Operating Systems and Systems Programming
Semester 1, 2023
Eike Ritter

The University of Birmingham
17 October, 2023

Exercise 2

Deadline: 13 November, 11:00am

The Task

Write a set of programs which maintains the firewall configuation. You should provide two programs:

1. The first one is a server program. Once started, the server should run forever. It should listen on a port and wait for requests from clients, process them and send a specified response back to the client.

   The server maintains a collection of firewall rules. Clients may request operations on these rules. For each rule in the collection of rules, the server also maintains a list of pairs of IP addresses and ports which have been queried and found to match this rule. The server program should accept and process the following requests and return a response to the client making the request:

   - Adding a rule to the stored rules. This request expects a string which contains a firewall rule as specified in exercise 1. If the rule is valid, the rule is added the set of firewall rules. The server returns the string `Rule added` if the rule has been added, and `Invalid rule` if the string is not a valid rule.
   - Checking whether a given IP address and port are allowed according to the rules. If this is the case, the IP address and port are added to list of matched queries for this rule. If the given address and port are allowed according to several rules, you should add the IP address and port to only one rule. If the given IP address and port is checked again, they may be added to a different rule. If the input does not provide a valid IP address and a valid port, the server should return `Illegal IP address or port specified`. If the IP address and port are valid and should be accepted according to the rules, the server should return `Connection accepted`. If the IP address and port are valid and should be rejected according to the rules, the server should return `Connection rejected`.
   - Delete a valid rule from the stored rules. This should also delete the list of IP addresses and ports stored for this rule. The rule should only be deleted if the server stores exactly the same rule. In all other cases, no action should be taken except returning a suitable message. If the specification of the rule is invalid, the server should return `Rule invalid`. If the rule is successfully deleted, the server

should return `Rule deleted`. If the rule is valid but not found on the server, the server should return `Rule not found`.

- Return all rules and for each rule list all IP addresses and ports which are stored with it. More precisely, for each rule, the server should return the line

  ```
  Rule: <rule>
  ```

  where `<rule>` is the specification of the rule as given in exercise 1, followed by the line

  ```
  Query: <IPaddress> <port>
  ```

  for each query stored with this firewall rule.
- For any other request, the server should return `Illegal request`.

The server program should be started with

```
<ServerProgram> <port>
```

where `<ServerProgram>` is the way to call the server program, typically `./server`, and `<port>` is the port on which the server listens for incoming connections.

2. The second program is a client program. This program makes it possible send requests to a server program and display the results. The client should be called with

```
<ClientProgram> <serverHost> <serverPort> A <rule>
```

for adding a rule,

```
<ClientProgram> <serverHost> <serverPort> C <IPAddress> <port>
```

for checking an IP address and port,

```
<ClientProgram> <serverHost> <serverPort> D <rule>
```

for deleting a rule and

```
<ClientProgram> <serverHost> <serverPort> L
```

for showing the current firewall rules.

All other ways of calling the client program should return an error message.

`<ClientProgram>` is the way to call the client program, typically `./client`, `<serverHost>` is the hostname of the server, and `<serverPort>` is the port on which the server is listening.

The specification of a firewall rule is the same as for exercise 1. It is repeated here.

A rule is of the form

```
<IPAddresses> <ports>
```

where `<IPAddresses>` is either a single IP address, which has the form `xxx.xxx.xxx.xxx`, where `xxx` is a number between 0 and 255, or `<IPAddress1>-<IPAddress2>`, where `<IPAddress1>` and `<IPaddress2>` are IP addresses, and `<IPAddress1>` is smaller than `<IPaddress2>`. Assume that IP addresses and ports are separated by exactly one space character.

Similarly, `<ports>` is either a single port number, which is a number between 0 and 65535, or `<port1>-<port2>`, where `<port1>` and `<port2>` are ports and `<port1>` is smaller than `<port2>`.

Examples of such rules would be

```
147.188.193.0-147.188.194.255 21-22
147.188.192.41 443
```

Your server program should maximise the degree of concurrency and not contain any memory leaks.

Submission and Marking

- You should submit a zip-file on canvas containing a directory called `ex2`. This directory should contain all the sources. The marking scripts will execute the command `make` to produce all required binaries.
- Do not add files or directories to the source that start with `test`, since we use those files for the marking scripts.
- A basic test script is provided on canvas. We will use more advanced scripts for marking which test all the specified functionality, including concurrency and memory leaks. If the basic test script does not work, all other scripts are likely to fail as well.
- Code which does not compile on the virtual machine provided will be awarded 0 marks.

Hints

You may use the code provided in the lectures in your assignments as you see fit, in particular the solutions to the first exercise and the multi-threaded server.