HTTP

Robert Grimm
New York University
Linux servers running JDK 1.4.1
  - class[20-25].scs.cs.nyu.edu
  - You should have accounts within a week
Assignment 1, the HTTP client, is due 9/23/03 before class!
We’ll talk about groups at the end of class…
**HTTP/1.0 Performance**

- Main message: Performance is suboptimal
  - Interaction latency
  - Server scalability

- Side message: tcpdump’s output is not illustrative
Interaction between HTTP and TCP

- Three-way handshake
  - SYN, SYN+ACK, ACK

- Slow start
  - Open congestion window for each successfully transmitted packet
  - Then send successive packets without waiting for acknowledgements

- Nagle’s algorithm
  - Delay transmission to collect additional data
    - telnet, rlogin

- TIME_WAIT state
HTTP/1.1 to the Rescue!

- Extensibility
- Caching
- Bandwidth optimization
- Network connection management
- Message transmission
- Internet address conservation
- Error notification
- Security, integrity, and authentication
- Content negotiation
HTTP/1.1 to the Rescue!

- Extensibility
- Caching
- Bandwidth optimization
- Network connection management
- Message transmission
- Internet address conservation
- Error notification
- Security, integrity, and authentication
- Content negotiation
HTTP/1.1 Extensibility

- Goal: full backwards compatibility
  - HTTP/0.9
  - HTTP/1.0
  - HTTP/1.1 draft implementations
- Via header
  - Collect end-to-end path information
- OPTIONS method
  - Query for supported features
- Upgrade header
  - Switch to different protocol
HTTP/1.1 Caching

- Reduces latency for cached resources
- Reduces bandwidth consumption
- Indirectly reduces latency for uncached resources
- Indirectly reduces load on origin servers
HTTP/1.1 Caching

- Goal: semantic transparency
- HTTP/1.0
  - Based on timestamps
    - Limited resolution (1 second)
    - Unsynchronized clocks
- HTTP/1.1
  - Based on relative times
    - max-age in Cache-Control header
  - Based on opaque tokens
    - ETag
    - If-None-Match, If-Match
HTTP/1.1 Bandwidth Optimizations

- Goal: conserve bandwidth
- Range requests
  - Only transmit necessary data
- Expect and 100 (Continue)
  - Ask for permission before transmitting large resources
- Compression
  - Use more compact on-the-wire representation
    - Content-Encoding: end-to-end
    - Transfer-Encoding: hop-by-hop
HTTP/1.1 Network Connection Management

- Goal: be more friendly to TCP
- Connection header
  - Declare what headers are hop-by-hop
- Persistent connections
  - Send many request/response interactions over the same TCP connection
- Pipelining
  - Do not wait for response before sending next request
HTTP/1.1 Message Transmission

- Goal: reduce buffering requirements
  - Content-Length header requires resource size
- Chunked transfer-coding
  - Break resource into many pieces
- Trailers
  - Send headers after resources
    - Content-MD5
Goal: turn one server into many servers
  - Treat DNS-to-IP mapping as many-to-one
    - IPv4 addresses are scarce, aren’t they?

Host header
  - Declare DNS name of targeted host
  - Though, HTTP/1.0 allows for absolute URLs
    - Interact with proxies

Unintended benefit (?)
  - Amortize management effort over many sites
HTTP/1.1 Error Notification

- Goal: support advisory information in addition to status code
- Warning header
  - Expose status of caches
    - Disconnected cache
- New status codes
  - 24 in all, including 100 (Continue), 206 (Partial content), 300 (Multiple choices)
HTTP/1.1 Authentication

- Goal: authorize users
- Based on WWW-Authenticate, Authorization headers
- HTTP/1.0: Basic authentication
  - User name, password in the clear
- HTTP/1.1: Digest authentication
  - Based on shared secret (user name, password pair)
  - Sends cryptographically secure checksum (MD5)
    - Username, password, nonce, HTTP method, URL
- HTTP/1.1: Proxy authentication
HTTP/1.1 Privacy

- Goal: respect privacy of users
- Rules for when to use Referer [sic] header
- Rules for how to use cookies (RFC 2965)
  - HTTP is stateless, yet we want state
  - Cookies to the rescue
    - Collections of name/value pairs
    - Issued by server on first access
    - Returned by client on subsequent accesses
Goal: support different variant of same resource

Server-driven negotiation

- Client declares preferences, server chooses
  - Different headers to distinguish properties
    - Media types, character sets, content encodings, human languages
  - Quality values (0.000-1.000) to weigh alternatives
  - Wildcards to express indifference
    - Accept: audio/*; q=0.2, audio/basic
HTTP/1.1 – Some Issues

- How to name a resource?
  - HTTP/1.0: URL
  - HTTP/1.1: URL + headers
    - Vary header to list relevant headers

- End-to-end or hop-by-hop?
  - Caches should be semantically transparent
  - Yet, they may require user interaction
    - Proxy authentication
    - Advisory information

- Stateless or stateful?
  - Cookies are a separate RFC, yet widely used
HTTP/1.1 to the Rescue

- What do you think?
HTTP/1.1 Performance

- Main message
  - Pipelined persistent connections work
Main message

- Pipelined persistent connections work
  ... but only if you are careful
HTTP/1.1 Performance
Experimental Methodology

- Synthesized web site ("Microscape")
  - One HTML page (42 KB)
  - 42 inlined GIF images (total of 125 KB)
- Three connections
  - LAN – 10 MBit Ethernet
  - WAN – MIT LCS to LBL
  - PPP – 28.8k modem to MIT LCS
- Software
  - Server: Jigsaw and Apache
  - Client: libwww
HTTP/1.1 Performance Experimental Methodology

- Hardware
  - Sun SPARC Ultra-1 server running Solaris
  - Digital AlphaStation and Windows NT clients
HTTP/1.1 Performance Tuning

- Initial test results
  - HTTP/1.1 reduces number of sockets and packets
  - But significantly increases latency
- Buffer management is key
  - Flush after initial request
    - Send as quickly as possible to receive HTML
  - But buffer requests for inline images
    - Pipeline as much as possible to utilize available bandwidth
- Also need to manage connections carefully
  - Only half-close connection
    - Close server sending side, but still accept client data
HTTP/1.1 Performance Results

- HTTP/1.0 with concurrent connections is slow
- HTTP/1.1 without pipelining is even slower
- HTTP/1.1 with pipelining is faster
- HTTP/1.1 with pipelining and compression is even faster, especially on low bandwidth links

- It’s hard to read results presented in (8) tables
HTTP/1.1 Performance
The Need for Compression

- Goal: fully utilize TCP connection
  - Complete first request quickly to create more work
  - Pipeline additional requests to exploit bandwidth
- Compression can help fit more HTML into a single packet
  - Tag case matters

- Are there better solutions?
Advantages of CSS
- Reuse over many resources
- Elimination of image abuse
  - Symbols, spacers

PNG vs. GIF
- More compact representation
Questions, Discussion
PS: Groups

- **ZZ Top**: Chris Natail, Ajay Haridasani, Zelin Lu, Luigi Zoso
- **Mangoes**: Mrudang Rawal, Sri Prasad Tadimalla, Zeno Lee, MaoJen Hsu
- **Loki**: Ken Lin, Peter Liu, Jonathan Miller, Brad Wellington
- **Optimus**: Dmitriy Mindich, Alexandre Lounev, Oleg Shevelenko, Natalia Gorelik, ???
- Jian Kang, Sajid Raza, Juan Cheng, Ihsin Lee