CSc 450/550
Computer Networks
Web and More

Jianping Pan

Summer 2007

Last lecture

- HTTP basics
  - the application-layer protocol for the Web
  - follow the client-server model
  - (stateless) request-reply transaction
    - HTTP request: GET / HTTP/1.0
    - HTTP response: HTTP/1.0 200 OK
  - the service expected from lower layers
    - reliable data transfer
    - normally by TCP

Today's topics

- HTTP: advanced topics
  - fit better on TCP
    - improve HTTP efficiency
  - become stateful
    - server and client can know/remember each other
  - deal with scalability
    - web caching and content delivery
- DNS name space
  - you say “www.google.com”, I say “66.102.7.104”

HTTP/TCP

- TCP connection establishment
  - client: connect(); server: accept()
- HTTP transaction
  - request: client: send(); server: recv()
  - response: server: send(); client: recv()
- TCP connection release
  - server: close(); client: close()
- Client is to retrieve the embedded objects

Q: round-trips?

Example

- http://www.a.com/index.html
  <html>
  <img src="http://www.a.com/x.gif">

  <img src="http://www.a.com/z.gif">
  </html>

Non-persistent HTTP

- One object per TCP connection
  - default behavior in HTTP/1.0
  - network cost: ~2*RTT per object
  - end-host cost: 1 socket() for each object
- Performance improvement
  - parallel/concurrent connections
    - e.g., an HTML page with 2 embedded objects

Q: cost reduced?

Persistent HTTP

- Multiple objects through a TCP connection
  - between the **same** client/server
  - default behavior in HTTP/1.1
    - for HTTP/1.0: Connection: Keep-Alive
  - network cost: ~RTT per object for many objects
  - to disable/close: Connection: Close
    - client/server Keep-Alive timeout
- Performance improvement: pipelining
  - ~2*RTT for all objects from the same server

Q: pros and cons?

Client-server states

- HTTP itself is stateless
  - request-reply **transactions**
- Many applications require states
  - cookie issued by server's *backend* servers
    - HTTP response header: Set-Cookie
  - client can *choose* to keep cookie
  - client represents cookie in subsequent requests
    - HTTP request header: Cookie

Tracking client?!

- Between web servers
  - HTTP request header: Referer
  - Referrer!
  - referrer spamming and referrer spoofing
  - de-referring
- User security and privacy
  - e.g., cookie theft, cookie poisoning, web bug
  - browsing anonymizing

Web caching

- Scalability issues with the client-server model
  - one server, many clients, concurrent requests
  - server load
  - network traffic
- Web caching: aggregate user requests
  - by caching responses to previous requests
  - explore locality: same requests may occur soon!
  - reduce response time and traffic load

Consistency control

- Objects retrieved from the cache
  - may be staled due to updates at origin servers
- Strong consistency
  - HTTP request header: If-Modified-Since
  - HTTP response: HTTP/1.0 304 Not Modified
    - reduce traffic load if hit
- Weak consistency
  - time-to-live (TTL)
    - reduce traffic load and response time if "hit"

Content delivery

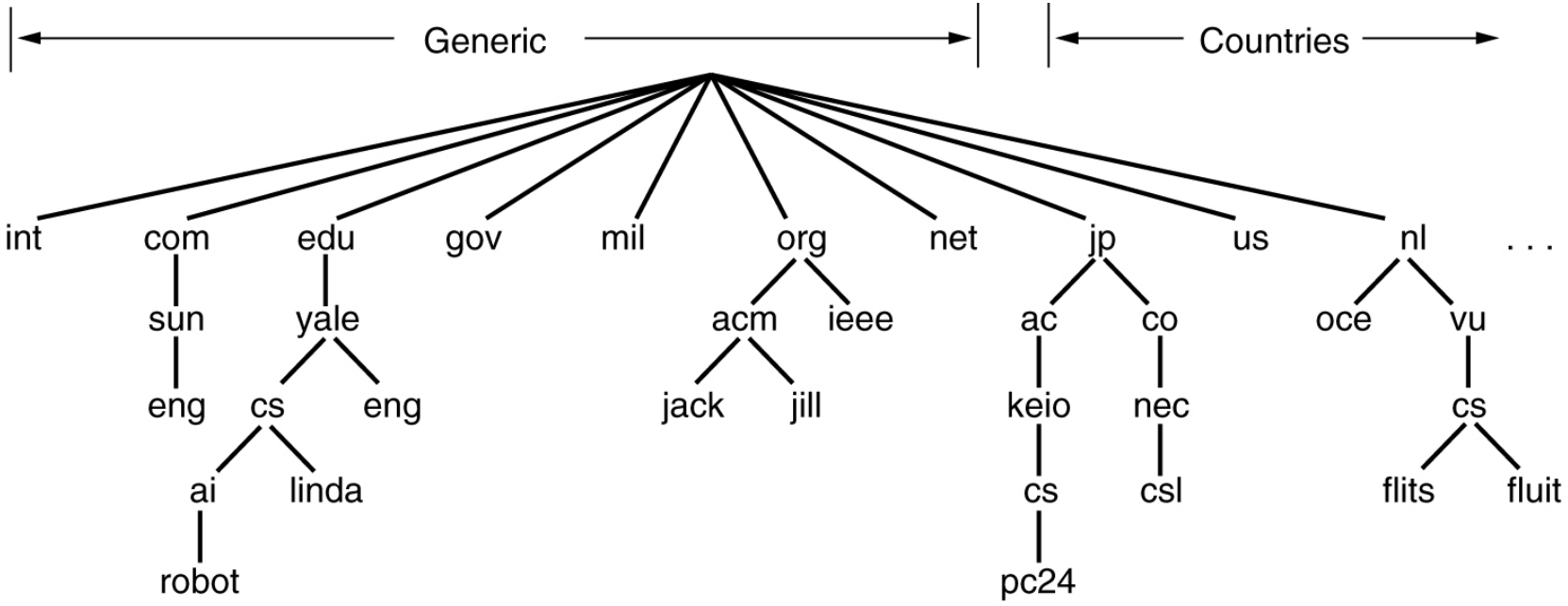
- Move content closer to end users
  - content distribution
- Redirect users to closer servers
  - information retrieval
  - how do they do that?
    - e.g., Akamai
    - DNS-based server selection

Domain names

- You say “www.google.com”
  - host name: www
  - domain name: google.com
- I say “66.102.7.104”
  - IPv4 address (4 bytes)
- Name-address mapping
  - initially, centralized hosts.txt
  - doesn’t scale!

Name space

- Hierarchical, distributed
  - gTLD: generic top-level domain
  - ccTLD: country-code top-level domain



Name hierarchy

- www.cs.uvic.ca.
  - root: .
  - ccTLD: ca.
  - UVic: uvic.ca.
  - CS@UVic: cs.uvic.ca.
- address hierarchy: 142.104.100.111
  - UVicNet: 142.104
  - EngrNet: 142.104.96~127
    - moving to 142.104.64~95

DNS: client view

- Local DNS resolver: gethostbyname()
  - name resolution configuration: /etc/host.conf
    - order hosts,bind
  - static name resolution: /etc/hosts
    - 1.2.3.4 nameserver
  - info about local DNS server: /etc/resolv.conf
    - nameserver 1.2.3.4
- Local DNS server
  - DNS “proxy”

This lecture

- HTTP
  - (non-)persistence, pipelining, cookies, referrers
  - web caching and content delivery
- DNS name space
- Explore further
  - How do your favorite web browsers and servers support advanced HTTP features?
  - **bonus** features in your P1?

Next lecture

- May 28: DNS
  - DNS server hierarchy
  - DNS resolution process
  - DNS-based server selection
- May 31: 1st in-class midterm exam
  - cover up to May 28 inclusive