_4 = QtWidgets.QLineEdit(self.centralwidget)
        self.lineEdit_4.setGeometry(QtCore.QRect(280, 250, 431, 23))
        self.lineEdit_4.setText("")
        self.lineEdit_4.setObjectName("lineEdit_4")
        self.pushButton = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton.setGeometry(QtCore.QRect(260, 300, 281, 41))
        font = QtGui.QFont()
        font.setFamily("Liberation Serif")
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.pushButton.setFont(font)
        self.pushButton.setObjectName("pushButton")
        self.label_2 = QtWidgets.QLabel(self.centralwidget)
```

```
        self.label_2.setGeometry(QtCore.QRect(130, 170, 161, 16))
        font = QtGui.QFont()
        font.setFamily("Noto Sans CJK SC")
        self.label_2.setFont(font)
        self.label_2.setObjectName("label_2")
        self.label_3 = QtWidgets.QLabel(self.centralwidget)
        self.label_3.setGeometry(QtCore.QRect(130, 210, 161, 16))
        font = QtGui.QFont()
        font.setFamily("Noto Sans CJK SC")
        self.label_3.setFont(font)
        self.label_3.setObjectName("label_3")
        self.label_4 = QtWidgets.QLabel(self.centralwidget)
        self.label_4.setGeometry(QtCore.QRect(130, 250, 161, 16))
        font = QtGui.QFont()
        font.setFamily("Noto Sans CJK SC")
        self.label_4.setFont(font)
        self.label_4.setObjectName("label_4")
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 832, 20))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
```

```
        MainWindow.setStatusBar(self.statusbar)
        self.pushButton.clicked.connect(self.transferClick)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
        self.fileSelect.setText(_translate("MainWindow", "Select File"))
        self.label.setText(_translate("MainWindow", "File Transfer
Client"))
        self.pushButton.setText(_translate("MainWindow", "Start
Transfer"))
        self.label_2.setText(_translate("MainWindow", "Enter Receiver 1 IP
:"))
        self.label_3.setText(_translate("MainWindow", "Enter Receiver 2 IP
:"))
        self.label_4.setText(_translate("MainWindow", "Enter Receiver 3 IP
:"))

    def fileClick(self):
        self._new_window = File()

    def transferClick(self):
```

```
        self.host.append(self.lineEdit_2.text())
        self.host.append(self.lineEdit_3.text())
        self.host.append(self.lineEdit_4.text())
        print(self.host)
        super().client(self.fileName)


    def ipClick(self, no):
        print("inside ip host")



if __name__ == '__main__':
    app = QApplication(sys.argv)
    ex = Ui_SenderWindow()
    sys.exit(app.exec_())
```

Receiver.py

```
# -*- coding: utf-8 -*-


# Form implementation generated from reading ui file 'recv.ui'

#

# Created by: PyQt5 UI code generator 5.9.2

#

# WARNING! All changes made in this file will be lost!


from PyQt5 import QtCore, QtGui, QtWidgets

from PyQt5.QtWidgets import QPushButton

import socket

import tqdm

import os

import argparse

from PyQt5.QtCore import QThread, pyqtSignal

import time




SEPARATOR = "<SEPARATOR>"

BUFFER_SIZE = 1024
```

```
class Ui_RecvWindow(object):

    def setupUi(self, MainWindow):

        MainWindow.setObjectName("MainWindow")

        MainWindow.resize(722, 363)

        self.centralwidget = QtWidgets.QWidget(MainWindow)

        self.centralwidget.setObjectName("centralwidget")

        self.label = QtWidgets.QLabel(self.centralwidget)

        self.label.setGeometry(QtCore.QRect(120, 10, 611, 131))

        font = QtGui.QFont()

        font.setFamily("Noto Serif CJK SC")

        font.setPointSize(36)

        font.setUnderline(True)

        self.label.setFont(font)

        self.label.setObjectName("label")

        self.label_2 = QtWidgets.QLabel(self.centralwidget)

        self.label_2.setGeometry(QtCore.QRect(40, 150, 181, 41))

        self.label_2.setObjectName("label_2")

        self.lineEdit = QtWidgets.QLineEdit(self.centralwidget)

        self.lineEdit.setGeometry(QtCore.QRect(170, 160, 501, 23))

        self.lineEdit.setObjectName("lineEdit")

        self.progressBar = QtWidgets.QProgressBar(self.centralwidget)

        self.progressBar.setGeometry(QtCore.QRect(40, 220, 631, 41))

        self.progressBar.setProperty("value", 0)

        self.progressBar.setObjectName("progressBar")
```

```
        self.pushButton = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton.setGeometry(QtCore.QRect(250, 280, 281, 41))
        font = QtGui.QFont()
        font.setFamily("Liberation Serif")
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.pushButton.setFont(font)
        self.pushButton.setFont(font)
        self.pushButton.setObjectName("pushButton")
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 722, 20))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)
        self.pushButton.clicked.connect(self.server)


        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)


    def retranslateUi(self, MainWindow):
```

```
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
        self.pushButton.setText(_translate("MainWindow", "Start
Receiving"))
        self.label.setText(_translate("MainWindow", "File Transfer
Server"))
        self.label_2.setText(_translate("MainWindow", "Connection status
:"))
        self.lineEdit.setText(_translate("MainWindow", "No connection
yet"))


    def server(self):
        SERVER_HOST = "0.0.0.0"
        SERVER_PORT=5001
        s=socket.socket()
        s.bind((SERVER_HOST,SERVER_PORT))
        s.listen(1)
        value = 0
        print(f"[*] Listening as {SERVER_HOST}:{SERVER_PORT}")
        client_socket, address = s.accept()
        print(f"[+] {address} is connected.")
        self.lineEdit.setText(f"Connected to {address[0]}.")


        received = client_socket.recv(BUFFER_SIZE).decode()
        filename, filesize = received.split(SEPARATOR)
```

```
        filename = os.path.basename(filename)

        filesize = int(filesize)



        progress = tqdm.tqdm(range(filesize), f"Receiving {filename}",
unit="B", unit_scale=True, unit_divisor=1024)

        with open(filename, "wb") as f:

            for _ in progress:

                bytes_read = client_socket.recv(BUFFER_SIZE)

                value = value + 1024

                self.progressBar.setProperty("value", value/filesize*100)

                if not bytes_read:

                    break

                f.write(bytes_read)

                progress.update(len(bytes_read))

        client_socket.close()

        s.close()




if __name__ == "__main__":

    import sys

    app = QtWidgets.QApplication(sys.argv)

    MainWindow = QtWidgets.QMainWindow()

    ui = Ui_RecvWindow()

    ui.setupUi(MainWindow)
```

```
    MainWindow.show()

    sys.exit(app.exec_())
```

Output

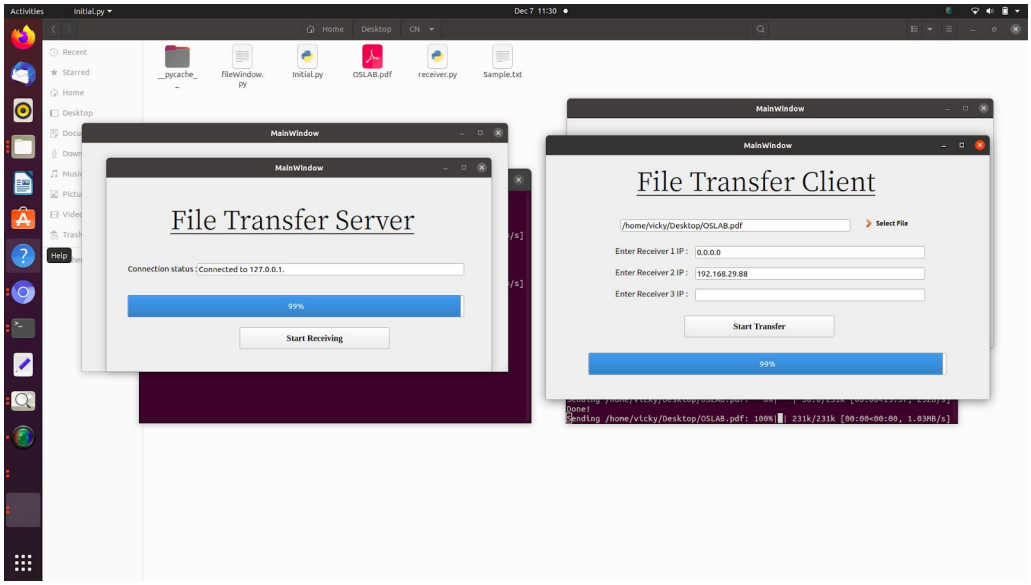
We have attached two videos that demonstrate the working of our program here.

In the first video, we show file transfer within the same machine, and in the second video we show it across different machines - we send it from one computer and receive it in another.

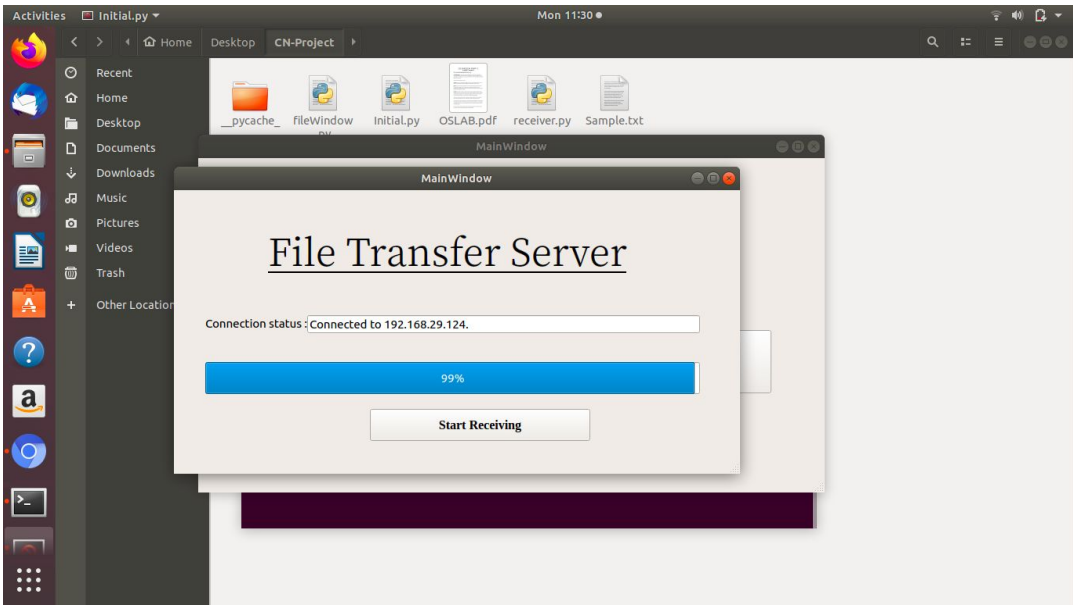
The first video shows a sender and a local host as a receiver(0.0.0.0), wherein the sender chooses to send a text file to this receiver, as well as a receiver on another machine connected through LAN. The second video shows the receiver across LAN receiving the file when the sender sends it (machine to machine transmission). This is similarly demonstrated for a pdf file, which is sent to a receiver in the same machine as that of the sender, as well as to a receiver in a different machine.

Output Screenshots

Machine 1 (sender as well as receiver) :



Machine 2(Another receiver):



Conclusion

Thus, we have implemented an inter network file transfer system that sends and receives files with status and progress updates for the sender and receiver on the files in transit.