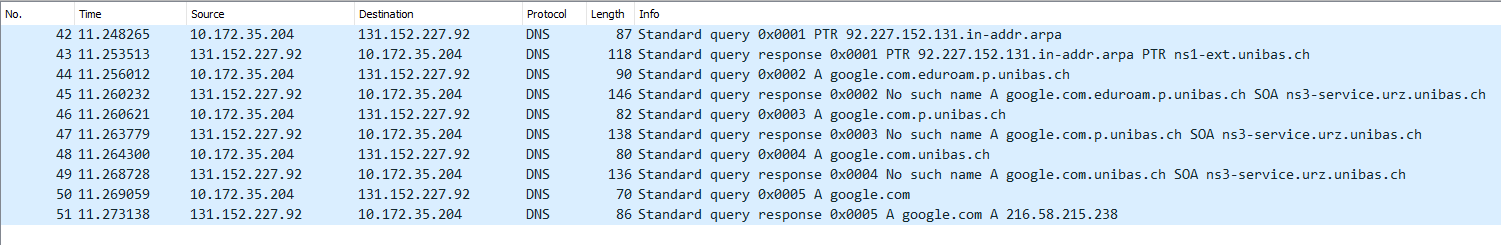
# Sheet 2

# 1) DNS



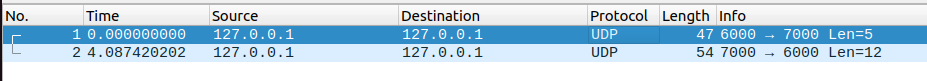
First, we do a reverse DNS lookup, which means that we want to get a domain name from an IP-address. Here we're asking for the name of the DNS server we're communicating with.

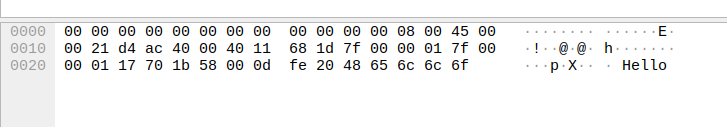
Next a query goes out to look up Google's IP. Since the server at google.com.eduoram.p.unibas.ch doesn't know the IP, we go one level up to google.com.p.unibas.ch. When this one doesn't know either we go further up and so forth. Finally, we reach the DNS server of google.com which can give us the IP we need.

## 2) Comparison TCP, UDP and UDP Multicast

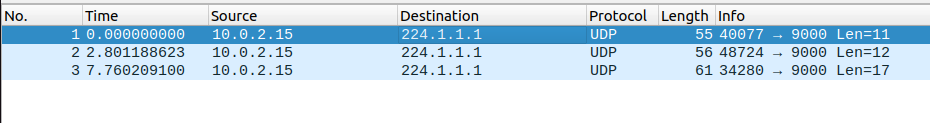
### 2a) UDP & TCP

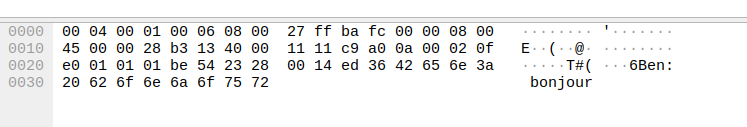
UDP:

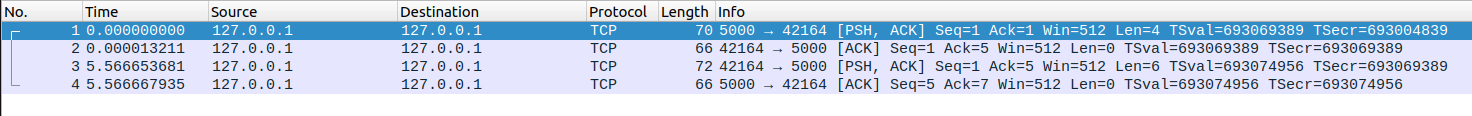


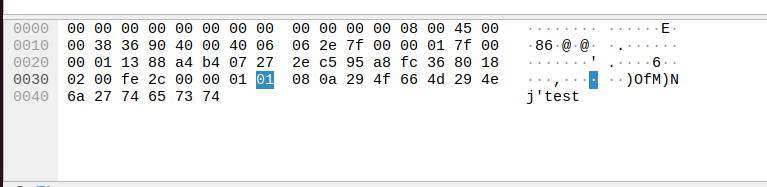


UDP Multicast:





TCP:



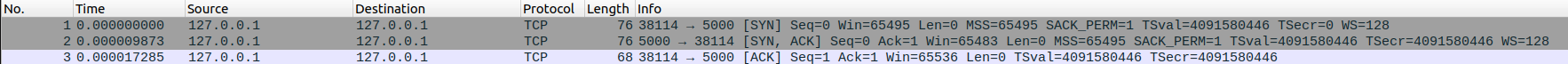
When two clients are talking the source and destination are both set to localhost, not matter whether

We use TCP or UDP. If we use UDP multicast our source is 10.0.2.15 which is a private IP used for local networks and our destination is the IP of the multicast room the client has entered (here: 224.1.1.1). The UDP messages don't get an answer to sent packages while the TCP communication always sends and answer with an ACK in it back. All three versions let us see the sent message inside the packets, nothing is encrypted.

### 2b) Advantages & Disadvantages

UDP has the advantage that there are less packets being sent, it "looks cleaner" and causes half the amount of traffick TCP causes. TCP meanwhile has the clear advantage in that there are less packets lost and we know when packets don't reach their destination due to the response the receiver sends back. Also UDP is not being encrypted, which is obviously worse than having encrypted messages using TCP.

### 2c) TCP Handshake



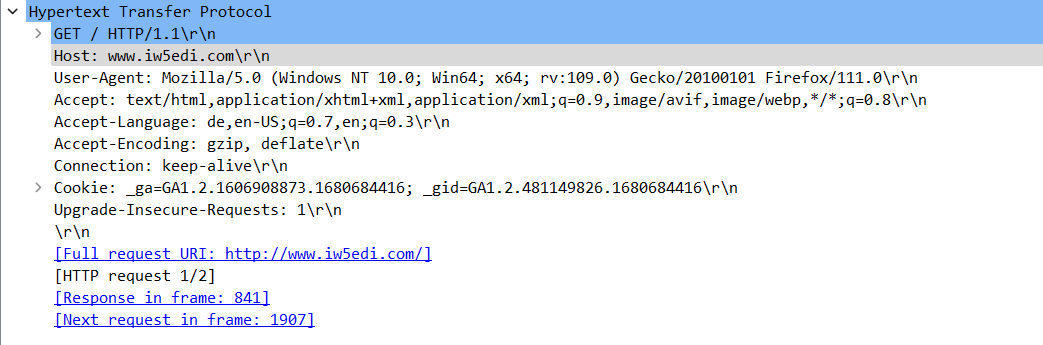
Here we have a client connecting to a server, both have the same IP here since it's run on localhost. First, the client sends the server a message having set the SYN flag on which means it wants to connect to the server. Next, the server answers and sets the SYN and ACK flag on, he agrees to connect. Lastly, the client sends a message including a set ACK flag as well to acknowledge that he understood. This three-way handshake causes way less problems than a two-way handshake, for this reason the last message with the ACK flag is sent by the client after the server has already agreed to connect.

## 3) HTTP vs. HTTPS

### 3a) HTTP



The http request for the website is easily trackable with wireshark. We just have to filter for the IP of the web-server and that we want HTTP packets. We can see the /GET requests for the html and the answers from the server.



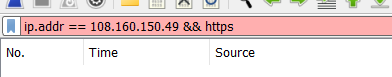
If we dive further and open the first sent package we can look under "Hypertext transfer Protocol" and can see things like the used browser (firefox), that we use windows, which languages we accept (german and english) and so forth.



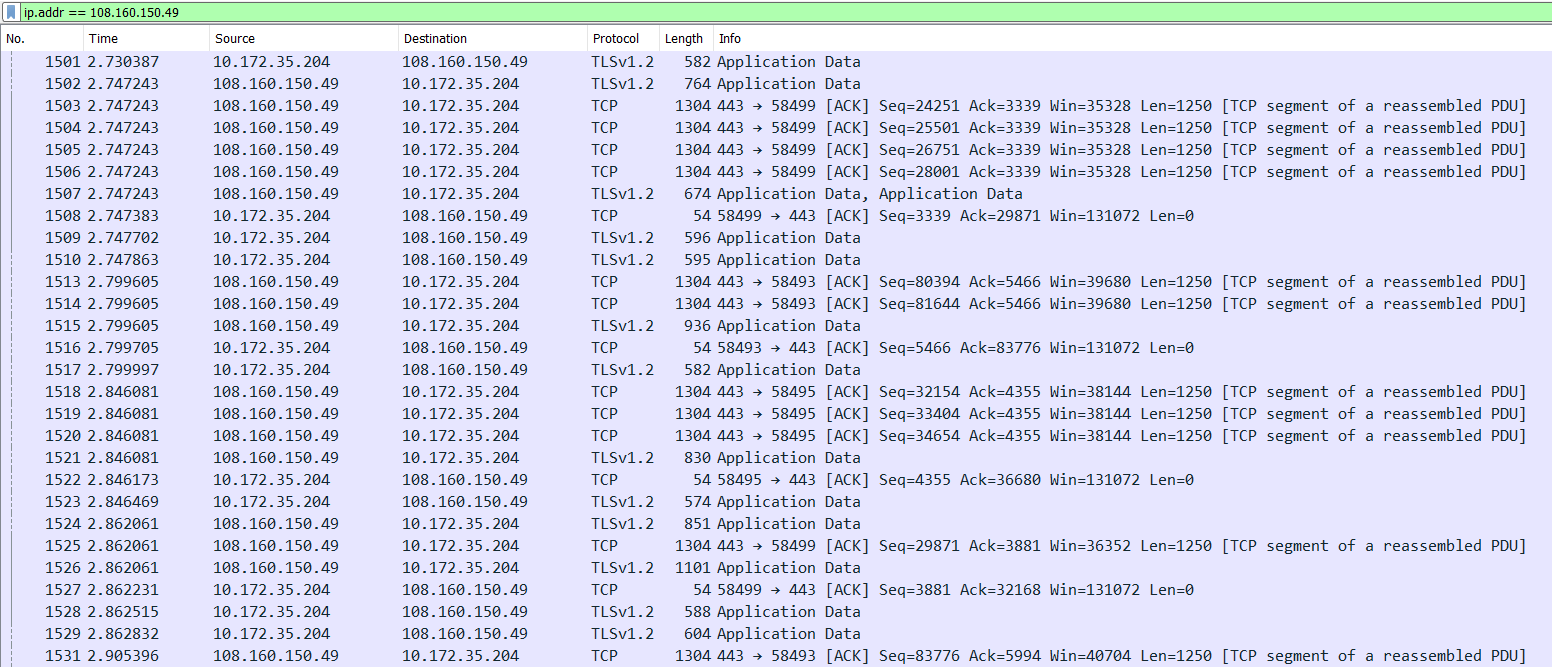
We searched "test" on the website and as we can see our search request shows up in the package info for everyone to see.

### 3b) HTTPS

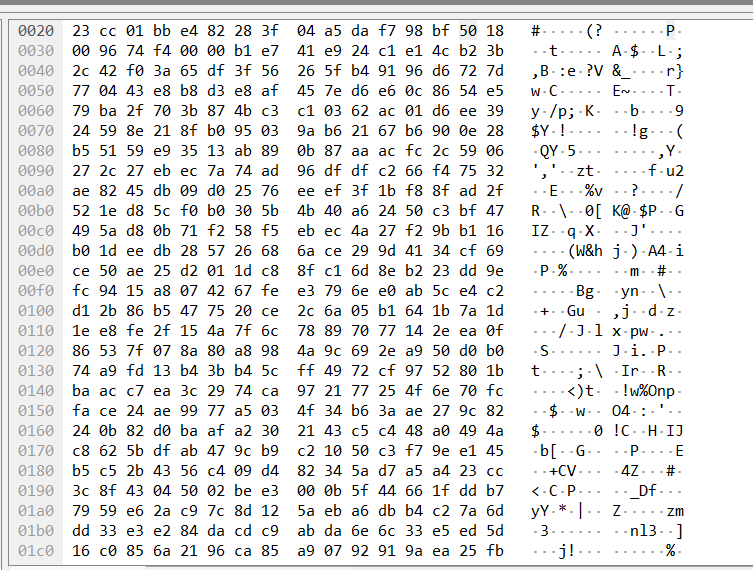
We can no longer just filter wireshark for the web-server IP and HTTPS since nothing will show up.



If we filter for the IP address we get something like this:



Our messages now get encrypted with TLS. While we can look at TLS packets we will only see encrypted data



The reasons to use HTTPS are pretty obvious from these findings. Not only can you see who is communicating with who but also things like what browser was used for requests and even what exactly was looked up on certain websites. Using HTTPS will solve those things by encrypting sent messages.