Chapter 3

Application Layer

If you ever write internet code, almost surely it will be application layer code.

3.1 HTTP

Today a large majority of all data is transferred using the HTTP protocol, with a share of over 80%. HTTP is used by websites such as Facebook, Wikipedia, or Google, but also by popular content streaming services such as Netflix or Youtube.

Definition 3.1 (URL)**.** *A Uniform Resource Locator (URL) is a string that uniquely identifies a resource on a server. In its most common form it consists of a* ***scheme*** *detailing the protocol to use, the* ***host identifier*** *on which the resource is located and a* ***path*** *detailing the location on the server.*

Remarks:

- The scheme in this section will either be *http* for unencrypted HTTP or *https* for HTTP over an encrypted connection. If the scheme is http, then by default the client will contact the server on port 80. For https the client uses port 443.
- HTTPS and HTTP only differ in the underlying transport protocol. While HTTP directly communicates over a TCP connection, HTTPS further encrypts its traffic by inserting an encryption layer between HTTP and the TCP connection. Please refer to Section 3.6 for more details on layers and Chapter 12 for the encryption. for details about encryption.
- The host identifier may either be the server's IP address or a domain name, which would then be translated to the IP address using a DNS lookup. Refer to Section 3.3 for details about DNS.
- There are a number of optional fields that may also be part of a URL. These are *username* and *password* for simple authentication, a *port* if a non-default port is used, a *query*-string to pass arguments to a resource, and a *fragment* describing which part of the resource is of interest: `scheme://user:password@host:port/path?query#fragment`.

Protocol 3.2 (HTTP)**.** *The Hypertext Transport Protocol (HTTP) is a network protocol built on top of TCP, used by a client to interact (retrieve, store, etc.) with resources on a server.*

Remarks:

- HTTP has simple request-response semantics, i.e., the client issues a request and the server responds. The requests and responses are in a human readable format.
- HTTP is a client-server protocol, in which the client requests an operation, called *method*, on the resource and the server returns a response. The most common method for HTTP requests is `GET` which is used to retrieve a resource from the server. Other methods include `HEAD`, used to retrieve metadata about a resource from the server, `POST` and `PUT`, commonly used to update an existing resource or create a new resource on the server, and `DELETE`, used to remove a resource from the server.
- Both request and response comprise a header and a payload separated by an empty line.
- The request header contains a request line, consisting of a method, a URL specifying which resource it should be applied to, and an HTTP version. In addition the header may include a number of request options, i.e., `Key: Value` pairs with additional information about the client and the request, each on a new line. The request header is terminated by two successive newline characters: `\n\n`.
- The `PUT` and `POST` methods include a *payload*, i.e., some data that is transferred along with the request. Semantically this corresponds to the content of the resource that the client is attempting to create or update. If a payload is included in the request, a `Content-Length` header option specifies the length of the payload in bytes, and the payload is appended after the two newline characters.
- The server response header consists of a response line containing a protocol version, a numeric return code and a human readable response code. Additionally it may include response options, in the same format as the request options. Options usually include the resource `Content-Type` and `Content-Length`. The response header is terminated by two successive newline characters and is followed by the payload, i.e., content, of the resource.
- The numeric return codes returned by the server are 3-digit numbers organized into groups according to their first digit:
  - *1xx: Informational* Not used, but reserved for future use
  - *2xx: Success* The action was successfully received, understood, and accepted, e.g., 200, indicating a successful request returning the requested resource.

  - *3xx: Redirection* Further action must be taken in order to complete the request
  - *4xx: Client Error* The request contains bad syntax or cannot be fulfilled, e.g., 404, indicating that a non-existing resource was requested.
  - *5xx: Server Error* The server failed to fulfill an apparently valid request

- A resource may be any content that can be serialized and returned in response to a request. This includes textual as well as media contents.

- Most resources on the internet are HTML pages and associated media resources, such as images and videos. A *browser* on the user's computer is used to render the HTML pages and associated media into a webpage and display it to the user. However, the pervasive deployment of HTTP for user content has resulted in HTTP becoming the primary transport mechanism for other resources as well. Standards such as Representational State Transfer (REST), XML-RPC, Webservices (SOAP), and JSON-RPC are nowadays used to enable application to application communication on top of HTTP.

- The server may respond to a request with a continuous stream of data. Conceptually streaming is no different from a download, the only difference is that the server does not announce the size of the returned response in the response options and it does not close the connection.

- HTTP is a stateless protocol.

Definition 3.3 (Stateless Protocol)**.** *A protocol is stateless if each request is treated independently. In other words, the communication consists of independent request-response pairs. A stateless protocol relieves the server from storing information about its clients.*

Remarks:

- Stateless protocols are simpler but limited in applicability. For example, authenticating a user is tricky in a stateless protocol.

- To allow stateful applications on top of HTTP, a `Cookies`-option has been introduced by browsers. This option can be used to send small amounts of data (session identifier, authentication token, ...) from the client to the server with every request, allowing the server to recognize the user, storing state about the user.

Protocol 3.4 (HTTP 1.0)**.** *HTTP 1.0 is the first version of HTTP.*

```
GET /index.html HTTP/1.0
User-Agent: Mozilla/5.0
Accept: text/html
```

```
HTTP/1.0 200 OK
Content-Type: text/html; charset=UTF-8
Content-Length: 312

<html>...
```

Figure 3.5: Simple HTTP 1.0 interaction retrieving the *index.html* resource on the server. The client request on the left is terminated by an empty line. On the right side, the response header and content are sent back by the server.

Remarks:

- HTTP 1.0 was only intended to transfer hypertext and some embedded resources such as images.
- In HTTP 1.0 a client opens a new connection for each request, which is closed once the response is delivered. Due to the increasing number of resources required to render a webpage, this results in a large number of (slow start) TCP connections between client and server.

Protocol 3.6 (HTTP 1.1)**.** *HTTP 1.1 is the second released version of HTTP, which includes a number of incremental changes such as request pipelining, protocol upgrade capabilities, range request, better support for caching and compression.*

Remarks:

- Pipelining uses persistent connections that are reused for multiple requests before being closed. The client opens a connection once and issues multiple requests on that connection. A browser may still open multiple concurrent connections, but the overall number of connections is reduced, and the overhead of establishing the connection and the TCP slow start is amortized over multiple requests.
- If supported by the server, range requests allow the client to specify a byte-range that is to be returned in the request options. Without range requests it is not possible to resume an interrupted download.
- Protocol updates are used to switch to completely different protocols over the same connection. The server responds to the current request and then the protocol is switched to the protocol indicated in the request options.
- One example of a protocol upgrade allows are WebSockets. For security reasons, scripts running in the browser are not allowed to open TCP connections to servers. Protocol upgrades allow an existing HTTP connection to be used as a WebSocket, allowing bidirectional communication between a client script and the server, including server-side pushes of messages.

Protocol 3.7 (HTTP/2)**.** *HTTP/2 or HTTP 2.0 is the latest major release of HTTP. It includes a number of changes that are not backward compatible.*

Remarks:

- While in previous versions of HTTP headers are human readable, HTTP/2 uses a binary format to reduce the size of the request and the response, resulting in lower latency and a smaller protocol overhead.

- HTTP/2 introduces multiplexing of logical streams onto a single connection. Each request-response pair forms a logical stream. The streams are split into frames that may be interleaved arbitrarily and sent over a single shared connection. This allows a single connection between client and server to be used for an arbitrary number of concurrent requests, thus eliminating the need to open multiple connections to the same server to load resources concurrently. Multiplexing furthermore enables dynamic prioritization of requests and server-side pushes, e.g., a server could push additional resources for a webpage without waiting for the client to request them.

3.2 HTML

Definition 3.8 (HTML)**.** *Hypertext Markup Language (HTML) is a markup language used to annotate plain text with hyperlinks and rendering instructions, enabling the creation of* ***structured documents****.*

Definition 3.9 (Element)**.** *An HTML element is a part of a document delimited by a pair of opening and closing* ***tags****. Tags themselves are delimited by angle brackets (*`<` *and* `>`*).*

Remarks:

- Figure 3.2 displays a simple HTML document. HTML elements are used both as rendering directives, e.g., the above `<strong>` tag will result in bold text, and semantic markup, e.g., `<head>` delimits a region containing the metadata of the document.

- Elements can be nested arbitrarily, e.g., a bold text as part of an italic text. The resulting structure is a rooted tree of elements in which child elements span parts of a parent element. Notice that this implies that tags must be closed in the inverse order they have been opened.

- The content of a tag is defined as the text between the opening and closing tags. The content may also be empty, i.e., the opening tag is directly followed by a closing tag. Empty elements may use the `<tag/>` shorthand instead of `<tag></tag>`. An example of an empty tag is the linebreak `<br/>` which forces a line break and does not have any content itself.

- Elements may have attributes that specify the behavior of the element. For example a hyperlink has a *content*, i.e., the clickable text, and a *location* which the browser should visit when clicked. Such a link has the following markup: `<a href='location'>content</a>`

```
<html>
  <head>
    <title>This is a title</title>
  </head>
  <body>
    <p><strong>Hello</strong> world!</p>
  </body>
</html>
```

Figure 3.10: Simple HTML document with header and a formatted body.

- Most browsers have a basic interpretation of how an element is to be rendered. This behavior can be arbitrarily modified by applying *styles* to the elements.

Definition 3.11 (Cascading Style Sheets)**.** *Cascading style sheets (CSS) can be used to override the rendering defaults included in a browser.*

Remarks:

- A cascading style sheet consists of a set of *rules*. Each rule has a *selector*, which specifies which elements the rule should be applied to, and a *declaration*, specifying how the rendering should be changed from the browser's defaults.
- Simple selectors could for example match all elements with a given element, e.g., if the text in a `<strong>` element should also be italic.
- Selectors may make use of the tree structure of the document, e.g., a rule may match all `<strong>` elements whose parent is a `<p>` using `p > strong` as a selector.
- The declaration is a block of key-value pairs, each specifying one aspect of the rendering.
- The cascading style sheet is either inlined into the HTML document as a `<style>` element in the `<head>`, or it is stored in a separate file and bound to the document using a `<link>` element in the head.

```
p > strong {
  color: red;
}
```

Figure 3.12: A cascading style sheet with one rule matching all strong-elements that are direct children of a p-element. The declaration specifies that the text in all matched elements should be red.

3.3 DNS

When visiting a website, do you enter an IP address like 195.176.255.237 or do you prefer using a name like www.google.com? In the latter case you are using DNS.

Protocol 3.13 (DNS)**.** *The domain name system (DNS) maps human readable domain names to the IP addresses of servers which are serving requests for those domains.*

Remarks:

- Translating a domain name to IP addresses is called *resolving* the domain name.
- Besides mapping domain names to IP addresses, DNS also provides the inverse mapping from IP addresses to domain names.

Definition 3.14 (Nameserver)**.** *A server in the domain name system is called a nameserver. It offers the domain name resolution service to its clients.*

Remarks:

- DNS is a request-response based protocol. The client sends a request packet to the nameserver, which looks up the matching record and returns it as a response to the client. The protocol is not human readable and is composed of two single packets which hold all fields for both request and response.
- Despite being part of the core functionality of the internet, DNS is implemented in the application layer, as a protocol on top of UDP.
- DNS is implemented as a distributed database in order to guard against single points of failure, to distribute traffic over a large number of servers, and to enable local caching for faster domain name resolution.
- The distributed database stores *resource records* (RRs). Resource records may have a multitude of types. Common types are `A` and `AAAA` for domain name to IPv4 and IPv6 mappings respectively, `CNAME` for canonical names, `NS` to delegate the domain name to another nameserver, and `MX` to locate the mailserver responsible for the queried domain.
- There is no single server holding all the database's records, instead each nameserver only knows a subset of domain names.

Definition 3.15 (Authoritative Nameserver)**.** *A server that is responsible for a domain name is called the domain's authoritative nameserver. The authoritative nameserver is the server in which the mapping can be configured by the domain name owner. It is the server that should be contacted to ultimately resolve the domain name.*

Remarks:

- In order to resolve the domain name of a server we first need to locate the authoritative nameserver for that domain name. This chicken-and-egg problem is solved by introducing a hierarchy of nameservers that delegate the resolution of domain names until the authoritative nameserver is found.
- The hierarchy results in a tree structure with well known static nameservers at the root, which then delegate the resolution to their descendants.

Definition 3.16 (Root Nameserver)**.** *A root nameserver is a nameserver that serves as the reliable point of contact when resolving a domain name. A list of root nameserver IPs is included with the operating system of all nameservers. Root nameservers are contacted if no better nameserver is known.*

Remarks:

- There are a total of thirteen root nameservers in the world, operated by a multitude of organizations. Having the root nameservers operated by a diverse group of organizations should ensure that the system is resilient against failures.
- Some root nameservers share the same IP address. IP anycast routing assures that the closest server is reached.
- Upon receiving a request, a nameserver checks whether it has the required information to respond to the request. Should the nameserver not have the necessary information, it will delegate the request to another nameserver, or drop the request if it does not know a better nameserver. The delegating server returns a `NS` response containing the next nameserver the client should contact.
- The first level below the root servers are the regional nameservers responsible for the top level domains (TLDs). Regional nameservers then delegate to a number of registrars which cater to the end users. Further levels are introduced to divide the load and the responsibility of a domain to a number of nameservers.
- If we were to always perform resolution by contacting the root nameservers, they would become bottlenecks. For this reason clients and nameservers cache the responses and return these if a matching request is received. To this end each response contains a time-to-live (TTL) field which specifies how long, in seconds, the response may be cached. The higher levels in the hierarchy specify a high TTL, since they have very few changes over time. Lower levels, where the majority of changes are performed specify lower TTL values.
- It is desirable for some nameservers to act as a local cache, e.g., an ISP nameserver may directly serve its customers to reduce latency and reduce the number of requests. Instead of delegating the request to another nameserver, these nameservers perform the resolution on

behalf of the client. This mechanism is called *recursive resolution* and allows the nameserver to cache the response and later serve similar requests locally.

3.4 Mail

Definition 3.17 (Mailserver)**.** *A mailserver is a server that routes mail messages from the sender to the recipient and stores incoming mail messages for its users.*

Remarks:

- A mail address is composed of a username and a domain, separated by an `@`. Mail destined to users of the same domain is delivered to the same mailserver. The mailserver that is responsible for handling a domain's mail is specified by an `MX` record returned by the nameserver of that domain. Mail on a mailserver awaiting to be retrieved by the user is said to be *spooled*.
- When sending a mail, the sending user will look up the responsible mailserver for the recipient's domain through DNS and contacts it to deliver the mail. The delivery from the user to the receiving mailserver is done using *SMTP*.
- Most modern clients may be configured to use a single mailserver for outgoing mail. When sending a mail the client contacts its outgoing mailserver instead of the destination mail server. The outgoing mailserver then delivers the mail on behalf of the client, allowing a number of advanced features such as sender authentication and automatically reattempting delivery should the destination mailserver be unreachable. In this scenario SMTP is used in both interactions, from the client to the outgoing mailserver and the outgoing mailserver to the destination mailserver.

Protocol 3.18 (SMTP)**.** *The simple mail transfer protocol (SMTP) is a protocol used to deliver mails from a client to a mailserver.*

Remarks:

- SMTP is an interactive human readable protocol over TCP connections. A mailserver listens to port 25 for unencrypted incoming connections, while port 465 is commonly used for encrypted incoming connections. Similar to HTTPS the encryption is implemented by adding an SSL layer inbetween the TCP layer and the SMTP layer. An interactive succession of requests and responses is called a *session*.
- The client issues requests in the form of text commands, while the server responds with a numerical response code and a textual response.
- Common commands include `HELO` to initiate a session, `RCPT TO` to specify the mail recipient, `MAIL FROM` to specify the mail sender and `DATA` to specify the mail content.

- With the exception of `DATA`, the commands are terminated by a newline. The `DATA` command is followed by the content of the mail, hence it allows multiple lines to follow, and is terminated by a `\n.\n`, i.e., a punctuation mark on an otherwise empty line.
- Similar to HTTP, the response codes are grouped into function groups:
  - *2xx: Success* The command issued by the client succeeded.
  - *3xx: Start mail input* In response to a `DATA` command, the client may send the mail body.
  - *4xx: Client error* The client issued an invalid command.
  - *5xx: Server error* The server failed to act on the command.
- Initially SMTP did not include any form of authentication, since it is solely used for mail delivery, i.e., it is not possible to retrieve someone else's mails over SMTP. The destination mailservers often still accept mails from unauthenticated sessions, however outgoing mailservers today require the user-agent to authenticate to reduce the risk of denial-of-service attacks and spam.
- Multipurpose Internet Mail Extensions (MIME) is a standard that describes how the content of a mail may be encoded. MIME enables HTML formatted messages, attachments, and avoids some problems with plaintext mails.

Protocol 3.19 (POP)**.** *The post office protocol (POP) is a protocol that enables a client to retrieve and manipulate spooled mail messages on a mailserver.*

Remarks:

- The first version of POP was introduced in 1984. Later versions introduced incremental changes. The latest version, POP3, includes an extension mechanism and a flexible authentication mechanism.
- POP3 is the retrieval counterpart of SMTP. It is therefore not surprising that the protocols are very similar with text based commands and text based responses. However, unlike SMTP, the responses do not contain a numeric response code, instead responses are prefixed with `+OK` if the command succeeded and failure causes the connection to be closed.
- Common commands include `USER` and `PASS` for authentication, `STAT` to retrieve overall statistics of the mailbox, `LIST` to retrieve a list of messages including the message ID and its length, `RETR` to retrieve a message and `DELE` to delete a message.
- Similar to the `DATA` command in SMTP, some responses may return multiple lines, in which case they are terminated using a punctuation mark on an otherwise empty line.
- Using SMTP and POP3, the mailserver is used for temporary storage of mails until they are retrieved. If a user uses multiple computers, mails retrieved on one client will not be synchronized with the other client.

- Modern alternatives to POP3 include the Internet Message Access Protocol (IMAP) and web-based mail clients.

Protocol 3.20 (IMAP)**.** *The Internet Message Access Protocol (IMAP) is a protocol used by mail clients to access mail messages from a mailserver over a TCP connection. IMAP was designed with the goal of permitting complete management of a mailbox by multiple mail clients, therefore, clients generally leave messages on the server until the user explicitly deletes them.*

Remarks:

- An IMAP server typically listens on port number 143. IMAP over SSL (IMAPS) is assigned port number 993.
- In contrast to POP, IMAP focuses on organizing the mail directly on the mailserver as opposed to downloading them and organizing them locally. For this purpose IMAP introduces the concept of folders into which mail can be organized and retrieved from. Keeping the mail on the mailserver furthermore allows multiple synchronized clients.
- IMAP is a human readable protocol that follows a similar format as POP. IMAP prefixes the last line of a request-response pair with an alphanumeric unique *tag*. The tag is generated by the client and sent with the request, and the server will reply with the same tag in its response. This uniquely identifies the last line of a response and hence reduces the ambiguity of having some responses span multiple lines.

3.5 Socket API

Definition 3.21 (Socket)**.** *A socket is the principal abstraction for network communication exposed in programming languages. A socket exposes the necessary information and methods to a user application, to exchange data between clients and servers.*

Remarks:

- A socket is endpoint of a connection between two applications. Data sent through a socket on one end is received by the socket on the other end.
- In order to establish a connection, the server must have a socket listening for incoming connections, and the client creates an outgoing socket connecting to the listening server socket.
- TCP and UDP are the most common transport layer protocols used when communicating between applications, however a multitude of other protocols exist with various tradeoffs in terms of latency, security and consistency.

```
1  import java.io.*;
2  import java.net.*;
3
4  public class HelloWorldClient {
5      public static void main(String[] args) throws
           IOException {
6          if (args.length != 3) {
7              System.err.println(
8                  "Usage: java HelloWorldClient <name> <host
                       > <port number>");
9              System.exit(1);
10         }
11         String name = args[0];
12         String host = args[1];
13         int port = Integer.parseInt(args[2]);
14
15         Socket sock = new Socket(host, port);
16         PrintWriter out = new PrintWriter(
17             sock.getOutputStream(), true);
18         BufferedReader in = new BufferedReader(
19             new InputStreamReader(sock.getInputStream()));
20
21         out.print(name + "\n");
22         out.flush();
23         System.out.println(in.readLine());
24     }
25 }
```

Listing 3.22: Simple greeting client in Java.

Remarks:

- Our example client (Listing 3.22) takes three arguments: a name, a host and a port. The client connects to the specified host and port (line 15), sends the provided name (line 21), and reads the server's response (line 22). Both the request and response are terminated by a newline character, which allows the use of the readLine method to retrieve the contents.

- Java buffers most of its I/O operations. This includes network communication. After calling the print method of the PrintWriter on line 21, the written data is buffered as more data may be added. Calling flush instructs the underlying socket implementation to construct a TCP packet and send the data that was buffered so far. Manually flushing is not always necessary and happens automatically if the size of the buffered data exceeds the maximum packet size or a timeout is triggered. Splitting data into individual packets requires that the data is reassembled on the receiving side.

```
1  import java.net.*;
2  import java.io.*;
3
4  public class HelloWorldServer {
5      public static void main(String[] args) throws
           IOException {
6          int portNumber = 31337;
7
8          ServerSocket serverSocket = new ServerSocket(
9              portNumber);
10         Socket clientSocket = serverSocket.accept();
11
12         PrintWriter out = new PrintWriter(
13             clientSocket.getOutputStream(),
14             true);
15         BufferedReader in = new BufferedReader(
16             new InputStreamReader(
17                 clientSocket.getInputStream()));
18
19         String name = in.readLine().trim();
20         out.print("Hello " + name + "\n");
21         out.close();
22         in.close();
23     }
24 }
```

Listing 3.23: Simple greeting server in Java.

Remarks:

- The server in Listing 3.23 listens for an incoming connection on port 31337. Once a connection is established, the server reads a line, i.e., the name of the client terminated by a newline character, responds with a greeting terminated by a newline, and closes the connection.
- Our program only accepts a single connection and then terminates. Real servers continually accept incoming connections and hand off the actual interaction to a thread.
- Notice that the server does not flush the buffered data, but the close method on the PrintWriter also causes the buffered data to be flushed.

3.6 Protocol Layers

The structure of network protocols (Chapter 1), transport protocols (Chapter 2) and application protocols (Chapter 3) is similar. Let's formalize this similarity by introducing a layered architecture for network communication.

Definition 3.24 (Internet Protocol Suite)**.** *The internet protocol suite is a computer networking model, and set of communications protocols, used in the internet as well as similar computer networks. It divides the responsibility of*

*individual protocols into **layers** that expose an abstract interface to higher up protocols.*

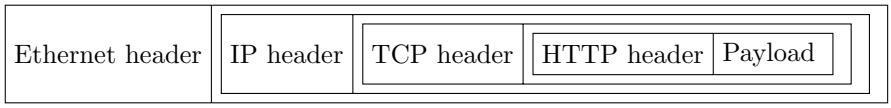

Figure 3.25: An example of nested layers for an HTTP request. Each layer has its own header and a payload consisting of the next higher layer.

Remarks:

- The internet protocol suite defines the following layers:
  - Application layer
  - Transport layer
  - Network layer
  - Link layer
- Layers are stacked so that higher layer protocols do not need to deal with the implementation details of lower layers, instead they may rely on the abstract interface exposed by those layers. Protocols in a layer should be designed in such a way that they are interchangeable. For example the format of an IP packet is independent of the physical medium it is transferred on, e.g., wire or air.
- The layering is implemented by repeatedly nesting packets of a layer in the next lower layer. Each layer is composed of a header and a payload. The payload simply contains the next higher layer packet. Figure 3.25 shows an example of how a link layer packet, Ethernet in this case, looks like.
- Over time the network has consolidated to just a few protocols that are in use, and the predominant network layer protocol is IP.

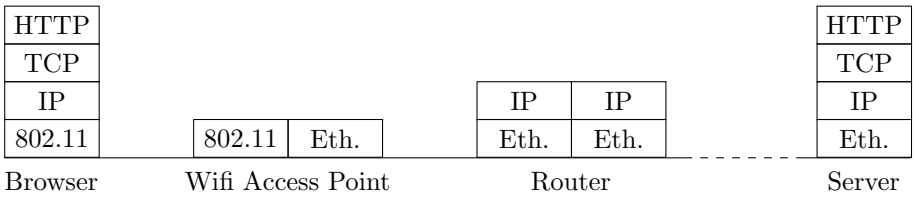

Figure 3.26: The browser issues an HTTP request over Wifi (802.11 link layer protocol) which is received by an access point and forwarded over Ethernet to a router. The router reads the IP header to identify the next hop and forwards the packet accordingly. Only the server needs to parse the entire stack of layers to get the original HTTP request.

Remarks:

- Separating the protocols into layers also allows keeping routing devices simple: a network switch does not need to understand the application protocol in order to correctly route a packet.
- Figure 3.26 shows an example of how packets can be routed without parsing the entirety of the packets.
- It may not always be possible to uniquely assign a protocol to a layer. Lower layer protocols may sometimes be tunnelled by higher layer protocols. In Section 1.7 we have seen that IP packets can be tunnelled through a VPN connection with IPsec by wrapping IP packets in IPsec packets. If we were to establish a TCP connection over IP tunnelled through IPsec, then the IP layer may be categorized as the transport layer or the network layer depending on whether we consider the point of view of the TCP connection or the IPsec connection.

Chapter Notes

Application layer protocols are specified in Requests for Comments (RFCs) which are the canonical specification of internet protocols. This allows a number of manufacturers to implement these protocols and guarantee compatibility.

Mail is one of the earliest applications, it even predates DNS. Mail relies on a number of protocols, some of which we have discussed in this chapter. SMTP was introduced in 1982 with RFC 821 [6], and initially required users to log into the nameserver where the mails were stored in a file structure to read and send mails. POP was introduced in 1984 with RFC [7], allowing users to retrieve mail on a mailserver and read it locally, while still using SMTP to send outgoing mails.

DNS was introduced in 1983 in RFC 883 [5]. It initially did not include any authentication of the responses, meaning that it was trivial for an intermediary nameserver to impersonate authoritative nameservers and redirect users to the wrong server. Domain Name System Security Extensions (DNSSEC) published in RFC 2065 [3] introduces authentication through digital signatures authenticating responses, and a public key infrastructure to authenticate the domain name owners.

HTTP 1.0 was released in 1996 in RFC 1945 [2] and was quickly superseded in 1997 by version 1.1 in RFC 2068 [4]. In 2009 Google announced a new transport protocol called speedy (SPDY) between its Chrome browser and its own services. After successful deployment of SPDY by Google it was picked up by the IETF and evolved into HTTP/2, which will eventually supersede both HTTP/1 and SPDY. HTTP/2 was finalized and published in 2015 as RFC 7540 [1].

This chapter was written in collaboration with Christian Decker.

Bibliography

[1] M. Belshe, R. Peon, and M. Thomson. RFC 7540: Hypertext Transfer Protocol Version 2 (HTTP/2), 2015.

[2] T. Berners-Lee, R. Fielding, and H. Frystyk. RFC 1945: Hypertext Transfer Protocol – HTTP/1.0, 1996.

[3] D. Eastlake and C. Kaufman. RFC 2065: Domain Name System Security Extensions, 1997.

[4] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee. RFC 2068: Hypertext Transfer Protocol – HTTP/1.1, 1997.

[5] P. Mockapetris. RFC 883: Domain Names - Implementation and Specification, 1983.

[6] Jonathan B. Postel. RFC 821: Simple Mail Transfer Protocol, 1982.

[7] J. K. Reynolds. RFC 918: Post Office Protocol, 1984.