Assignment 3

3-06-2019

Programmazione di Reti

Ing. Chiara Contoli, PhD
chiara.contoli@unibo.it

Corso di Laurea Triennale in Ingegneria e Scienze Informatiche

Homework

Goal. The goal of this assignment is to leverage the experience acquired so far with socket programming in order to build a simple measurement protocol. Measurements concern Round Trip Time and Throughput in a TCP connection.

The **RTT** is the time elapsed between a segment is sent and its corresponding ACK is received.

The **Throughput** is the total quantity of information sent in the unit time.

Proper formulae exist for both indicator, but you will emulate the measurement as described in the following, i.e., adopting a very simplified version of those formulae.

Your assignment

The assignment contains two parts: an analysis/design phase (*Task1*) and the implementation phase (*Task2*).

Task1: In this task, you are asked to provide a Finite State Machine (FSM) of the behavior of both the sender (Client) and the receiver (Server) side. In terms of "how to build" the FSM, you can get inspiration from Kurose-Ross examples (see the example *rdt* FSM, page 181 | the TCP sequence states at page 218).

This means that you are free to choose to build an FSM of one of the following two types:

- Transitions described as pseudo-code (as in the *rdt* example, page 181);
- Transitions described in plain text (as in the TCP example, page 218).

Regardless the type you choose, the FSM **must** contain:

- Main states;
- In transitions, the input message and the corresponding output action.

Your assignment

Task2: in this task, you are asked to implement a TCP Client and a TCP Server to implement the behavior, and the measurement protocol, described in the following. As a starting point for your implementation, you can leverage part of the TCP Client and Server source code provided at lab classes, slides 2_TCP_socket.pdf .

The protocol you are asked to implement consists of 3 phase:

- Hello phase (HP);
- Measurement phase (MP);
- Bye phase (BP).

Each phase is composed of an (exact) message format, and a certain Client/Server interaction.

It is implied that, before carry out those phases, the Client and the Server has to establish the TCP connection via normal three-way handshake.

Your assignment

Hello phase: is the first phase of the protocol, and is used by the Client to inform the Server that it want to carry out a network measurement to compute the RTT or the Throughput along the path between them.

Client behavior: once the TCP connection is established, Client **must** send a single message to the Server adopting the following format:

<*protocol_phase*> <*sp*> <*measure_type*> <*sp*> <*n_probes*> <*sp*> <*msg_size*> <*sp*> <*server_delay*>\n

<*protocol_phase*>: in this first phase, will be denoted by '***h***'; the server will use this value to differentiate between the 3 protocol phases.

<*measure_type*>: used by Client to tell Server what it wants to compute: '***rtt***' for RTT, '***thput***' for Throughput.

<*n_probes*>: used by Client to tell Server the number of measurement probes the Server should expect to receive. The Server will have to echo back all these probe messages so that the Client will be able to take a measurement of each one. Once all those measurement has been collected, the Client should compute the mean (average) RTT, or mean Throughput, depending on the type of measurement being performed.

The probe message's format will be described in the Measurement Phase.

Your assignment

<*msg_size*>: number of bytes contained in the probe’s payload.

<*server_delay*>: amount of time used by Server as a waiting time before echoing the message back to the client. This parameter wants to emulate the propagation delay along the path, and by default should be set to 0. You will then vary this parameter later, in order to emulate path with higher propagation delay. **Note** that such delay actually emulate the processing time at the server, but it will be used as something that affect also the propagation delay.

<*sp*>: it is a single white space used as fields separator (this will be useful for parsing purposes).

\n: new line char, indicating the end of the message.

Server behavior: upon Hello message reception, Server has to parse and validate the message. If the Hello message is valid, the Server has to respond with a text message containing “200 OK - Ready”, telling the Client that it can proceed to the next phase. If the Hello message is incomplete or invalid (e.g., invalid message type | number of probes under threshold | other types of unexpected values), the Server has to respond with a text message containing the string “404 ERROR – Invalid Hello message”, and then terminate the (TCP) connection.

Your assignment

Measurement phase: this protocol phase is used by Client to start sending probe messages to the Server in order to make appropriate measurements required for computing the mean RTT or the mean throughput of the path between each others.

Client behavior: it sends the specified number of probe messages to the Server with an increasing sequence number starting from 1. The probe message format to be adopted is as follow:

<*protocol_phase*> <*sp*> <*probe_seq_num*> <*sp*> <*payload*>\n

<*protocol_phase*>: in this phase, will be denoted by ‘*m*’.

<*probe_seq_num*>: sequence number of the probe message, starting from 1 and up to the number of probes specified in the Hello message (field <*n_probes*>).

<*payload*>: acts as probe’s payload and can be any text string whose size was specified in the Hello message (field <*msg_size*>).

Your assignment

Server behavior: it should echo back to the client every probe message received. As additional task, it should keep track of the probe sequence numbers in order to check that it's incremented by 1 each time, and also to check that the number of probes do not exceed the number of probes specified in the Hello message (field <*n_probes*>). If the probe message is incomplete or invalid (e.g., the sequence number is incorrect), the Server should not echo the message back to the Client. In such case, the Server has to respond with a text message containing the string "404 ERROR – Invalid Measurement message", and then terminate the connection.

Bye phase: used by both Client and Server to gracefully close the connection.

Client behavior: unless the connection was already terminated due to an error in the previous phase, the Client should send a termination request to the Server and wait for the response. Once the response is received, the client should terminate the TCP connection. The Bye message format **must** be as follows:

<*protocol_phase*>\n

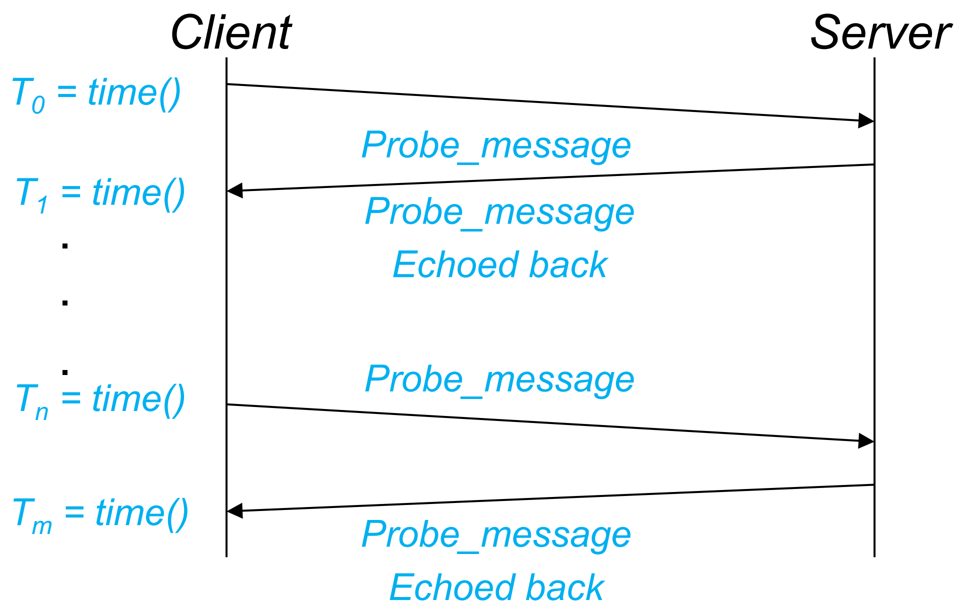
<*protocol_phase*>: in this phase, will be denoted by '*b*'

Server behavior: if the message format is correct, the server has to respond with a text message containing the string "200 OK – Closing".

Suggestions

To measure the RTT, you will have to send and receive messages of size 1, 100, 200, 400, 800 and 1000 bytes; whereas, to measure the throughput, you will send and receive messages of size 1K, 2K, 4K, 16K and 32K bytes. **Note** that RTT and Throughput, for the set of messages size, has two different units.

For each measurement and for each message size, the Client will have to send 20 probe messages to the Server, which will have to be echoed back.



RTT will be evaluated as the difference between, e.g., T_1 and T_0 . The average can be evaluated as the arithmetic mean of all the evaluated RTTs.

Suggestions

The average throughput will be evaluated as the ratio between the size of the (echoed back) message probe (i.e., the size of the whole probe message) over the average RTT). The size can be computed by simply making the sum of the parts composing the message (spaces included).

Note that, if the computation of the Throughput is requested, in order to evaluate its mean, the RTT needs to be evaluated anyway.

20 acts as threshold for the number of probes.

Multiple resolution strategies exists.

Your assignment

Client and Server behavior notes:

Server. At start-up time, server takes as input the port on which it will listen for incoming connection requests. As soon as a client is connected, server **must** print client's IP address and port.

Whenever the server receive/send a message from/to the client, it **must** print received/sent message (whether things goes wrong or successful).

Client. At start-up time, client takes as input the IP address and the port number of the server. Whenever the client receive/send a message from/to the server, it **must** print received/sent message (whether things goes wrong or successful).

During Measurement phase, Client **must** print, for each probe:

- Probe sequence number;
- Its RTT;
- At the end of the phase, the evaluated mean RTT (in case requested measurement was "rtt"), the throughput (in case requested measurement was "thput").

RTT will have to be reported in *milliseconds* (ms), whereas the Throughput will have to be reported in *kilobit per seconds* (kbps).

What to submit (and how)

Whoever fails in following these simple rules will incur a penalty in the grading process.

Submit by your institutional e-mail a zip file of a directory containing the **whole** assignment. Mail sent from others e-mail will not be considered. The *object* of the e-mail *must* indicate *which* assignment you are submitting and *all* group members full name. Also, whoever of the group member submit the assignment <u>*must*</u> put in CC <u>*all*</u> the other member of the group.

The directory **must** contain:

1. The source code of the Client and the Server;
2. A report of the **whole** assignment in *.pdf* format(i.e., both Task1 and Task2); report has to be written in *neat* and *clear* Italian.

What to write in the Report (1/2)

Length is not always a synonymous of quality: keep your report relevant to the topic you are required to write about, and try to be clear and effective in your explanation.

For **Task1**, report **must** contain:

- The FSM of both Client and Server;
- A brief description of the behavior.

For **Task2**, report **must** contain:

- A description of the adopted resolution strategy (multiple resolution strategies exists to implement Client and Server behavior);
- Two graphs that, when generating these plots, server delay (<*server_delay*> field) should be set to the default value 0:
  - One for TCP mean RTT as a function of probe's payload size;
  - One for TCP Throughput as a function of message size.
- Few comments on obtained plots.

In order to plot those graphs, it is at your convenience to chose:

- The most appropriate way to keep mem of measured values (e.g., by logging);
- The plotting program (e.g., Gnuplot, Excel, or others).

What to write in the Report (1/3)

Note that parameters of the Hello message needs to be provided to Client in some way (once TCP connection is established): feel free to chose a configuration file approach or the standard input.

Optional. As an optional subtask, for Task 2, you can vary server delay parameter (<*server_delay*> field), and comment on how/why varying such parameter affects the shape of the plots.

If you chose to carry out this optional subtask, include two additional plots, for different server delay values, to support your comment/conclusions.