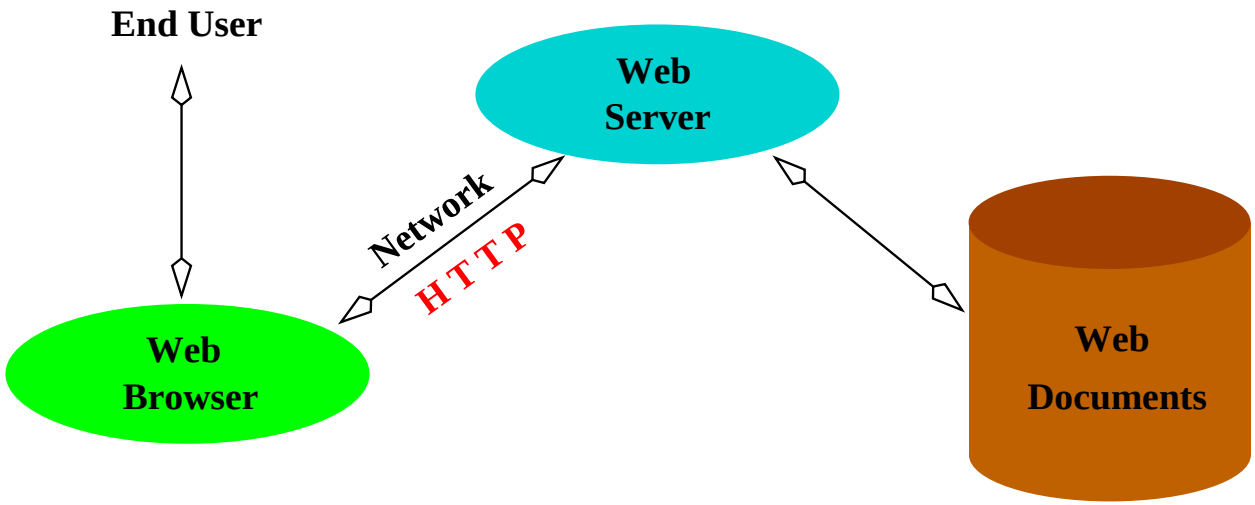
# An Overview of HTTP

Paul S. Wang
Institute for Computational Mathematics
Kent State University

pwang@mcs.kent.edu
http://icm.mcs.kent.edu/~pwang

# Hyper-text Transfer Protocol



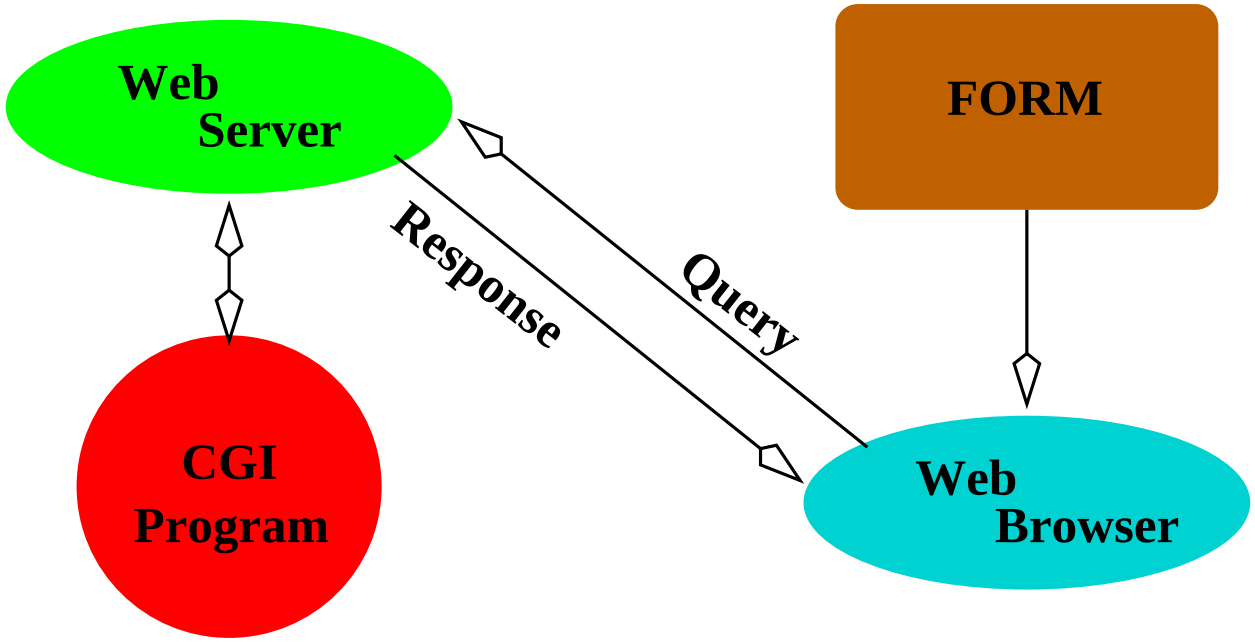
## Evolution of HTTP

- Early 1990s — simple HTTP, adopting MIME content types, ...
- Mid 1990s — HTTP 1.0 (rfc1945 in 1996), `POST` query, ...
- July 1996 — HTTP 1.1 drafted
- July 97 — HTTP-NG project begins
- November 97 — HTTP 1.1 Rev-01 moves HTTP closer to IETF Draft Standard
- HTTP 1.1 is becoming a standard, HTTP-NG will be revolutionary.

## HTTP Basics

- When communicating with a Web server, a program becomes a client of the Web server.
- HTTP protocol governs how the client and the server exchange information.
- HTTP employs the connection-oriented TCP/IP.
- A framework of HTTP transactions:
  1. *Connection* — A client opens a connection to a server.
  2. *Query* — The client requests a resource controlled by the server.
  3. *Processing* — The server receives and processes the request.
  4. *Response* — The server sends the requested resource back to the client.
  5. *Termination* — The transaction is done and the connection is closed.

# Form Processing Data Flow



## HTTP Message Format

HTTP governs the format of queries and responses:

| | |
|---|---|
| *initial line* | (different for query and response) |
| *HeaderKey1*: *value1* | (zero or more header fields) |
| *HeaderKey2*: *value2* | |
| *HeaderKey3*: *value3* | |
| | (an empty line separating header and body) |

*Optional message body contains query or response data.*
*The amount and type of data in the body is specified in the headers.*

The header part is textual and each line in the header should end in RETURN and NEWLINE but it may end in just NEWLINE.

## The Query Line

The initial line identifies the message as a query (request) or a response.

- A query line has three parts, separated by spaces: a *query method* name, a local path of the requested resource, and a HTTP version number.

```
GET    /path/to/file/index.html   HTTP/1.0
POST   /path/script.cgi           HTTP/1.0
HEAD   /path/to/file/index.html   HTTP/1.0
```

- The `GET` method simply request the specified resource and does not allow a message body.
- The `HEAD` query is like `GET` but requests only the header part of the response.
- The `POST` method allows a message body and is designed to work with a CGI URL.

# HTTP Query Syntax

```
<METHOD> <URI> "HTTP/1.0" <crlf>
{<Header>: <Value> <crlf>}*
<crlf>
```

Query headers are in RFC-822 format. Most common are `Accept` headers which tells the server which object types (MIME) the client can handle and `User-Agent` which gives the browser name/version.

## The Response Line

A response (or status) line also has three parts separated by spaces: a HTTP version number, a status code, and a textual description of the status. Typical status lines are:

```
HTTP/1.0    200    OK
```

for a successful query, or

```
HTTP/1.0    404    Not Found
```

when the requested resource cannot be found.

# HTTP Response Syntax

```
"HTTP/1.0" <result-code> [<message>] <crlf>
{<Header>: <Value> <crlf>}*
<crlf>
```

# HTTP Response Headers

| Header | Meaning |
|---|---|
| `Location` | New document location |
| `User-Agent:` | Name/version of program sending query |
| `SERVER:` | Name/version of server sending response |
| `Last-Modified:` | Last modification time of resource retrieved |
| `Content-Type:` | The type of the body in the message |
| `Content-Length:` | Length in bytes of the body |

## The POST Query

- A `POST` query includes:
  1. A URL specifying a CGI program
  2. `Content-Type` header
  3. `Content-Length` header
  4. Message body that includes *URL-encoded* material
- A Web server usually processes the incoming data by calling a CGI program.
- A CGI program receives the message body through standard input and processes it.

## Sample POST Query

```
POST /cgi-bin/register-user HTTP/1.0
HOST: SymbolicNet.mcs.kent.edu
From: jDoe@great.enterprise.com
User-Agent: NetScape 4.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 132

name=John Doe&address=678+Main+Street&...
```

## URL and Message Body Encoding

- HTTP uses special characters for certain protocol purposes.
- URLs are transmitted under *URL-encoding*.
- `POST` query data is usually in the form of a series of entries separated by `&`. Each entry is a *name*`=`*value* pair.
- Each SPACE within an entry is replaced by a `+`.
- Special characters in the name and value parts must be URL-encoded to avoid misinterpretation.
- URL-encoding basically means replacing each non-alphanumeric character by `%`*hh* where *hh* is its ASCII code in hexadecimal.

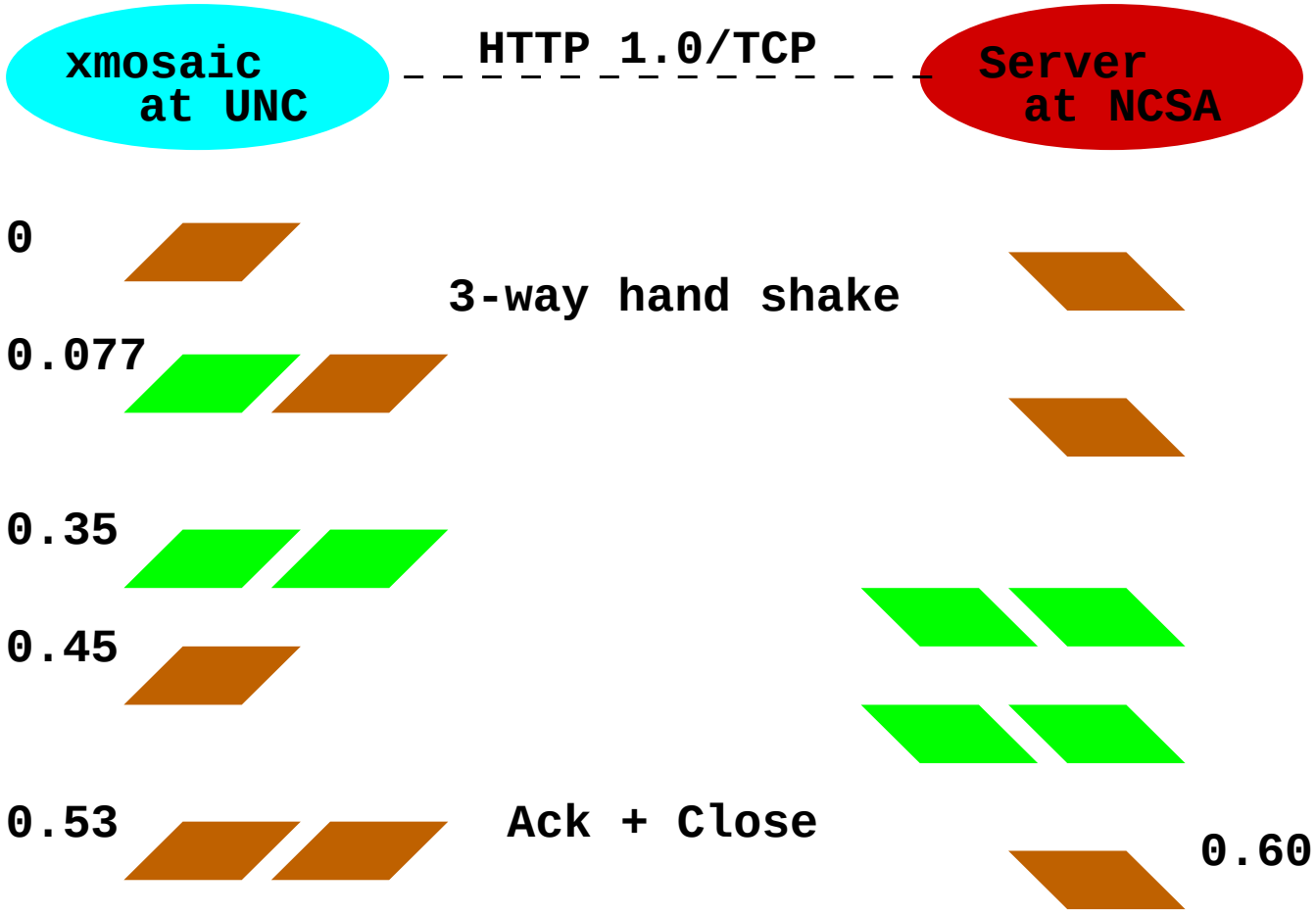
| | |
|---|---|
| `=` | `%3D` or `%3d` |
| `+` | `%2B` or `%2b` |
| `&` | `%26` |

# HTTP 1.0 Performance Analysis

# Typical Access Pattern

1. Client request a Web page, sending many "`Accept`" headers with it.
2. Same client then issues a series of requests to the same server for icons and images contained in the page
3. After displaying the page, a user usually clicks on a link which leads to another page at the same site

# HTTP 1.0 Transaction Traffic



## HTTP 1.0 Transaction Traffic

*Xmosaic* access of NCSA homepage: request 1130 bytes, response 1668 bytes, these are sent via TCP packets, each packet with a *max segment size* (MSS) of 536 to 1460 bytes.

- 0 – Client sends TCP connection request at port 80 of target host
- 0.077 — Server sends TCP connection response to client. Client acknowledges the connect response and sends the first 536 bytes of the HTTP request (A).
- 0.35 — Server acknowledges (A). The client sends 2nd part of request, and without waiting for an acknowledgement, sends the 3rd and final part of the request (B).
- 0.45 — Server sends a packet with acknowledgement of (B) and 512 bytes of the HTTP response, followed by a second packet with another 512 bytes of the response. The client acknowledges the receipt of the packets.

- 0.53 — Server sends 3rd and 4th response packet, and an indication to close the connection. The client acknowledges and announces that it is closing the connection.
- 0.60 — Server acknowledges the close.

## Metrics in the Example

- MSS — 536 byte default used
- RTT — round-trip time of about 70 milliseconds
- Bandwidth — 512 bytes/ 3.57 milliseconds (about 1.15 Mbps)

## HTTP 1.0 Performance Problems

- HTTP 1.0 interacts badly with TCP and spends most of the time waiting rather than transmitting data.
- HTTP 1.0 incurs frequent round-trip delays due to connection establishment.
- Suffers frequent slow starts in both directions for short duration connections (*slow start* refers to the slow increase of the number of unacknowledged packets a sender is allowed to send.)
- Incurs heavy latency penalties due to mismatch of typical access profiles with single-request model.
- Too many opened then closed connections present a problem for servers to keep `TIME_WAIT` state for closed connections (for 240 seconds after closing).

## HTTP 1.1 Features

HTTP 1.1 is becoming a standard to replace HTTP 1.0 in the near term. Generally speaking, HTTP 1.1 is a superset of HTTP 1.0.

- HTTP 1.0 defines 16 headers, none are required.
- HTTP 1.1 defines 46 headers, and one (`Host:`) is required in requests.
- HTTP 1.1 features:
  - Faster response, by allowing multiple transactions to take place over a single persistent connection.
  - Faster response and great bandwidth savings, by adding cache support.
  - Faster response from CGI scripts, by supporting *chunked encoding*, which allows a response to be sent before its total length is known.
  - Efficient use of IP addresses, by allowing multiple domains to be served from a single IP address.

# HTTP 1.1 Client

- must include the `Host:` header with each request
- must accept responses with *chunked data*
- must either support persistent connections, or include the `Connection: close` header with each request
- must handle the `100 Continue` response

## HTTP 1.1 Server

- must require the `Host:` header from HTTP 1.1 clients
- must accept absolute URLs in a request
- must accept requests with *chunked data*
- must either support persistent connections, or include the `Connection: close` header with each response
- must use the `100 Continue` response appropriately
- must include the `Date:` header in each response
- must handle requests with `If-Modified-Since:` or `If-Unmodified-Since:` headers

## HTTP Implementations

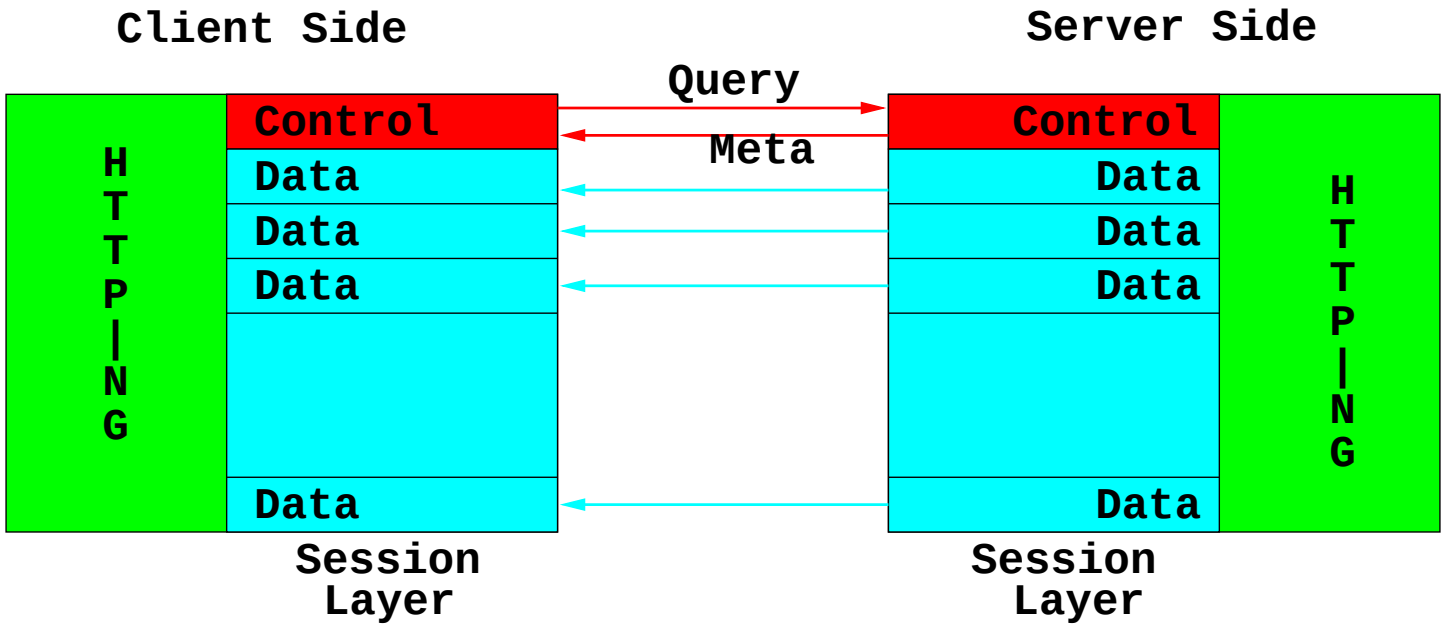
- `libwww` — W3C's client library in C and other tools on-top
- `jigsaw` — W3C's sample implementation of HTTP Jigsaw is a full blown HTTP/1.1 server entirely written in Java
- `apache` — Apache Group's HTTP/1.1 server and proxy
- `NetScape Navigator and Communicator` — full-featured browser/editor
- `HTTPClient` — Aas Software's client library in Java
- `CL-HTTP` — MIT AI Lab's large server, proxy, client, Web walker, package written in Common-Lisp
- For more details see `www.w3.org/Protocols/HTTP/Forum/Reports/`

## HTTP-NG the Next Generation

HTTP-NG aims to replace previous HTTPs. Major features include:

- Binary encoded, rather than text-based, messages using ASN.1 (*abstract syntax notation*) and PER (*Packed Encoding Rules*).
- Standard object types referenced by bit vectors
- Multiple objects (URLs) in a single `GET` request
- Multiple independent request-response streams:
  - To allow many different requests to be sent over a single persistent connection asynchronously
  - To allow concurrent responses to many requests and the interweaving of response packets
  - To separate *control* from *transaction* messages

# HTTP-NG Session Layer



## HTTP Specification and References

- RFC-1945 — HTTP 1.0
- RFC-2068 — HTTP 1.1
- `www.w3.org` — HTTP-NG
- RFC 822 — structure of Internet text messages, including header fields
- RFC 1738 — definition of URL syntax
- RFC 1808 — definition of relative URL syntax
- RFC 1521 — definition of MIME and of MIME types