Project course 89-350 Introduction to Networking
Department of Computer Science Bar Ilan University
Lecturer: Hemi Leibowitz
TA: Erez Hartuv

Diplomatic cables from Homeland to Embassy

The task is to implement a reliable and efficient data (file) transfer between Homeland foreign office (H) and Embassy (E), which are connected (only) through Relay-Station (R). R serves as a router in the routing path between H and E.
Since the embassy is in a distant foreign country, messages that are been sent to it and from it, might have big propagation delay, due to using of satellite communication. Furthermore, some packets may be lost because of the small packets queue of the RelayStation, some other packets may be lost, altered or reordered by countermeasures of the foreign country.
The project was designed to allow quick implementation in parallel to the course; please begin working seriously as soon as possible to ensure success and avoid contention with other tasks.

1. Students will implement communication protocol between Homeland foreign office and Embassy via Relay-Station (see figure below). H and E may issue packets to R only, using IP (and protocols in the upper layers). Students are to use UDP on port 10000 or TCP on port 11000 to send and receive data to/from R.

   You are only required to implement H and E. R's functionality is provided to you as an executable file and it will not to be written or changed by you.

   Students need to implement the protocol for H and for E (in separate files). The protocol has to work between the student's pair of H and E only; that is, one student's protocol for H and E is **NOT** expected to be compatible with another student's H and E. The design of the protocol is an important part of the challenge of this project; students are **NOT** allowed to compare design details such as their packet format, length etc. Discussion about general design principles is allowed; we encourage you to do it using the course forum (in the Moodle).

   IP spoofing is prevented by the infrastructure rules, don't try changing source addresses.

2. The Relay-Station R (provided).

R does not initiate transactions, but rather responds to such by forwarding the received payload to the suitable destination, on either UDP (port 10000) or TCP (port 11000).
The Relay R implements Store and Forward (see [5]).
Note again, H, E must establish connections with R to start the route of the packets between H and E via R (for TCP is automatic, for UDP is by sending a packet to R).
After R establish a connection both with H and E, R will send to both sides “hello” message; after receiving these messages you may assume the connection between H and E via R is established.

R is provided and being run in the checking system.
R’s executable file is provided for students use, and is invoked by:

- user$ ./udp.out PortNumber LossRate QueueDelay PropagationDelay QueueSize AlterRate ReorderScheme

- user$ ./tcp.out PortNumber LossRate QueueDelay PropagationDelay QueueSize AlterRate ReorderScheme

    - PortNumber - Relay-Station listens to this port, e.g. 10000 for udp, 11000 for tcp.
    - LossRate - x% of the packets will be lost (H→R→E, H←R←E).
    - QueueDelay - Maximum queue delay of R (seconds).
    - PropagationDelay – Maximum propagation delay (seconds) on the channel R ←→ E.
    - QueueSize – Queue size of R (packets number)
    - AlterRate - y% of the packets will be altered
    - ReorderScheme - 0-none, 1-medium, 2-high

There are 32/64 bit versions of the R tcp/udp implementations.
You may need to run this command "chmod 777 *output_file*", before you can run the output file. Use the *sudo* prefix if needed for execution of the file.

Note: avoid choosing port number that other student chose and used at this moment. It could disrupt your scripts and lead to unexpected behavior.
Use the command: netstat -ln | grep ': port_number '
This guideline refers to running your project on BIU’s machines (u2, planet). It’s a good practice anyway, even on your private PC to avoid ports that other apps use.

3. Embassy E (to implement)

E retrieves the file sent by H, output it and terminate. E is not allowed to initiate any data transfer other than via R; all other communication will be blocked. Upon receive,

E should store the file as *output.txt* in the directory where it runs. The file E outputs needs to be **identical** to the original file (*input.txt*) sent by H.

E is invoked by:
user$ python embassy.py ip_r
Where “ip_r” is the IP address of R, e.g. 1.2.3.4 .

4. Homeland H (to implement)

H sends a file input.txt to E and terminate. H is not allowed to initiate any data transfer other than with R; all other communication will be blocked.

H is invoked by:
user$ python homeland.py ip_r

5. The order of execution is as follows:

1. The Relay-Station
2. The Embassy
3. Homeland

Make sure that the Embassy will connect to the Relay-Station **before** Homeland. In the SUBMIT (see section 9.v below) it will be done for you, but on your private test environments you must execute in that order.

6. Details

i. Data in the input file, will be in binary format. Students may assume the data is random (i.e. no special rule applies to the data). The transferred file’s size is limited to 100 KBytes.

ii. The MTU is restricted to 500 Bytes. This implies shorter payload length (depending on protocol, TCP vs. UDP).

iii. After submission, the code will be run with several test inputs and disruption schemes (loss rates etc.), which will be checked against the received outputs.

iv. Disruption (packet loss etc.) probabilities will remain fixed throughout a run, although it may differ between directions (H→R→E vs. H←R←E). The students must deal with these losses and errors; to avoid any doubt, the file received by E must match EXACTLY the file sent by H.

v. The project must be coded using Python 2.7.
Use of the following libraries (only) is allowed:

- *Sys*
- *Socket*
- *Time*
- *Struct*

no other libraries or published code are to be used.

7. Efficiency

The implementation should be efficient, that is, the file should be delivered to the destination with high efficiency (reasonable data overhead and transition time). Part of the project's grade will be determined according to the efficiency of the implementation, also in comparison to other students projects efficiency.

While we do not formally restrict computation time or use of memory, this is merely since the task is not computationally or memory intensive; implementation which are very wasteful will cost point reduction, or may fail to run on the test system (such failures, or any other `technical' failure, are not acceptable and would be considered project failure)

A time limit of 4 minutes per run is imposed, exceeding this limit is a fail in the run. **It is not acceptable to fail in runs**, fail in run will result a 0 points on this run's grade. Note that disruptions (packet loss etc.) are randomized, hence, results will vary from run to run. A good protocol will have good results in almost every run!

Students may use any protocol to implement a reliable data transfer, and we recommend you begin with a simple protocol such as Stop and Wait (see [9], [10]). However, correct, well-thought-of design is essential to achieve good or even acceptable efficiency.

8. Development Platform

From a HW perspective, there are several options in choosing the development platform. Keep in mind that your final submission must run on BIU's provided checking system, running Linux OS. This system will provide three machines: H, R, E. See the infrastructure scheme provided above.
Students may choose to develop the project using VM (recommended), Cloud-provided platforms (see [11], [12]) or actual physical infrastructure.
The project code must be developed and run by Python 2.7 (make sure the version/release are correct!).

9. Technical and Administration Requirements

i. The submission dates including bonuses and penalties for early / late submission of the project (both parts of the implementation - H and E) are in section number 10 below.

Note that the time given for the project already includes possible server downtime, overloads, etc. In particular, due to high server load nearing the submission date, we strongly encourage students to run and test their project in advance, in contrast to running it the night before the deadline on an overloaded server. Failure to take this into account is likely to result in failure to meet the deadline.

ii. Each file should begin with the name, surname and ID of the student that developed the project, as follows: /* Harry Potter 123456789 */

iii. Each student should also submit a documentation file README.pdf, describing the protocols that were implemented for file transfer. The document should also begin with the name, surname and ID of the submitting student. The document should be named README.pdf, and should be written in English.

iv. All the code pertaining to the project should be developed using Python 2.7.

v. Submission will be done through BIU's faculty SUBMIT system: https://submit.cs.biu.ac.il/cgi-bin/welcome.cgi

in two stages:

- Choose in the SUBMIT system-"project", in order to test that your project is running on the faculty's UNIX accounts. This is not the final submission for grading, just to help you check the validity of your project. When your project is running move to the second stage (final submission).

- Choose in the SUBMIT system "project final" in order to submit your project for grading (it will then be run with our full set of testing).

vi. Projects will not be tested on other operating systems, and will not be graded based on running on any other system.

vii. The project must be developed and submitted in singles.

viii. Any use of external libraries or other published code (except the libraries mentioned in section 6.iv above) is forbidden.

10. Grading

- The project constitutes 15% of the final grade.

- A bonus / penalty will be added for early / late submissions:
  - 12.1.2017 (10 points **bonus**).
  - 19.1.2017 (0 points bonus).

- 26.1.2017 (10 points **penalty**).

- The implementation must be documented decently. Undocumented and sloppy code may have a negative impact on the grade of the project.

- **Ethics**: copying code (or parts thereof) from other projects, or the Internet, is strictly forbidden, and will be checked. Detected copied projects will lead to failing the course and to disciplinary actions against both parties. Therefore, you are highly encouraged to maintain confidentiality, in order to prevent others from copying your code or design. Note that in addition, you may not use existing implementations, other projects or open source projects (certainly not after changing them!). In discussion about the project, you are only allowed to discuss design principles, not exact values, specific decisions, and certainly without sharing code.

11. Recommended Reading Material and Tools

[1] IPv4 description:
https://en.wikipedia.org/wiki/IPv4

[2] Python Programming Language:
http://www.python.org/

[3] Wireshark - Network Protocol Analyzer:
http://www.wireshark.org/

[4] Transport Layer
http://www.tcpipguide.com/free/t_TransportLayerLayer4.htm

[5] Store and Forward
http://en.wikipedia.org/wiki/Store_and_forward

[6] Repetition Code
http://en.wikipedia.org/wiki/Repetition_code

[7] Forward Error Correction Code
http://en.wikipedia.org/wiki/Error-correcting_code

[8] Cyclic redundancy check
http://en.wikipedia.org/wiki/Cyclic_redundancy_check

[9] Stop and Wait ARQ (Wiki)
http://en.wikipedia.org/wiki/Stop-and-wait_ARQ

[10] Stop and Wait (another source)

http://ecomputernotes.com/computernetworkingnotes/communication-networks/stop-and-wait-protocol

[11] Amazon Cloud; possible to get free time-limited platforms. Make sure you read the instructions regarding free usage limitations!
http://aws.amazon.com/ec2/

[12] Amazon EC2: Getting Started guide
http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html