System architecture and design choices

The goal of the system is to implement a reliable data transmission between a client and a server, based on the unreliable UDP protocol. To add reliability some basic functions of the TCP protocol, such as the use of *Acknowledgements* and retransmissions, were implemented in the application.

- **Software Platform**
  The system was developed on Unix-like systems using Sublime Text and executed using the terminal. No special application is requested to correctly execute the program, except for the necessary use of *mate terminal* to print a progress bar during transmission.

- **Implementation**
  The system is separated in a client part and a server part, both stored in different directories with their own files, to simulate to different hosts.

  The server supports three different requests:
    - LIST, which provides a list of the files the server can transmit to the client.
    - GET, which permits the download of a certain file from the server to the client.
    - PUT, which permits the upload of a new file from the client to the server.

  Files are transmitted using sockets set up with the UDP protocol.

  Reliability is obtained using packets containing the type of the message, the data transmitted, the sequence number referring to the byte stream and the ack number. Packets transmitted get stored in a transmission window, whose base packet gets associated with a timeout struct to manage retransmissions.

  The size of the transmission window was set to 10 packets, while the timeout interval was implemented as adaptive so that whenever an ACK is received for a known packet, the estimation of the RTT permits to recalculate the interval according to the traffic on the channel.

  Retransmission also occurs when the sender receives three duplicated ACKs.

  An artificial packet loss was also added in the implementation to simulate a real transmission channel. Since the system was developed and tested on one device, the effect of the artificial loss is added to the natural loss due to the speed with which the transmitted packets arrive at their destination.

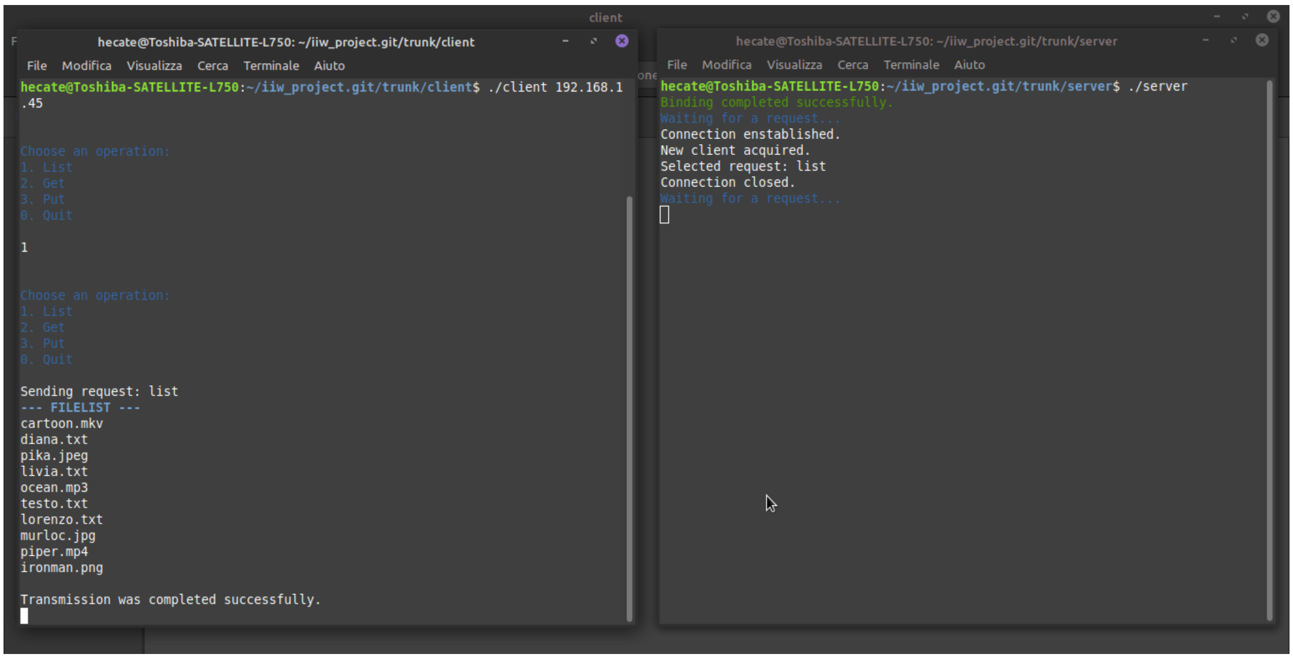
  The loss probability is a fixed value that is used along side the `rand` function: whenever a packet must be sent, the system checks whether a random value is a multiple of the loss probability and, if so, the packet is not transmitted.

  To provide a connection between the hosts, before the actual file transmission, a mechanism of 3-way handshake was implemented. After the handshake, the server adds a particular client to the queue of clients in service. One of the working threads of the server will take care of responding to the request of the new client.

  When a transmission terminates, the connection gets closed with the final exchange of messages and the client is removed from the server's queue.

  Both the client and the server are implemented with a multithreaded structure. The server spawns ten working threads that will wait for incoming requests. The client spawns a new thread for every concurrent request that it will send to the server. Every thread of the client will use a new socket to make the requests distinguishable on the server side.

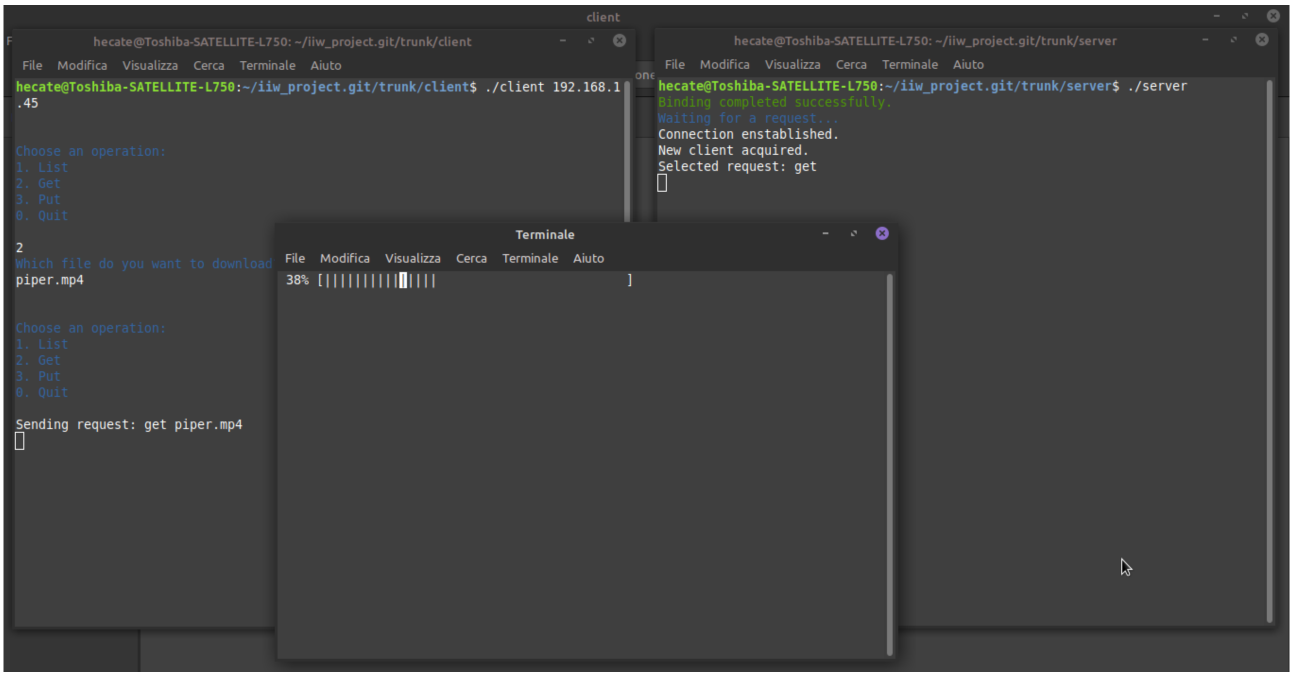
Examples


client
hecate@Toshiba-SATELLITE-L750: ~/iiw_project.git/trunk/client
File Modifica Visualizza Cerca Terminale Aiuto
hecate@Toshiba-SATELLITE-L750: ~/iiw_project.git/trunk/server
File Modifica Visualizza Cerca Terminale Aiuto

```
hecate@Toshiba-SATELLITE-L750:~/iiw_project.git/trunk/client$ ./client 192.168.1
.45


Choose an operation:
1. List
2. Get
3. Put
0. Quit

1


Choose an operation:
1. List
2. Get
3. Put
0. Quit

Sending request: list
--- FILELIST ---
cartoon.mkv
diana.txt
pika.jpeg
livia.txt
ocean.mp3
testo.txt
lorenzo.txt
murloc.jpg
piper.mp4
ironman.png

Transmission was completed successfully.
```

```
hecate@Toshiba-SATELLITE-L750:~/iiw_project.git/trunk/server$ ./server
Binding completed successfully.
Waiting for a request...
Connection enstablished.
New client acquired.
Selected request: list
Connection closed.
Waiting for a request...
```

List

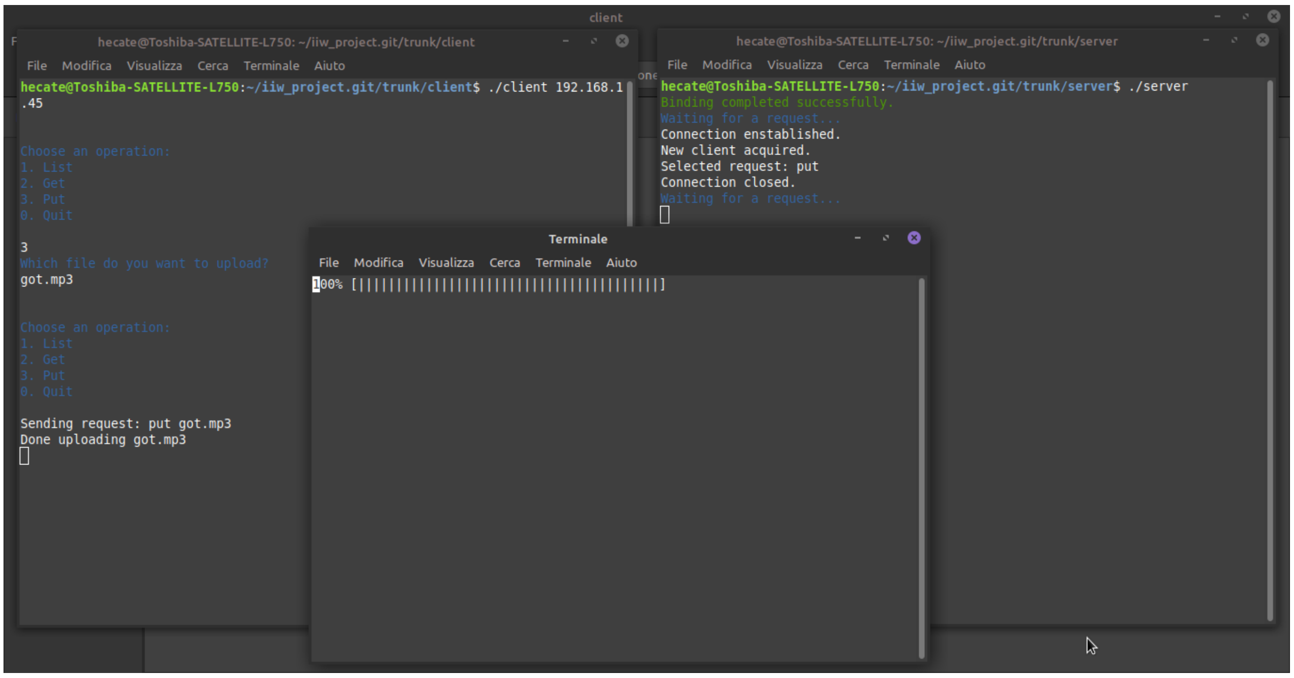

client
hecate@Toshiba-SATELLITE-L750: ~/iiw_project.git/trunk/client
File Modifica Visualizza Cerca Terminale Aiuto
hecate@Toshiba-SATELLITE-L750: ~/iiw_project.git/trunk/server
File Modifica Visualizza Cerca Terminale Aiuto
Terminale
File Modifica Visualizza Cerca Terminale Aiuto

```
hecate@Toshiba-SATELLITE-L750:~/iiw_project.git/trunk/client$ ./client 192.168.1
.45

Choose an operation:
1. List
2. Get
3. Put
0. Quit

2
Which file do you want to download
piper.mp4


Choose an operation:
1. List
2. Get
3. Put
0. Quit

Sending request: get piper.mp4
```

```
hecate@Toshiba-SATELLITE-L750:~/iiw_project.git/trunk/server$ ./server
Binding completed successfully.
Waiting for a request...
Connection enstablished.
New client acquired.
Selected request: get
```

```
38% [|||||||||||||||                                    ]
```

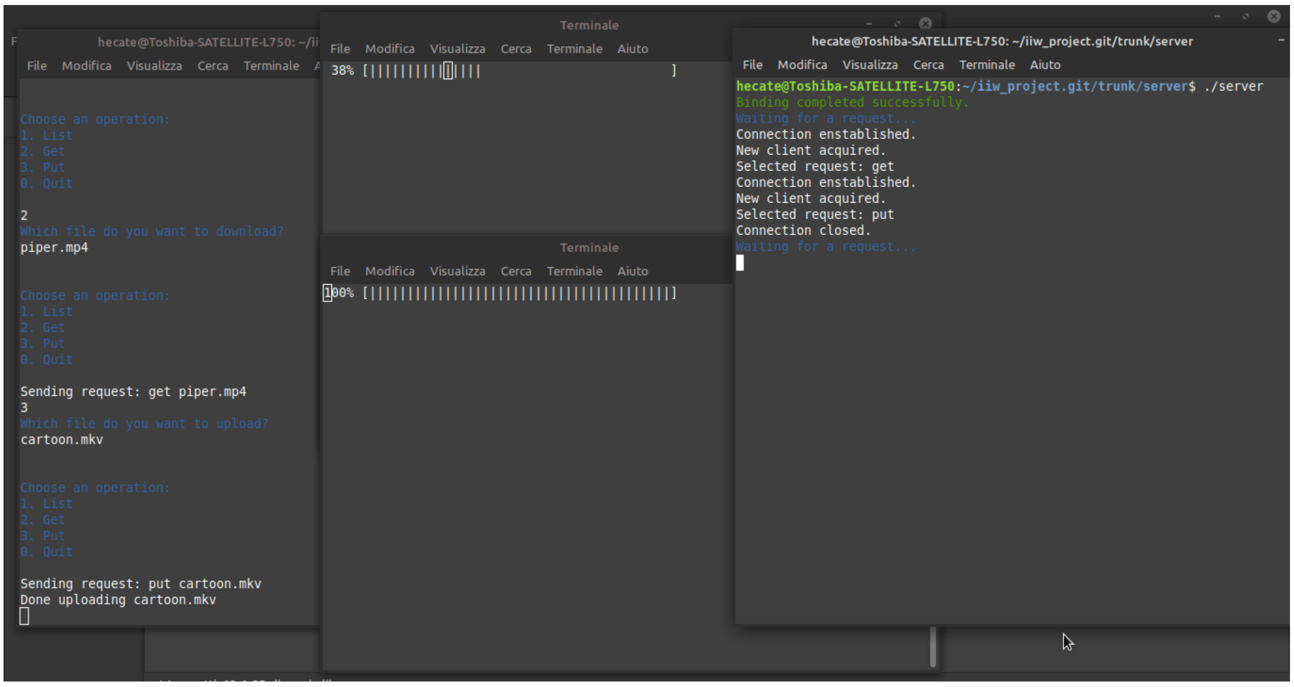
Get


```
hecate@Toshiba-SATELLITE-L750:~/iiw_project.git/trunk/client$ ./client 192.168.1
.45


Choose an operation:
1. List
2. Get
3. Put
0. Quit

3
Which file do you want to upload?
got.mp3


Choose an operation:
1. List
2. Get
3. Put
0. Quit

Sending request: put got.mp3
Done uploading got.mp3
```

```
100% [||||||||||||||||||||||||||||||||||||||||]
```

```
hecate@Toshiba-SATELLITE-L750:~/iiw_project.git/trunk/server$ ./server
Binding completed successfully.
Waiting for a request...
Connection enstablished.
New client acquired.
Selected request: put
Connection closed.
Waiting for a request...
```

Put

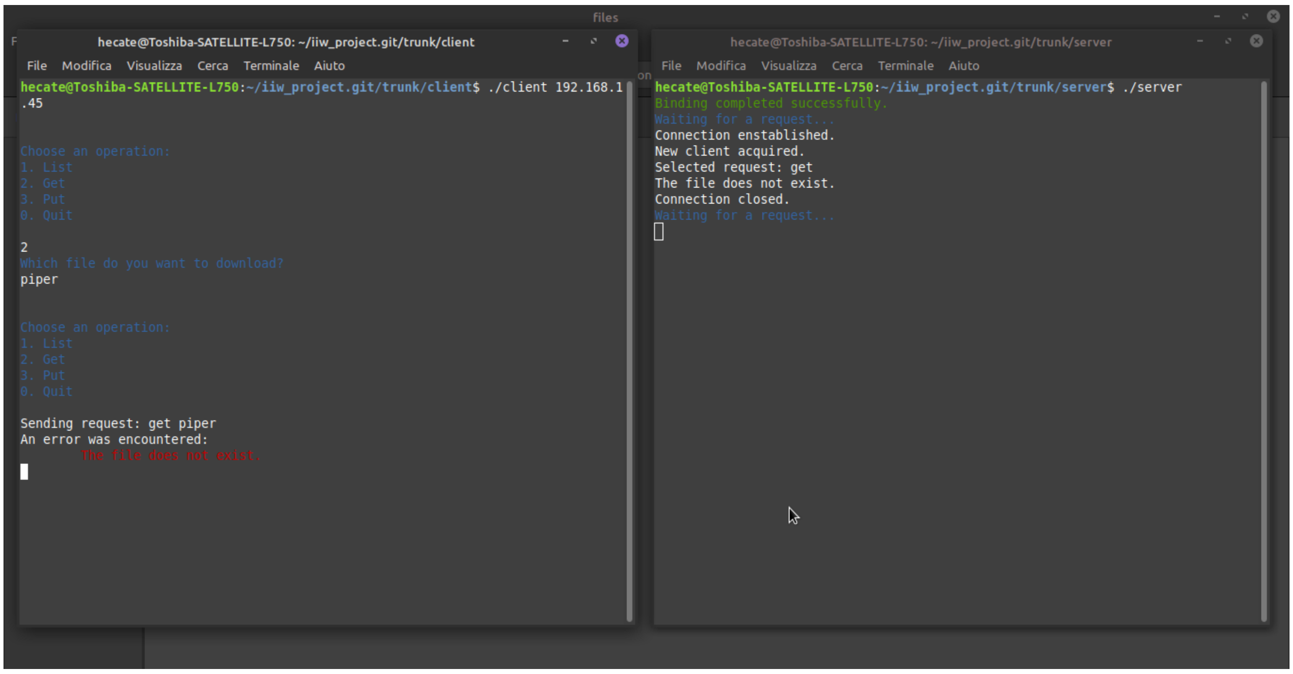

```
Choose an operation:
1. List
2. Get
3. Put
0. Quit

2
Which file do you want to download?
piper.mp4


Choose an operation:
1. List
2. Get
3. Put
0. Quit

Sending request: get piper.mp4
3
Which file do you want to upload?
cartoon.mkv


Choose an operation:
1. List
2. Get
3. Put
0. Quit

Sending request: put cartoon.mkv
Done uploading cartoon.mkv
```

```
38% [|||||||||||||||                          ]
```

```
100% [||||||||||||||||||||||||||||||||||||||||]
```

```
hecate@Toshiba-SATELLITE-L750:~/iiw_project.git/trunk/server$ ./server
Binding completed successfully.
Waiting for a request...
Connection enstablished.
New client acquired.
Selected request: get
Connection enstablished.
New client acquired.
Selected request: put
Connection closed.
Waiting for a request...
```

Concurrent requests (Put and Get)


files
hecate@Toshiba-SATELLITE-L750: ~/iiw_project.git/trunk/client
File Modifica Visualizza Cerca Terminale Aiuto
hecate@Toshiba-SATELLITE-L750: ~/iiw_project.git/trunk/server
File Modifica Visualizza Cerca Terminale Aiuto

```
hecate@Toshiba-SATELLITE-L750:~/iiw_project.git/trunk/client$ ./client 192.168.1
.45


Choose an operation:
1. List
2. Get
3. Put
0. Quit

2
Which file do you want to download?
piper


Choose an operation:
1. List
2. Get
3. Put
0. Quit

Sending request: get piper
An error was encountered:
        The file does not exist.
```

```
hecate@Toshiba-SATELLITE-L750:~/iiw_project.git/trunk/server$ ./server
Binding completed successfully.
Waiting for a request...
Connection enstablished.
New client acquired.
Selected request: get
The file does not exist.
Connection closed.
Waiting for a request...
```

Error handler

Performance evaluation

The system uses three crucial parameters:

- Size of the transmission window (*N*), which is used to store the packets “in flight”.
- Loss probability (*P*), which is used to simulate the transmission over a lossy channel.
- Timeout (*T*), which is used to implement the retransmission mechanism.

While *N* and *P* are fixed values, *T* was implemented as adaptive along with a periodic RTT estimation.

As these parameters vary, the performances of the system will vary with them.

To test this variation, different situations were considered. The function used to test the system is GET, which resulted in being the slowest one.

- **Test 1**: values used normally (*N* = 10, *P* = 50, *T* adaptive)
    - `real 0m58,864s`
    - `user 0m0,103s`
    - `sys  0m0,179s`

- **Test 2**: lower dimension of the transmission window (*N* = 5, *P* = 50, *T* adaptive)
    - `real 1m5,289s`
    - `user 0m0,098s`
    - `sys  0m0,201s`

- **Test 3**: higher dimension of the transmission window (*N* = 20, *P* = 50, *T* adaptive)
    - `real 1m52,277s`
    - `user 0m0,145s`
    - `sys  0m0,210s`

- **Test 4**: higher probability of packet loss (*N* = 10, *P* = 10, *T* adaptive)
    - `real 1m3,146s`
    - `user 0m0,134s`
    - `sys  0m0,139s`

- **Test 5**: lower probability of packet loss (*N* = 10, *P* = 100, *T* adaptive)
    - `real 0m47,241s`
    - `user 0m0,069s`
    - `sys  0m0,181s`

- **Test 6**: fixed timeout interval of 1 second (*N* = 10, *P* = 50, *T* 1s)
    - `real 0m57,011s`
    - `user 0m0,079s`
    - `sys  0m0,191s`

- **Test 7**: fixed timeout interval of 3 seconds (*N* = 10, *P* = 50, *T* 3s)
    - `real 2m2,196s`
    - `user 0m0,065s`
    - `sys  0m0,200s`

Instructions

The source code is divided into two folders. To run the program, two different shells must be opened, one in the client directory and the other in the server directory. Both have their own `Makefile` to compile every module.

The server must be the first to be executed, using the command `./server` after compiling with `make`. After this, the client will be executed in the other shell with `./client <local IP-address>`.

The client will choose which service to request using an integer from 1 to 3 (0 to exit the program) and the server will respond consequently. When requesting the upload or download of the file, the name of the file must be provided, including its extension.

To avoid having errors caused by the management of the progress bar used during the PUT and GET requests, *mate-terminal* must be used on the device that executes the program.