Advanced Networks — Laboratory 11

Juliusz Chroboczek

13 May 2025

Exercice 1. The goal of this exercice is to write a program to determine your public IPv4 address by sending a Bind request to a STUN server and parsing the reply.

1. Write a program that sends a UDP packet with a STUN binding request to the server given on the command line. The message you send will consist of the header given in Section 6 of RFC 5389, with the fields having the following values:
   - *STUN Message Type* is 1 (*Binding Request*);
   - *Message Length* is 0 (empty body);
   - *Magic Cookie* is the string of bytes `0x21, 0x12, 0xa4, 0x42`;
   - *Transaction ID* is a random value that will be echoed in the reply.

   Test your program against both the STUN server at `stun.l.google.com` port 3478 and the one at `galene.org` port 1194 (the latter is less permissive). Use *tcpdump* or *WireShark* to verify that you receive a reply.
2. Modify your program to receive and parse the reply. You should first verify that the *Magic Cookie* and *Transaction ID* have the expected values, and that the *STUN Message Type* is 0x20 (*Binding Reply*). You should then walk the body, which consists of a sequence of TLVs as defined in Section 15 of the RFC. One of the TLVs should be of type `XOR-MAPPED-ADDRESS` (Section 15.2) and contain your public address.
3. Modify your program to deal with packet loss using an exponential backoff mechanism.

Exercice 2. We define the *Funny Quotation Protocol*, a UDP-based protocol where all messages have the following structure:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |    Length     |     Body...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

Fourd message types are defined:

- *NoOp* = 0, this message has no effect.
- *FunnyMessageRequest* = 1, requests a funny message. The receiver will send a *FunnyMessage* to the sender.

- *FunnyMessage* = 2: the body contains a funny message. encoded as UTF-8.
- *ThirdPartyRequest* = 3, requests that the receiver send a *NoOp* message to a third party. If the body is empty, then the *NoOp* should be sent to the sender of the request. If the body is 6 bytes long, then it contains an IPv4 socket address (IP and port) to which the *NoOp* should be sent. If the body is 18 bytes long, then it contains an IPv6 socket address.

1. Write a program that sends an HTTP `GET` request to `https://galene.org:8449/peers/` to find out the socket addresses of all funny quotation peers. For every returned peer, send a *FunnyQuotationRequest* up to five times, with exponential backoff. Did you receive a reply from every peer? Why?

The server accepts `POST` requests of the form `https://galene.org:8449/request?from=from&to=to` with an empty body. When the server receives such a request, it sends a *ThirdPartyRequest* message to the address given by the field `destination` with the address given in the field `requestor` in the body. (You may use the method `Request.URL.Query().add()` in order to generate the request.)

2. Why is the server able to contact the peer even though the originator is not?
3. Modify your program so that it requests a *NoOp* message from the peer while simultaneously sending a *FunnyQuotationRequest* (with exponential backoff). Are you able to reach the previously unreachable peers?

Exercice 3. The provided peer registers with the server using an undocumented request. Use tcpdump to reverse-engineer the undocumented part of the protocol, and modify your program to serve quotations to other peers.