Task 3: UDP-over-TCP tunnel
SKJ (2018)

Introduction

The task consists in writing a pair of tasks – an agent and a relay, implementing the functionality of a UDP-over-TCP tunnel that transmits data from the UDP protocol using the TCP protocol. The relay application is run on a remote computer somewhere in the network. The agent application is run on the user's computer. After starting the agent, it establishes the TCP connection with the specified relay, then configures it, and then opens UDP ports and listens on them. If any packet is read, it remembers the parameters of a client that sent it, and then transmits the data to the relay, which is supposed to send this data with the UDP protocol to the recipient's (according to its initial configuration) specific port. The relay, if it receives feedback data from the computer to which it transmits data, is obliged to transfer this data to the agent, which then sends the data to the client who initiated the connection to a particular port. In this way, we get the opportunity to connect using the UDP protocol in cases when (for some reason) a direct connection with UDP is impossible, but it is possible to communicate over TCP with a remote computer on which the relay process works.

Specification

The application consists of two processes: an agent and a relay.

A Relay

A relay process is run on a computer in a network, which is reachable over the TCP protocol from the user's machine. It requires one parameter at its startup, which is the TCP port number for communication with the agent (this number can be hardcoded in its code and in such case it is not necessary to give it as a parametr). Its further configuration is remote. This process is a 'delegate' of the user's computer used to effectively send UDP packets from the place from which they have a chance to reach the destination (eg in case when the local network does not allow sending UDP data to specific ports).

- After starting, the relay process opens a **specific** TCP port for listening. The port's number must be known to agents (if the author wishes, he can pass this number as a parameter, but the same parameter must be passed on to an agent later). The relay then waits for an agent connection on this TCP port.
- After establishing the connection, the relay is configured by an agent. The configuration consists of the computer address, **to which** the data should be forwarded and **from which** the answer should be received. If required by the protocol, the configuration may include other elements. Additionally, the relay opens a UDP port (with a random number), which is later used to forward data or receive responses.
- After configuration, the relay waits for messages from the agent. Such a message may contain:
    1. The command to disconnect and end work. After receiving it, the relay finished its work for a given agent, disconnects from it and goes back to the waiting mode (for connection with another agent).
    2. The data together with the UDP port number to which these data should be forwarded. Upon receipt, the relay generates a UDP message containing this data and sends it to the remote computer, according to its configuration (see above).
- Additionally, the relay waits for any data on the UDP port that it has opened. It should only expect messages from a computer that conformes to the relay configuration. After receiving them, it passes them along with the UDP port number from which these data have come to the agent (via the TCP channel).

An Agent

An agent process is run on the user's computer. It is a proxy that receives data from a user's process, and then it sends it to a remote relay process, which shall pass this data on to the real target.

- During execution, the process requires a number of parameters:
    1. An IP address or name of the computer on which the relay process works.
    2. A TCP port number on which the relay works (if this value is permanently set and embedded in the code, this parameter is not required).
    3. An IP address or computer's name to which the UDP data is to be forwarded by the relay.
    4. A list of at least one UDP port on which the agent listens for data from a process on this machine.

- After starting, the agent connects via TCP with the relay working on the given address and port. After establishing the connection, it provides it with configuration data, specifying the IP address or name of the remote computer to which the UDP data should be transferred (plus any other data required by the protocol).
- After the configuration of the relay is complete, the agent opens the UDP ports with the numbers specified by its parameters and starts to wait on them for the data. We assume that one UDP port will be used by only one process (hereinafter referred to as the client process). A client process for a given UDP port is determined, when the first message is received by this port.
- Each UDP packet received from the client process should be forwarded to the relay via the previously established TCP connection. Such a message must contain the port number on which the data was captured (this number is the same as the port number to which the relay must forward this data). Receiving UDP packets and forwarding them is an iterative activity, which should be repeated over and over again.
- At the same time, the agent listens on the established TCP channel to data from the relay. Each received message must contain the port number from which it was sent by the remote computer to the relay process (sender's port number). This message should be sent via a UDP port with the same number as the sender port number passed to the client process that communicates through this port.
- The user can enter the agent's termination command at any time. In this case, the agent first sends a message to the relay, terminating the connection, and then closes all its ports and ends its work.

Grading and requirements

In order to complete the task, one must design and implement your own communication protocol, allowing for the implementation of the functionality described above (communication between the agent and the relay).

1. Fully and correctly implemented project is worth **8 points**. One can also obtain **one additional point** for implementation of the additional functionality (as described below). Implementing each of the following functionalities is rewarded up with the specified number of points:
    - **2 points** for precise description of the designed communication protocol for data exchange beterrn an agent and a relay.
    - **2 points** for obtaining tunnel functionality for one UDP port operating according to a question↔answer scheme (that is, after receiving one UDP packet from the client process and forwarding it to the remote computer we expect the response that we deliver back to the client, then we repeat it).

- **2 points** for adding the functionality of an asynchronous tunnel with one forwarded port, which means the possibility of simultaneous transmissions of any number of packets in both directions.
- **2 points** for adding the functionality of simultaneous asynchronous packet forwarding from multiple ports through one tunnel (full functionality).
- For expanding the relay's ability to work simultaneously for multiple agents, one can get **one additional point**.

  It should be assumed that only one agent can communicate with a specific target host and its specific port at a time (this allows to avoid con icts with responses in the case, when that two agents want to send data to the same recipient). This assumption must be verified during configuration by the relay after connecting (eg by sending by the agent a list of port numbers of supported ports and comparing them with the list of ports already reserved on the relay).

2. The program should be written in Java, in compliance with Java 8 standard (JDK 1.8). Only basic UDP socket classes can be used for implementing the network functionality.

3. The projects should be saved in appropriate directories of EDUX system not later than on 27.01.2019 (the deadline can be moved by the teaching assistant).

4. The archived project file should contain:

   - The file `Documentation_(indexNo)_Task3.pdf`, describing what was implemented, what wasn't, what were the difficulties and what errors are still present.
   - Source files (JDK 1.8) (together with all the libraries used by the author that are not the part of Java standard library). The application should compile and run on PJA lab's computers.

   NOTICE: THE DOCUMENTATION FILE IS NECESSARY. PROJECTS WITHOUT DOCUMENTATION WILL NOT BE GRADED.

5. Solutions are evaluated with respect to the correctness and compliance with the specification. The quality of code and following software engineering rules can in uence the final grade.

6. UNLESS SPECIFIED OTHERWISE, ALL THE AMBIGUITIES THAT MIGHT ARISE SHOULD BE DISCUSSED WITH TEACHING ASSISTANT. INCORRECT INTERPRETATION MAY LEAD TO REDUCING THE GRADE OR REJECTING THE SOLUTION.