Application Layer Improvements

Contents

- Our focus will be on two main applications, specifically web browsing and periodic light traffic such as voice.

1 Web browsing

- The functional components of the World Wide Web are illustrated in Figure 1
- **HyperText Markup Language (HTML)**
  - A text language used to define hypertext documents.
  - The idea behind HTML was to add simple constructs, called tags, to regular text documents, to enable the linking of one document to another, as well as to allow special data formatting and the combining of different types of media.
  - HTML has become the standard language for implementing information in hypertext, and has spawned the creation of numerous other related languages.

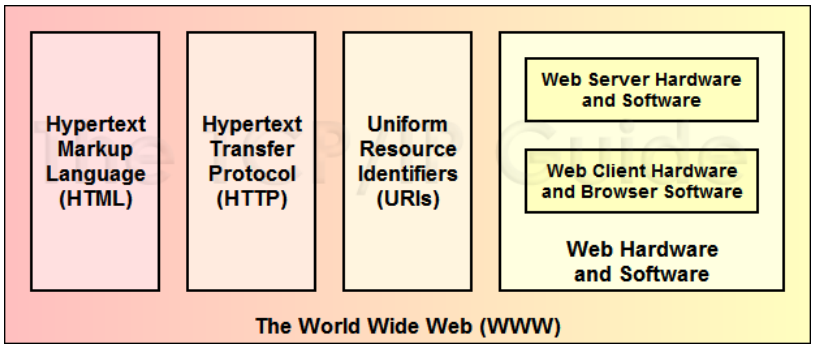


Figure 1: WWW Components

```
<html>
<head>
(head elements go here…)
</head>
<body>
(body elements go here…)
</body>
</html>
```

- **Hypertext Transfer Protocol (HTTP)**
  - HTTP is the main application protocol used to transfer webpages
  - HTTP began as a very crude protocol for transferring HTML documents between computers, and has evolved to a full-featured and sophisticated messaging protocol as will be shown below.
- **Uniform Resource Identifiers (URIs)**
  - . URIs were originally developed to provide a means by which the users of the Web could locate hypertext documents so they could be retrieved.
- **Web servers**
  - They are computers that run special server software to allow them to provide hypertext documents and other files to clients who request them.
  - Well-known HTTP server include Apache, lightpd, Nginx [2].
- **Web browsers**
  - They are HTTP client software (e.g. Firefox, Chrome, IE, Opera, ...) that run on TCP/IP client computers to access Web documents on Web servers.
  - These browser programs retrieve hypertext documents and display them, and also implement many of the Web's advanced features, such as caching.
  - Today's browsers support a wide variety of media, allowing the Web to implement many different functions aside from simply hypertext document transfer. Examples include displaying images, playing sounds and implementing interactive programs.

1.1 HTTP Overview

- What happens when you request a page?
  - an HTTP client establishes a connection to an HTTP server using TCP.
    - convert server name to an IP address
  - The client then issues a single "GET" request specifying a resource to be retrieved.

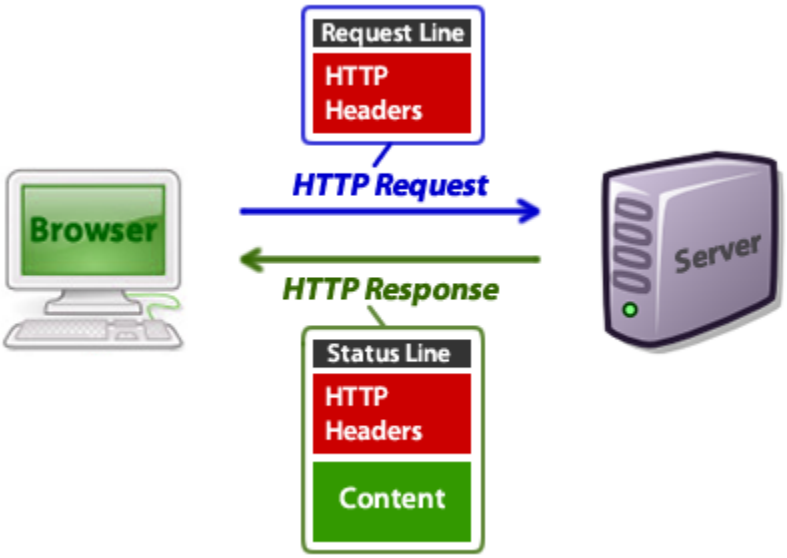


Figure 2: HTTP Dialog

– The server responds by sending the file as a stream of text bytes, and the connection is terminated.

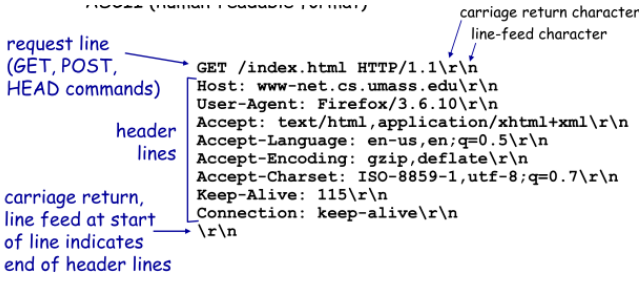
WIRESHARK DEMO

- HTTP GET request

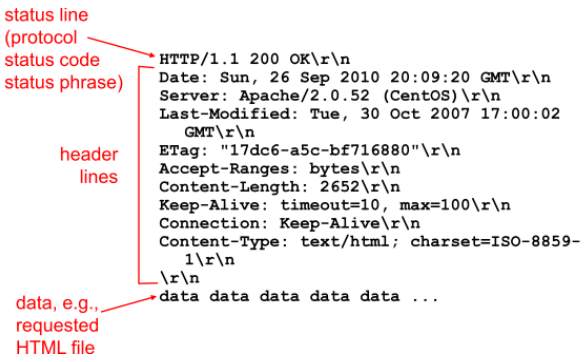
  For example, suppose the user enters a URL such as this:

  http://www.myfavoritewebsite.com:8080/chatware/chatroom.php

  We obviously don't need to send the "http:" to the server. The client would take the remaining information and split it so the URI was specified as "/chatware/chatroom.php" and the Host line would contain "www.myfavoritewebsite.com:8080". Thus, the start of the request would look like this:



- Common messages: GET, HEAD, POST
- HTTP Response and status code



- 2xx messages indicate successful operations
  - 200 OK
- 4xx indicates a client error,
  - 404 (page not found)
  - 400 bad request
- 5xx indicates server error
  - 505 HTTP version not supported

1.2 HTTP Connections

- **Transitory Connections**
  - These connections, called transitory due to their short-lived nature, were the only type supported by the original HTTP/0.9, and the same model was maintained in the more widely-deployed HTTP/1.0.
  - :) The advantage of this connection model is its conceptual simplicity
  - :(the problem with it is that it is inefficient when the client needs to make many requests to the same server.
    - This is often the case with modern hypertext documents, which usually carry inline references to images and other media.
    - Recent measurement studies on show that webpages have an average of 20 objects.
- **Persistent Connections**
  - Introduced by ***HTTP/1.1***
  - allows a client to send multiple requests for related documents to a server in a single TCP session.
  - This greatly improves performance over HTTP/1.0, where each request required a new connection to the server.
  - Persistent connections can be used with and without ***request pipelining***
- HTTP 1.1 also introduced several features of interest

  - ***Partial Resource Selection:*** In HTTP/1.1, a client can ask for only part of a resource rather than the entire document, which reduces the load on the server and saves transfer bandwidth.
  - ***Better Caching and Proxying Support:*** HTTP/1.1 includes many provisions to make caching and proxying more efficient and effective than they were in HTTP/1.0. These techniques can improve performance by providing clients with faster replies to their requests while reducing the load on servers, as well as enhancing security and implementing other functionality.
  - ***Content Negotiation:*** A negotiation feature was added that allows the client and server to exchange information to help select the best resource or version of a resource when multiple variants are available.

- HTTP/1.1 continues to be the current version of the Hypertext Transfer Protocol, even though it is now several years old. This may seem somewhat surprising, given how widely used HTTP is. Then again, it may be the fact that so many millions of servers and clients implement HTTP/1.1 that no new version has been created.

1.2.1 Example

A user is interested in an HTML page containing 20 embedded objects. Calculate the time required to fetch the page in the following cases

1. HTTP 1.0
2. HTTP 1.1 using persistent connection and no pipelining and a single connection
3. HTTP 1.1 using persistent connection and pipelining and a single connection
4. HTTP 1.1 using persistent connection and pipelining and multiple connections

Assume

- geostationary satellite
- assuming that all the content is located on the same server.
- the average object size is k kilobytes
- assume the link speed is c kbps assume n_c parallel connections

This example shows the dramatic dependence on RTT of the communication link. Hence, optimizations tricks should be performed to optimize the data transfer over long delay link. Among these tricks

- PEP optimizations
- Web caching

2 PEP optimizations

- TCP acceleration
- compression
- trans-coding
  - images
  - video

2.1 Case Study: XIP-LINK solution for satellite systems

Check xiplink pdf.

References

[1] HTTP Standard (RFC 2616)

[2] Web Servers

[3] http://www.tcpipguide.com