CSC 458
HTTP + Web
DNS

Stefan Saroiu
http://www.cs.toronto.edu/syslab/courses/csc458

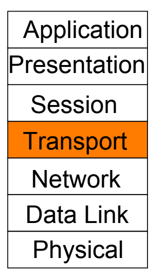
University of Toronto at Mississauga

Exercise in Democracy

- Old scheme for project grades:
  - 4 projects, 9% each
- New scheme for project grades:
  - First 3 projects, 12% each
  - 4th project, 0%
    - Effectively canceled

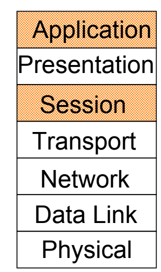
Last Time …

- The Transport Layer
- Focus
  - How does TCP <u>share bandwidth</u>?
- Topics
  - AIMD
  - Slow Start
  - Fast Retransmit / Fast Recovery

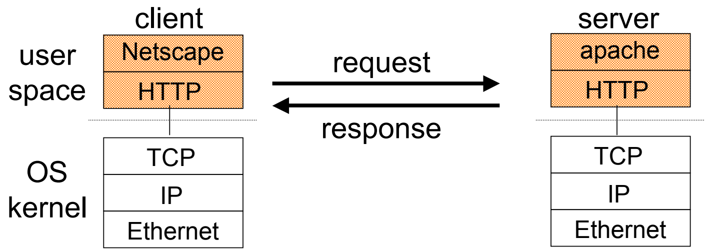


This Lecture

- HTTP and the Web (but not HTML)
- Focus
  - How do Web transfers work?
- Topics
  - HTTP, HTTP1.1
  - Performance Improvements
    - Protocol Latency
    - Caching



Web Protocol Stacks



- To view the URL http://server/page.html the client makes a TCP connection to port 80 of the server, by it's IP address, sends the HTTP request, receives the HTML for page.html as the response, repeats the process for inline images, and displays it.

HTTP Request/Response

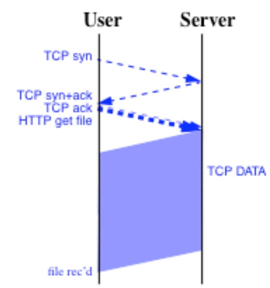
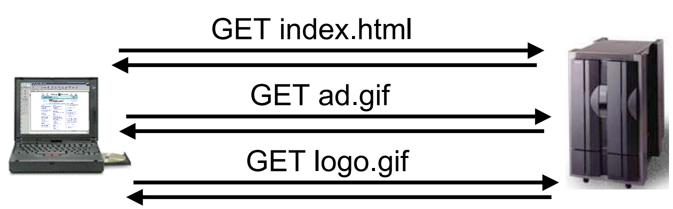


FIGURE 3. HTTP File Transfer

1 RTT channel OPEN
0.5 RTT send request
0.5 RTT file starts to arrive
Ftrans time to transmit the file
----
2 RTT + Ftrans
= time to get a file in HTTP

Simple HTTP 1.0



- HTTP is a tiny, text-based language
- The GET method requests an object
- There are HTTP headers, like "Content-Length:", etc.
- Try "telnet server 80" then "GET index.html HTTP/1.0"
  – Other methods: POST, HEAD,… google for details

HTTP Request/Response in Action

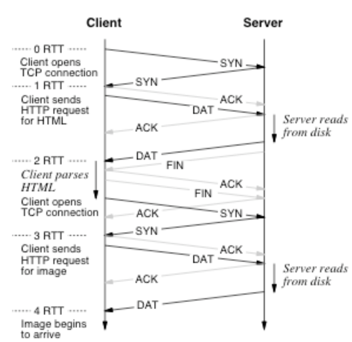


Figure 3-1: Packet exchanges and round-trip times for HTTP

Problem is that:
- Web pages are made up of many files
  - Most are very small (< 10k)
- files are mapped to connections

For each file
- Setup/Teardown
  - Time-Wait table bloat
- 2RTT "first byte" latency
- Slow Start+ AIMD Congestion Avoidance

The goals of HTTP and TCP protocols are not aligned.
- Implications

TCP Behavior for Short Connections Over Slow Networks

RTT=70ms

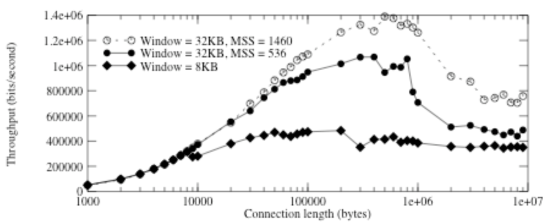


Figure 3-2: Throughput vs. connection length, RTT = 70 msec

Figure 3-2 shows that, in the remote case, using a TCP connection to transfer only 2 Kbytes results in a throughput less than 10% of best-case value. Even a 20 Kbyte transfer achieves only about 50% of the throughput available with a reasonable window size. This reduced throughput translates into increased latency for document retrieval. The figure also shows that, for this 70 msec RTT, use of too small a window size limits the throughput no matter how many bytes are transferred.

It's the RTT

RTT=1ms

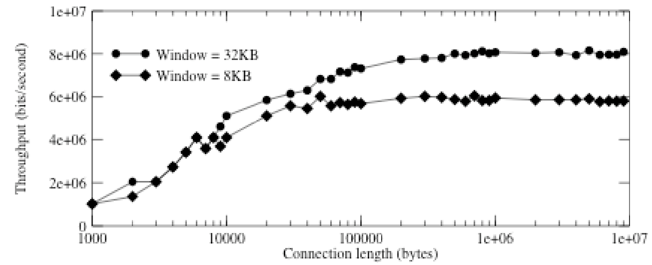


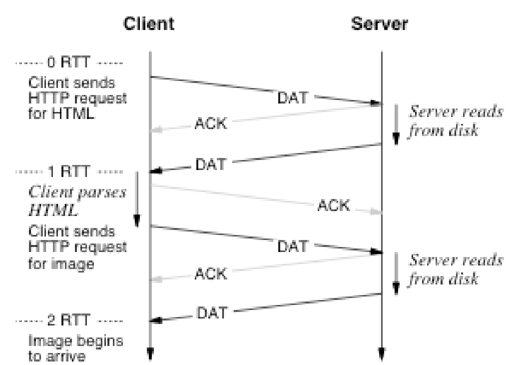
Figure 3-3: Throughput vs. connection length, RTT near 0 msec

No slow start here (ULTRIX LAN)

HTTP1.1: Persistent Connections

- Bright Idea: Use one TCP connection for multiple page downloads (or just HTTP methods)
- Q: What are the advantages?
- Q: What are the disadvantages?
  - *Application layer multiplexing*

HTTP/1.1



Effect of Persistent HTTP

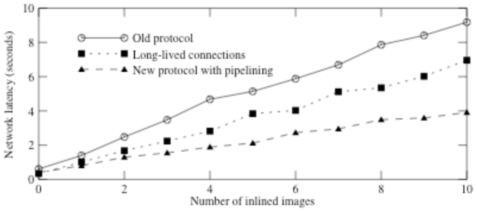


Figure 6-1: Latencies for a remote server, image size = 2544 bytes

Image size=2544

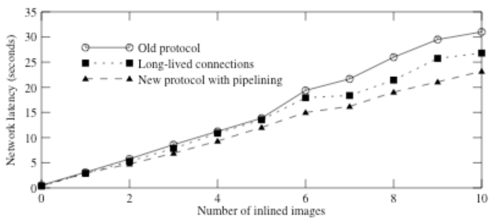


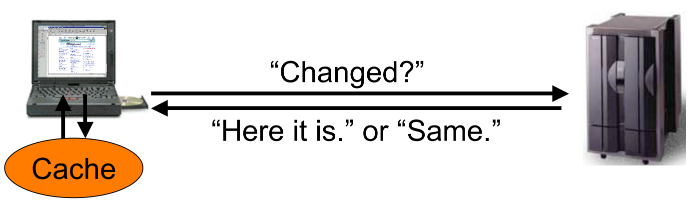
Figure 6-2: Latencies for a remote server, image size = 45566 bytes

Image size=45566

Caching

- *It is faster and cheaper to get data that is closer to here than closer to there.*
- *"There" is the origin server. 2-5 RTT*
- *"Here" can be:*
  - *Local browser cache (file system) (1-10ms)*
  - *Client-side proxy (institutional proxy) (10-50)*
  - *Content-distribution network (CDN -- "cloud" proxies) (50-100)*
  - *Server-side proxy (reverse proxy @ origin server) (2-5RTT)*

Browser Caches

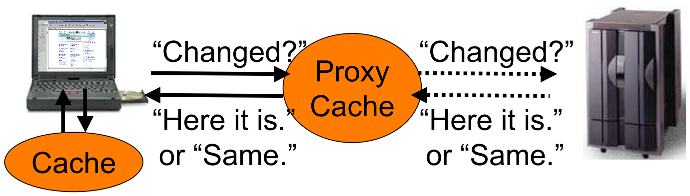


- Bigger win: avoid repeated transfers of the same page
- Check local browser cache to see if we have the page
- GET with If-Modified-Since makes sure it's up-to-date
- Q: What are the advantages and disadvantages?

Consistency and Caching Directives

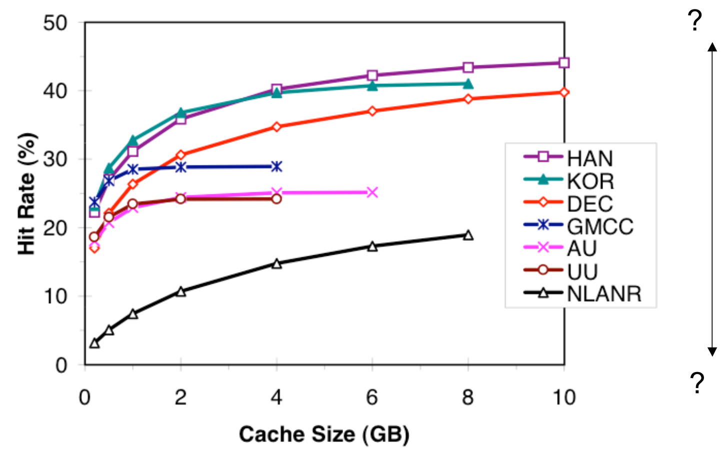
- Key issue is knowing when cached data is fresh/stale
  - Otherwise many connections or the risk of staleness
- Browsers typically use heuristics
  - To reduce server connections and hence realize benefits
  - Check freshness once a "session" with GET If-Modified-Since and then assume it's fresh the rest of the time
  - Possible to have inconsistent data.
- Caching directives provide hints
  - Expires: header is basically a time-to-live
  - Also indicate whether page is cacheable or not

Proxy Caches



- Insert further levels of caching for greater gain
- Share proxy caches between many users (not shown)
  - If I haven’t downloaded it recently, maybe you have
- Your browser has built-in support for this

Proxy Cache Effectiveness



Hit Rate Follows Request Rate

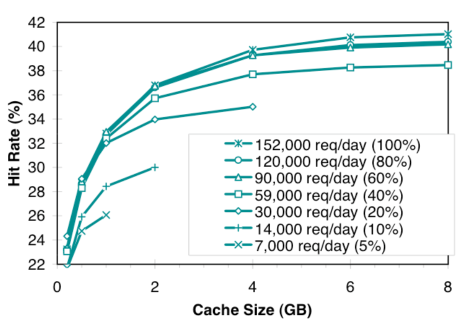


Figure 3: Cache hit rate for KOR as a function of cache size for a range of request rates.

Sharing, Not Locality, Drives Effectiveness

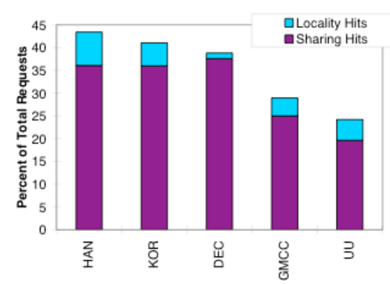


Figure 9: Hit rate divided into hits due to sharing and due to locality of a single client.

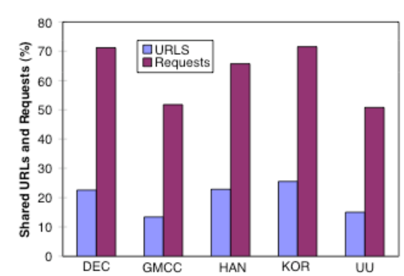
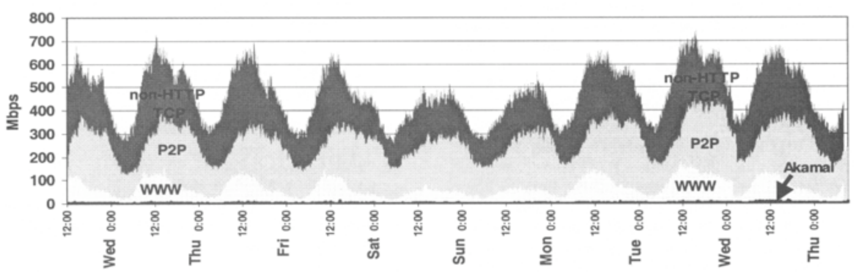


Figure 10: The percent of a total URLs in a trace requested by two or more clients and the percent of total requests to these shared objects.

The Trends

- HTTP Objects are getting bigger
- But Less important



Next Steps?

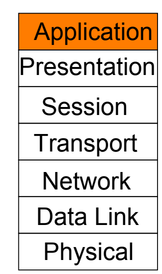
- Different types of content (streaming media, XML)
- Content Delivery Networks (caching alternative)
- Security (for all those purchases)

Key Concepts

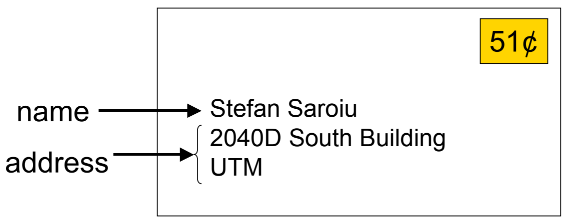
- HTTP and the Web is just a shim on top of TCP
  - Sufficient and enabled rapid adoption
  - Many “scalability” and performance issues now important

This Lecture

- Naming
- Focus
  - How do we name hosts etc.?
- Topics
  - Domain Name System (DNS)
  - Email/URLs



Names and Addresses



- Names are identifiers for objects/services (high level)
- Addresses are locators for objects/services (low level)
- Binding is the process of associating a name with an address
- Resolution is the process of looking up an address given a name
- But, addresses are really lower-level names; many levels used

Naming in Systems

- Ubiquitous
  - Files in filesystem, processes in OS, pages on the web, …
- Decouple identifier for object/service from location
  - Hostnames provide a level of indirection for IP addresses
- Naming greatly impacts system capabilities and performance
  - Ethernet addresses are a flat 48 bits
    - flat → any address anywhere but large forwarding tables
  - IP addresses are hierarchical 32/128 bits
    - hierarchy → smaller routing tables but constrained locations

Internet Hostnames

- Hostnames are human-readable identifiers for end-systems based on an administrative hierarchy
  - cleo.slup.cs.toronto.edu is my desktop machine
- IP addresses are a fixed-length binary encoding for end-systems based on their position in the network
  - 192.12.174.140 is cleo's IP address
- Original name resolution: HOSTS.TXT
- Current name resolution: Domain Name System
- Future name resolution: ?

Original Hostname System

- When the Internet was really young …
- Flat namespace
  - Simple (host, address) pairs
- Centralized management
  - Updates via a single master file called HOSTS.TXT
  - Manually coordinated by the Network Information Center (NIC)
- Resolution process
  - Look up hostname in the HOSTS.TXT file

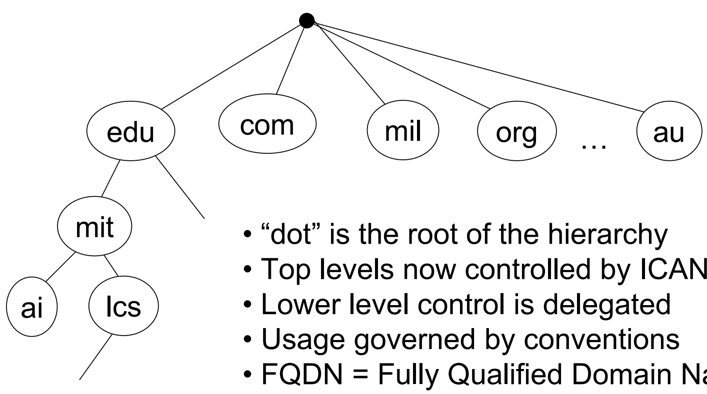
Scaling Problems

- Coordination
  - Between all users to avoid conflicts
- Inconsistencies
  - Between update and distribution of new version
- Reliability
  - Single point of failure
- Performance
  - Competition for centralized resources

Domain Name System (DNS)

- Designed by Mockapetris and Dunlap in the mid 80s
- Namespace is hierarchical
  - Allows much better scaling of data structures
  - e.g., cleo.slup.cs.toronto.edu
- Namespace is distributed
  - Decentralized administration and access
  - e.g., *.toronto.edu managed by CSC
- Resolution is by query/response
  - With replicated servers for redundancy
  - With heavy use of caching for performance

DNS Hierarchy



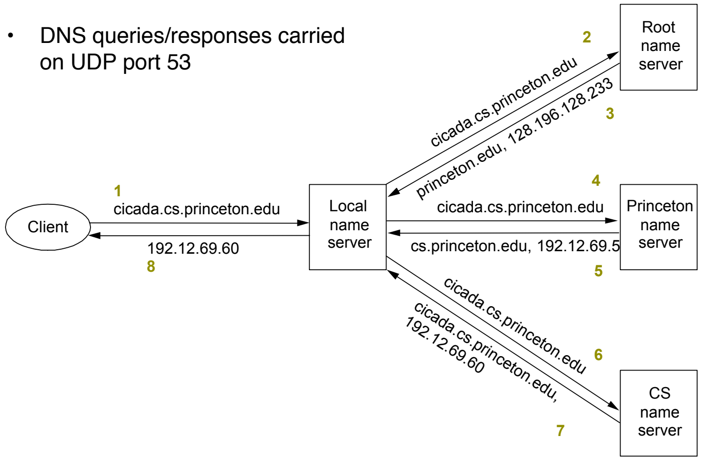
- “dot” is the root of the hierarchy
- Top levels now controlled by ICANN
- Lower level control is delegated
- Usage governed by conventions
- FQDN = Fully Qualified Domain Name

DNS Distribution

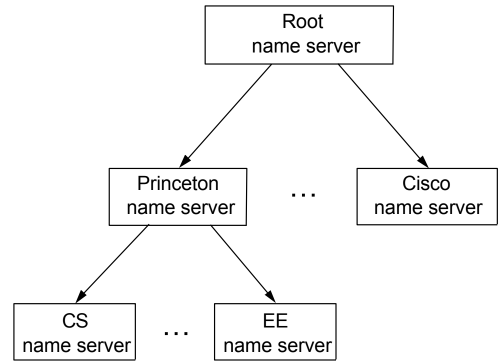
- Data managed by zones that contain resource records
  - Zone is a complete description of a portion of the namespace
  - e.g., all hosts and addresses for machines in toronto.edu with pointers to subdomains like cs.toronto.edu
- One or more nameservers manage each zone
  - Zone transfers performed between nameservers for consistency
  - Multiple nameservers provide redundancy
- Client resolvers query nameservers for specified records
  - Multiple messages may be exchanged per DNS lookup to navigate the name hierarchy (coming soon)

DNS Lookups/Resolution

- DNS queries/responses carried on UDP port 53



Hierarchy of Nameservers



Caching

- Servers and clients cache results of DNS lookups
  - Cache partial results too (e.g., server for princeton.edu)
  - Greatly improves system performance; lookups the rare case
- Cache using time-to-live (TTL) value from provider
  - higher TTL means less traffic, lower TTL means less stale info
- Negative caching is used too!
  - errors can cause repeated queries for non-existent data

DNS Bootstrapping

- Need to know IP addresses of root servers before we can make any queries
- Addresses for 13 root servers ([a-m].root-servers.net) handled via initial configuration (named.ca file)

Building on the DNS

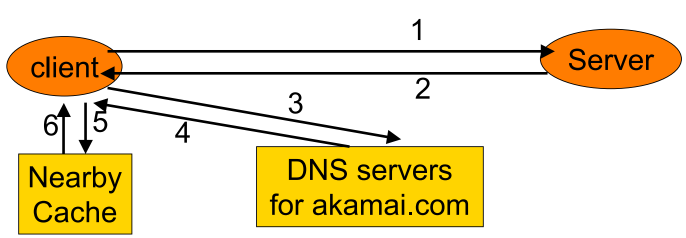
- Other naming designs leverage the DNS
- Email:
  - e.g., stefan@cs.toronto.edu is stefan in the domain cs.toronto.edu
- Uniform Resource Locators (URLs) name for Web pages
  - e.g., www.cs.toronto.edu/~stefan
  - Use domain name to identify a Web server
  - Use “/” separated string to name path to page (like files)

Future Evolution of the DNS

- Design constrains us in two major ways that are increasingly less appropriate
- Static host to IP mapping
  - What about mobility (Mobile IP) and dynamic address assignment (DHCP)
- Location-insensitive queries
  - What if I don’t care what server a Web page comes from, as long as it’s the right page?
  - e.g., a yahoo page might be replicated

Akamai

- Use the DNS to effect selection of a nearby Web cache



- Leverage separation of static/dynamic content
- Beware DNS caching

Key Concepts

- The design of names, addresses and resolution has a significant impact on system capabilities
- Hierarchy, decentralization and caching allow the DNS to scale
  - These are general techniques!