CS472

Homework 01

Amir Omidi# CS472

Homework 01

Amir Omidi

Questions:

1. We will take the exchange between a food truck and a student into consideration:
    1. The messages begin with a greeting message, this can take on many forms such as:
        - Hello
        - Hi
        - Hey there A "greeting" message is expected back from the other party. After that the messages can go two directions:
        1. Person placing order starts ordering the food.
        2. Food truck owner asks what they want, and then part one happens.

        Once that message sequence is complete, the food is prepared. Once that is complete the person buying the food asks how much they need to pay, an exchange of money happens and a "farewell" message is sent between the two parties.
    2. A food truck can have a list of food where you're expected to order from that list.
    3. Natural language has indicators where the sentence has completed. It is also routine so that the two parties are expecting certain cues to find out the other party has finished talking.
    4. As explained in number 1, there are different "states". Greetings, Order placement, Order preparation, Bill/Money, Farewell.
    5. The assumption is that the parties have done this before, and are speaking in a language that the both parties agree to (eg English).
    6. For ordering food, there are no "security" phases. It wouldn't make sense for this conversation to have a security phase.
    7. In natural languages, looking at it from a "Standard" perspective would not be helpful.
    8. In person and over the phone are quite similar, except over the phone the ordering state is extended where it asks for your address. The billing phase is usually done in person. Over TCP/IP this is different, the "messages" are just bits of information in a Data Exchange Format (eg json). Over TCP/IP this would most likely be encrypted under SSL/TLS because of billing information.
    9. Final points: When talking about natural languages, we're not as prone to error as with connections over internet. However when an erroneous state happens either party can respond with a "error catch phrase" (eg "Sorry I didn't catch that") to rollback the conversation.
2. Lets think about Android's update service. To make things easier, we're going to forget about DNS queries and assume we have an active TCP connection to Android's update server.
    1. The client starts with sending a hello and doing a TLS key exchange. Encryption is used in this purpose to verify the integrity of this update. After the handshake, the client sends its version information and some other information (eg Rooted or not, system partition modified or not and etc). The server takes this information, checks to see if there are any updates for this specific version and responds to the client with that update file. There could be an erroneous

state here where the server can not find any updates for that specific combination of information. To make this part easier, we will assume that the update isn't sent in chunks and rather a stream of packets for one file. The update starts the stream and sends it along with the hash of the entire update package. Once the download is complete from client side the client checks the hash locally and if it doesn't match what the server said it will request the file from the server again, this is how it catches errors in the transport layer.

2. There are standard questions and answers. For example when asking for the update package, the server has a predefined set of answers it can return and the client uses that set of response to show a user-friendly version of it to the user.
3. Since this is over the internet, a special null packet can be sent, or a specific message can be sent. This is all up to what the client and server agreed to beforehand.
4. There are definitely different states. Handshake, version information exchange, download exchange, and a goodbye to close connection.
5. The assumptions are that both the client and the server know the flow of the conversation. Another assumption is that both the client and the server are able to connect to eachother.
6. There is definitely security in this protocol in the form of TLS.
7. There definitely is a standard for this protocol due to the sheer number of android devices used globally. They should all be able to connect to the updating server (Such as play store) and be able to work.
8. This wouldn't really be possible in person since we're talking about a protocol that was made to be over the internet. However if we consider a real-life software vendor, we could ask them for a new disk for a specific software and get it from them.

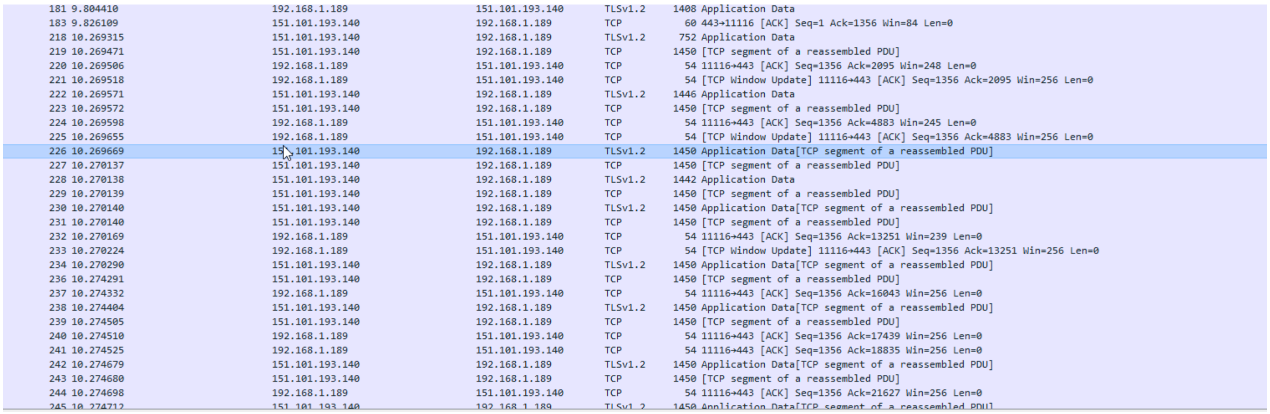
1. Non Encrypted: I have applied the filter `ip.addr==94.182.146.195` which is what was resolved for www.asriran.com.
   - We can see that the `HTTP GET / HTTP/1.1` was requested, so we can see the exact connection in plain text:

51	2.894198	192.168.1.189	94.182.146.195	TCP	54	11177→80 [FIN, ACK] Seq=1 Ack=1 Win=251 Len=0
52	2.894271	192.168.1.189	94.182.146.195	TCP	66	11265→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK
66	2.896919	192.168.1.189	94.182.146.195	TCP	66	11279→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK
67	2.897006	192.168.1.189	94.182.146.195	TCP	66	11280→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK
68	2.897089	192.168.1.189	94.182.146.195	TCP	66	11281→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK
69	2.897202	192.168.1.189	94.182.146.195	TCP	66	11282→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK
70	2.897283	192.168.1.189	94.182.146.195	TCP	66	11283→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK
91	3.098114	94.182.146.195	192.168.1.189	TCP	66	80→11279 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460
95	3.098225	192.168.1.189	94.182.146.195	TCP	54	11279→80 [ACK] Seq=1 Ack=1 Win=65536 Len=0
99	3.098548	192.168.1.189	94.182.146.195	HTTP	843	GET / HTTP/1.1
108	3.103406	94.182.146.195	192.168.1.189	TCP	66	80→11265 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460
110	3.103465	192.168.1.189	94.182.146.195	TCP	54	11265→80 [ACK] Seq=1 Ack=1 Win=65536 Len=0
111	3.103525	94.182.146.195	192.168.1.189	TCP	66	80→11280 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460
112	3.103551	192.168.1.189	94.182.146.195	TCP	54	11280→80 [ACK] Seq=1 Ack=1 Win=65536 Len=0
113	3.105418	94.182.146.195	192.168.1.189	TCP	60	80→11180 [ACK] Seq=1 Ack=2 Win=249 Len=0

   - It was responded to about 1.1 seconds later (The server is in Iran so this type of latency is to be expected):

228	4.268029	94.182.146.195	192.168.1.189	TCP	1514	[TCP segment of a reassembled PDU]
229	4.268101	192.168.1.189	94.182.146.195	TCP	54	11279→80 [ACK] Seq=790 Ack=65701 Win=65536 Len=0
230	4.268126	94.182.146.195	192.168.1.189	TCP	1514	[TCP segment of a reassembled PDU]
231	4.268144	192.168.1.189	94.182.146.195	TCP	54	11279→80 [ACK] Seq=790 Ack=67161 Win=65536 Len=0
233	4.272323	94.182.146.195	192.168.1.189	TCP	1514	[TCP segment of a reassembled PDU]
234	4.272324	94.182.146.195	192.168.1.189	HTTP	1303	HTTP/1.1 200 OK (text/html)
235	4.272380	192.168.1.189	94.182.146.195	TCP	54	11279→80 [ACK] Seq=790 Ack=69870 Win=65536 Len=0
1048	33.313289	94.182.146.195	192.168.1.189	HTTP	266	HTTP/1.0 408 Request Time-out (text/html)
1049	33.313393	192.168.1.189	94.182.146.195	TCP	54	11280→80 [ACK] Seq=1 Ack=214 Win=65280 Len=0
1050	33.318191	94.182.146.195	192.168.1.189	HTTP	266	HTTP/1.0 408 Request Time-out (text/html)
1051	33.318226	192.168.1.189	94.182.146.195	TCP	54	11283→80 [ACK] Seq=1 Ack=214 Win=65280 Len=0

2. Encrypted: I have applied the filter `ip.addr==151.101.193.140` which is what was resolved for www.reddit.com.
   - As you can see, there are no indications of what type of request is being made due to the TLS encryption. This way anyone listening in on the connection (eg ISP, "bad people") would only know you're connected to that IP, not what you're doing on that IP.



181	9.804410	192.168.1.189	151.101.193.140	TLSv1.2	1408 Application Data
183	9.826109	151.101.193.140	192.168.1.189	TCP	60 443→11116 [ACK] Seq=1 Ack=1356 Win=84 Len=0
218	10.269315	151.101.193.140	192.168.1.189	TLSv1.2	752 Application Data
219	10.269471	151.101.193.140	192.168.1.189	TCP	1450 [TCP segment of a reassembled PDU]
220	10.269506	192.168.1.189	151.101.193.140	TCP	54 11116→443 [ACK] Seq=1356 Ack=2095 Win=248 Len=0
221	10.269518	192.168.1.189	151.101.193.140	TCP	54 [TCP Window Update] 11116→443 [ACK] Seq=1356 Ack=2095 Win=256 Len=0
222	10.269571	151.101.193.140	192.168.1.189	TLSv1.2	1446 Application Data
223	10.269572	151.101.193.140	192.168.1.189	TCP	1450 [TCP segment of a reassembled PDU]
224	10.269598	192.168.1.189	151.101.193.140	TCP	54 11116→443 [ACK] Seq=1356 Ack=4883 Win=245 Len=0
225	10.269655	192.168.1.189	151.101.193.140	TCP	54 [TCP Window Update] 11116→443 [ACK] Seq=1356 Ack=4883 Win=256 Len=0
226	10.269669	15[illegible]101.193.140	192.168.1.189	TLSv1.2	1450 Application Data[TCP segment of a reassembled PDU]
227	10.270137	151.101.193.140	192.168.1.189	TCP	1450 [TCP segment of a reassembled PDU]
228	10.270138	151.101.193.140	192.168.1.189	TLSv1.2	1442 Application Data
229	10.270139	151.101.193.140	192.168.1.189	TCP	1450 [TCP segment of a reassembled PDU]
230	10.270140	151.101.193.140	192.168.1.189	TLSv1.2	1450 Application Data[TCP segment of a reassembled PDU]
231	10.270140	151.101.193.140	192.168.1.189	TCP	1450 [TCP segment of a reassembled PDU]
232	10.270169	192.168.1.189	151.101.193.140	TCP	54 11116→443 [ACK] Seq=1356 Ack=13251 Win=239 Len=0
233	10.270224	192.168.1.189	151.101.193.140	TCP	54 [TCP Window Update] 11116→443 [ACK] Seq=1356 Ack=13251 Win=256 Len=0
234	10.270290	151.101.193.140	192.168.1.189	TLSv1.2	1450 Application Data[TCP segment of a reassembled PDU]
236	10.274291	151.101.193.140	192.168.1.189	TCP	1450 [TCP segment of a reassembled PDU]
237	10.274332	192.168.1.189	151.101.193.140	TCP	54 11116→443 [ACK] Seq=1356 Ack=16043 Win=256 Len=0
238	10.274404	151.101.193.140	192.168.1.189	TLSv1.2	1450 Application Data[TCP segment of a reassembled PDU]
239	10.274505	151.101.193.140	192.168.1.189	TCP	1450 [TCP segment of a reassembled PDU]
240	10.274510	192.168.1.189	151.101.193.140	TCP	54 11116→443 [ACK] Seq=1356 Ack=17439 Win=256 Len=0
241	10.274525	192.168.1.189	151.101.193.140	TCP	54 11116→443 [ACK] Seq=1356 Ack=18835 Win=256 Len=0
242	10.274679	151.101.193.140	192.168.1.189	TLSv1.2	1450 Application Data[TCP segment of a reassembled PDU]
243	10.274680	151.101.193.140	192.168.1.189	TCP	1450 [TCP segment of a reassembled PDU]
244	10.274698	192.168.1.189	151.101.193.140	TCP	54 11116→443 [ACK] Seq=1356 Ack=21627 Win=256 Len=0
245	10.274712	151.101.193.140	192.168.1.189	TLSv1.2	1450 Application Data[TCP segment of a reassembled PDU]

3. RFC's stand for Request For Comments. It is a method of creating standards created by the IETF. Writing an RFC requires following standards set forth by other RFCs (such as RFC2119). When an RFC is written, it is assigned a "status". This status can range from "Proposed Standard", to "Experimental" and even have "Historic". These RFCs can be put forth by different people and companies until they agree on a final format and that gets published and used as a standard. Sometimes some comments put forth an RFC and start implementing their own version without respecting subsequent RFCs making the protocol/standard better. If I were to implement a standard, I would have to have an introduction explaining what this protocol solves that other protocols so far haven't been able to. I'd have to explain or reference specific protocols (Such as TCP, UDP). Explain the sequence of messages and set of expected responses and explain how to handle errors.
4. There are lots of other protocols other than TCP and UDP. A one I'm very interested in is Interplanetary Internet (Covered by RFC 4838). This RFC presents a standard for an architecture where latency is always assumed to be very high. It explains quality of service for service for this type of protocol. Since this is a RFC for the transport layer, security doesn't need to be built into this and it could be added onto it seamlessly with different standards. However the RFC argues about security where nodes should drop unauthorized traffic from other nodes. They also explain security in terms of Confidentiality, Authentication and Error detection which is normal of every transport layer protocol.
5. I am interested in network surrounding the handling of big data. I am also interested in understanding scaling from a networking perspective.