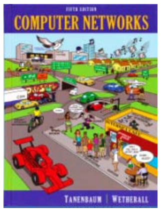
Solutions – Protocol Layers



The solutions below are based on our capture and use of tools. Your answers will differ in the details if they are based on your own capture and use of tools in a different network setting. Nonetheless, we expect our solutions to help you understand whether your answers are correct.

Step 3: Packet Structure

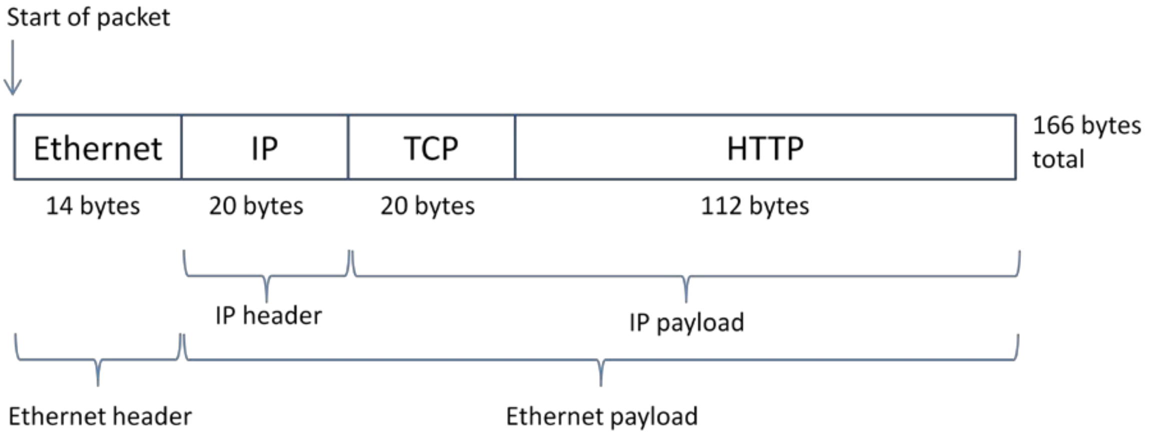


Figure 1: Protocol layer structure of the HTTP GET packet

There are several features to note:

- The order of the headers (Ethernet, IP, TCP, HTTP) is the protocol stack from the bottom upwards because the lower layers are outermost in the packet as it travels through the network.
- Observant students will note some differences between the Ethernet header size in Wireshark and in the text that will be explored in later labs.
- The size of the IP and TCP headers is normally around 20 bytes each, but it may be larger in some cases depending on the OS, e.g., IPv6 instead of IPv4 and optional TCP header fields might double these numbers.
- The size of the HTTP message will vary depending on what tool and URL is used to send the web request. For wget/curl, it is likely to be around 100-300 bytes.
- The Ethernet payload comprises everything beyond the Ethernet header. That is, Ethernet does not understand the IP / TCP / HTTP internal structure; it is up to higher layers to determine their headers and message boundaries.
- Similarly, the IP payload comprises everything beyond the IP header. Note that neither the IP header nor payload covers the Ethernet header.

Step 4: Protocol Overhead

For our trace, the large download packets are each 1484 bytes. Inspecting the TCP details in one of them shows it to carry 1430 bytes of higher-layer data. Alternatively, the Ethernet, IP, and TCP headers can be seen to total 54 bytes. This gives an overhead of 54/1484 or 3.6%. If you do the same for your trace you will get a slightly different number, but it is likely to be in the ballpark of 5%.

Now, the question asked for an estimate of the overhead over all downloaded packets, not just one of them. We would expect this to be somewhat higher, but not greatly. For our trace, we have:

HTTP data = 14937 bytes (seen in the HTTP packet)

Total (HTTP data + headers) = 66+60+ 1484*7+1282+1290*2+1281+60 = 15717 bytes

Overhead = (15717 – 14937) / 15717 = 5.0%

The overhead will be somewhat different for your trace, but should be comparable. It will be higher if the content that is downloaded is smaller or if the largest packet size is smaller.

The conclusion from this exercise is that the overhead of protocol layers on network transfers is not usually significant. At 5%, if it could be removed entirely (which it can't because it is there for good reasons) then we would speed up the network by just 1/0.95 or around 5%. We won't get a significant throughput gain by eliminating protocol headers.

Step 5: Demultiplexing

Answers to the questions:

1. The demultiplexing key for Ethernet is the Type field. It holds 0x800 when the higher layer is IP.
2. The demultiplexing key for IP is the Protocol field. It has value 6 when the higher layer is TCP.

Explore on your own

Answers to the questions:

- The packet is destined to the TCP protocol entity of the receiving computer, not to the application that implements HTTP. It is a control packet exchanged between TCP peers to manage their connection state.
- Only the first drawing will have an HTTP header inside the packet (unless the HTTP header is extremely long). The second drawing will have HTTP payload data but no HTTP header.
- With encryption, the lower protocol layer will rewrite the payload in its body as well as append its header. In this case, the internal structure of the higher protocols will no longer be visible while the packet is on the network; it will be revealed when the receiving peer protocol decrypts the message.
- With compression, the lower protocol layer will rewrite the payload in its body as well as append its header. Since it is compression, the rewritten payload will (normally) be shorter than the original payload. As with the case of encryption, the internal structure of the higher protocols will no longer be visible while the packet is on the network; it will be revealed when the receiving peer protocol decrypts the message.

[END]