Chapter 4
Application Layer & Sockets

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

4.1 HTTP

Definition 4.1 (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:
• 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.
• HTTP has simple request-response semantics: 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 create a resource on the server or to update an existing resource, 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 as Key: Value pairs which provide additional information about the client and the request, each on a new line. The request header is terminated by two successive newline characters. \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. 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.

Protocol 4.2 (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.
4.1. HTTP

- **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 4.3 (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, thus storing state about the user.

**Protocol 4.4 (HTTP/1.0).** HTTP/1.0 is the first version of HTTP.

- HTTP/2 only with encryption. Thus, the performance gains of HTTP/2 are only available to web sites using HTTPS. Remember this when creating your own web site!
Figure 4.10: Simple HTML document with header and a formatted body.

4.2 HTML

Definition 4.8 (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 4.9 (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 4.10 displays a simple HTML document. HTML elements are used both as rendering directives, e.g., the <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. Some 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>
- 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 4.11 (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 4.12: 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.

4.3 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 4.13 (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 4.14 (Nameserver). A server in the domain name system is called a nameserver. It offers the domain name resolution service to its clients.
4.3. DNS

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 4.15 (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 4.16 (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.

4.4 Mail

Definition 4.17 (Mailserver). A mailserver is a server that routes mail messages from the sender to the recipient and stores incoming mail messages for its users.
4.4. MAIL

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 4.18 (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. The response code indicates one of three possibilities: if the command succeeded and failure causes the connection to be closed.

• 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.\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.

4.5. POST OFFICE

Remarks:

• 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 mail servers often still accept mails from unauthenticated sessions, however outgoing mail servers 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 4.19 (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 4.20 (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.
4.5 Socket API

Definition 4.21 (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.

Remarks:

- Our example client (Listing 4.22) 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 23). 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.

Remarks:

- The server in Listing 4.23 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.

```java
import java.net.*;
import java.io.*;

public class HelloWorldServer {
    public static void main(String[] args) throws IOException {
        int portNumber = 31337;
        ServerSocket serverSocket = new ServerSocket(portNumber);
        Socket clientSocket = serverSocket.accept();
        PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
        BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
        String name = in.readLine().trim();
        out.println("Hello "+name+"\n");
        out.close();
        in.close();
    }
}
```

Listing 4.23: Simple greeting server in Java.

- 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.

### 4.6 Protocol Layers

The structure of network protocols (Chapter 2), transport protocols (Chapter 3) and application protocols (Chapter 4) is similar. Let’s formalize this similarity by introducing a layered architecture for network communication.

**Definition 4.24** (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.*

**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 4.25 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 4.25: An example of nested layers for an HTTP request. Each layer has its own header and a payload consisting of the next higher layer.

- Transport layer
- Network layer
- Link layer

- The Internet protocol suite defines the following layers:
  - Application layer
  - Transport layer
  - Network layer
  - Link layer

- Over time the network has consolidated to just a few protocols that are in use, and the predominant network layer protocol is IP.

Figure 4.26: 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 identity 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 4.26 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 2.7 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 918 [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