Community Chat System

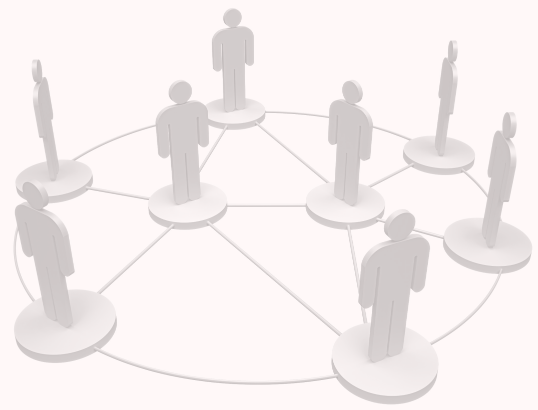
Worapol Leerunyakul (st116391)
Sariya Tangthamniyom (st116401)
Delani Rushanka Perera (st117810)
Teera Kritpranam (st117895)

Outline

Introduction

Objectives

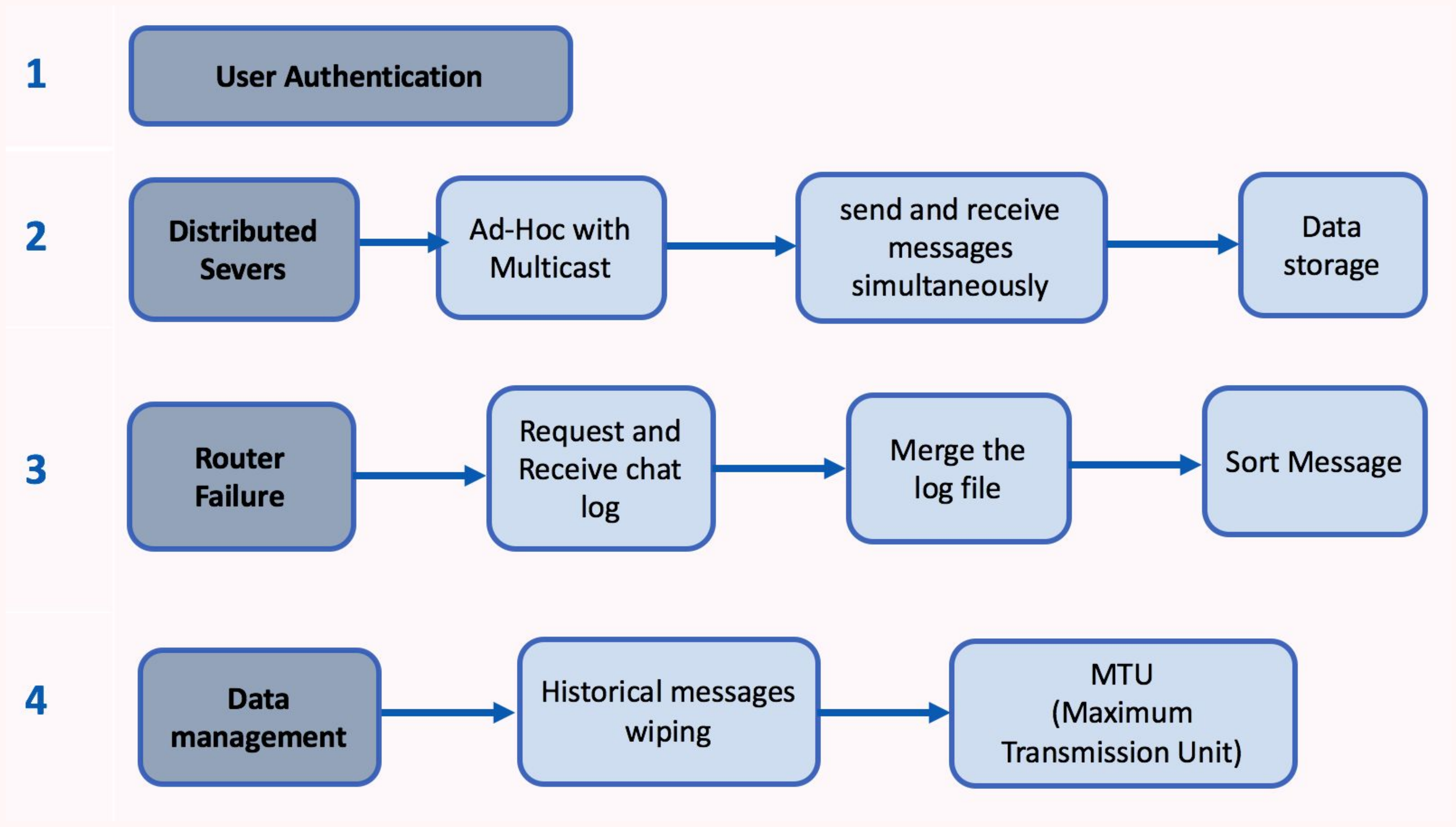
- ❖ Create Ad-hoc network and implement a method to share text message
- ❖ Implement multiple chat rooms
- ❖ Merge and synchronize message in every router
- ❖ Handle router failures and manage message history



Challenges & Techniques

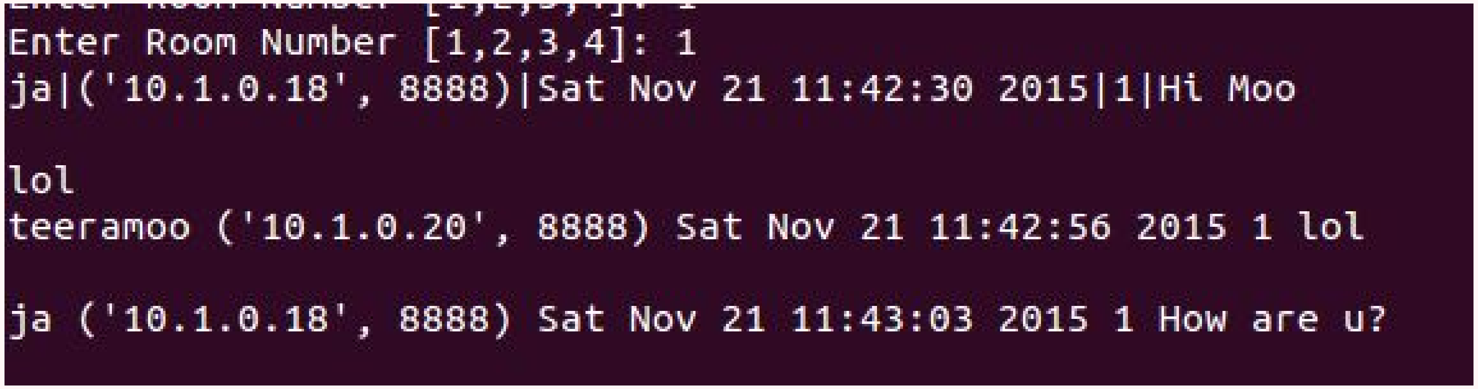
Challenges	Techniques
Broadcasting limitation (only one hop)	Multicast mechanism
Data storage limitation (No DBMS)	multiple text files
Data synchronization	Storing text files in routers, write once they received
Failure handling	Pulling data once routers started
Data retrieval & decrease loss rates	Applying MTU fragmentation, merge and sorting algorithm by datetime

System Design and Implementation- 4 Stages



User Authentication : ***Why do we need this?***

To identify users in chat rooms



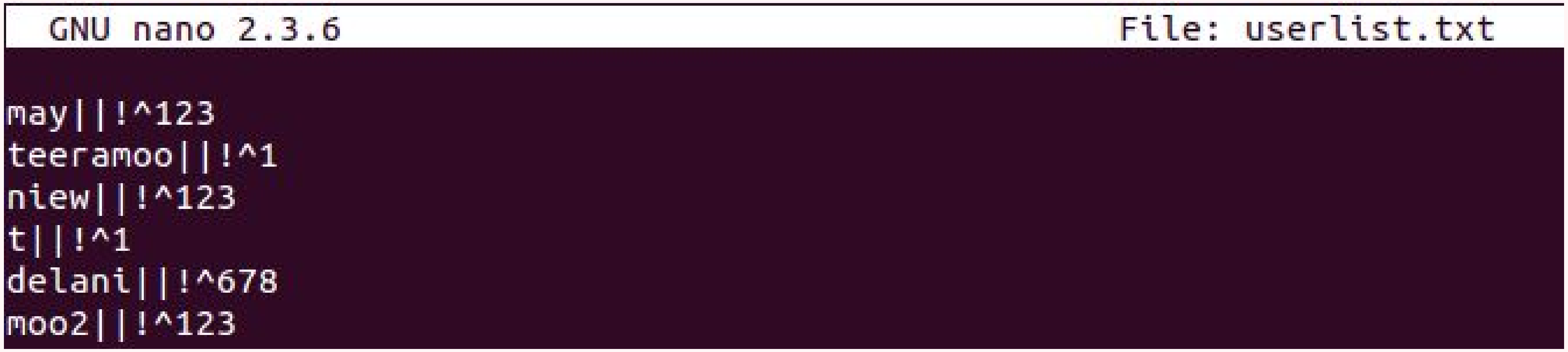
```
Enter Room Number [1,2,3,4]: 1
ja|('10.1.0.18', 8888)|Sat Nov 21 11:42:30 2015|1|Hi Moo

lol
teeramoo ('10.1.0.20', 8888) Sat Nov 21 11:42:56 2015 1 lol

ja ('10.1.0.18', 8888) Sat Nov 21 11:43:03 2015 1 How are u?
```

User Authentication : ***How user data is recorded***

———

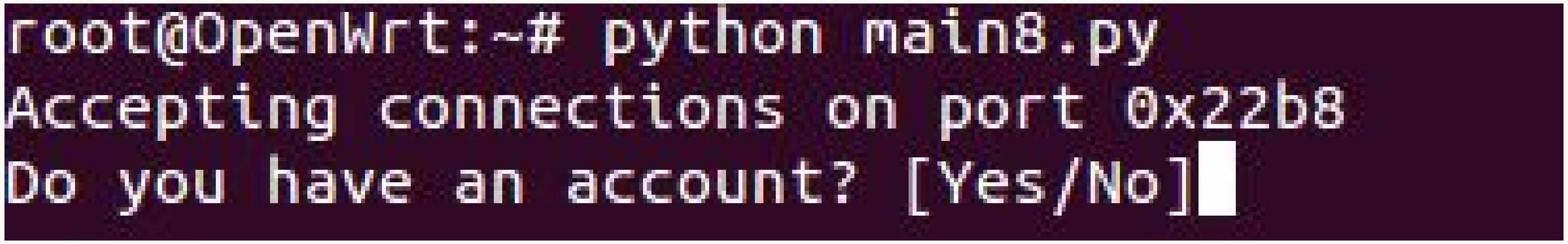
username + delimiter + password

In this case, the delimiters is “||!^”



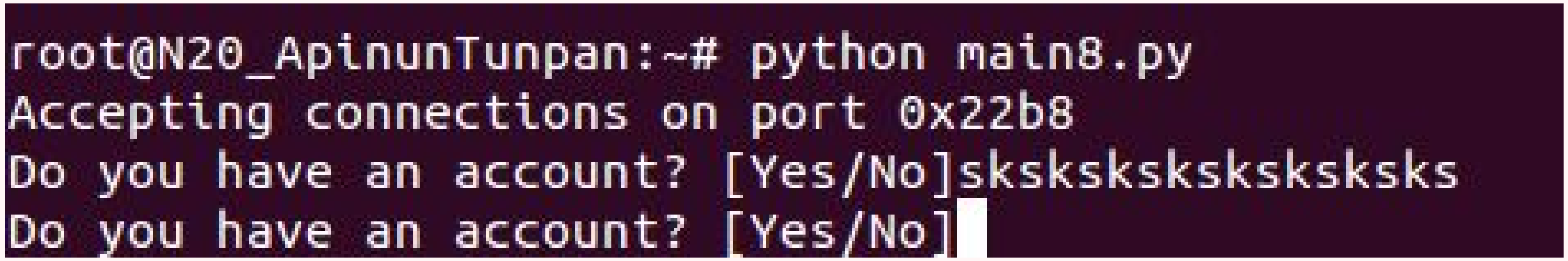
```
  GNU nano 2.3.6                                    File: userlist.txt

may||!^123
teeramoo||!^1
niew||!^123
t||!^1
delani||!^678
moo2||!^123
```

User Authentication : ***How does it work?***



```
root@OpenWrt:~# python main8.py
Accepting connections on port 0x22b8
Do you have an account? [Yes/No]
```

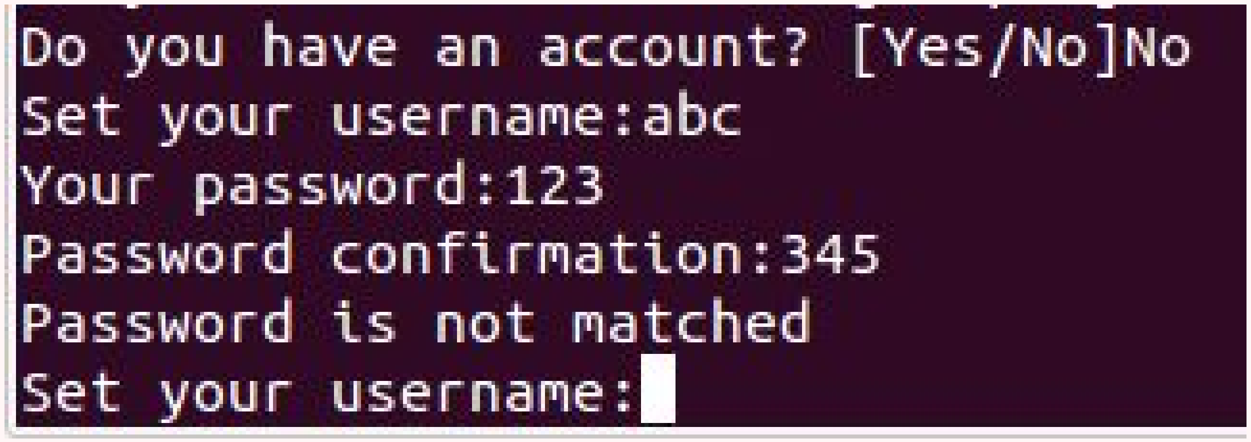
Other answers rather than
Yes and No



```
root@N20_ApinunTunpan:~# python main8.py
Accepting connections on port 0x22b8
Do you have an account? [Yes/No]sksksksksksksks
Do you have an account? [Yes/No]
```

User Authentication : ***How does it work?***

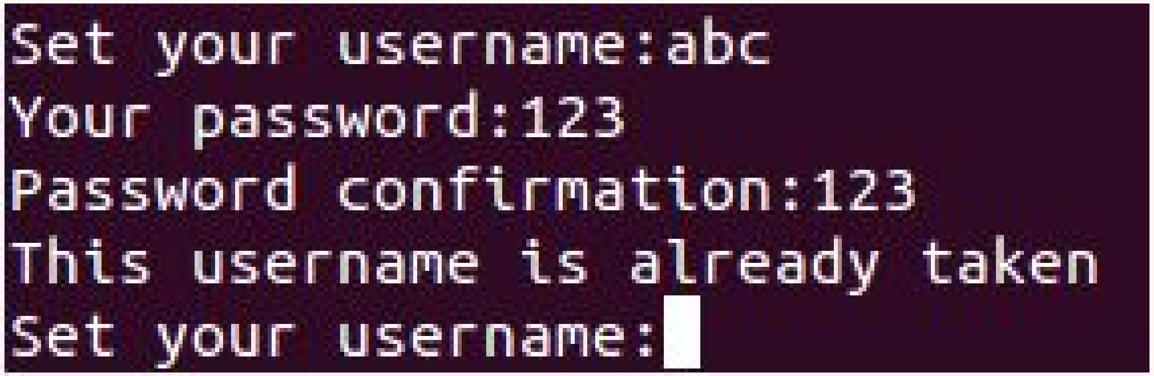
Answer is No. Password confirmation is not satisfied



```
Do you have an account? [Yes/No]No
Set your username:abc
Your password:123
Password confirmation:345
Password is not matched
Set your username:
```

User Authentication : ***How does it work?***

Answer is No. Username is already taken

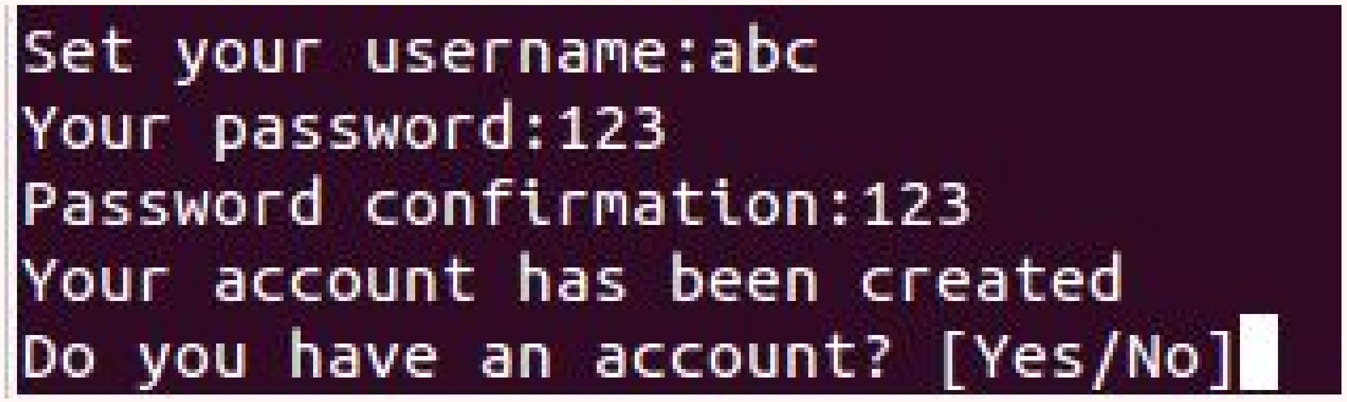


```
Set your username:abc
Your password:123
Password confirmation:123
This username is already taken
Set your username:
```

User Authentication : ***How does it work?***

———

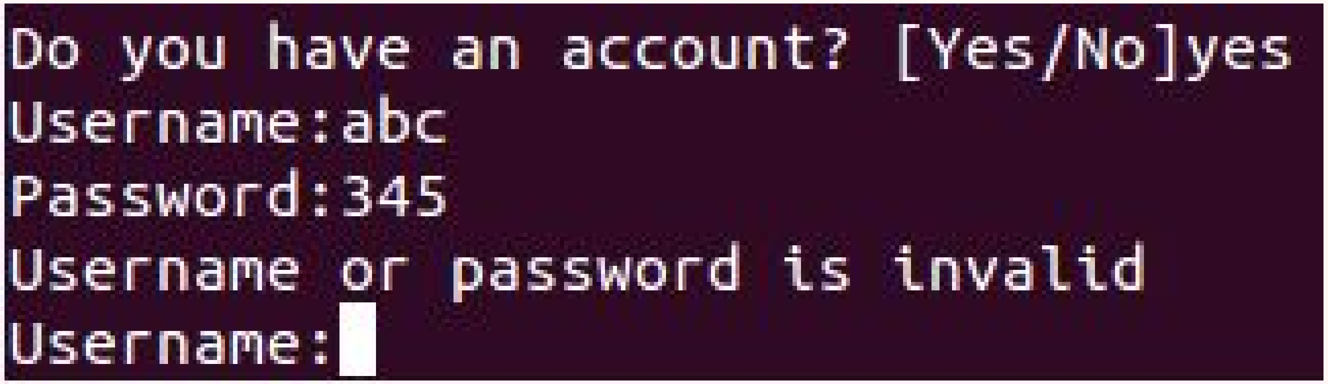
Answer is No. Username is unique and password confirmation is satisfied.



```
Set your username:abc
Your password:123
Password confirmation:123
Your account has been created
Do you have an account? [Yes/No]
```

User Authentication : ***How does it work?***

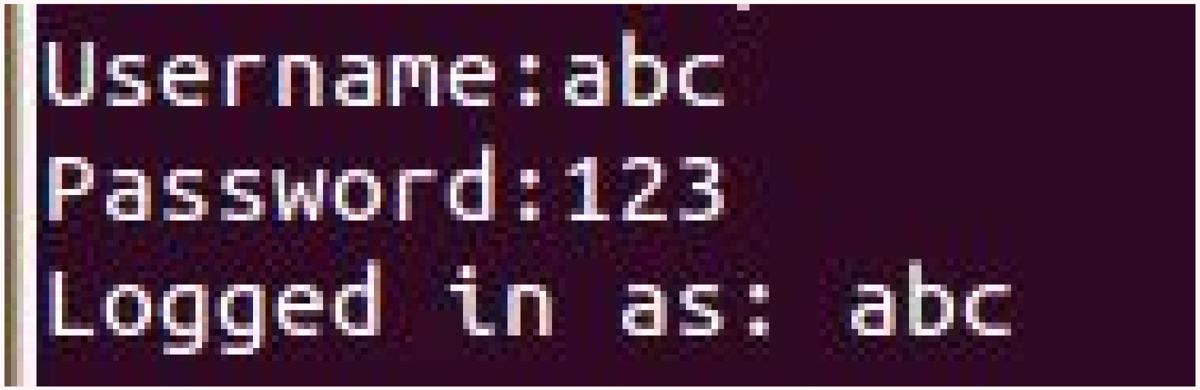
Answer is Yes but username and password is not matched



```
Do you have an account? [Yes/No]yes
Username:abc
Password:345
Username or password is invalid
Username:
```

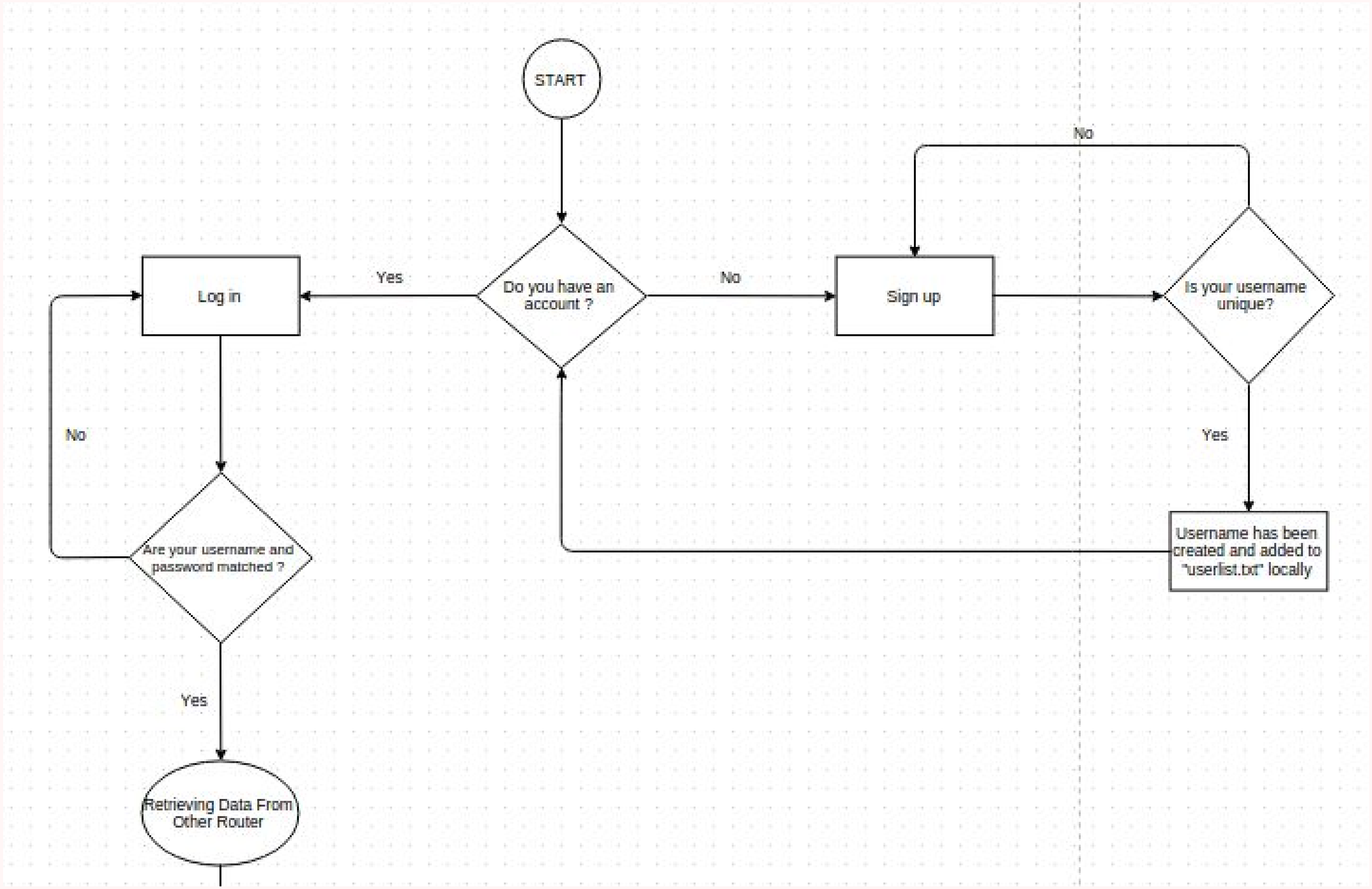
User Authentication : ***How does it work?***

Answer is Yes and username and password is matched

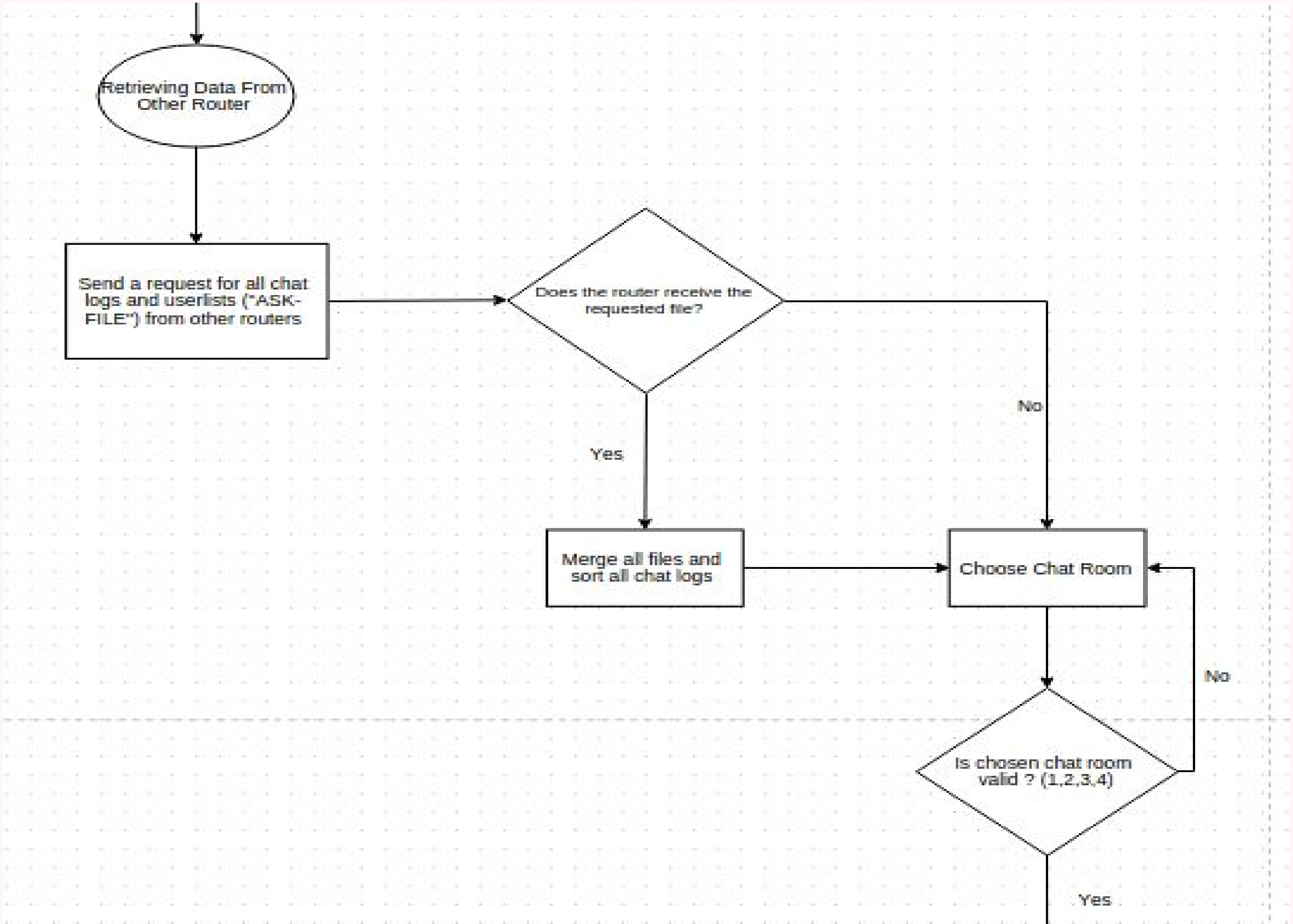


```
Username:abc
Password:123
Logged in as: abc
```

User Authentication : ***How does it work?***


START
No
Yes
Do you have an account ?
No
Log in
Sign up
Is your username unique?
No
Yes
Are your username and password matched ?
Username has been created and added to "userlist.txt" locally
Yes
Retrieving Data From Other Router

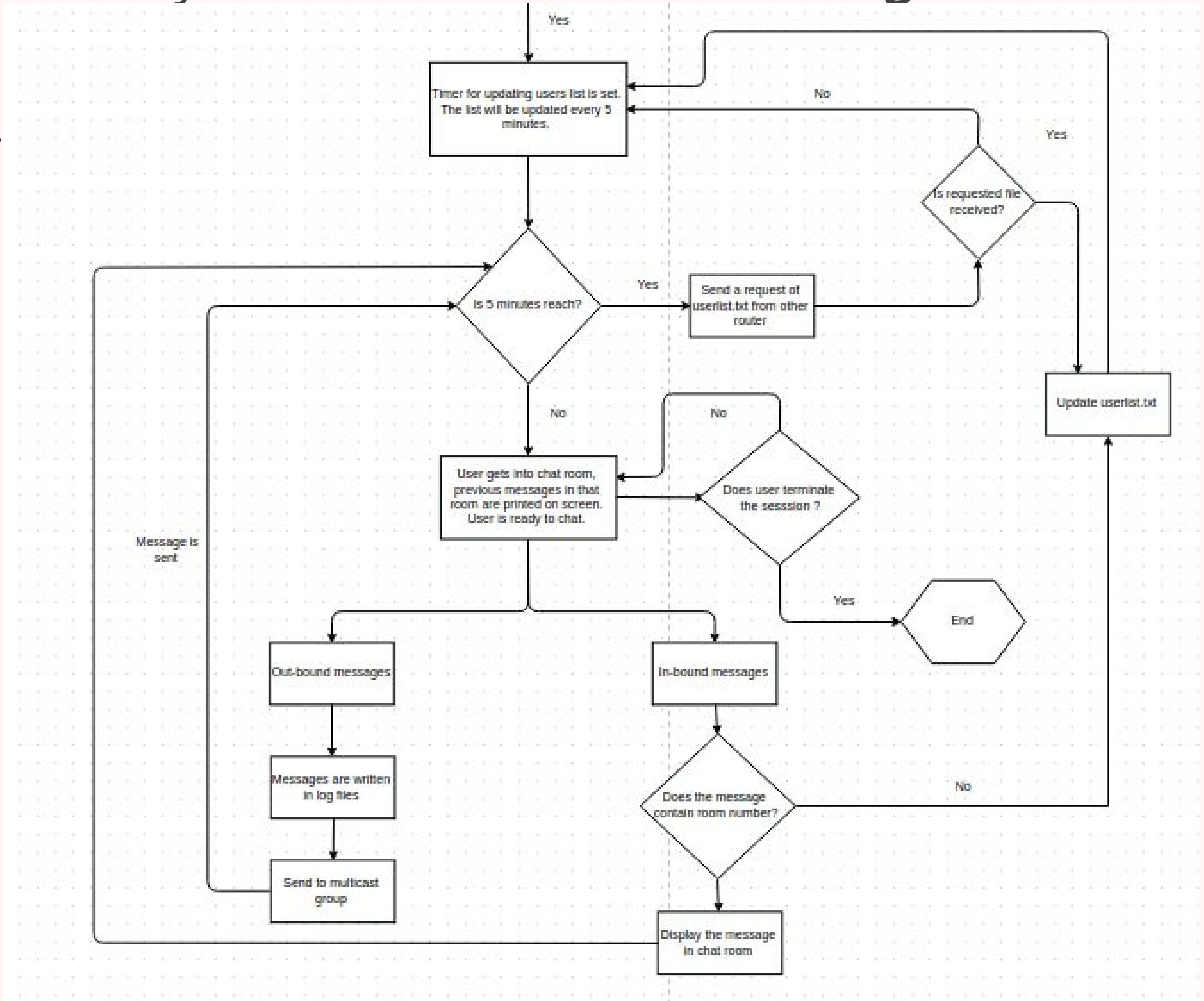
User list synchronization : ***Before selecting chat room***



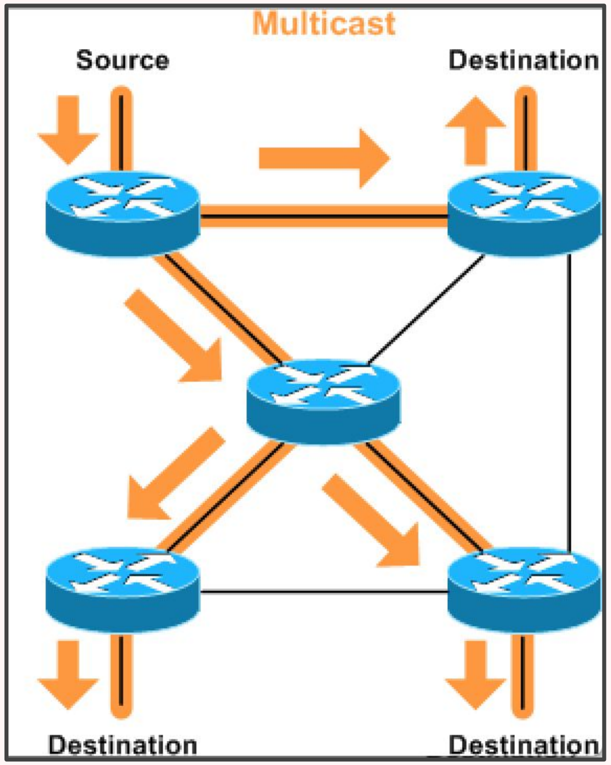
Retrieving Data From Other Router
Send a request for all chat logs and userlists ("ASK-FILE") from other routers
Does the router receive the requested file?
Yes
No
Merge all files and sort all chat logs
Choose Chat Room
No
Is chosen chat room valid ? (1,2,3,4)
Yes

User list synchronization : ***After selecting chat room***

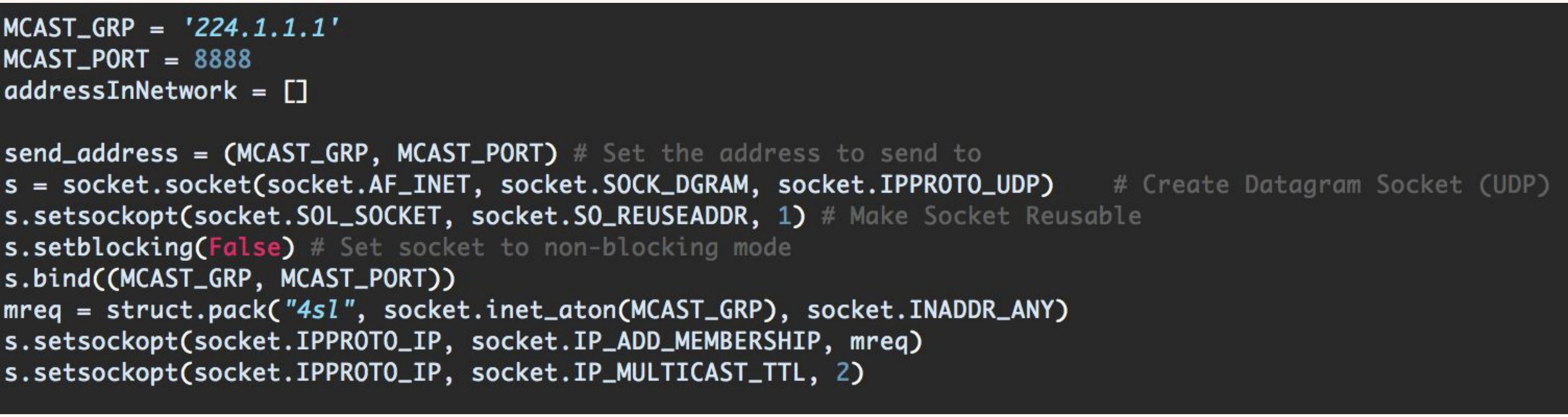
–


Yes
Timer for updating users list is set. The list will be updated every 5 minutes.
No
Yes
Is requested file received?
Is 5 minutes reach?
Yes
Send a request of userlist.txt from other router
Update userlist.txt
No
No
User gets into chat room, previous messages in that room are printed on screen. User is ready to chat.
Does user terminate the sesssion ?
Message is sent
Yes
End
Out-bound messages
In-bound messages
Messages are written in log files
Does the message contain room number?
No
Send to multicast group
Display the message in chat room

Distributed servers : ***Multicasting***

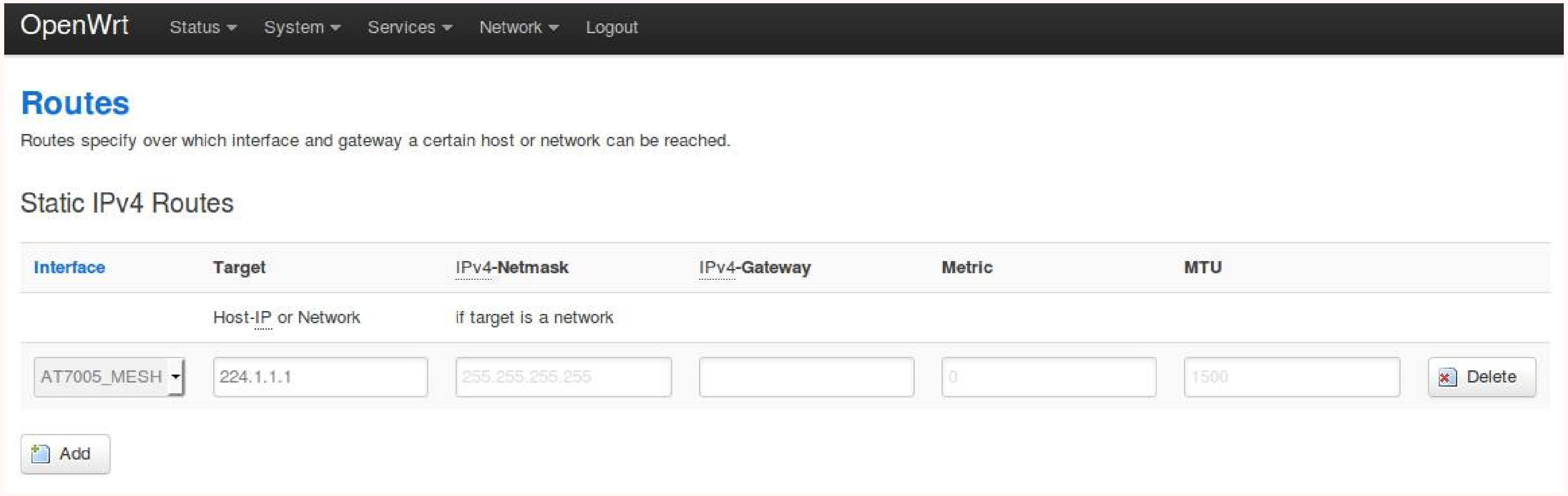


Distributed servers : ***Multicasting*** *cont.*



```
MCAST_GRP = '224.1.1.1'
MCAST_PORT = 8888
addressInNetwork = []

send_address = (MCAST_GRP, MCAST_PORT) # Set the address to send to
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)    # Create Datagram Socket (UDP)
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1) # Make Socket Reusable
s.setblocking(False) # Set socket to non-blocking mode
s.bind((MCAST_GRP, MCAST_PORT))
mreq = struct.pack("4sl", socket.inet_aton(MCAST_GRP), socket.INADDR_ANY)
s.setsockopt(socket.IPPROTO_IP, socket.IP_ADD_MEMBERSHIP, mreq)
s.setsockopt(socket.IPPROTO_IP, socket.IP_MULTICAST_TTL, 2)
```

Distributed servers : ***Multicasting*** *cont.*



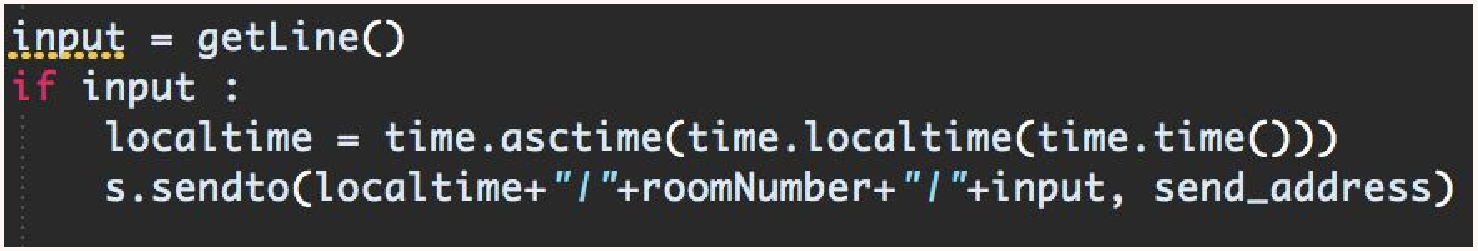
Routes

Routes specify over which interface and gateway a certain host or network can be reached.

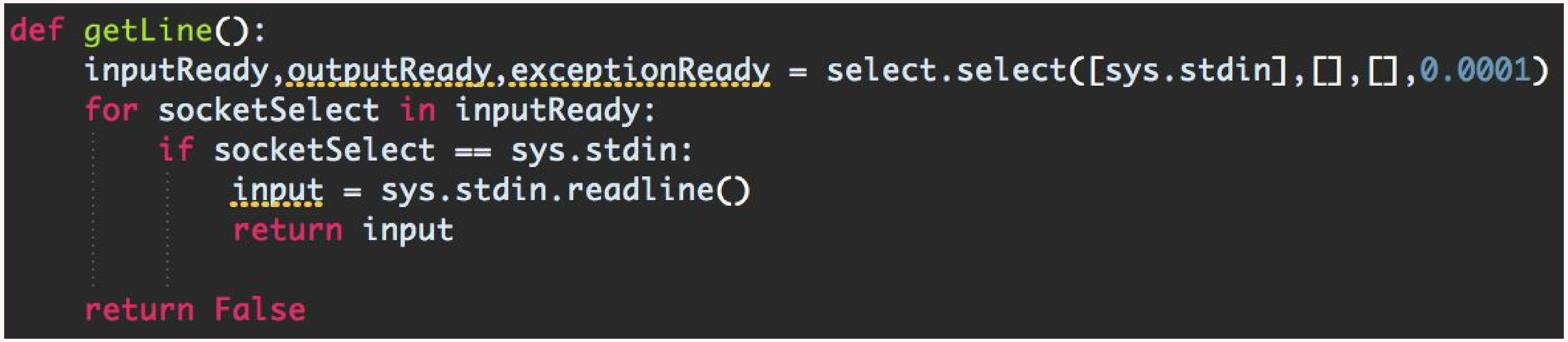
Static IPv4 Routes

Interface	Target	IPv4-Netmask	IPv4-Gateway	Metric	MTU	
	Host-IP or Network	if target is a network				
AT7005_MESH	224.1.1.1	255.255.255.255		0	1500	Delete

Distributed servers : ***Chatting***

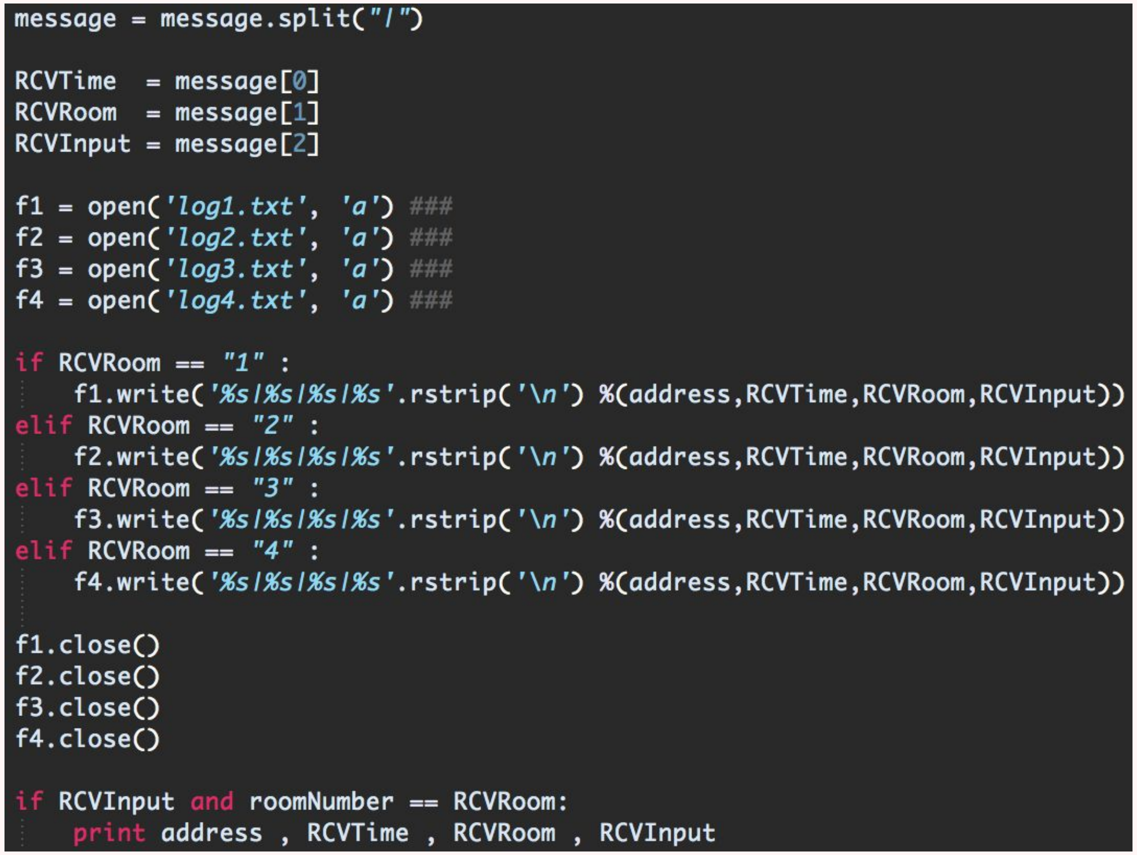


```
input = getLine()
if input :
    localtime = time.asctime(time.localtime(time.time()))
    s.sendto(localtime+"/"+roomNumber+"/"+input, send_address)
```



```
def getLine():
    inputReady,outputReady,exceptionReady = select.select([sys.stdin],[],[],0.0001)
    for socketSelect in inputReady:
        if socketSelect == sys.stdin:
            input = sys.stdin.readline()
            return input

    return False
```

Distributed servers : ***Data store***

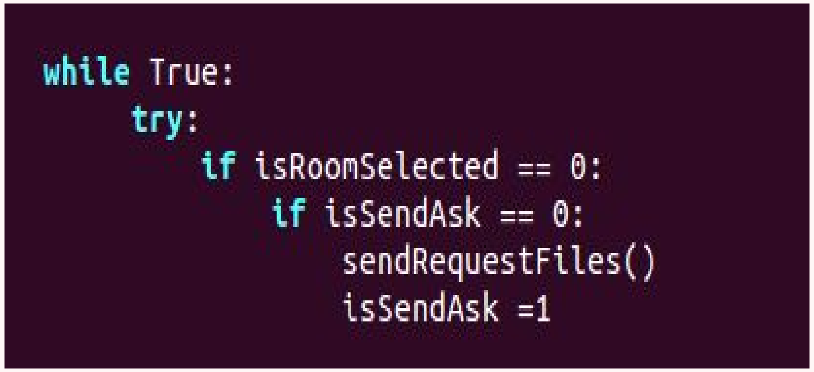


```
message = message.split("/")

RCVTime  = message[0]
RCVRoom  = message[1]
RCVInput = message[2]

f1 = open('log1.txt', 'a') ###
f2 = open('log2.txt', 'a') ###
f3 = open('log3.txt', 'a') ###
f4 = open('log4.txt', 'a') ###

if RCVRoom == "1" :
    f1.write('%s|%s|%s|%s'.rstrip('\n') %(address,RCVTime,RCVRoom,RCVInput))
elif RCVRoom == "2" :
    f2.write('%s|%s|%s|%s'.rstrip('\n') %(address,RCVTime,RCVRoom,RCVInput))
elif RCVRoom == "3" :
    f3.write('%s|%s|%s|%s'.rstrip('\n') %(address,RCVTime,RCVRoom,RCVInput))
elif RCVRoom == "4" :
    f4.write('%s|%s|%s|%s'.rstrip('\n') %(address,RCVTime,RCVRoom,RCVInput))

f1.close()
f2.close()
f3.close()
f4.close()

if RCVInput and roomNumber == RCVRoom:
    print address , RCVTime , RCVRoom , RCVInput
```

Router Failure Handling :
Request and receive chat history messages

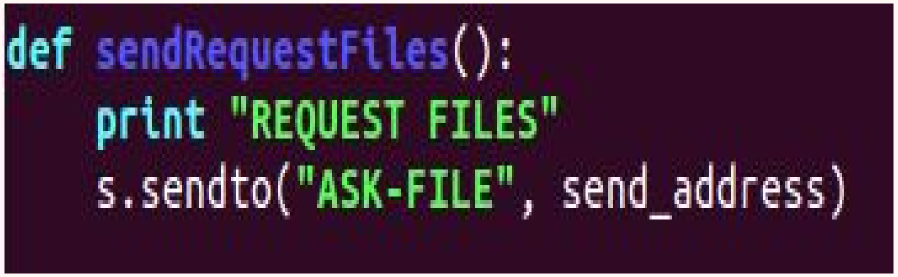
Every time the router turn on, the router will always call **sendRequestFiles()** function first to request the chat log files by sending the message “ASK-FILE” to other routers that are currently running on that time.

main()



```
while True:
    try:
        if isRoomSelected == 0:
            if isSendAsk == 0:
                sendRequestFiles()
                isSendAsk =1
```

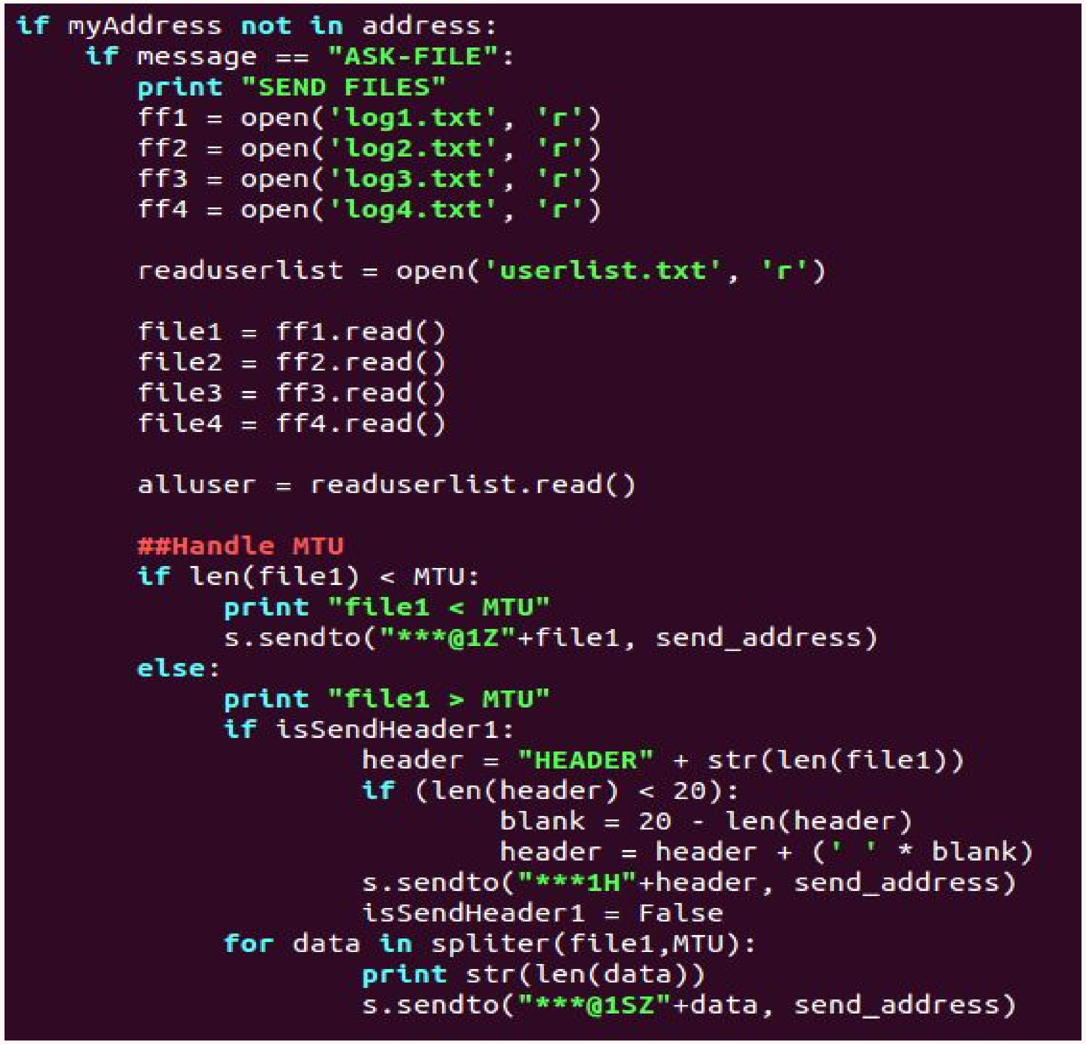
sendRequestFiles()



```
def sendRequestFiles():
    print "REQUEST FILES"
    s.sendto("ASK-FILE", send_address)
```

Router Failure Handling : **Request and receive chat history messages**

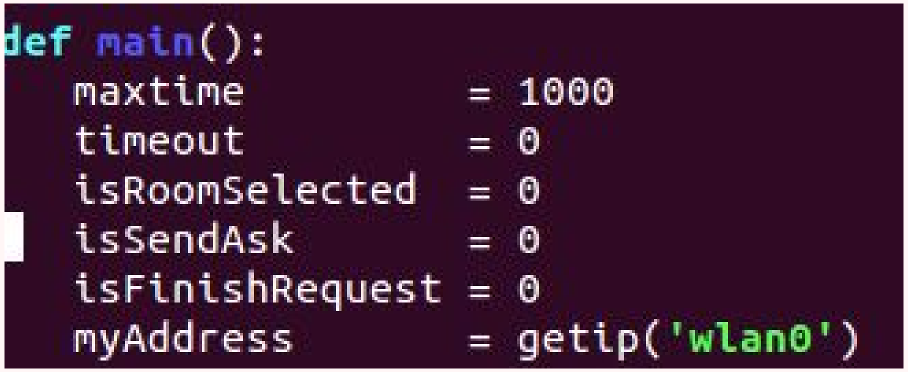
Sending
Chat
Messages
(Ex. log1.txt)



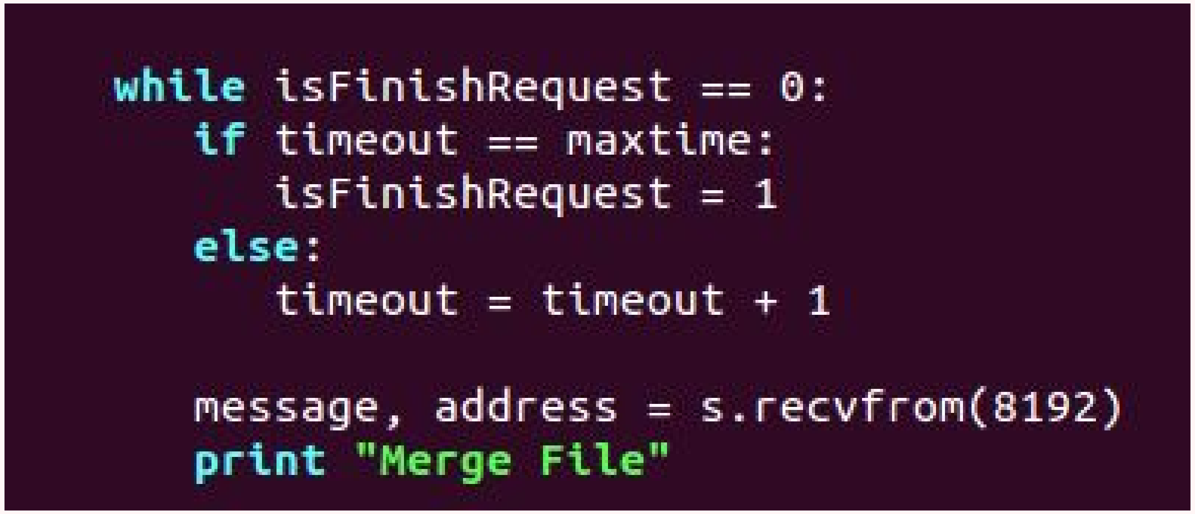
```
if myAddress not in address:
    if message == "ASK-FILE":
        print "SEND FILES"
        ff1 = open('log1.txt', 'r')
        ff2 = open('log2.txt', 'r')
        ff3 = open('log3.txt', 'r')
        ff4 = open('log4.txt', 'r')

        readuserlist = open('userlist.txt', 'r')

        file1 = ff1.read()
        file2 = ff2.read()
        file3 = ff3.read()
        file4 = ff4.read()

        alluser = readuserlist.read()

        ##Handle MTU
        if len(file1) < MTU:
             print "file1 < MTU"
             s.sendto("***@1Z"+file1, send_address)
        else:
             print "file1 > MTU"
             if isSendHeader1:
                     header = "HEADER" + str(len(file1))
                     if (len(header) < 20):
                             blank = 20 - len(header)
                             header = header + (' ' * blank)
                     s.sendto("***1H"+header, send_address)
                     isSendHeader1 = False
             for data in spliter(file1,MTU):
                     print str(len(data))
                     s.sendto("***@1SZ"+data, send_address)
```

Router Failure Handling :
Request and receive chat history messages

Receiving
Chat
Messages



```
def main():
    maxtime         = 1000
    timeout         = 0
    isRoomSelected  = 0
    isSendAsk       = 0
    isFinishRequest = 0
    myAddress       = getip('wlan0')
```

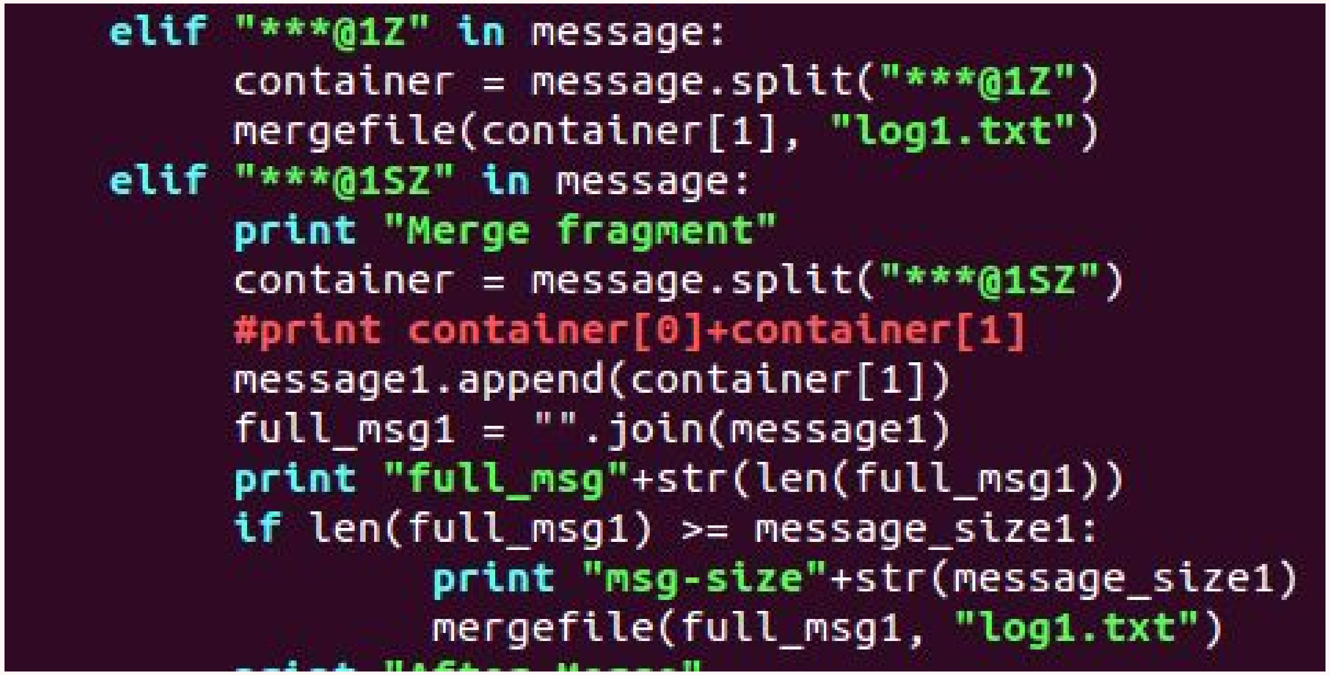


```
    while isFinishRequest == 0:
        if timeout == maxtime:
            isFinishRequest = 1
        else:
            timeout = timeout + 1

        message, address = s.recvfrom(8192)
        print "Merge File"
```

Router Failure Handling :
Request and receive chat history messages

Receiving
Chat
Messages

(Ex. log1.txt)

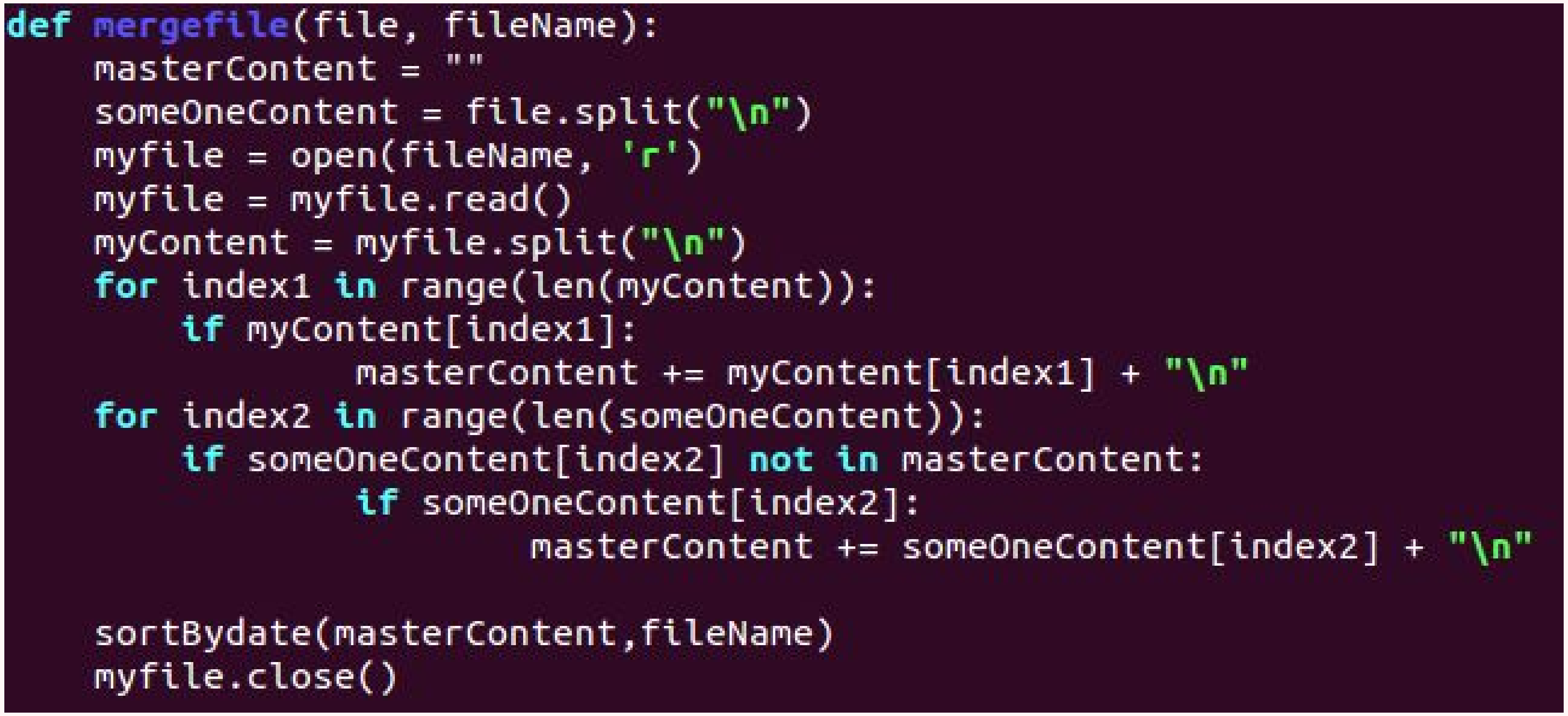


```
    elif "***@1Z" in message:
        container = message.split("***@1Z")
        mergefile(container[1], "log1.txt")
    elif "***@1SZ" in message:
        print "Merge fragment"
        container = message.split("***@1SZ")
        #print container[0]+container[1]
        message1.append(container[1])
        full_msg1 = "".join(message1)
        print "full_msg"+str(len(full_msg1))
        if len(full_msg1) >= message_size1:
            print "msg-size"+str(message_size1)
            mergefile(full_msg1, "log1.txt")
```

Router Failure Handling :
Merging messages

———

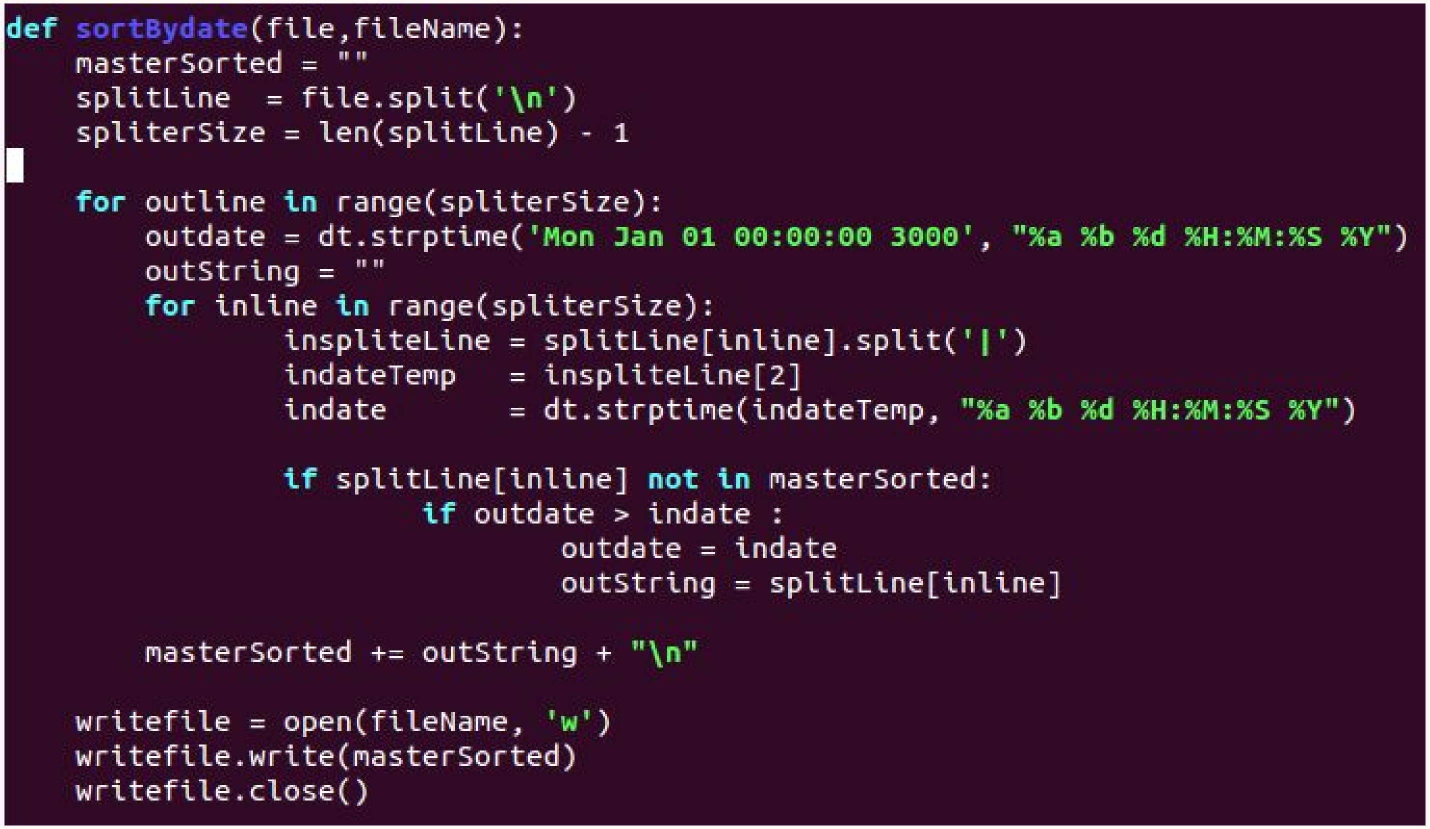
In the **mergefile(file,fileName)** function, it splits the content of chat messages that we have just requested line by line, and then compare with our local log file.



```
def mergefile(file, fileName):
    masterContent = ""
    someOneContent = file.split("\n")
    myfile = open(fileName, 'r')
    myfile = myfile.read()
    myContent = myfile.split("\n")
    for index1 in range(len(myContent)):
        if myContent[index1]:
                masterContent += myContent[index1] + "\n"
    for index2 in range(len(someOneContent)):
        if someOneContent[index2] not in masterContent:
                if someOneContent[index2]:
                        masterContent += someOneContent[index2] + "\n"

    sortBydate(masterContent,fileName)
    myfile.close()
```

Router Failure Handling :
Sorting messages

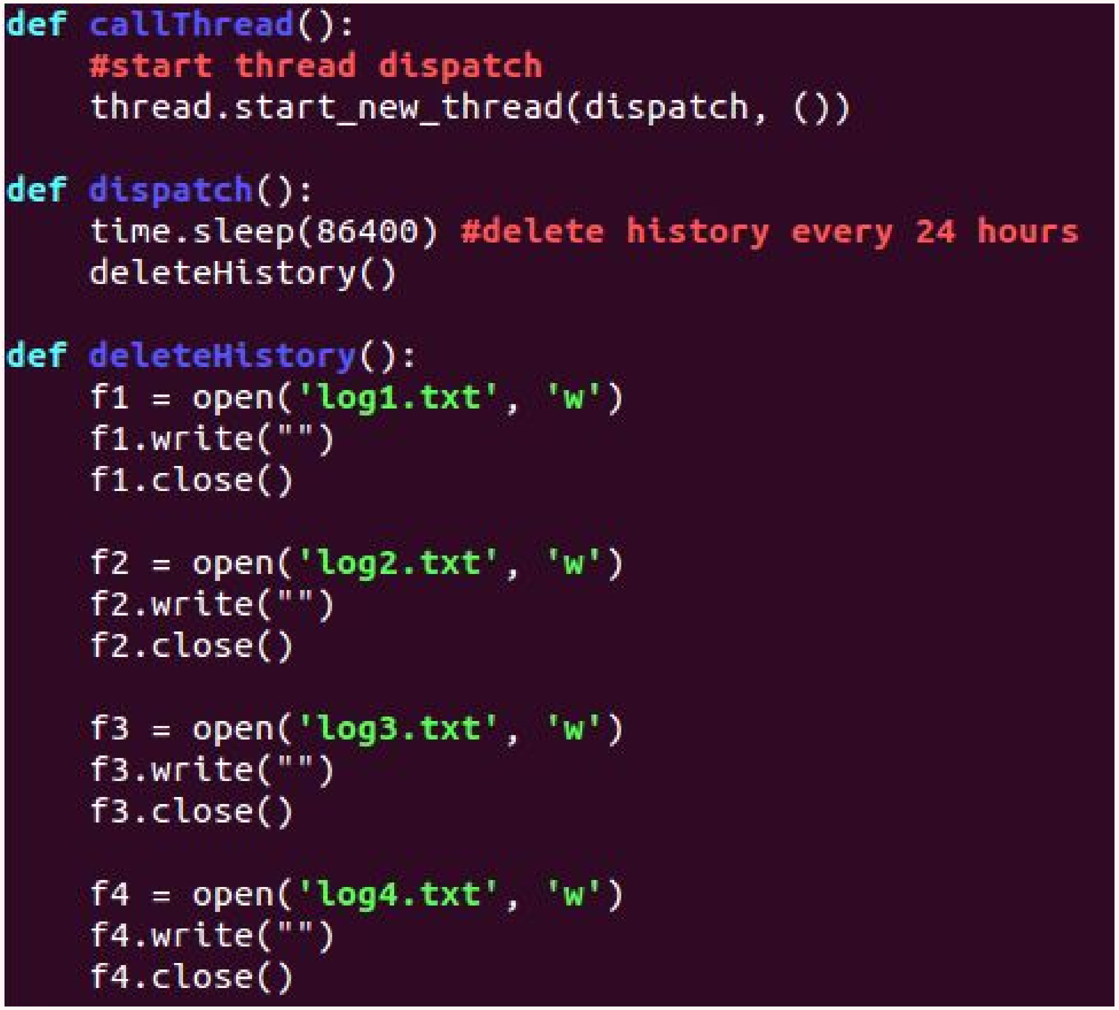
The **sortBydate(file,fileName)** function handles sorting the merged messages to be according to date&time before overwrite them in our local log file.



```
def sortBydate(file,fileName):
    masterSorted = ""
    splitLine  = file.split('\n')
    spliterSize = len(splitLine) - 1

    for outline in range(spliterSize):
        outdate = dt.strptime('Mon Jan 01 00:00:00 3000', "%a %b %d %H:%M:%S %Y")
        outString = ""
        for inline in range(spliterSize):
                inspliteLine = splitLine[inline].split('|')
                indateTemp   = inspliteLine[2]
                indate       = dt.strptime(indateTemp, "%a %b %d %H:%M:%S %Y")

                if splitLine[inline] not in masterSorted:
                        if outdate > indate :
                                outdate = indate
                                outString = splitLine[inline]

        masterSorted += outString + "\n"

    writefile = open(fileName, 'w')
    writefile.write(masterSorted)
    writefile.close()
```

Data Management Control :
Historical messages wiping

In this chat system we design to wipe our chat history everyday.



```
def callThread():
    #start thread dispatch
    thread.start_new_thread(dispatch, ())

def dispatch():
    time.sleep(86400) #delete history every 24 hours
    deleteHistory()

def deleteHistory():
    f1 = open('log1.txt', 'w')
    f1.write("")
    f1.close()

    f2 = open('log2.txt', 'w')
    f2.write("")
    f2.close()

    f3 = open('log3.txt', 'w')
    f3.write("")
    f3.close()

    f4 = open('log4.txt', 'w')
    f4.write("")
    f4.close()
```

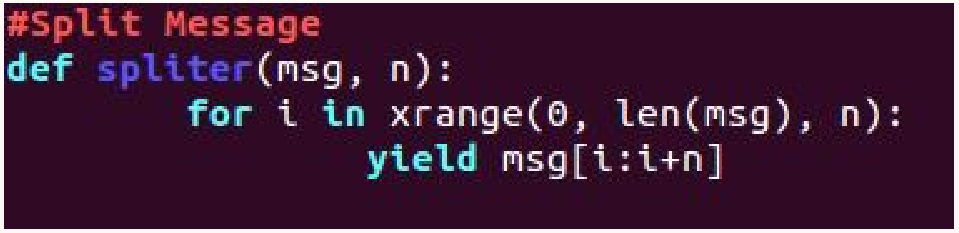
MTU Fragmentation:
Maximum Transmission Unit

———

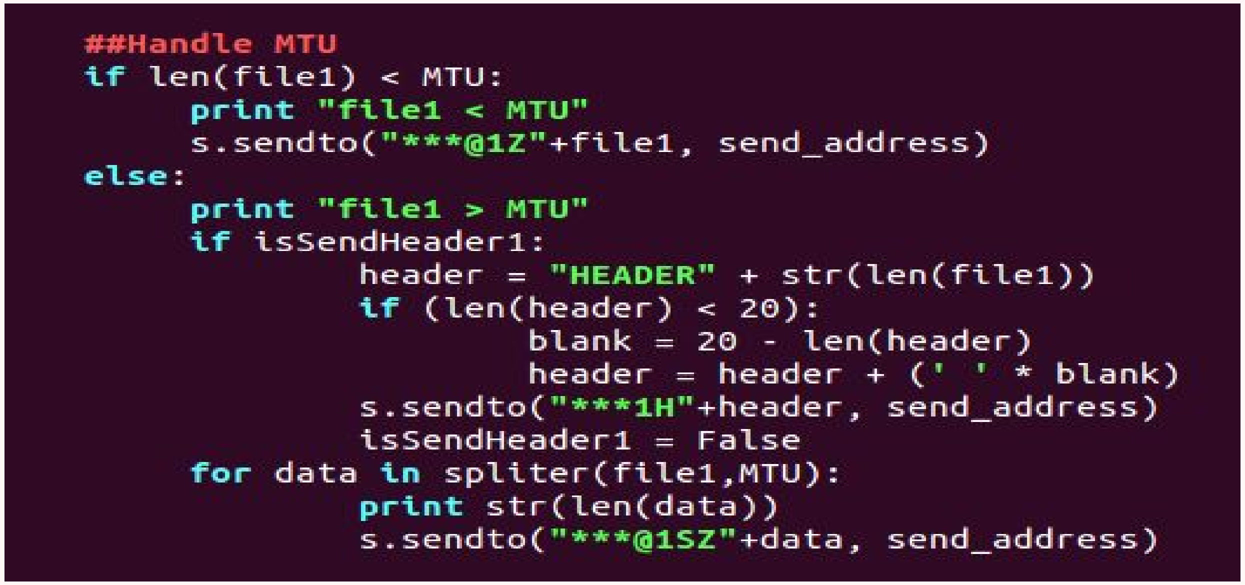
Fragment messages!!

(Ex. log1.txt)

MTU = 1500 bytes (wlan0)



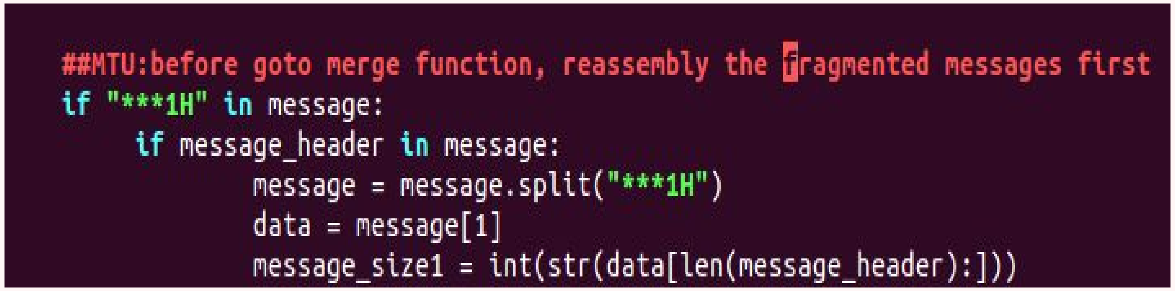
```
#Split Message
def spliter(msg, n):
        for i in xrange(0, len(msg), n):
                yield msg[i:i+n]
```



```
##Handle MTU
if len(file1) < MTU:
        print "file1 < MTU"
        s.sendto("***@1Z"+file1, send_address)
else:
        print "file1 > MTU"
        if isSendHeader1:
                header = "HEADER" + str(len(file1))
                if (len(header) < 20):
                        blank = 20 - len(header)
                        header = header + (' ' * blank)
                s.sendto("***1H"+header, send_address)
                isSendHeader1 = False
        for data in spliter(file1,MTU):
                print str(len(data))
                s.sendto("***@1SZ"+data, send_address)
```

MTU Fragmentation:
Maximum Transmission Unit

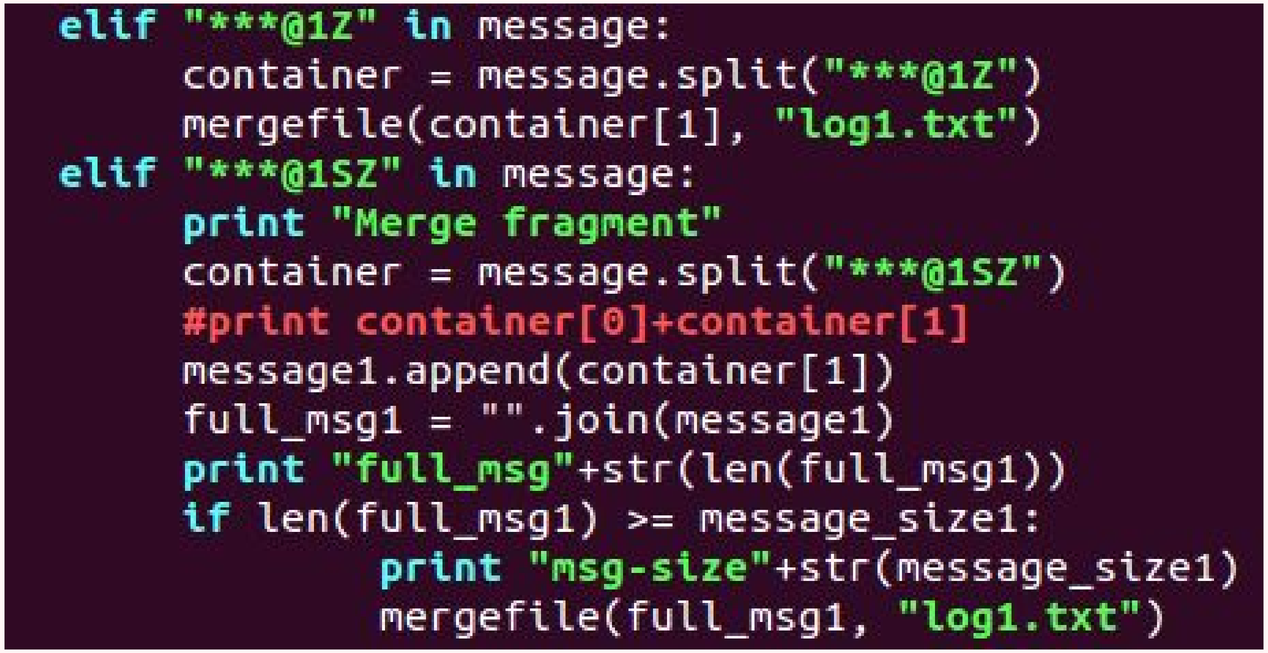
MTU = 1500 bytes (wlan0)



```
##MTU:before goto merge function, reassembly the fragmented messages first
if "***1H" in message:
    if message_header in message:
            message = message.split("***1H")
            data = message[1]
            message_size1 = int(str(data[len(message_header):]))
```

Reassembly messages!!

(Ex. log1.txt)



```
elif "***@1Z" in message:
        container = message.split("***@1Z")
        mergefile(container[1], "log1.txt")
elif "***@1SZ" in message:
        print "Merge fragment"
        container = message.split("***@1SZ")
        #print container[0]+container[1]
        message1.append(container[1])
        full_msg1 = "".join(message1)
        print "full_msg"+str(len(full_msg1))
        if len(full_msg1) >= message_size1:
                print "msg-size"+str(message_size1)
                mergefile(full_msg1, "log1.txt")
```

Demo

Conclusion & Future work

Conclusion:

- Every core challenges are successfully done
  - Do multiple hops transferring by multicast
  - Sync and order the messages by write once & datetime
  - Handle router failure by pulling mechanism -> datetime merge & sort
  - Reduce loss rate from exceeding MTU by datagram fragmentation

Future work:

- Plan to implement a UI
- Apply Erasure coding for data protection & recovery (n=k+m)

Thank You