Advanced Networks — Laboratory 2

Juliusz Chroboczek

5 March 2025

A chat server is running at `https://galene.org:8445`

Exercice 1. Post a chat message from the web interface.

Exercice 2. In what programming language is the client implemented? Can you examine its source code?

The server implements the following REST-like API:

- a `GET` request to the URL `/chat/` returns the list of identifiers of chat messages, one per line;
- a `POST` request to the URL `/chat/` creates a new message and returns its identifier ;
- a `GET` request to the URL `/chat/`*`id`* returns the body of a chat message; the date of the message is returned in the header `Last-Modified` ;
- a request `DELETE` to the same URL deletes the message.

Exercice 3. This API has no session state on the server side. What does that sentence mean?

If you connect to your own instance of the server rather than the one on `galene.org`, the client will likely complain about an invalid TLS certificate. In that case, you will need to add the option `-k` to every invocation of `curl`. You will need to add the following to your clients written in Go:

```
transport := &*http.DefaultTransport.(*http.Transport)
transport.TLSClientConfig = &tls.Config{InsecureSkipVerify: true}
client := &http.Client{
    Transport: transport,
    Timeout:   50 * time.Second,
}
```

Instead of `http.Get(...)`, you will use `client.Get(...)`, etc.

Exercice 4.

1. Use the command `curl -i` (or `curl -ki`, see above) to examine the list of chat messages.
2. Use `curl -d` to post a new message to the chat.

Exercice 5. Write a program in Go that displays the last 50 messages in the chat (or all the messages if there are fewer than 50). How many HTTP requests does your program make? How many round-trips? Would it be possible to send multiple requests in parallel? What is the minimum number of round-trips? What problem could we have?

Exercice 6.

1. Modify your program so that it displays new chat messages as they are posted. Since HTP does not provide server notifications, you will need to perform periodic `GET` requests, for example every 10 seconds.
2. Why would it be useful to use an adaptative timeout? Propose an algorithm to compute an adaptative timeout, describe it, and implement it.

Exercice 7. Modify your program so that it doesn't need to receive unmodified data by using an `If-None-Match` header on your request. Why is that header preferable to `If-Modified-Since`?

Exercice 8. Use the `flag` package to add options to your program to:

- display message identifiers together with the messages;
- post a new message, read from standard input;
- delete a given message.