HTTP

This lecture gives an overview of HTTP, a typical application-level protocol. 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

This lecture describes, HTTP, Hypertext Transfer Protocol. HTTP is the protocol used to retrieve hypertext (usually HTML) and other documents 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

Web browsers are one of the most popular Internet applications. A browser displays hypertext. Hypertext is text that has been combined with "markup" instructions that allow the reader to follow "links" to other resources, including other hypertext documents.

Application Protocol Overview

All of these protocols work in a similar fashion. A client application establishes a TCP connection to a server and sends a request. The server responds with a result code and the requested data. The request/response sequence is repeated (typically, but not always, using the same TCP connection) until the client or server terminates the session with an appropriate request/response or by closing the TCP connection.

The messages exchanged between the client and server are lines of text, each terminated with a CR-LF pair. Each request or response typically consists of one line in a protocol-specific format followed by a sequence of header lines. A request or response often includes data and a blank line is used to separate the headers from the data. 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 the request for a web page might involve the client setting up a TCP connection to the server and sending the following three lines:

```
GET /about/ HTTP/1.1
Host: www.bcit.ca

```

where the line with GET is the request, the line with Host: is a header and the blank line terminates the request.

Exercise 1: What might be some advantages and disadvantages of using text-based protocols?

URIs and URLs

URI and URLs are 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 2: 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 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 within the URL.

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.

Exercise 3: URL-encode the string `Hi's?`, escaping all characters other than letters.

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) 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 4: What might the content-type `text/csv` mean?

HTTP Protocol

HTTP is defined in various RFCs. The most commonly-used version, HTTP 1.1, is defined in RFC 2616.

The HTTP protocol follows the general Internet application protocol described above.

The HTTP request consists of an initial line with the request type (most often GET, but can also be PUT, POST and others), additional header lines, a blank line to terminate the headers and then the data (if required by the request, for example, for POST requests).

For HTTP version 1.1 GET request only one header, Host:, is required and no additional data needs to be supplied. Thus the simplest HTTP request would be:

```
GET / HTTP/1.1
Host: www.example.com
```

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

The response from the server consists of a response line with the protocol version and 3-digit number followed by additional header lines and, optionally, the content (separated from the headers with a blank line). For example, the response to the above request might be:

```
HTTP/1.1 200 OK
Date: Mon, 05 May 2014 18:20:13 GMT
Server: Apache/2.2.16 (Debian)
Accept-Ranges: bytes
Vary: Accept-Encoding
Keep-Alive: timeout=10, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html

<!DOCTYPE html PUBLIC "-//W3C//DTD ...
... lots of HTML ...
```

HTTP responses often include headers that indicate the server software, the date, the encoding of the content, its length, and updated cookie information to be stored by the client and sent along with future requests.

In some cases this meta-information is embedded in the HTTP document itself rather than the header.

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 for completeness.

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.