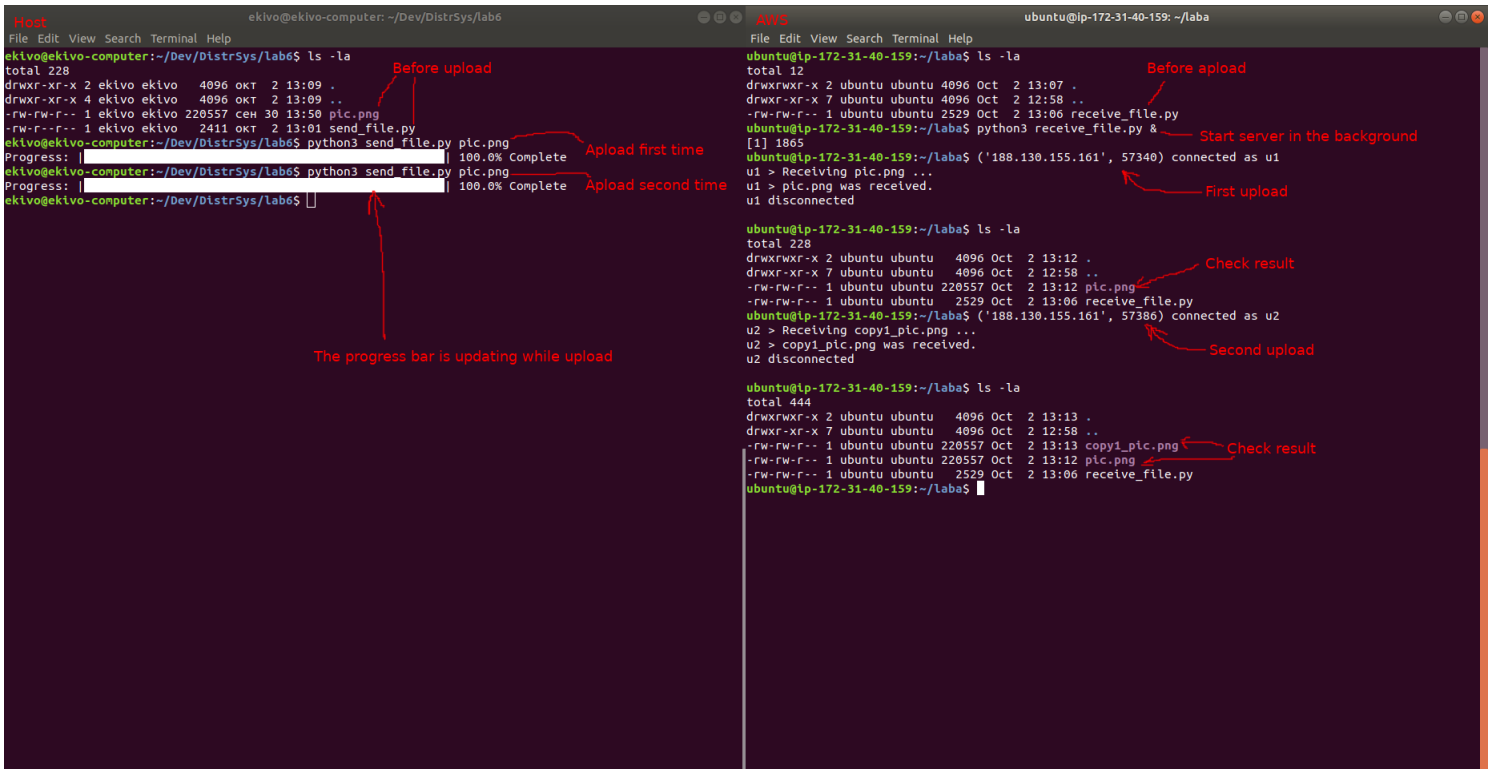
Lab 6: Message-oriented Communication

Kivotova Evgenia, DS-01

October 2019

1 Sending process + ls -la before and after upload



2 Server source code

```python
import socket
from threading import Thread
import os.path

clients = []


# Thread to listen one particular client
class ClientListener(Thread):
    def __init__(self, name: str, sock: socket.socket):
        super().__init__(daemon=True)
        self.sock = sock
        self.name = name

    # clean up
    def _close(self):
        clients.remove(self.sock)
        self.sock.close()
        print(self.name + ' disconnected')

    def run(self):
        # read the length of filename
        l_filename = int(self.sock.recv(4))
```

```
        # read the name of file
        data = self.sock.recv(1_filename)
        if data:
            filename = data.decode('utf-8')

            #check if file exists and add 'copyi_' to the begining of its name
            i = 0
            extra_part = ''
            while os.path.isfile(extra_part+filename):
                i += 1
                extra_part = 'copy'+str(i)+'_'
            filename = extra_part+filename
            f = open(filename, 'wb')
            print( self.name,'>','Receiving',filename,'...')
        else:
            # if we got no data     client has disconnected
            self._close()
            # finish the thread
            return
        while True:
            # try to read 1024 bytes from user
            # this is blocking call, thread will be paused here
            data = self.sock.recv(512)
            if data:
                f.write(data)
            else:
                # if we got no data     client has disconnected
                print(self.name,'>',filename,'was received.')
                f.close()
                self._close()
                # finish the thread
                return


def main():
    next_name = 1

    # AF_INET     IPv4, SOCK_STREAM     TCP
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    # reuse address; in OS address will be reserved after app closed for a while
    # so if we close and imidiatly start server again     we'll get error
    sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    # listen to all interfaces at 8800 port
    sock.bind(('localhost', 8800))
    sock.listen()
    while True:
        # blocking call, waiting for new client to connect
        con, addr = sock.accept()
        clients.append(con)
        name = 'u' + str(next_name)
        next_name += 1
        print(str(addr) + ' connected as ' + name)
        # start new thread to deal with client
        ClientListener(name, con).start()


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

3 Client source code

```
import socket
import sys
import os

# define server
SERVER_DOMAIN = 'ec2-18-212-138-166.compute-1.amazonaws.com'
SERVER_PORT = 8800

# define filename
FILE_NAME = sys.argv[1]

# Print file sending progress (taken from elekt.tech)
def printProgressBar (iteration, total, prefix = '', suffix = '', decimals = 1, length = 100, fill = '
    '):
    """
```

```
    Call in a loop to create terminal progress bar
    @params:
        iteration   - Required  : current iteration (Int)
        total       - Required  : total iterations (Int)
        prefix      - Optional  : prefix string (Str)
        suffix      - Optional  : suffix string (Str)
        decimals    - Optional  : positive number of decimals in percent complete (Int)
        length      - Optional  : character length of bar (Int)
        fill        - Optional  : bar fill character (Str)
    """
    percent = ("{0:." + str(decimals) + "f}").format(100 * (iteration / float(total)))
    filledLength = int(length * iteration // total)
    bar = fill * filledLength + '-' * (length - filledLength)
    print('\r%s |%s| %s%% %s' % (prefix, bar, percent, suffix), end = '\r')
    # Print New Line on Complete
    if iteration == total:
        print()

# check the file exists
if not os.path.isfile(FILE_NAME):
    print('ERROR:',FILE_NAME, 'is not in the current derictory.')
    exit()

# connect to server
cli_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
cli_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
cli_socket.connect((SERVER_DOMAIN,SERVER_PORT))

# define the length of the filename, extend it if needed
name_length = bytes(str(len(FILE_NAME)), 'utf-8')
name_length = (b'0')*(4-len(name_length)) + name_length

# send name length to server
cli_socket.send(name_length)

# send file name to server
cli_socket.send(bytes(FILE_NAME, 'utf-8'))

# define the length of file
file_length = os.path.getsize(FILE_NAME)

# send file
with open(FILE_NAME,'rb') as f:
    # print enpty progress
    printProgressBar(0, file_length, prefix = 'Progress:', suffix = 'Complete', length = 50)
    data = f.read(512)
    sent = 0
    while data:
        # send data from file
        sent += cli_socket.send(data)
        #print progress
        printProgressBar(sent, file_length, prefix = 'Progress:', suffix = 'Complete', length = 50)
        data = f.read(512)
    # close file
    f.close()
#close connection
cli_socket.close()
```

4 GitHub

https://github.com/Genvekt/Distributive_Systems_F19