Computer Networks 2 and Introduction to Cybersecurity

Marco Sgobino

March 2, 2022

Contents

Part I

Low-Level Network Protocols

Chapter 1

TCP

1.1 Brief recap of TCP

TCP is a procotol that has the following properties,

- allows *connection between processes*;
- is *connection-oriented*: before transmitting data, a connection must be established;
- is *reliable*: it assures all segments are correctly delivered through use of `ACK` mechanism, and **at most once**;
- offers a *sliding window* mechanism for congestion control and stream control. This assures read and send buffers are well-optimized in both sender and receiver;
- is *byte-oriented*: the byte stream is fragmented into multiple segments, and composed again after getting to destination.

The logical structure is the following one. There are client and server. The client first authenticates to the server, after that the server opens the connection and client executes send-receive loop. Both server and client create a *socket s*, and the client connect *s* to IP-srv, port-srv. The communication takes place on *s* by means of application protocol. It is *reliable*: no losses, no packet loss, packet arrive in the same order as they are sent.

A very simplified pseudo-code for TCP is as following,

```
 int s; s := socket(...);
connect(s, IP-srv, port-srv,...);
...
send (s, msgl, ...);
...
msg2 := receive(s,...);
...
```

The logical structure at server side is quite different. A server creates socket $s1$, chooses a port number to bind to that socket, then it declares willingness to accept connections on $s1$, and finally it awaits for connection requests on $s1$. Server remains on *sleep* until a connection is requested.

```
s1 := socket(...);
bind(s1, portsrvm ...);
listen(s1,...);
s2 := accept(s1,...); // another socket
...
msg1 := receive(s2,...);
...
send(s2,msg2,...);
...
```

In TCP, communication is *bidirectional*, with a pattern that depends on the application protocol.

The send-receive patterns depend on the application itself – browser send-receive sequences are very different from, let's say, an e-mail client send-receive sequence.

1.1.1 TCP Implementation

IP operates between *nodes*. It is *connectionless*, **unreliable**, and is *message-oriented*. The Maximum Transmission Unit size of an IP packet is $MTU = 64KB$. TCP lies on top of IP: to overcome the unreliable aspect of IP, countermeasures should be adopted.

TCP layers communicate between themselves in terms of *segments*. A segment is a *message between TCP layers*, and contains a *TCP header* and – eventually – data *payload*. Payload can either be 0 byte or carry some information useful for application layers. An important property is that *it must be small enough to fit in a single IP packet*, hence IP header + TCP segment size should be no greater than $64KB$.

A segment is thus composed by a IP header, whose payload is a TCP segment. The TCP segment is composed by a TCP header, followed by eventual application data. Usually, IP header size is usually 20 bytes, as well as TCP header that is 20 bytes. The IP datagram can be greater up to $64KB$, with the first $40, 50$ bytes reserved to headers.

Segments can carry portions of data (for instance, in a video stream many segments should be sent to client in order to carry enough information and let application layer reconstruct the video correctly).

In application layer, one application message could correspond to *many segments* in TCP layer, in **both** directions. In fact, at TCP level multiple segments are usually required in order to send a single application-level message.

Each TCP layer represents a connection as (<id>, <state>). The <id> is the <IP-local, port-local, IP-remote, port-remote>, while the <state> refers to the state of the TCP connection. Conceptually there is a single table storing both <id> along with connection <state>.

IP addresses are extracted from the IP header, while port numbers are extracted from TCP header. Packets are thus sorted accordingly. The connection <state> includes information on the *Maximum Segment Size* (MSS), which is the maximum size of the *data part* of a segment that the other part is willing to achieve. The MSS is negotiated upon connection opening. This value is, in practice, identical in both direction and is not arbitrary. In most cases, there are only 2 possible values for historical reasons:

- on different networks (through internet), MSS is 536 bytes (MTU=576), that is the maximum segment size that can fit in the smallest possible packet;
- on same network (ethernet), MSS is 1460 bytes (MTU=1500), which corresponds to ethernet MTU minus the IP header and TCP header.

The core idea is that each segment must be sufficiently small to fit in one packet along the full path, in order to prevent fragmentation.

TODO Add figure that recaps IP header + TCP header.

Part II

Network Tools

Chapter 2

Wireshark

Wireshark is a free and open-source tool for *recording* and *analyzing* network messages. Recording takes place in a local interface, while analysis can be done anywhere by loading saved sessions in a file. Wireshark is useful for analysis, troubleshooting and understanding of a network's behavior.

2.1 Network analysis with pre-captured session