# Hierarchical Name Services: DNS

March 29, 2020

# Roadmap

# DNS Names and IP Addresses

Problem: IP addresses are not easy to memorize (even in v4, let alone v6);

Solution: use **names** instead of addresses:

- identify objects;
- (may) help locate objects;
- (may) specify a role;
- (may) provide access permissions

# Roadmap

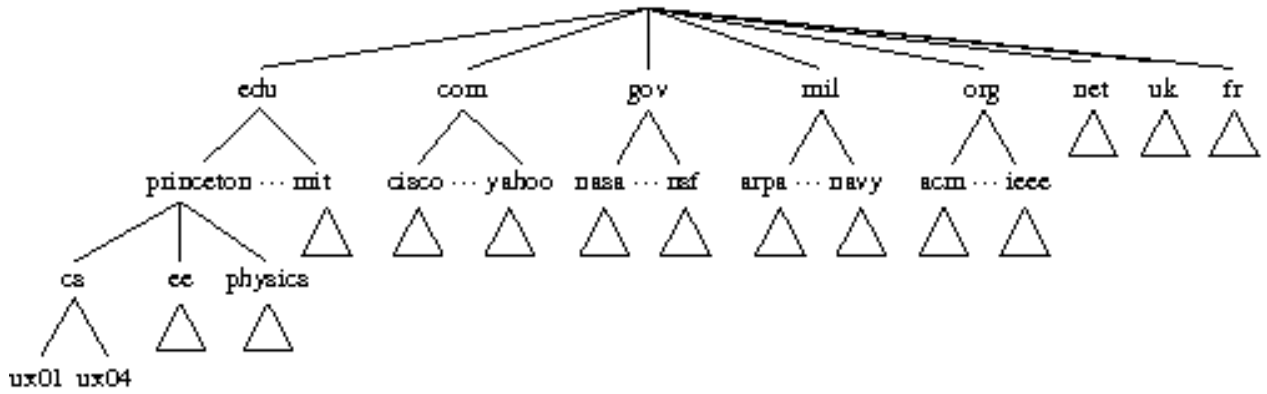
# DNS: Domain Name System

- Is used on the Internet to identify objects (not only hosts):
  - DNS uses a hierarchical name space;
  - This space is maintained in a distributed and hierarchical way by several servers
- Before DNS, the Internet used a file that was maintained by the Internet Network Information Center (NIC), and initially distributed periodically to all computers. This was a:
  - centralized
  - non scalable

  solution

# DNS: Fundamental Concepts

Hierarchy:



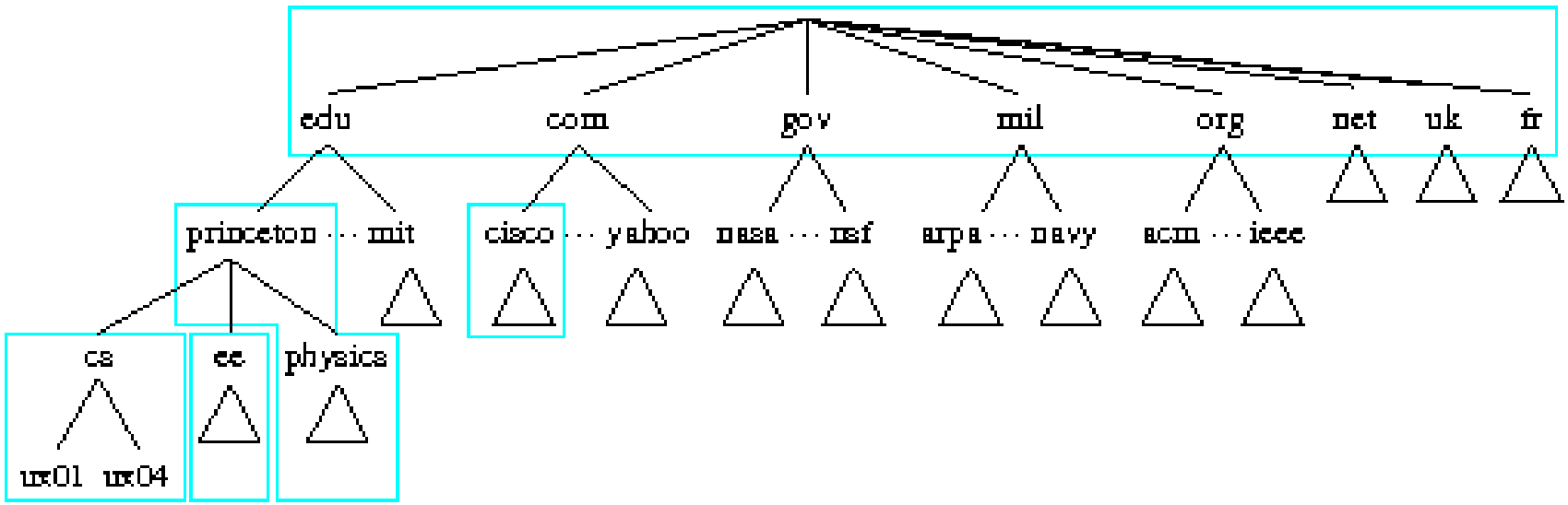

Name: `up.pt` or `fe.up.pt` (starts at the node);

Domain (subdomain): sub-tree under a name: `up.pt` (`fe.up.pt`).

# Name Space Implementation: Zones

Zone: is a sub-tree that stems from the partition of the DNS hierarchy

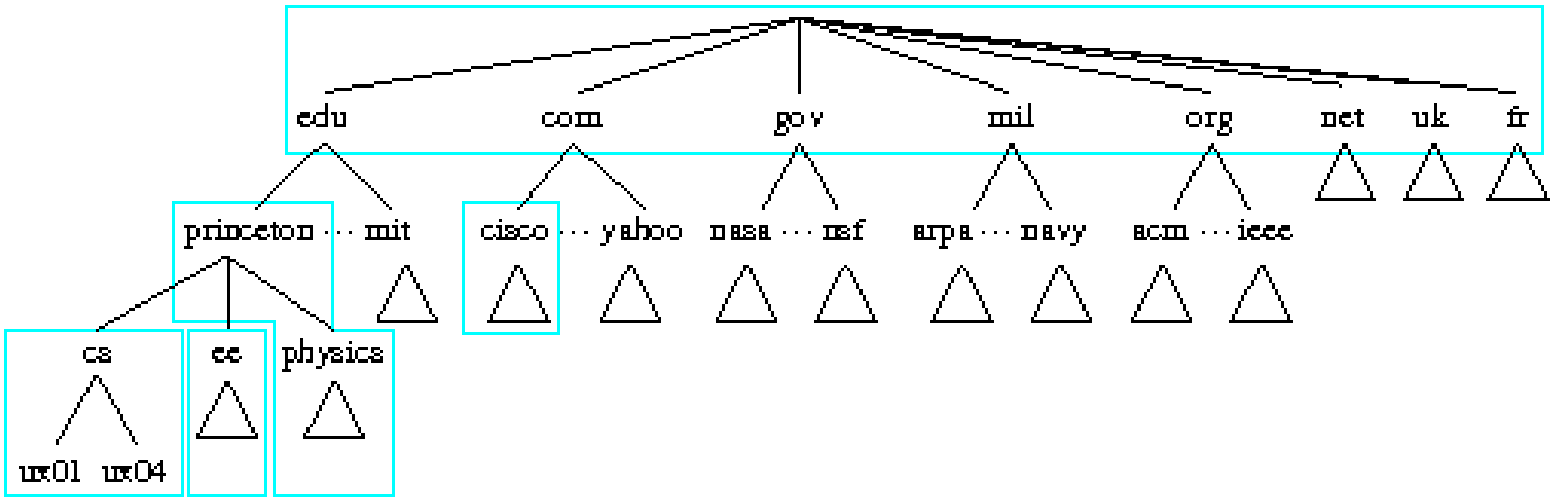
- A zone may not be a subdomain. E.g. the `princeton.edu` zone




- 2 zones never overlap
- A zone corresponds to an administrative authority

# DNS Server

- For each zone there is one server, the primary, that is responsible for the information of that zone




- A server may contain information of more than one zone.
- For availability reasons, a zone's information must be replicated at least in another (secondary) server

**IMP-** Primary and secondary servers should be located so as to maximize the availability of the zone's information

# Roadmap

# Resource Records

- The information on each node in the DNS tree is kept in resource records
- A resource record maps the node's name to a value:

  *(name, value, type, class, ttl)*

  where

  `name/value` are not necessarily host names and IP addresses

  `type` specifies how the value should be interpreted (it depends on the class);

  `class` specifies the name space (the `IN` class is, by far, the most used class);

  `ttl` *time-to-live* - is the validity time of the record in seconds (used for caching)

# Resource Records: `IN` Class Types

`A` IP address (`dig sifeup.fe.up.pt`)

*(sifeup.fe.up.pt, 193.136.28.205, A, IN)*

`NS` name of the zone's DNS server zona (`dig ns fe.up.pt`)

*(fe.up.pt, ns1.up.pt, NS, IN)*

`CNAME` alias (canonical name) (`dig cname www.fe.up.pt`):

*(www, sifeup.fe.up.pt, CNAME, IN)*

Cannonical names do not have type A RR

`MX` name of SMTP servers (`dig mx fe.up.pt`):

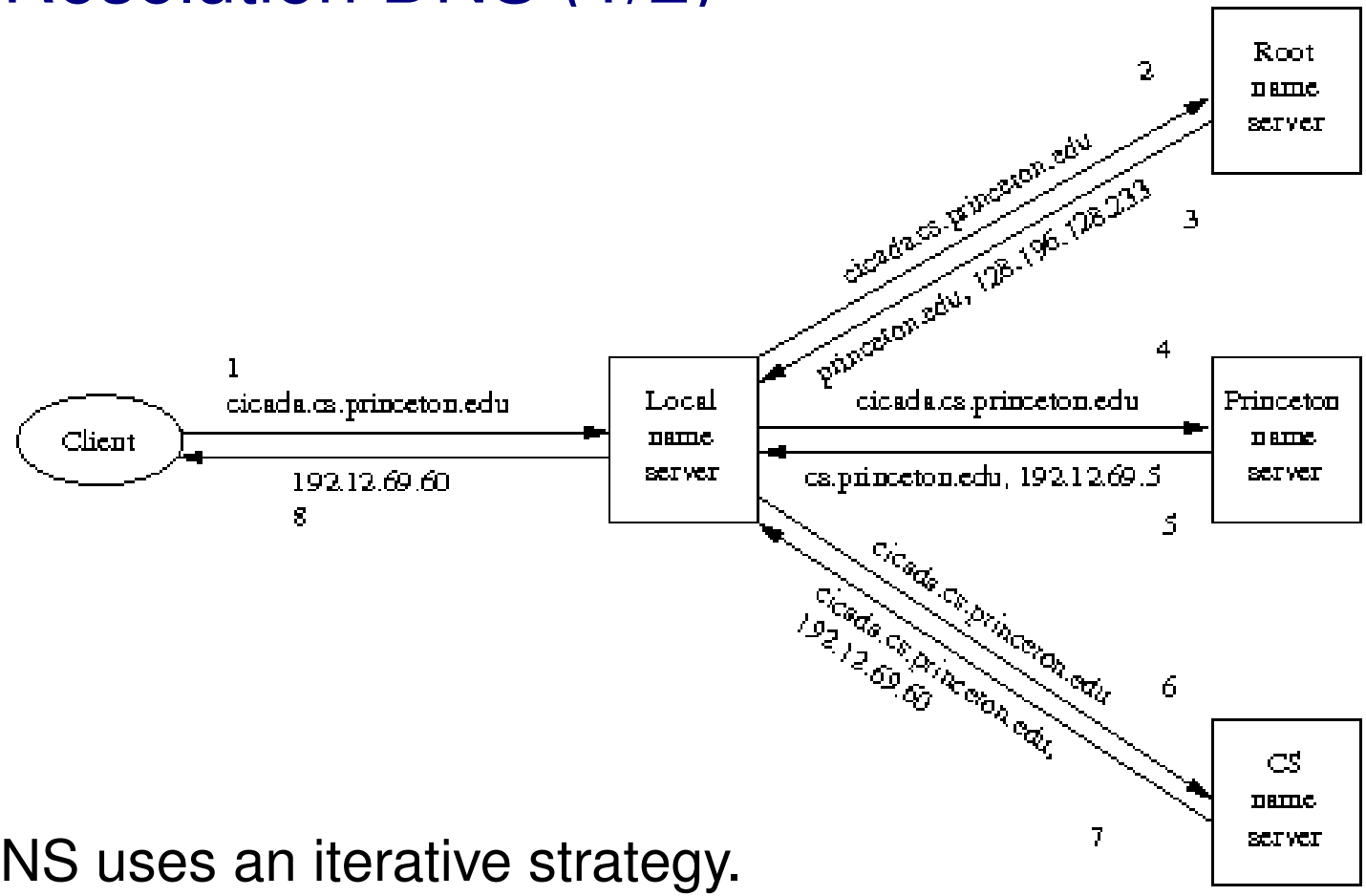
*(fe.up.pt, mx01.up.pt, MX, IN)*

...

*(fe.up.pt ,mx06.up.pt, MX, IN)*

# Roadmap

# Name Resolution DNS (1/2)




- DNS uses an iterative strategy.
- Resolution is based on **longest prefix matching**
- Every server must keep `(NS,A)` RR pairs for each of its direct subdomains
- Clients must be configured with either a local name server or a public name server, e.g. Google's

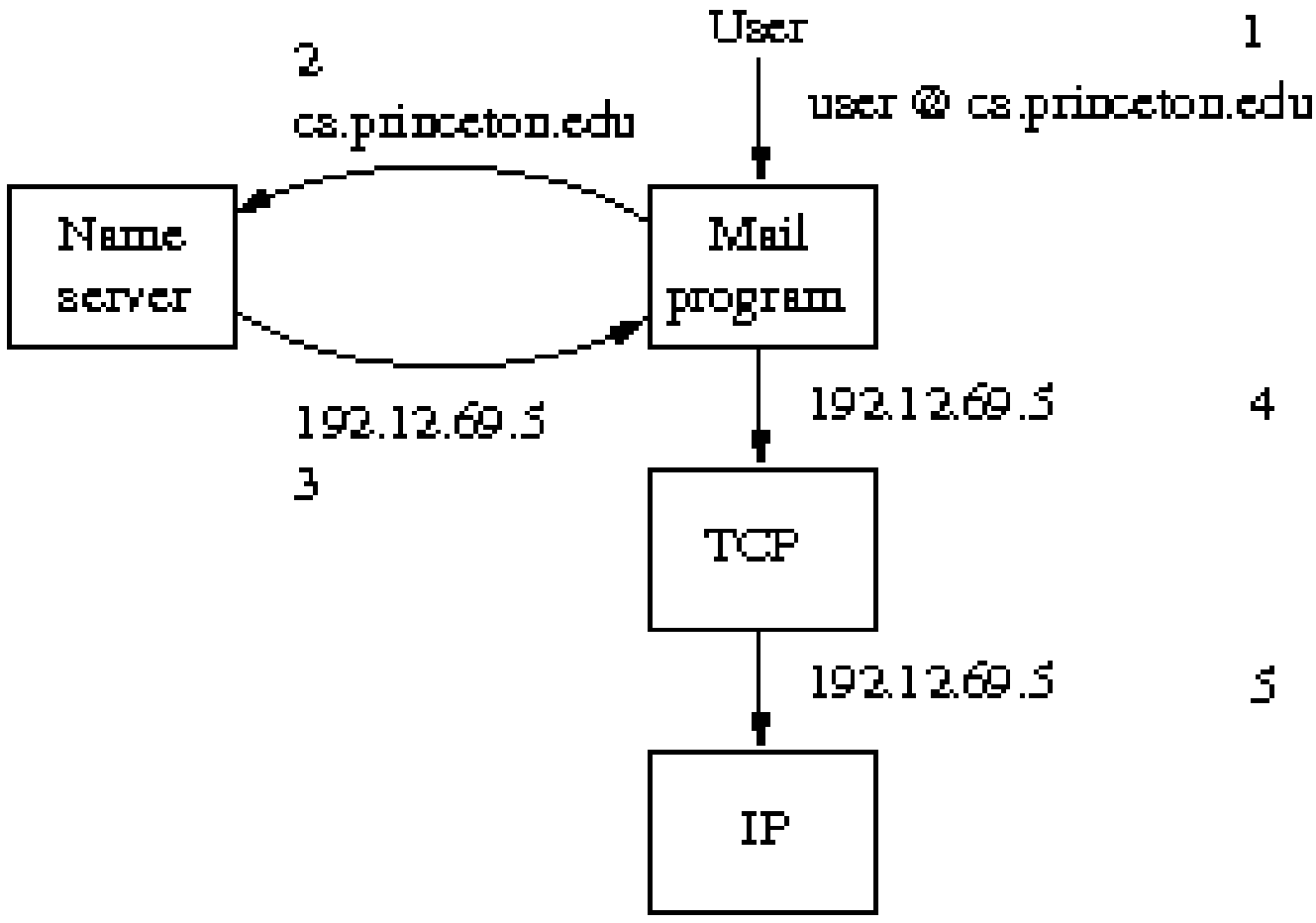
# Resolução de nomes em DNS (2/2)

- For efficiency reasons, i.e. avoiding communication, DNS **resolvers**, i.e. DNS clients, use caching extensively

  Libraries not as effective, but may avoid communication

  Servers , a.k.a. **recursive resolvers**
- Answers provided using caches are said to be **non-authoritative**
- Cached RRs are garbage-collected using their TTL field.

# Exemplo: Uso de DNS




Obs.- The client needs both the MX RR for `cs.princeton.edu` and the A RR of the SMTP server. However, it enough to send an MX query.

# Roadmap

# Replication in DNS (1/2)

- DNS relies heavily on replication for reasons of:
  - performance
  - availability
- Replication raises problems of inconsistency
  - Which are even harder when changes can be done on different copies
- DNS has some special features that simplify replication
  - The zone information is updated/added at only one server (the primary)
  - The replicas need not be updated synchronously, i.e. simultaneously. The use of stale data
    - Is usually detected upon use
    - Even if that is not the case, it usually does not affect the correction of the applications that use DNS

# Replication in DNS (2/2)

- In DNS a zone's RRs must have
  - One **primary server**
  - At least one **secondary server**
- Both primary and secondary servers must keep all RRs of the nodes in the zone
  - Replies by these servers are considered **authoritative**
- To detect changes to a zone's RR, each zone has a Start of Authority (SOA) RR with the following fields

  Serial a 32 bit that identifies the zone's "version"
  - Everytime there is a change to a Zone's RR, this field must be increased

  Refresh a 32 bit integer that specifies the maximum time (in seconds) between update attempts

# Update Detection and Zone Propagation

**Update detection** by comparing the Serial field of the SOA RR in the secondary with that of the SOA RR in the primary

**How can the secondary get the primary's SOA RR?**

**polling** the secondary issues a SOA query to the primary

**notification** every time the primary updates the serial number of its SOA, it notifies the secondary servers

- This is specified in RFC 1996, but it has not been approved yet

**Zone Transfer**

**Non-incremental** the secondary sends an `AXFR` query requesting for the transfer of an entire zone

**Incremental** the secondary requests the primary to transfer only the changes to the zone.

- This approach is specified in RFC 195, but it has not been approved yet

**Note** DNS requires zone transfer to use TCP

- For all other queries, DNS usually uses UDP, although it is possible to use TCP

# Roadmap

# Further Reading

- Subsection 5.3(.4) of van Steen and Tanenbaum, *Distributed Systems, 3rd Ed.*
- P. Mockapetris, *DOMAIN NAMES - CONCEPTS AND FACILITIES*, in RFC 1034, 1987
- P. Mockapetris, *DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION*, in RFC 1035, 1987
- M. Ohta, *Incremental Zone Transfer in DNS*, in RFC 1995, 1996
- P. Vixie, *A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)*, in RFC 1996, 1996