CSC2231: DNS

http://www.cs.toronto.edu/~stefan/courses/csc2231/05au

Stefan Saroiu

Department of Computer Science

University of Toronto

Administrivia

- **Project proposals due on Thursday**
  - Create Web page with brief project proposal (HTML,TXT)
    - What is the problem you are solving?
    - Why is the problem interesting?
    - Why is the problem hard?
    - How are you planning to solve the problem?
    - What is the related work?

Key Architectural Decisions in DS

- **Naming:**
  - What a user is looking for
- **Addressing**
  - Where the resource is
- **Routing**
  - How to get to the destination
- **Name lookup**
  - Binding between names and addresses
  - Resolve names to addresses

- **Name servers' API:**
  - address = resolve(name)
  - bind(name, address)

Incorporate Structure

- **Into names:**
  - Name syntax, types of records
- **Into system administration:**
  - Tree of name servers + local designated name server
- **Into name authority:**
  - Hierarchical name space composition
  - Based on delegating authority:
    - + simple and elegant
    - - higher-up authority resists delegating control
      - ICANN has been in the press a lot
      - CSLab raises security, administrative issues
      - Individual users have no binding power

Trade-offs Centralized/Distributed Naming Service

Trade-offs Centralized/Distributed Naming Service

- **+ simple database**
- **+ uniqueness and unambiguity easy to ensure**
- **+ could provide flat namespace**

- **- single point of failure**
- **- scalability bottleneck**
- **- performance bottleneck**
- **- administrative bottleneck**

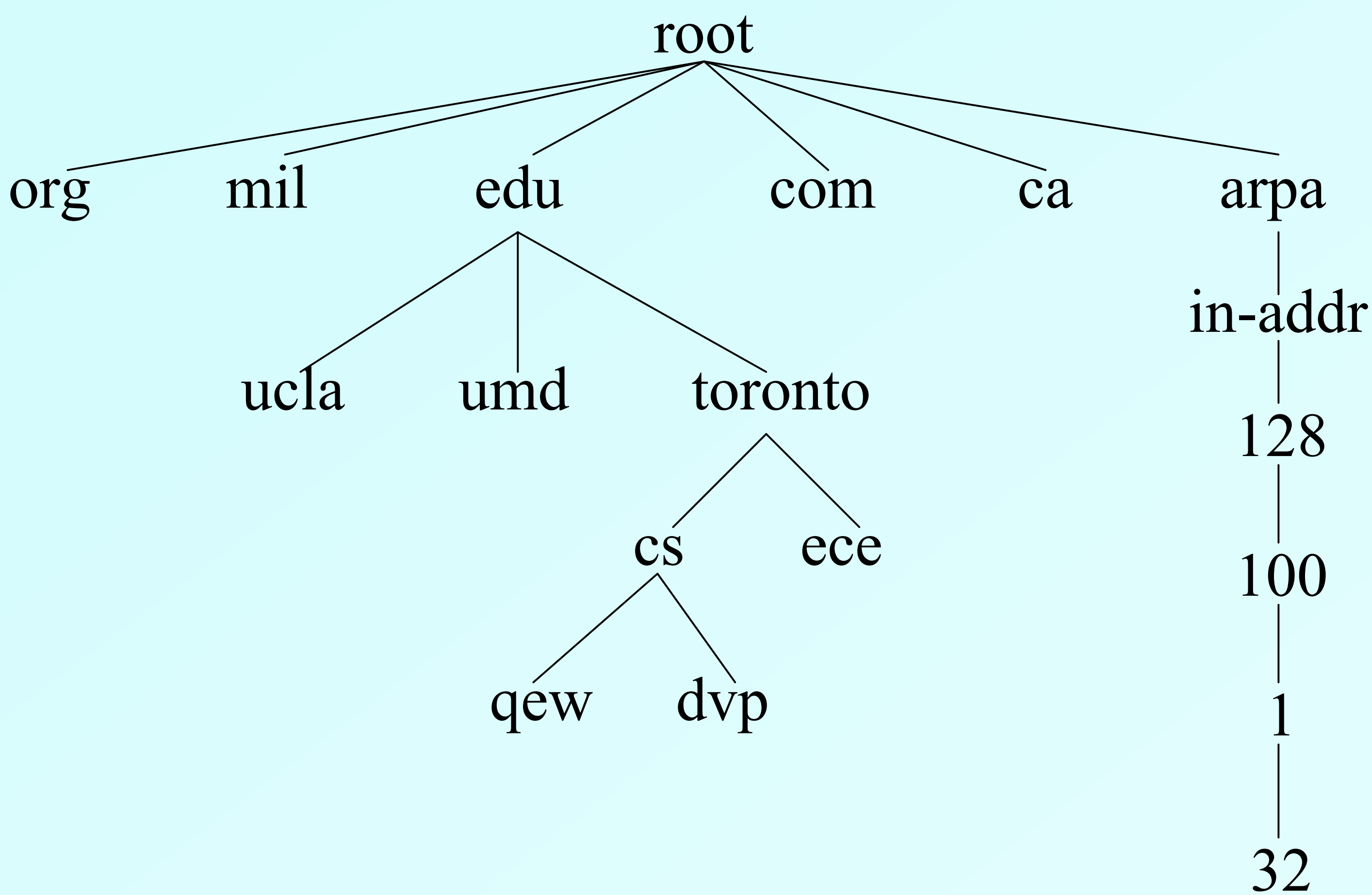
Key Architectural Issues for Distributed Naming Service

- **Distributing information among servers**
  - Models of lookup/resolution
    - Recursive
      - + simpler clients
    - Iterative
      - + simpler servers, client caching + timeout decisions
- **Figuring out the authority for a given name**
  - Bootstrapping problem: hardcode
- **Preserving adequate scalability and performance**
  - No linear growth (linear in # clients, servers, names)
    - DB state, metadata, server, name resolution cost
- **Maintaining consistency across replicas, caches**

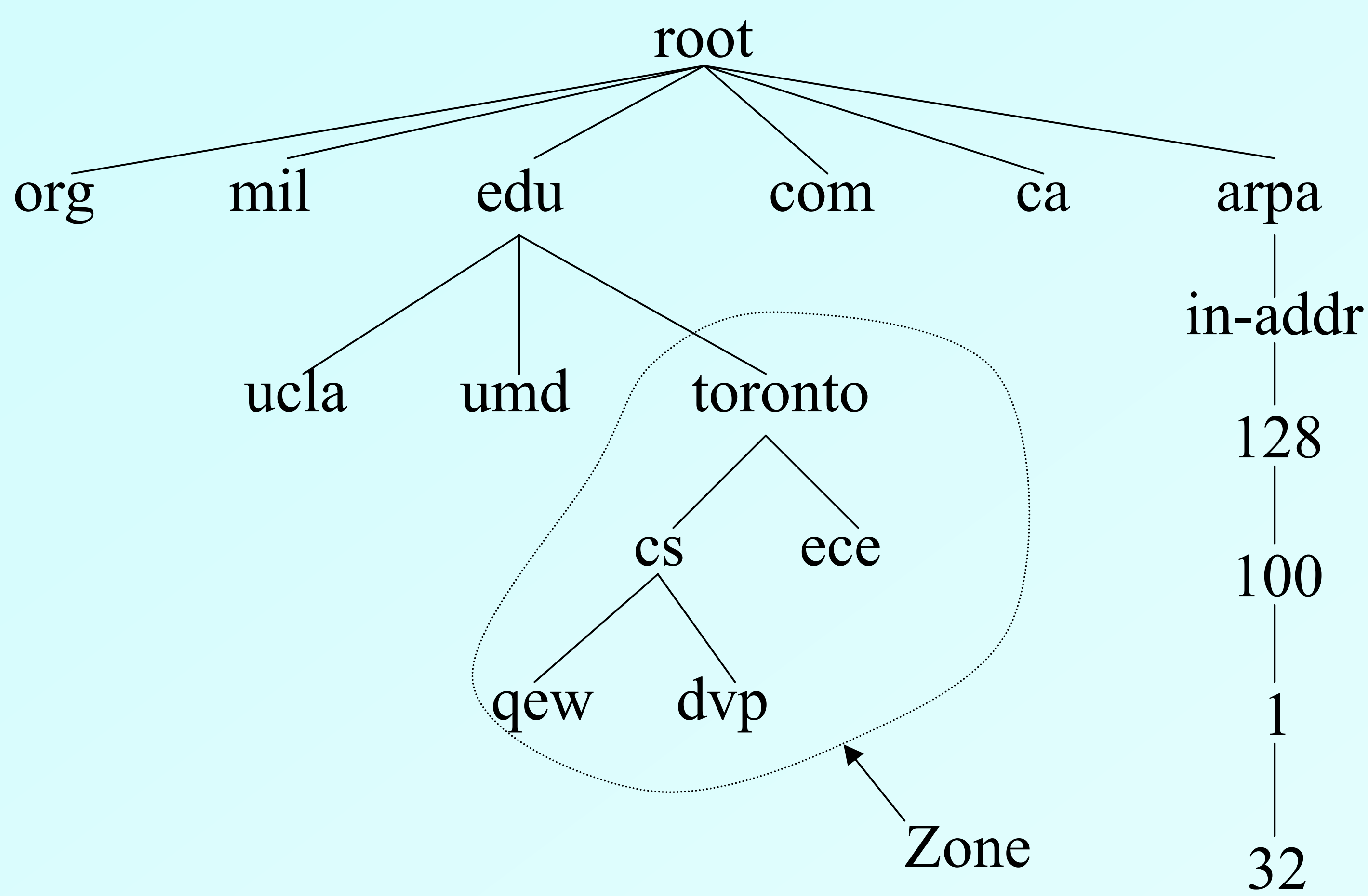
Domain Name System

- **Original ARPAnet: hosts.txt file**
- **DNS developed in early 1980s**
- **One of most successful distributed systems ever created**
- **Assumptions:**
  – Updates are infrequent
  – Weak/no support for atomic updates, consistency

Hierarchical Name Space



Hierarchical Name Space



DNS Workings

- **Many-to-many mapping between zones and servers**
- **Higher zones knows servers for lower zones**
- **Delegation of authority at zone boundaries**
  - SOA records

- **Name resolution algorithm:**
  - If name is in server's zone, do lookup in own db
  - If name is in delegated zone, pass lookup down
  - Otherwise, pass lookup to root server

Name Space Types

- **RR Types:**
  - A = address
  - CNAME = alias
  - MX = mail exchange records
  - PTR = reverse lookup
  - NS/SOA = name servers, SOA records

Example

```
[stefan@eon stefan]$ dig www.cs.toronto.edu

; <<>> DiG 9.2.1 <<>> www.cs.toronto.edu
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 31585
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 4

;; QUESTION SECTION:
;www.cs.toronto.edu.            IN      A

;; ANSWER SECTION:
www.cs.toronto.edu.     86400   IN      CNAME   christie.cs.toronto.edu.
christie.cs.toronto.edu. 86400  IN      A       128.100.1.32

;; AUTHORITY SECTION:
cs.toronto.edu.         86400   IN      NS      dns2.cs.toronto.edu.
cs.toronto.edu.         86400   IN      NS      dns1.cs.toronto.edu.

;; ADDITIONAL SECTION:
dns1.cs.toronto.edu.    86400   IN      A       128.100.3.250
dns1.cs.toronto.edu.    86400   IN      A       128.100.2.250
dns2.cs.toronto.edu.    86400   IN      A       128.100.2.251
dns2.cs.toronto.edu.    86400   IN      A       128.100.3.251

;; Query time: 3 msec
;; SERVER: 128.100.3.251#53(128.100.3.251)
;; WHEN: Sun Oct  2 11:51:44 2005
;; MSG SIZE  rcvd: 177
```

Example

```
[stefan@eon stefan]$ dig www.cs.toronto.edu

; <<>> DiG 9.2.1 <<>> www.cs.toronto.edu
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 31585
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 4

;; QUESTION SECTION:
;www.cs.toronto.edu.            IN      A

;; ANSWER SECTION:
www.cs.toronto.edu.     86400   IN      CNAME   christie.cs.toronto.edu.
christie.cs.toronto.edu. 86400  IN      A       128.100.1.32

;; AUTHORITY SECTION:
cs.toronto.edu.         86400   IN      NS      dns2.cs.toronto.edu.
cs.toronto.edu.         86400   IN      NS      dns1.cs.toronto.edu.

;; ADDITIONAL SECTION:
dns1.cs.toronto.edu.    86400   IN      A       128.100.3.250
dns1.cs.toronto.edu.    86400   IN      A       128.100.2.250
dns2.cs.toronto.edu.    86400   IN      A       128.100.2.251
dns2.cs.toronto.edu.    86400   IN      A       128.100.3.251

;; Query time: 3 msec
;; SERVER: 128.100.3.251#53(128.100.3.251)
;; WHEN: Sun Oct  2 11:51:44 2005
;; MSG SIZE  rcvd: 177
```

Example

```
[stefan@eon stefan]$ dig www.cs.toronto.edu

; <<>> DiG 9.2.1 <<>> www.cs.toronto.edu
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 31585
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 4

;; QUESTION SECTION:
;www.cs.toronto.edu.            IN      A

;; ANSWER SECTION:
www.cs.toronto.edu.     86400   IN      CNAME   christie.cs.toronto.edu.
christie.cs.toronto.edu. 86400  IN      A       128.100.1.32

;; AUTHORITY SECTION:
cs.toronto.edu.         86400   IN      NS      dns2.cs.toronto.edu.
cs.toronto.edu.         86400   IN      NS      dns1.cs.toronto.edu.

;; ADDITIONAL SECTION:
dns1.cs.toronto.edu.    86400   IN      A       128.100.3.250
dns1.cs.toronto.edu.    86400   IN      A       128.100.2.250
dns2.cs.toronto.edu.    86400   IN      A       128.100.2.251
dns2.cs.toronto.edu.    86400   IN      A       128.100.3.251

;; Query time: 3 msec
;; SERVER: 128.100.3.251#53(128.100.3.251)
;; WHEN: Sun Oct  2 11:51:44 2005
;; MSG SIZE  rcvd: 177
```

Example

```
[stefan@eon stefan]$ dig www.cs.toronto.edu ns

; <<>> DiG 9.2.1 <<>> www.cs.toronto.edu ns
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 31247
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;www.cs.toronto.edu.            IN      NS

;; ANSWER SECTION:
www.cs.toronto.edu.     86400   IN      CNAME   christie.cs.toronto.edu.

;; AUTHORITY SECTION:
cs.toronto.edu.         86400   IN      SOA     keele.cs.toronto.edu. hostmaster
.cs.toronto.edu. 2005093000 10800 1800 3628800 86400

;; Query time: 1 msec
;; SERVER: 128.100.3.251#53(128.100.3.251)
;; WHEN: Sun Oct  2 11:56:16 2005
;; MSG SIZE  rcvd: 112
```

Protocol uses UDP transmissions

- **Unreliable: up to the client to implement reliability**
- **Berkeley resolver: cycles through up to 3 servers per request, doubling timeout each try**
- **Berkeley name server**
  - 16 different addresses per request
  - Cycles through servers up to 3 times, doubling timeout
- **Sequence number per request to match response**

Update and Consistency Models

- **Manual update at primary server for zone**
  - Version number incremented each time
- **Secondary servers check with primary periodically**
  - SOA record specifies version, refresh, expiration
  - Secondary does zone transfer if needed
  - Not all "authoritative" servers may be up-to-date
- **TTL associated with each RR**
- **"Eventual" consistency**

Caching

- **Application level: browsers**
- **Stub resolver: BIND library**
- **Local server:**
  - Temporal locality within single user stream
  - Temporal locality within user population
  - Spatial locality
- **Caching results:**
  - Name popularity has Zipf distribution
  - Popular names have lower TTLs
  - Locality “saturates” with O(10-20) clients in a group. Why?

Lessons

- **Almost no issues in practice related to consistency, TTL settings, protocol specification:**
  - Root servers see bad queries or traffic due to unavailable servers
    - Use negative caching
  - Does UDP mattered?
    - Not clear DNS can keep up with TCP load
    - Data exchange fits in diagram, no need for ordering
    - Congestion window is not important, timers are
    - Connectionless mentality masked early TCP
      - Clients can ask queries in parallel
      - Rate of server retransmission independent of rate of clients' requests

Security Issues

- **Attacks:**
  - DNS servers can act as reflectors <-- DoS attack
  - Opportunistic responses to DNS queries
  - Subvert DNS server can cause lots of damage
- **Data integrity:**
  - What is reasonable?
    - Authority for namespace signs off records inside namespace
    - Is it enough?
- **Data confidentiality:**
  - Discover all names within a domain is bad
  - DNSSEC weakens confidentiality
    - Due to signing on answers

Discussion

- **How to support mobile hosts in DNS?**

Discussion

- **How can we use DNS for load-balancing?**

Discussion

- **Can we use DNS to implement a search engine?**