B/S体系软件设计

胡晓军

课程内容

- B/S开发基础
- HTML/CSS/JavaScript
- PHP/Node.js/Python
- Java EE
- ASP.NET
- Client技术的发展
- Web应用优化

参考书

- **HTML5与CSS3基础教程（第8版）**
  - **Elizabeth Castro / Bruce Hyslop，望以文译**
  - 人民邮电出版社
- **深入浅出Spring Boot 2.x**
  - 杨开振
  - 人民邮电出版社
- **Front-end Development with ASP.NET Core, Angular, Bootstrap**
  - **Simone Chiaretta**
  - **Wrox**

考核方式

- 完成一个大程

Internet概述

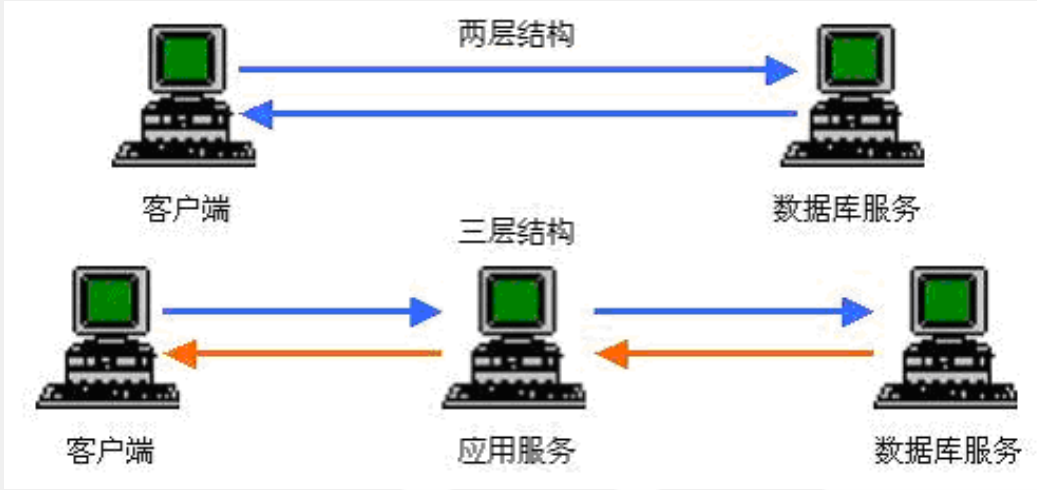
- Internet，因特网，互联网
- Internet历史
  - 20世纪60年代，ARPA Net
  - 20世纪80年代，TCP/IP
  - 1986，NSFNet
  - 20世纪90年代，Internet开始迅速发展
- Internet基础服务
  - WWW
  - Email
  - FTP
  - Telnet

B/S体系结构

- 主机模式:
  - all computation take place in the main computer, using dummy terminal
- C/S架构（Client/Server）
  - most computation take place in the server, client is a computer carrying part computation
- B/S架构（Browser/Server）
  - thin client, limited ability of client, using Web browser

B/S结构的优点

- 维护方便，能够降低总体拥有成本。
  - B/S比C/S的维护工作量大大减少了
  - B/S相对C/S能够降低总体拥有成本
- 选择更多
- 移动办公
- 系统整合



Internet网络协议

- **TCP/IP(Transmission Control Protocol/Internet Protocal)**
  - 网络层：**IP, IPv4, IPv6**
  - 传输层：**TCP(Transmission Control Protocol), UDP(User Datagram Protocol)**
  - 应用层：
    - **HTTP**
    - **FTP**
    - **SMTP**
    - **Telnet**
    - **DNS**
    - **…**
- **IP地址和域名**

World Wide Web (WWW)

- Core Components
  - Servers
    - Store files and execute remote commands
  - Browsers (i.e., clients)
    - Retrieve and display "pages" of content linked by hypertext
  - Networks
    - Send information back and forth upon request
- Problems
  - How to identify an object
  - How to retrieve an object
  - How to interpret an object

Semantic Parts of WWW

- URI (Uniform Resource Identifier)
  - protocol://hostname:port/directory/object
    - http://www.cs.iastate.edu/index.html
    - ftp://popeye.cs.iastate.edu/welcome.txt
    - https://finance.yahoo.com/q/cq?s=ibm&d=v1
  - Implementation: extend hierarchical namespace to include
    - anything in a file system
    - server side processing
- HTTP (Hyper Text Transfer Protocol)
  - An application protocol for information sending/receiving
- HTML (Hypertext Markup Language)
  - An language specification used to interpret the information receiving from server

HTTP Properties

- **Request-response exchange**
  - Server runs over TCP, Port 80
  - Client sends HTTP requests and gets responses from server
  - Synchronous request/reply protocol
- **Stateless**
  - No state is maintained by clients or servers across requests and responses
  - Each pair of request and response is treated as an independent message exchange
- **Resource metadata**
  - Information about resources are often included in web transfers and can be used in several ways

HTTP Commands

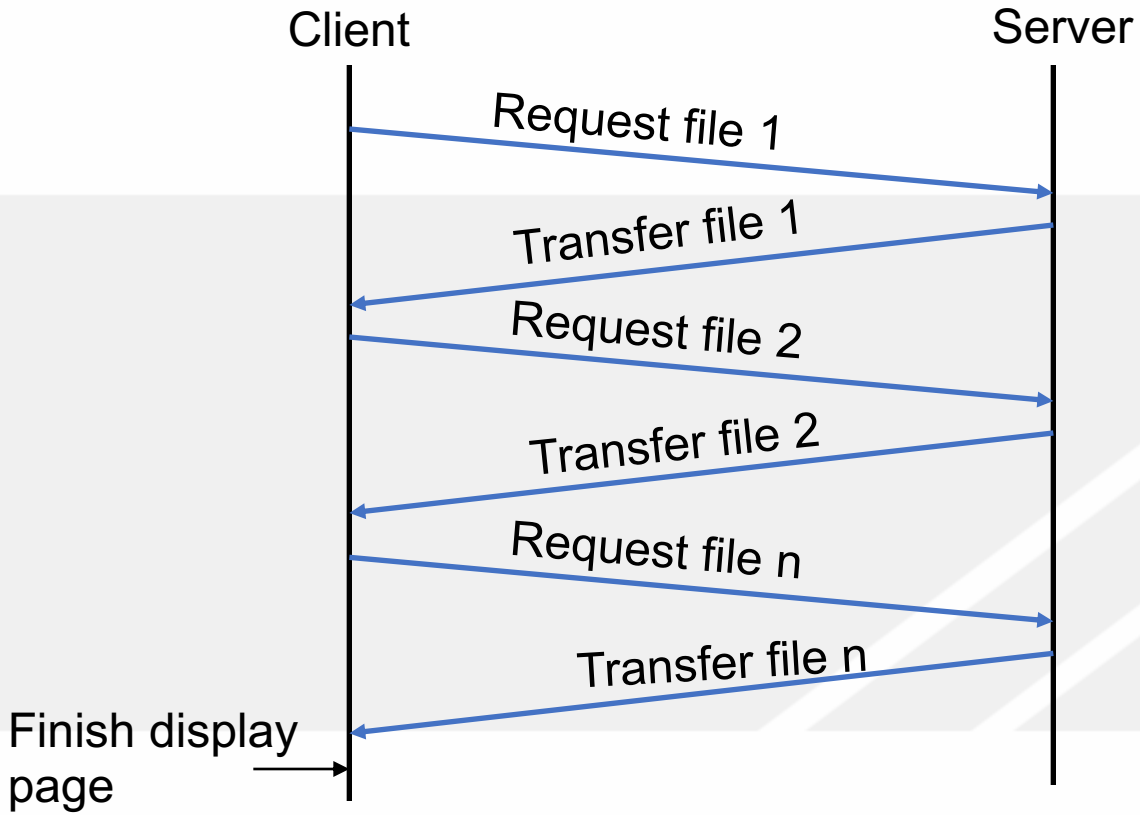
- **GET**
  - Transfer resource from given URL
- **HEAD**
  - Get resource metadata (headers) only
- **PUT**
  - Store/modify resource under a given URL
- **DELETE**
  - Remove resource
- **POST**
  - Provide input for a process identified by the given URL (usually used to post CGI parameters)

HTTP Request Processing

- The client
  1. Contact its local DNS to find out the IP address of www.cs.iastate.edu
  2. Initiate a TCP connection on port 80
  3. Send the get request via the established socket

     GET /index.html HTTP/1.0
- The server
  4. Send its response containing the required file
  5. Tell TCP to terminate connection
- The browser
  6. Parse the file and display it accordingly
  7. Repeat the same steps in the presence of any embedded objects

HTTP/1.0 Example



Response Code of HTTP 1.0

- 2xx success
- 3xx redirection
- 4xx client error in request
- 5xx server error; can’t satisfy the request

Server Response

```
HTTP/1.0 200 OK
Content-Type: text/html
Content-Length: 1234
Last-Modified: Mon, 19 Nov 2001 15:31:20 GMT
<HTML>
<HEAD>
<TITLE>CS Home Page</TITLE>
</HEAD>
…
</BODY>
</HTML>
```

HTTP/1.0 Caching

- A modifier to the GET request:
  - If-modified-since – return a "not modified" response if resource was not modified since specified time
- A response header:
  - Expires – specify to the client for how long it is safe to cache the resource
- A request directive:
  - No-cache – ignore all caches and get resource directly from server

Issues with HTTP/1.0

- Each resource requires a new connection
  - Large number of embedded objects in a web page
  - Many short lived connections
- Serial vs. parallel connections
  - Serial connection downloads one object at a time (e.g., MOSAIC) causing long latency to display a whole page
  - Parallel connection (e.g., NETSCAPE) opens several connections (typically 4) contributing to network congestion
- HTTP uses TCP as the transport protocol
  - TCP is not optimized for the typical short-lived connections
  - Most Internet traffic fit in 10 packets (overhead: 7 out of 17)
    - Too slow for small object
    - May never exit slow-start phase

Highlights of HTTP/1.1

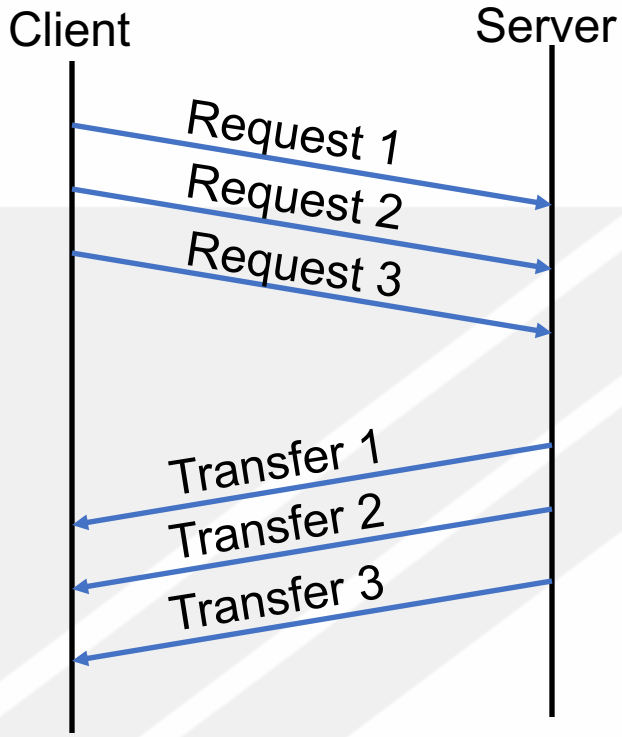
- Persistent connections
- Pipelined requests/responses
- Support for virtual hosting
- More explicit support on caching
- Internet Caching Protocol (ICP)
- Content negotiation/adaptation
- Range Request

Persistent Connections

- The basic idea was
  - reducing the number of TCP connections opened and closed
  - reducing TCP connection costs
  - reducing latency by avoiding multiple TCP slow-starts
  - avoid bandwidth wastage and reducing overall congestion
    - A longer TCP connection knows better about networking condition
- New GET methods
  - GETALL
  - GETLIST

Pipelined Requests/Responses

- Buffer requests and responses to reduce the number of packets
- Multiple requests can be contained in one TCP segment
- Note: order of responses has to be maintained
- Question: why not just send the embedded objects right away without being asked?

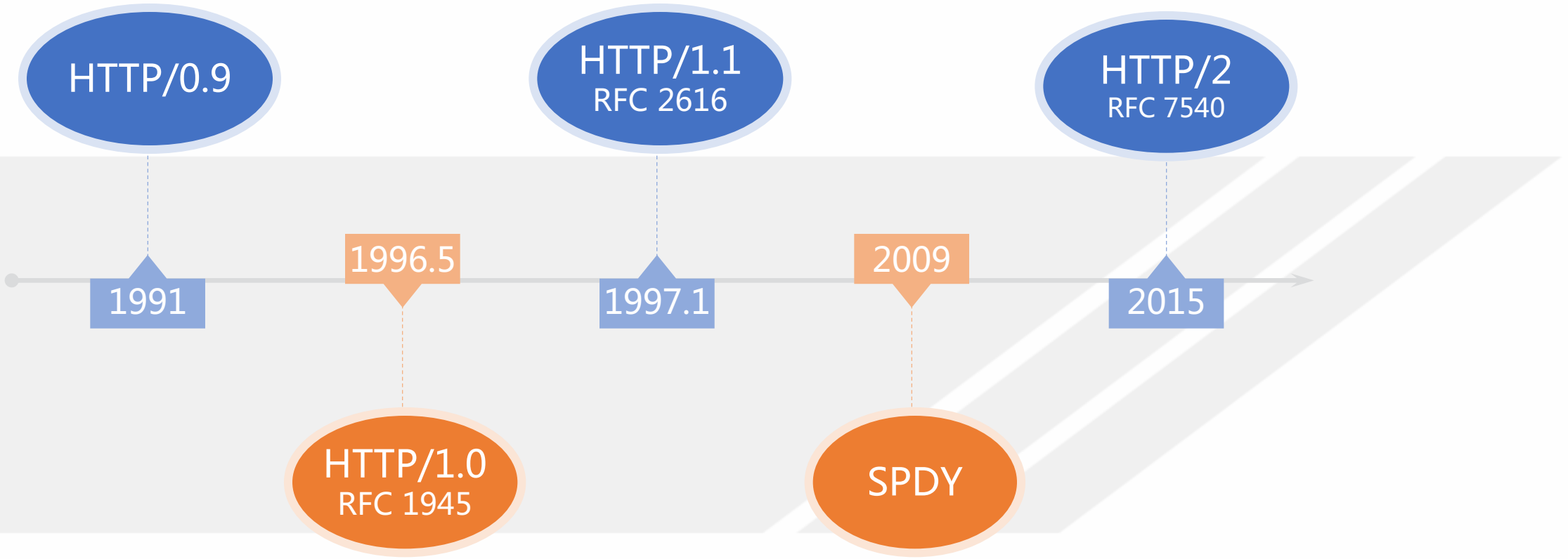


Support for Virtual Hosting

- **Problem – outsourcing web content to some company**
  - http://www.hostmany.com/A ⇔ http://www.A.com
  - http://www.hostmany.com/B ⇔ http://www.B.com
- **In HTTP/1.0, a request for http://www.A.com/index.html has in its header only:**
  - **GET /index.html HTTP/1.0**
- **It is not possible to run two web servers at the same IP address, because GET is ambiguous**
- **HTTP/1.1 addresses this by adding "Host" header**

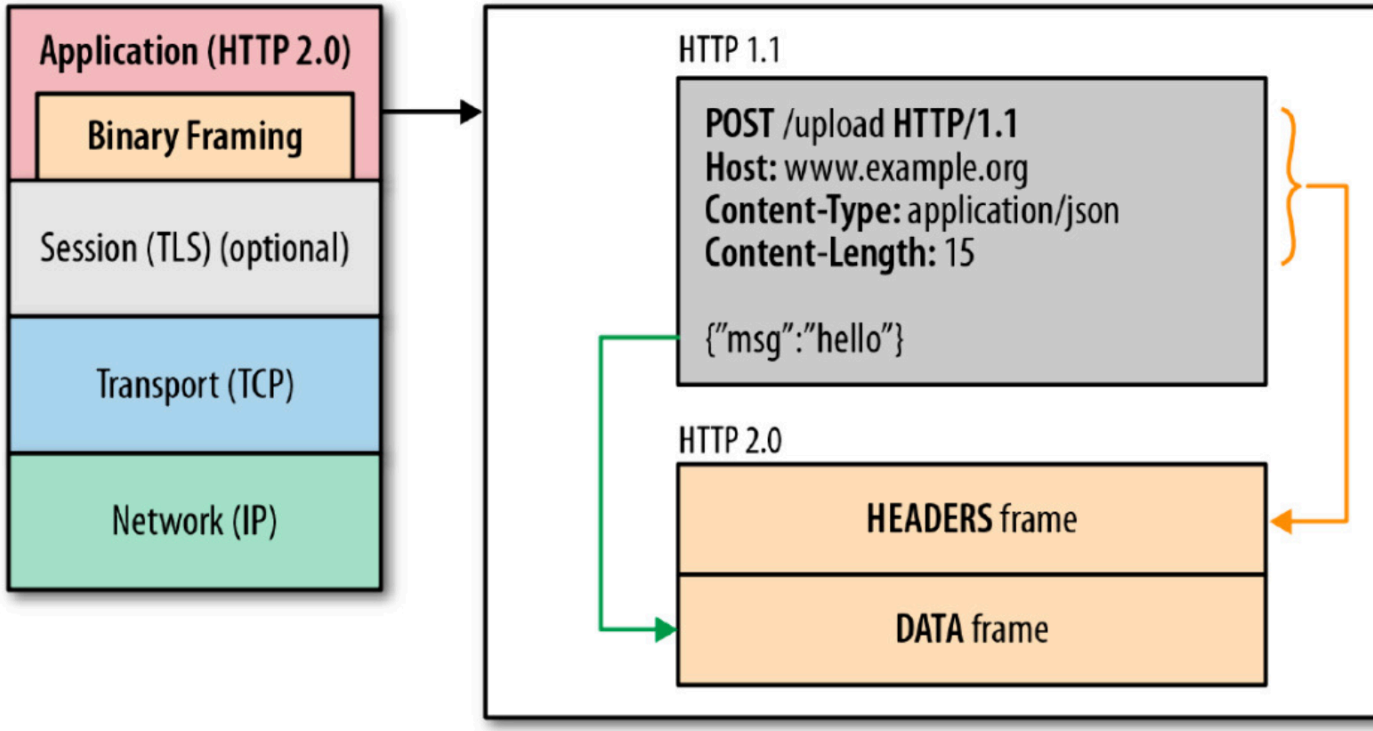
GET /index.html HTTP/1.1
Host: www.A.com

HTTP 2演进

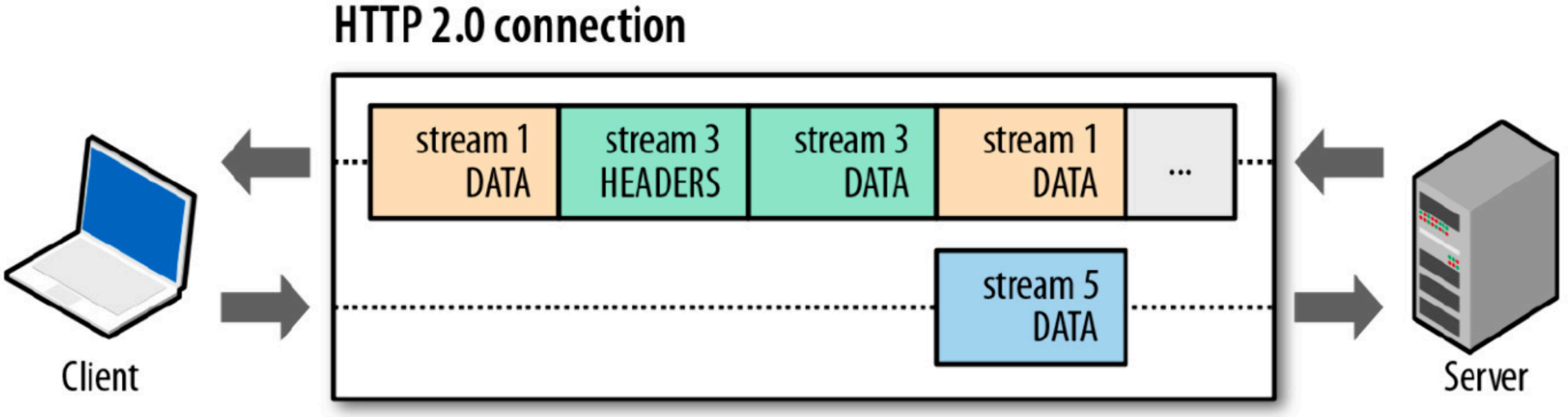

HTTP/0.9
HTTP/1.1
RFC 2616
HTTP/2
RFC 7540
1991
1996.5
1997.1
2009
2015
HTTP/1.0
RFC 1945
SPDY

HTTP 2.0

- Binary instead of textual
- Fully multiplexed instead of ordered and blocking
- Use one connection for parallelism
- Uses header compression to reduce overhead
- Allows servers to “push” responses proactively into client caches

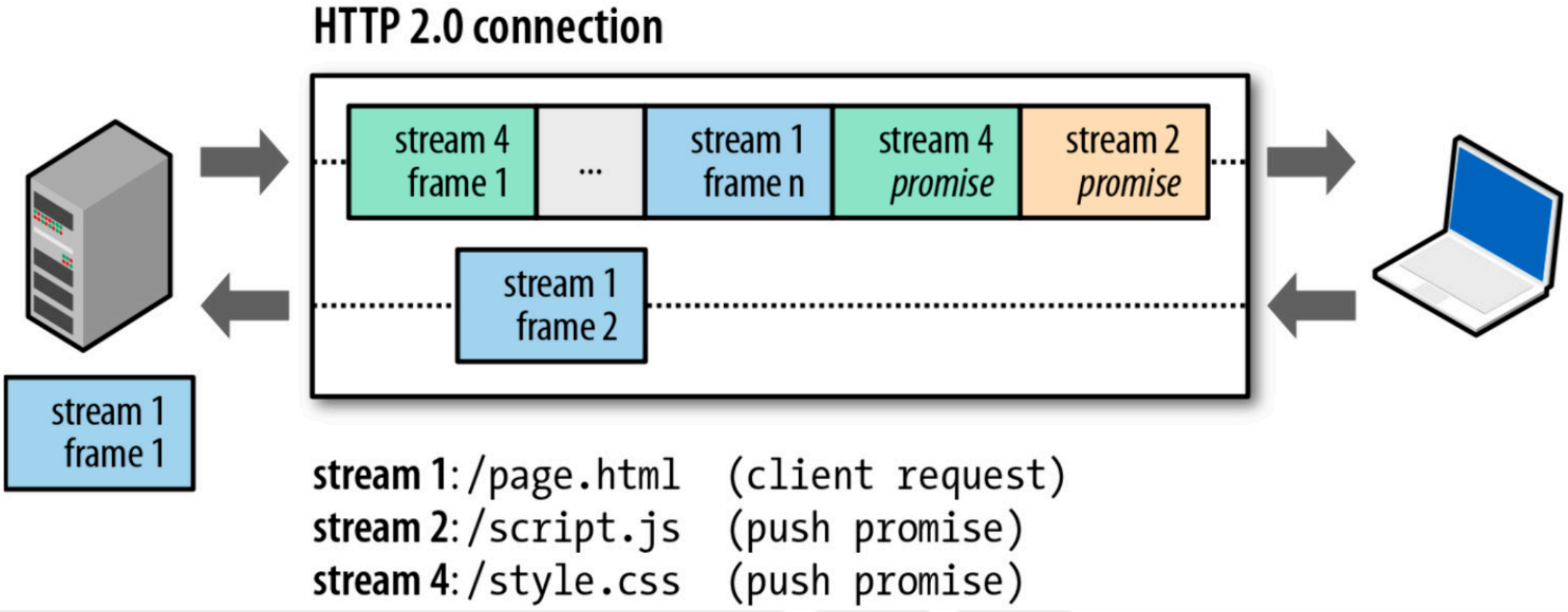


HTTP 2 Data Flow



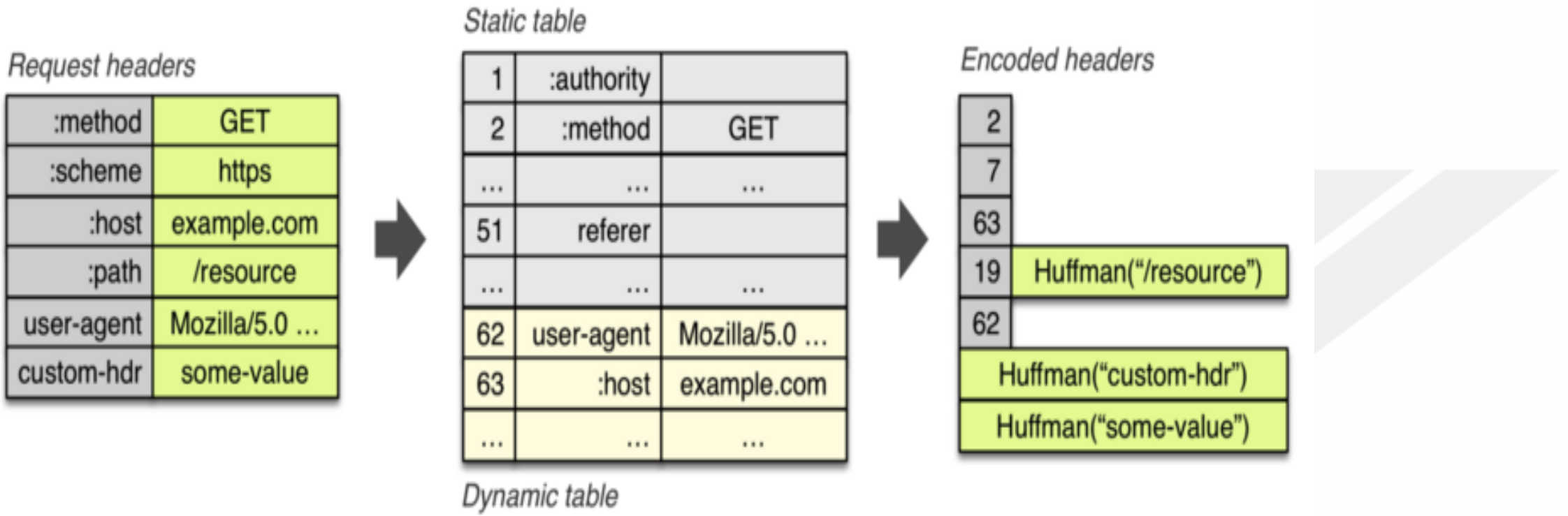
- **Streams are multiplexed by splitting communication into frames**
  - **Frames are sent over single TCP connection**
- **Frames are interleaved**
  - **Frames are prioritized**
  - **Frames are flow controlled**

HTTP 2 Push

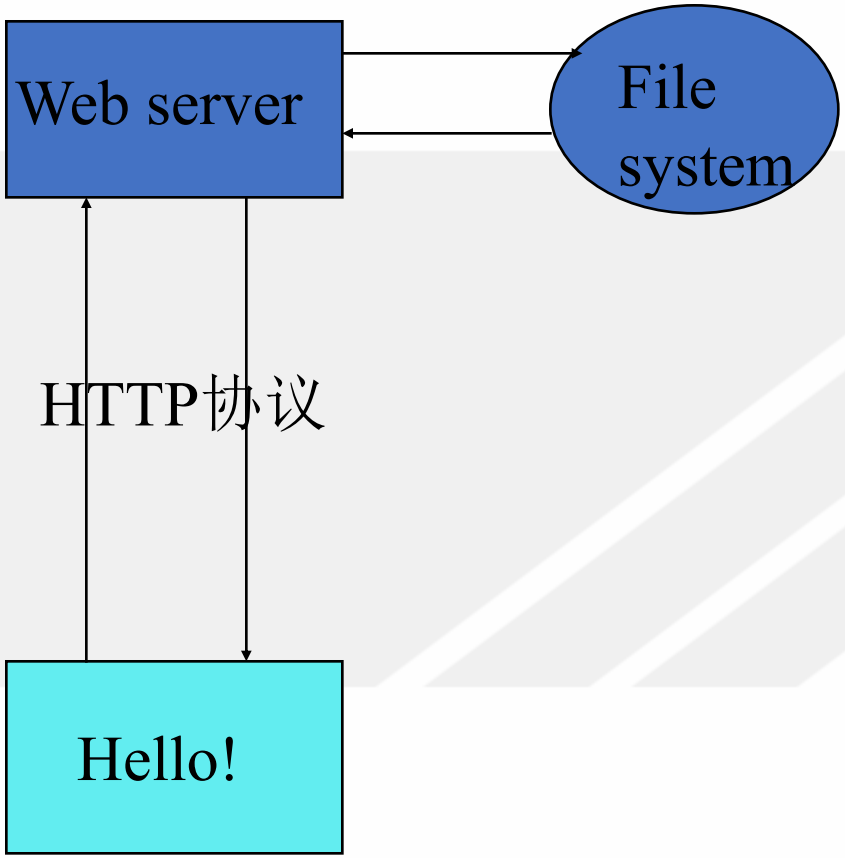


- **HTTP 2 Server Push is cacheable**
- **Client may cancel by sending RST_STREAM frame**

HTTP 2 Header Compression



基本Web服务器



HTML

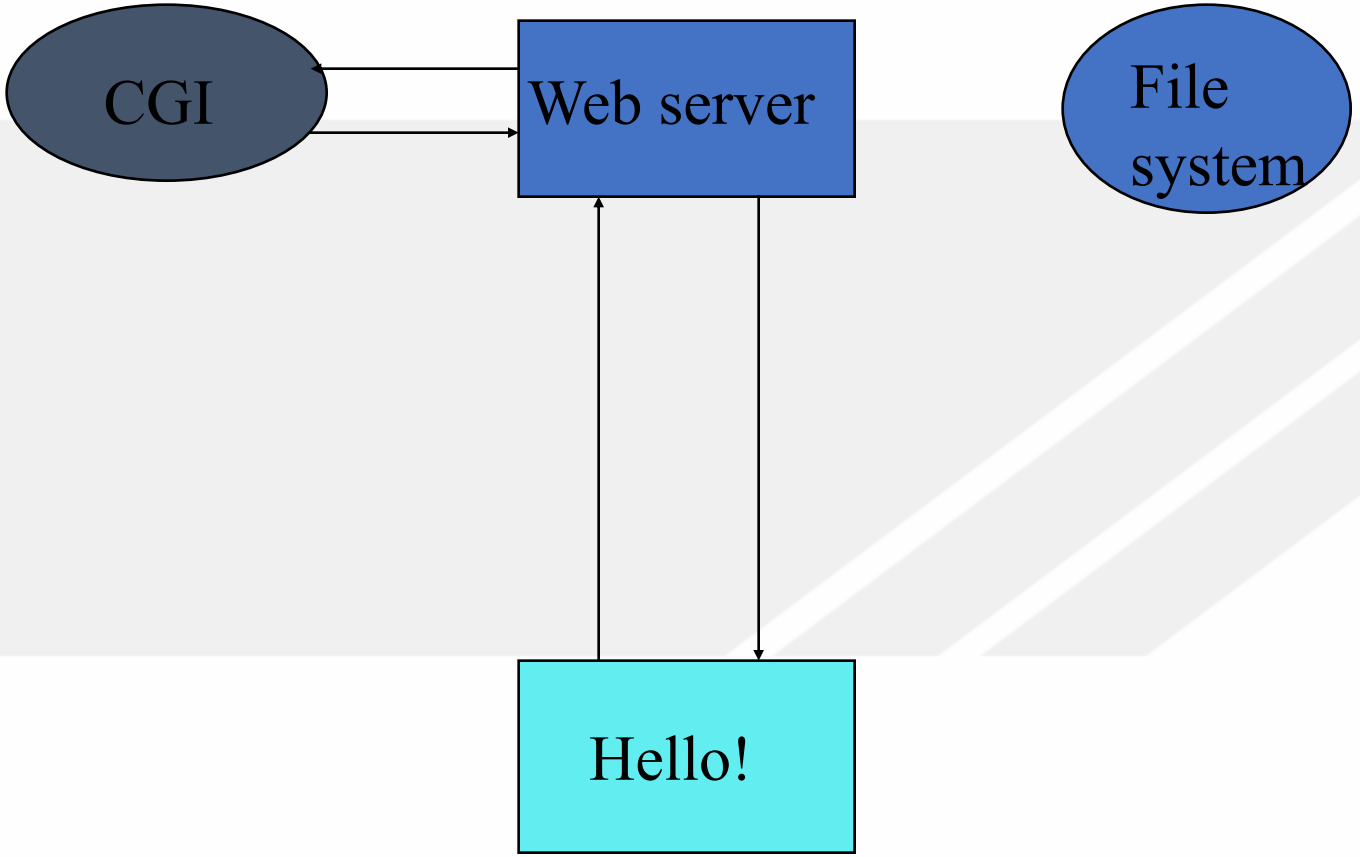
- Hyper-Text Markup Language
- Hyper-Text Transport Protocol
- Cascading Style Sheet

- XML

Client side script/program

- JavaScript
- VBScript
- Java Applet
- ActiveX

CGI(Common Gateway Interface)



Better CGI

- FastCGI
- Java Servlet

Server side module

- NSAPI
- ISAPI
- Apache: mod_perl

Server side script

- Server Side Include
- PHP
- Active Server Pages
- Java Server Pages

合适的技术选择

- **Client:**
  - **HTML/CSS**
  - **JavaScript**
- **Server**
  - **PHP**
  - **JSP**
  - **Java Servlet**
  - **ASP.NET**

Web服务器

- **IIS**
  - 安装
  - 配置
  - 运行控制
- **Nginx**
  - 安装
  - 配置
  - 运行控制
- 其他
  - **Java EE应用服务器**