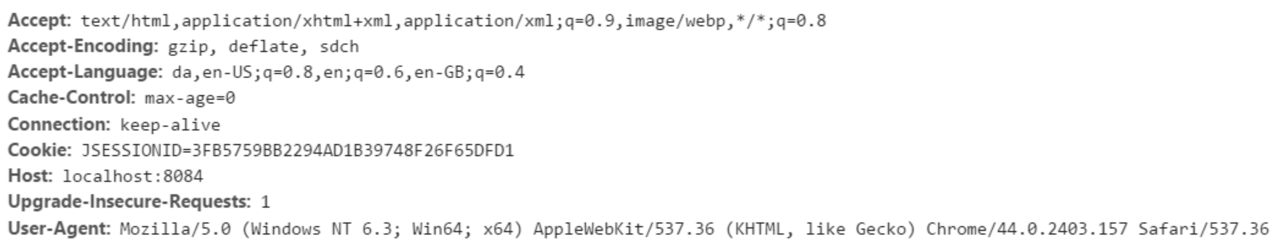
Designing an Application Layer Protocol (On top of TCP)

We know that, if a browser sends information like below, to a server:



```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch
Accept-Language: da,en-US;q=0.8,en;q=0.6,en-GB;q=0.4
Cache-Control: max-age=0
Connection: keep-alive
Cookie: JSESSIONID=3FB5759BB2294AD1B39748F26F65DFD1
Host: localhost:8084
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/44.0.2403.157 Safari/537.36
```

The Server can, along with many other things, detect:

- The mime types the browser will accept in the answer,
- The preferred language for the reply
- Lookup a session object that belongs to this request (state on top of a stateless protocol)
- Detect the browser etc.

This is possible, because both ends have agreed on a shared protocol (HTTP), so both know how to interpret the specific header values.

If you read the full HTTP specification you could build your own Browser, or your own Server that could interoperate with all existing Web Servers or Browsers.

If this is possible for something as complex as HTTP, it must be possible for us to define a simple chat protocol that (running on top of TCP) would allow us to:

- Allow clients to connect (we don't care about login, and assume that all users have a unique username)
- Can provide clients with real time info about all users online (constantly updated when new chatters connect or online chatters disconnect)
- Allow clients to send a message to a:
  - single user (Lars)
  - several users (Lars, Jens, Jan)
  - all users

Reflect about this

- How would you design a protocol that could implement this behaviour?
- What would be the first "protocol thing" sent between the two parts (by which end)?
- Which "protocol thing" could the server use to report about all current online users?
- Which "protocol thing" could a client use send a message in the ways described above?
- Which "protocol thing" could the server use to pass on a message to clients, including both the sender and message?
- Which "protocol thing" could a client use to indicate it would disconnect.

Think about a "protocol thing" as a way to send a text string with both a key (token), as for example Cache-Control (key) and max-age=0 (value) in the HTTP-example above, and a value.

Order will probably matter (not all protocol messages can be sent at all times). This will be a statefull protocol.

We will discuss this today, and design a protocol that all servers and clients will have to implement.