IP-Based Applications and HTTP

This lecture explains how most IP-based applications work by using HTTP, a typical application-level protocol, as an example. HTTP, Hypertext Transfer Protocol, is the application-level protocol used to retrieve hypertext information ("web pages") from a "web server".
After this lecture you should be able to: parse a URL URI into its components, URL-encode an arbitrary string, parse a media (MIME) content type into its components, generate the text for an HTTP 1.1 request given the URL and header values, generate the text for an HTTP response given the content and header values, and add A and B HTML tags to text to create a hypertext document.

Introduction

Hypertext Transfer Protocol, HTTP, is the protocol used to retrieve documents, usually in the HyperText Markup Language (HTML), from a server. We will study HTTP because it is similar to many other IP application protocols such as:

FTP (File Transfer Protocol) - used to transfer files

SMTP (Simple Mail Transfer Protocol) - used to send mail to a mail server

IMAP (Internet Message Access Protocol) and POP (Post Office Protocol) - used to retrieve e-mail from a server

SIP (Session Initiation Protocol) - a signalling protocol used to set up voice and video calls

In each of these protocols, a client application establishes a TCP connection to a server and sends a request. The server responds with a result code, typically numeric and the requested data. The request/response sequence can be repeated (typically, but not always, using the same TCP connection) until the client or server terminates the session with a terminating request or response or by closing the TCP connection.

Messages exchanged between the client and server are lines of text, each terminated with a CR-LF pair. Each request or response consists of one line in a protocol-specific format followed by a sequence of header lines. A blank line is used to separate the headers from the data that often follows a request or response. The header lines typically consist of a header name, a colon and header-specific data. The data is terminated in a protocol-specific manner (e.g. a terminator line or reaching a byte count specified in a header).

For example, here is the exchange required to send a short e-mail using the SMTP protocol:

```
MAIL FROM: edc@bcit.ca
250 2.1.0 Ok
RCPT TO: alice@example.net
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Subject: Test

Hello!
.
250 2.0.0 Ok: queued as 1A4FE1AE002D
QUIT
221 2.0.0 Bye
```

Exercise 1: Which of these lines were sent by the client and which by the server? What are the header lines and how are they terminated? What is the data and how is it terminated? How many different result codes are there? What caused the connection to terminate?

Exercise 2: What are some advantages and disadvantages of using text-based protocols?

HTTP Protocol Overview

Web browsing is one of the most common Internet applications. A web browser is a program that displays hypertext. Hypertext is text that has been combined with "markup" instructions that can embed resources such as images and also allows the reader to

follow "links" to other resources, often other hypertext documents.

We will look in more detail at HTTP as an example of an IP-based protocol. HTTP is defined in various RFCs. The most commonly-used version, HTTP 1.1, is defined in RFC 2616 and follows the general Internet application protocol described above.

For example the request for a web page to the host www.example.net might involve the client setting up a TCP connection to the server and sending the lines shown in Listing 1 where the first line (GET) is the request, lines following that are headers with additional information about the request and the blank line terminates the request.

HTTP request often that indicate the client software, the languages and types of content the client can handle, and session context information in the form of "cookies."

There are other requests in addition to GET. For example, PUT and POST request can be used to send data to the server.

Exercise 3: What might some of the headers in Listing 1 mean?

The response consists of one line with the status code (200), additional headers, a blank line and the the data associated with the response. Listing 2 shows a possible response to the previous request.

Exercise 4: What might some of the headers in Listing 2 mean?

URIs and URLs

A Universal Resource Identifier (URI) and Universal Resource Locator (URL) are the syntax used to identify resources which are typically, but not always, files that can be retrieved over the Internet. URIs and URLs are defined in RFC 3986. The syntax of a URI is:

scheme ":" hier-part ["?" query] ["#" fragment]

A URL is a specific type of URI. The format of a URL is:

scheme://domain:port/path?query_string#fragment_id

The fields are:

scheme - often called the protocol, this defines both the syntax of the rest of the URL and, in most cases the IP protocol used to retrieve the information (e.g. http, ftp, etc)

domain - this is the host or IP address

port - the TCP port number (defaults to well-known values)

path - the (virtual) location of the resource on the server

query - additional data to be passed to the web server, typically a something specific to this request such as text to be searched for

fragment - the portion of the requested document, typically a section in a document

Exercise 5: Parse the URL:

```
https://bcit.ca:85/files/public/?bydate#first-end
```

The web client uses the domain and port information to set up a connection to the server but the server interprets the remainder of the URL (possibly including the domain) as it sees fit.

The server typically also has access to additional information supplied by the client in protocol headers (IP, TCP or HTTP) such as the IP address, "cookies" and the history of previous requests. This means that the server's response to a particular query could depend on many factors, not just the URL itself.

URL Encoding

As shown in the above syntax, various characters (/, :, ?, #) are used to separate the URL into its parts and thus cannot be used in the content of the URL. Since spaces are used as terminators they also cannot be used.

Escape sequences beginning with the percent (%) character are used to include these special characters in URLs. The escape character is followed by two hexadecimal digits which define the value of the character. The byte sequence can also be a UTF-8 encoded string. For example, a space (hex 20) can be represented by %20. A plus sign ('+') can also be substituted for a space.

Exercise 6: URL-encode the string `Hi's?`, escaping all characters other than letters.

```
GET / HTTP/1.1
Host: www.example.net
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:37.0) Gecko/20100101 Firefox/37.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: keep-alive
```

Listing 1: Sample HTTP GET request. The last line is a blank.

```
HTTP/1.1 200 OK
Accept-Ranges: bytes
Cache-Control: max-age=604800
Content-Type: text/html
Date: Fri, 24 Apr 2015 22:10:01 GMT
Etag: "359670651"
Expires: Fri, 01 May 2015 22:10:01 GMT
Last-Modified: Fri, 09 Aug 2013 23:54:35 GMT
Server: ECS (pae/3796)
X-Cache: HIT
x-ec-custom-error: 1
Content-Length: 1270

<!doctype html>
<html>
<head>
...
```

Listing 2: Start of a sample response to HTTP GET. The first line is the status (200). The end of the data is determined by the byte count in the Content-Length: header.

Content Types

Resources retrieved over HTTP can be of different types (text, photographs, audio, etc) and each of these can be encoded differently. For example, a text document can be a plain text file or contain HTML markup or be in a particular Microsoft Word format. The pixels in a photograph can be encoded in various ways (raw pixel values, JPEG compression, ...).

There is a (sort of) standard for describing the content of a resource. The content type (also called a MIME type for Multimedia Internet Mail Extensions) is two strings separated by a slash. The first part specifies the media type and the second specifies how it is encoded. Examples of content types include:

- text/html - text using HTML markup
- text/plain - plain text
- video/x-flv - Flash video
- video/mp4 - MP4 compressed video
- application/octet-stream - arbitrary data generated by an application
- application/pdf - a document in PDF format that may contain other media types
- application/zip - a zip archive file

Exercise 7: What might the content-type `text/csv` mean?

HTML Markup

The topic of HTML and how it is displayed by browsers would be a course in itself, but a very brief introduction is given here.

The text supplied to the web browser consists of readable text and instruction for the browser in the form of markup 'tags'. In HTML the tags are enclosed in angle brackets (<, >) and are not displayed. Instead, the tags tell the browser how to display the text.

For example the bold (B) tag is indicated by placing the text to be displayed in a bold font between `<B>` and `</B>` tags.

The main advantage of hypertext is the ability to link to other documents. In HTML this is done with the anchor (A) tag. The tag includes a parameter containing the URL of the link. For example:

```
Some <B>bold</B> text.

And this is a link to
<a href="http://www.bcit.ca/">BCIT's web page</a>.
```

The web client will render the word 'bold' in a bold font and will indicate (e.g. by underlining) that the words "BCIT's web page" are a link to the indicated URL.

There are dozens of HTML tags that allow for things such as adding section headers (H1, H2, ...), dividing a page into sections (DIV), and embedding of images and other content into pages (IMG).

Modern web browsers have extended the Hypertext concept by allowing embedded software in the form of scripts written in the Javascript language. These scripts can manipulate the page's contents and retrieve additional content on their own. These features can be used to create web-based applications and user interfaces whose complexity rivals or exceeds those available on stand-alone computers.

Modern web pages also make use of a syntax called CSS (Cascading Style Sheets) to change the formatting of web pages rather than using embedding formatting tags such as <B>.