Understanding FTP

A. Objectives

1. Practice with ftp servers and learn how to measure network throughput
2. Learn about basic Python Network Programing
3. Basic Socket Programing
4. How to use Python programing and Python SOCKETS library to perform FTP
5. How to use Python programing to send an email

B. Time of Completion

This laboratory activity is designed for students with very little knowledge of network application development and socket programing. PART I is estimated to take about 3 hours to complete. PART II should be completed within 2 hours.

C. Requirements

Create a gmail account!
An FTP server is required to be available for PART II. See Appendix for more details.

D. Procedure

Pay attention to how the PC numbers are setup. Your actual IP addresses may be different. As you complete each part respond to each question. Submit only your responses. Make sure you include the question and its number in each case. All submissions must be **typed**.

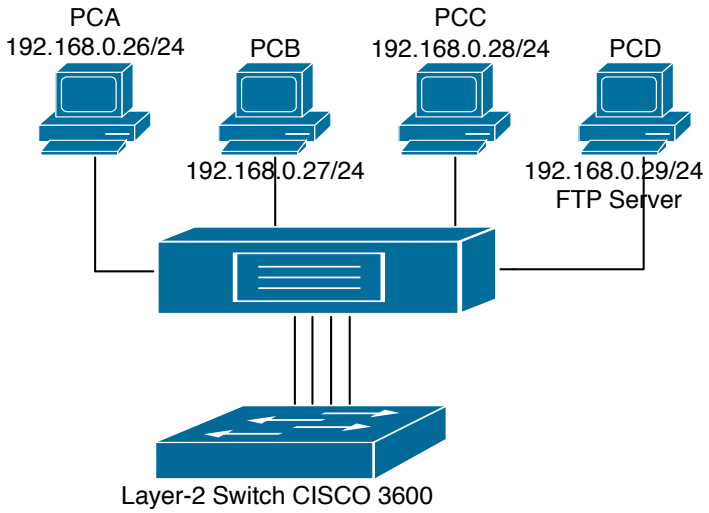

Figure 1 – Example of Group Configuration on Station A

PART I

In this section you are expected to setup your own ftp sever. Follow the steps below. For more information refer to https://help.ubuntu.com/10.04/serverguide/ftp-server.html.

1. Set the IP addresses: *sudo ifconfig eth0 192.168.0.xx netmask 255.255.255.0 broadcast 192.168.0.255*
2. Install Very Secure FTP (vsftpd) on the host using *sudo apt-get install vsftpd* – see figure 1
3. Start the ftp service to open the FTP port in your computer. Run *sudo start vsftpd*
4. Your FTP folder will be in */srv/ftp*.
5. In order to configure vsftpd, authenticate system users, and allow them to upload files edit /etc/vsftpd.conf using VI editor. Just make sure you remove the # from the line. Be **very careful** as you modify this file:
   local_enable=YES
   write_enable=YES
   anon_upload_enable=YES
6. Restart the ftp service to enable the FTP port on your computer: *sudo restart vsftpd*
7. Make sure the very secure ftp server is up and running: *ps -aef | grep vsftpd*
8. Using your browser download `Wireshark-win32.exe`. Check your Downloads folder.
9. Move the downloaded file into your FTP folder. Make sure the file is in the folder. The file must have "read" permission.
10. Change the file permission of the FTP folder to 644.

On a different machine (the client) setup the IP address. Connect the server and client to each other via the switch, as you did in part I.

11. From your machine issue: `ftp 192.168.0.xx` (this is the IP address of the FTP server you setup above). The user name and password maybe both `anonymous`. Read the link above for more information. Note: If you cannot get into the FTP server please check your IP address first, and make sure you can ping each other.
12. After you logged on, do `ls` and make sure the file `Wireshark-win32.exe` is there (your actual file name may be different). How large (in byte) is the file?
13. Exit from the ftp site by typing `quit` or using `Control C`.
14. Open the *`Wireshark`* program on your computer DO NOT start capturing.
15. At this point re-issue the ftp command: `ftp 192.168.0.xxx`.
16. Start the *`Wireshark`* program on your computer and start capturing packets.

17. Download a large file: `ftp> get Wireshark-win32.exe` from the ftp prompt! If it times out you must redo the ftp login process. Download the file TWO more times (total of three times). Once the download process is completed stop *wireshark*. Save the captured packets in a file; call it `Captured_FTP_FILE`.
18. Answer the following questions, using the data on your *ftp* terminal:
    a. On Average, how long did it take to download the files?
    b. On average, how many bytes you downloaded.
    c. What does command *get* do?
    d. How can you *upload* a file in the ftp server?

YOUR RESPONSE:

Open your `Captured_FTP_FILE` using *wireshark*. Answer the following questions.

19. Go to Statistics→IO Graphs. Note that the graph can be saved using the **SAVE** button in *jpeg* or any other format. Answer the following questions:
    a. Take a snapshot of the IO Graphs as shown by *wireshark.*
    b. How many **peaks** do you observe in the graph? Why?
    c. In terms of Bytes/Second what is the peak transmission rate for each peak?
    d. Go to Statistics→End Points. Click on IPV4. How many bytes your terminal received during this transition?
    e. Go to Statistics→End Points. Click on IPV4. How many packets your terminal received during this transition?
    f. What is the MAC address of the FTP server?

YOUR RESPONSE:

20. Upload a file called <YOUR_NAME> into the server and verify the file is there. Answer the following questions:
    a. What is the permission on the file?
    b. Can you change the permission to 666? Explain.
    c. Can you create a *directory* in the ftp server?

Show your results to the Instructor before you proceed!

In this section you need to write a program to display files in a public library in ftp.all.kernel.org.

1. Using your terminal, make sure you can ftp to ftp.all.kernel.org. use 'anonymous' for username and password. Follow the steps below. Make sure you see the files in the /pub directory.

```
50012-SALZ2010A:Chapter5 farid11$ ftp ftp.kernel.org
Trying 199.204.44.194...
Connected to ftp.all.kernel.org.
220 Welcome to kernel.org
Name (ftp.kernel.org:farid11): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd /pub
250 Directory successfully changed.
ftp> ls
```

```
FTP_SERVER_URL = 'ftp.kernel.org'

import ftplib

def test_ftp_connection(path, username, email):
    #Open ftp connection
    ftp = ftplib.FTP(path, username, email)

   #List the files in the /pub directory
    ftp.cwd("/pub")
    print "File list at %s:" %path
    files = ftp.dir()
    print files

    ftp.quit()
```

2. Using the following code you can automatically display the file in /pub library. This program automates the FPT action!
3. Modify the Python code to display all the files in the ROOT directory of ftp.all.kernel.org, e.g., "/"
4. Modify the Python code to display all the files in the root directory of YOUR ftp site in the lab.
5. Using Wireshark take a snapshot of what happen as you run the program.

Show your results to the Instructor before you proceed!

PART II – Simple Socket Programming

The purpose of this section is to learn how to write Python programs that communicate with the network using TCP/IP.

Quick Links:
http://www.tutorialspoint.com/python/python_networking.htm

We start with something simple program called *Tutorial.py*. Note that the purpose of this program is to display your host and IP address. In addition, the program can display the IP address of a remote machine, in this case called www.sonoma.edu.
In this program two basic functions are run:

- print_machine_info()
- get_remote_machine_info()

The functions are defined based on the Python SOCKET library[1]. SOCKETs are the endpoints of a bidirectional communications channel. Sockets may communicate within a process, between processes on the same machine, or between processes on different continents. The above functions are using two SOCKET libraries:

- socket.gethostname()
- socket.gethostbyname()

```
#!/usr/bin/env python
# This program is optimized for Python 2.7.

import socket

# Function to define the host and IP infomation
def print_machine_info():
    host_name = socket.gethostname()
    ip_address = socket.gethostbyname(host_name)
    print "Host name: %s" %host_name
    print "IP address: %s" %ip_address

#Function to define IP address of the remote_host
def get_remote_machine_info():
    remote_host = 'www.sonoma.edu'
    try:
        print "IP address of %s: %s" %(remote_host, socket.gethostbyname(remote_host))
    except socket.error, err_msg:
        print "%s: %s" %(remote_host, err_msg)

# This is where the finctions are called
# Note that _name_ is the name of the calling process
if __name__ == '__main__':
    print_machine_info()
    get_remote_machine_info()
```

[1] http://www.tutorialspoint.com/python/python_networking.htm

Here is how you run the above program called *Tutorial.py*:

```
50012-SALZ2010A:Chapter1 farid11$ python Tutorial.py
Host name: 50012-SALZ2010A
IP address: 192.168.1.73
IP address of www.sonoma.edu: 130.157.3.70
```

Question 1: Do the following:
Add the following function to the code above and explain what it does.

```
def find_service_name():
    protocolname = 'tcp'
    for port in [80, 25]:
        print "Port: %s => service name: %s" %(port, socket.getservbyport(port, protocolname))
```

Question 2: Do the following:
Carefully read the PORT assignment[2]. Change the program above to display the application that runs on fort 25. Print your results. What is port 53 used for?

For Practice Only – Do NOT Turn In This Section!
In this section you learn how to write a program to send an email to a valid user using your gmail SMTP server. The code (as shown below in appendix) logs into your gmail account and sends file/message to a specified user in the command line:

```
farid11$ python 5_6_send_email_from_gmail.py --sender=farid@gmail.com --recipient=farid@sonoma.edu
Enter your Google password:
Email
```

WARNING: Do not use your own personal gmail account! In some cases your gmail may be blocked when you use this program!

In this recipe, an e-mail message is created in the *send_email()* function. This function is supplied with a Google account from where the e-mail message will be sent. The message header object, *msg*, is created by calling the *MIMEMultipart()* class and then subject, recipient, and sender information is added on it.

The image attachment object, *img*, is then created with the *MIMEImage()* method from the *email.mime.image* module. A correct image header is added to this object and finally, the image is attached with the *msg* object created earlier. We can attach multiple image files within a for loop as shown in this recipe. We can also attach a plain text attachment in a similar way.

To send the e-mail message, we create an SMTP session. We call some testing method on this session object, such as *ehlo()* or *starttls()*. Then, log in to the Google SMTP server with a username and password and a *sendmail()* method is called to send the e-mail.

A- Make sure you can run the program.
B- Change the program to send your picture to the user. Show your code.

[2] http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers

```
#!/usr/bin/env python - email code
# This program requires Python 2.7 or any later version

import argparse
import os
import getpass
import re
import sys
import smtplib

from email.mime.image import MIMEImage
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText

SMTP_SERVER = 'smtp.gmail.com'
SMTP_PORT = 587

def send_email(sender, recipient):
    """ Send email message """
    msg = MIMEMultipart()
    msg['Subject'] = 'Python Email Test'
    msg['To'] = recipient
    msg['From'] = sender
    subject = 'Python email Test'
    message = 'Images attached.'
    # attach imgae files
    files = os.listdir(os.getcwd())
    gifsearch = re.compile(".gif", re.IGNORECASE)
    files = filter(gifsearch.search, files)
    for filename in files:
        path = os.path.join(os.getcwd(), filename)
        if not os.path.isfile(path):
            continue
        img = MIMEImage(open(path, 'rb').read(), _subtype="gif")
        img.add_header('Content-Disposition', 'attachment', filename=filename)
        msg.attach(img)

    part = MIMEText('text', "plain")
    part.set_payload(message)
    msg.attach(part)

    # create smtp session
    session = smtplib.SMTP(SMTP_SERVER, SMTP_PORT)
    session.ehlo()
    session.starttls()
    session.ehlo
    password = getpass.getpass(prompt="Enter your Google password: ")
    session.login(sender, password)
    session.sendmail(sender, recipient, msg.as_string())
    print "Email sent."
    session.quit()

if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Email Sending Example')
    parser.add_argument('--sender', action="store", dest="sender")
    parser.add_argument('--recipient', action="store", dest="recipient")
    given_args = parser.parse_args()
    send_email(given_args.sender, given_args.recipient)
```

REFERENCES

[1] Read about vsft server and how you can set is up in Ubuntu:
https://help.ubuntu.com/10.04/serverguide/ftp-server.html

[2] More information on setting up your network card: https://help.ubuntu.com/13.04/serverguide/network-configuration.html - Read about example commands for ethtool:
http://www.thegeekstuff.com/2010/10/ethtool-command/

[3] Later

Question 2: Printing the current time form the Internet time server

Network Time Protocol (NTP) is a networking protocol for clock synchronization between computer systems (more here)

Make sure you execute: su easy_install install ntplib - this will install the ntplib.

Links to the text:
http://www.learnr.pro/content/51372-python-network-programming-cookbook/134#1808914095:79421.8540424481

http://books.google.com/books?id=c9YrAwAAQBAJ&pg=PT71&lpg=PT71&dq=printing+current+time+from+time+server+python&source=bl&ots=mfZ_jjNr_l&sig=SyeaDPRKggIPLrbDRdqBFzY-zmM&hl=en&sa=X&ei=K28_VMWiC-eg8QGS8YGADA&ved=0CD8Q6AEwBQ#v=onepage&q=printing%20current%20time%20from%20time%20server%20python&f=false