Development of the Domain Name System -> 1988

Authors: Paul V. Mockapetris & Kevin J. Dunlap

Presented by Steve Sprecher

Problem

- IP addresses are hard to remember
- Keep track of hostname -> address mapping
- Centrally controlled
- Definitely not scalable
- Current domain name systems are Limited or host specific

Design Requirements & Constraints

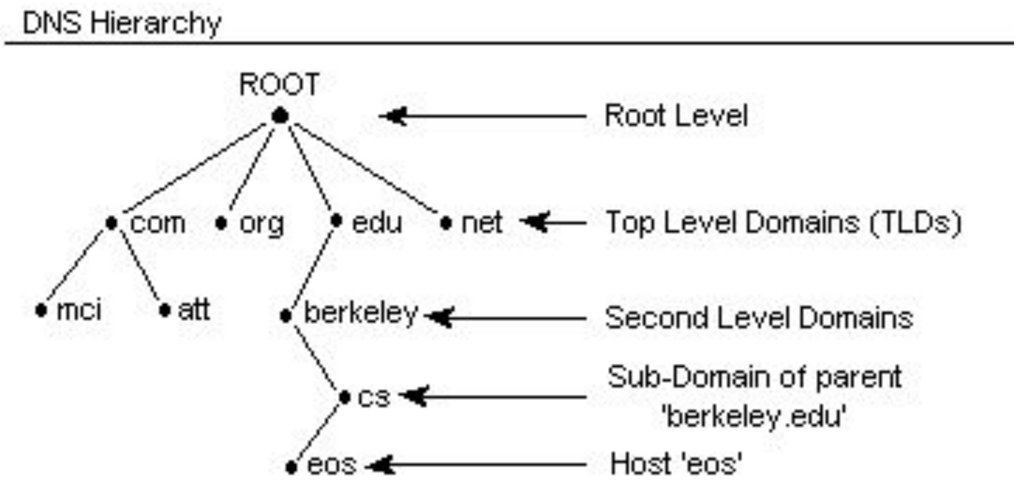
- DNS same or more info as HOSTS.TXT
- Distributed maintenance
- No obvious size limits
- Work cross Internet(s)
- Tolerable performance
- Must provide extensible services and work everywhere (to justify cost)
- Do not force an OS or organizational style

Key Insights and Design Details

- Hierarchical
- Distributed
- Cached

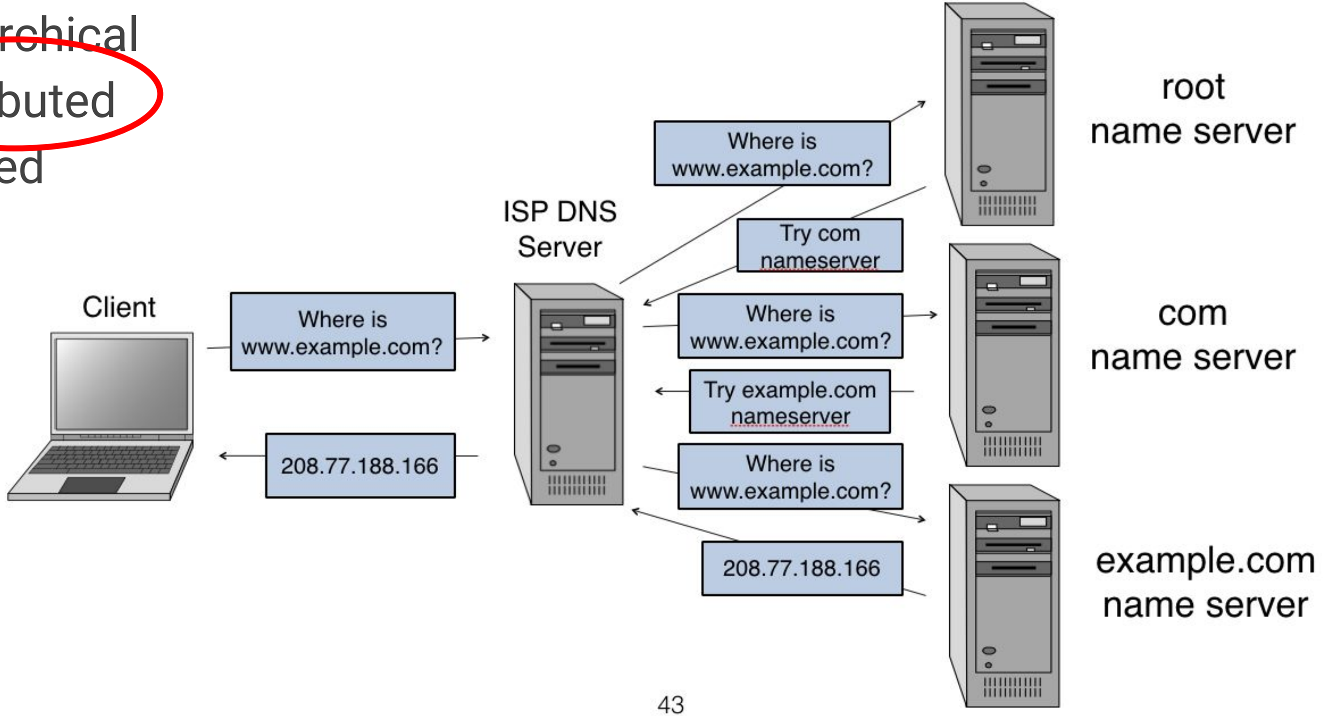
Key Insights and Design Details

- Hierarchical
- Distributed
- Cached



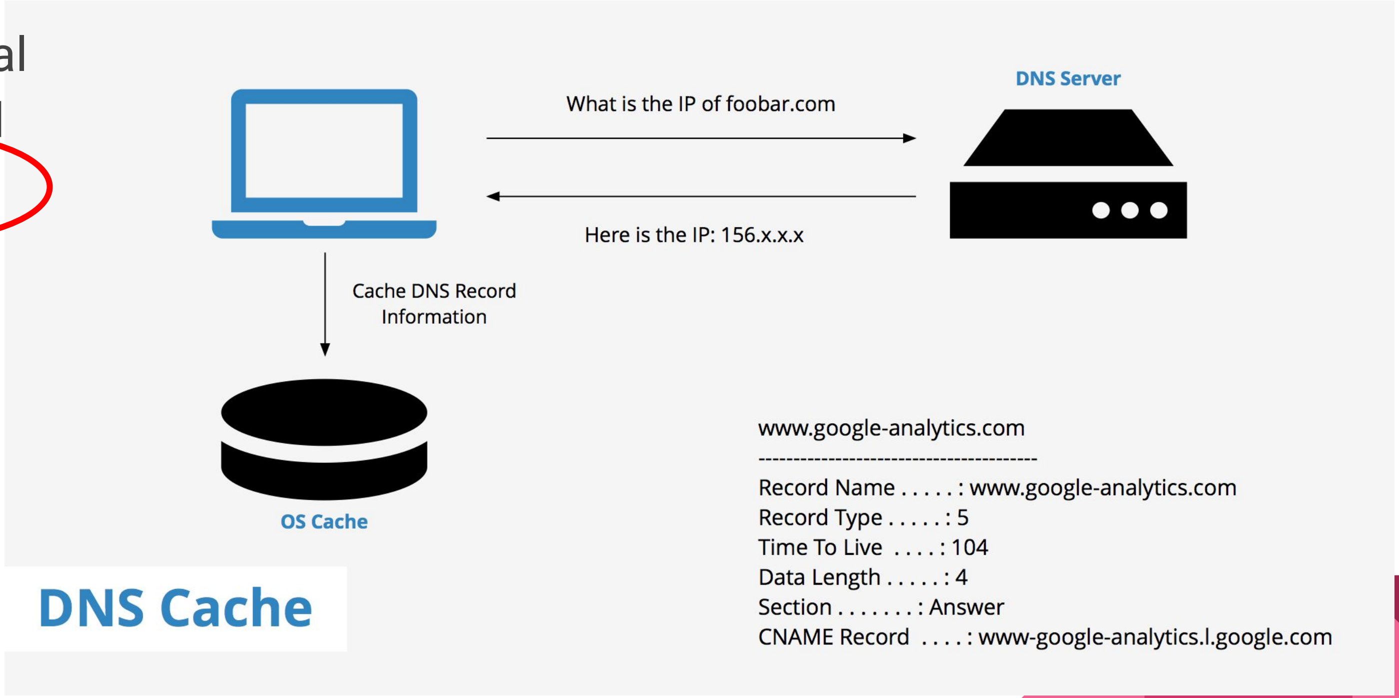
Key Insights and Design Details

- Hierarchical
- Distributed
- Cached



Key Insights and Design Details

- Hierarchical
- Distributed
- Cached



Results / Key Insights

- The form and content of information was well understood
    - Nope!
- Network performance wasn’t great
- Measurements were very hard to do
- Expected negative responses to be rare
    - Needed to add caching for this as well

Results / Lessons Learned

- Successes
  - Variable depth hierarchy (self organizing, different size orgs, easy encapsulation)
  - Organizational structuring of names
  - Datagram access
  - Additional section processing
  - Caching!
  - Mail address cooperation
- Shortcomings
  - Type and class growth (expected huge demand, got almost none)
  - Transition wasn't easy
  - Distribution of control ≠ Distribution of expertise / responsibility

Summarize the KEY results

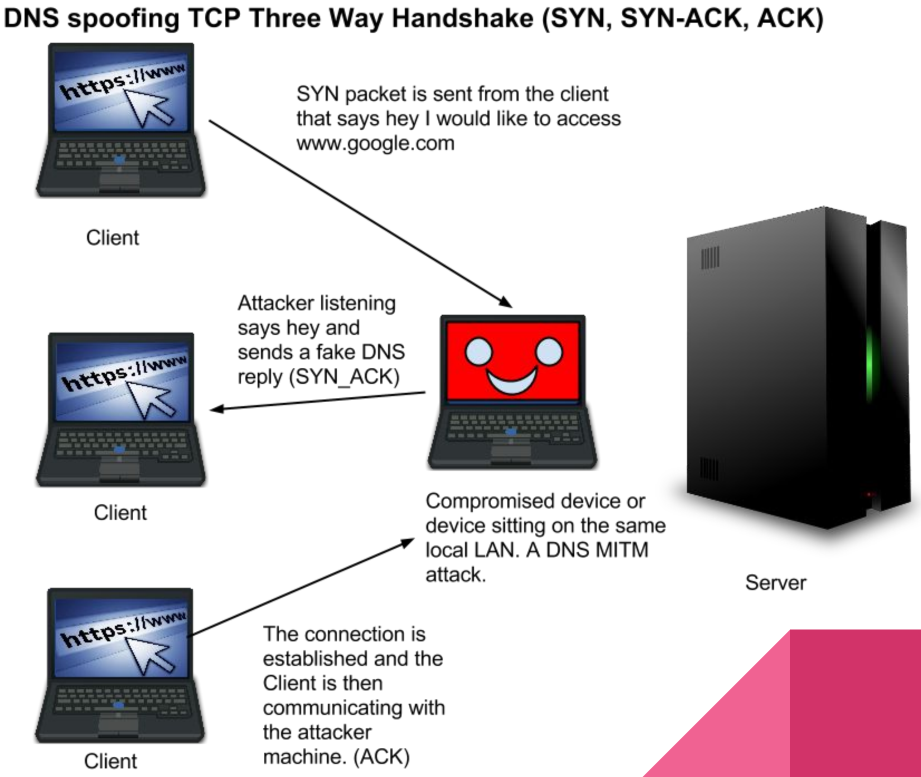
Types of Records

...

Shorthand	Resource Record	Description
A	Address Mapping records	IPv4
AAAA	IP Version 6 Address records	IPv6
CNAME	Canonical Name records	The CNAME record specifies a domain name that has to be queried in order to resolve the original DNS query
HINFO	Host Information records	General information about a host - not typically used on public servers
ISDN	Integrated Services Digital Network records	Telephone number
MX	Mail exchanger record	Mail exchange server for a DNS domain name
NS	Name Server records	The NS record specifies an authoritative name server for given host
PTR	Reverse-lookup Pointer records	Domain name -> IP address
SOA	Start of Authority records	The record specifies core information about a DNS zone
TXT	Text records	The text record can hold arbitrary non-formatted text string. Typically, the record is used by Sender Policy Framework (SPF) to prevent fake emails to appear to be sent by you

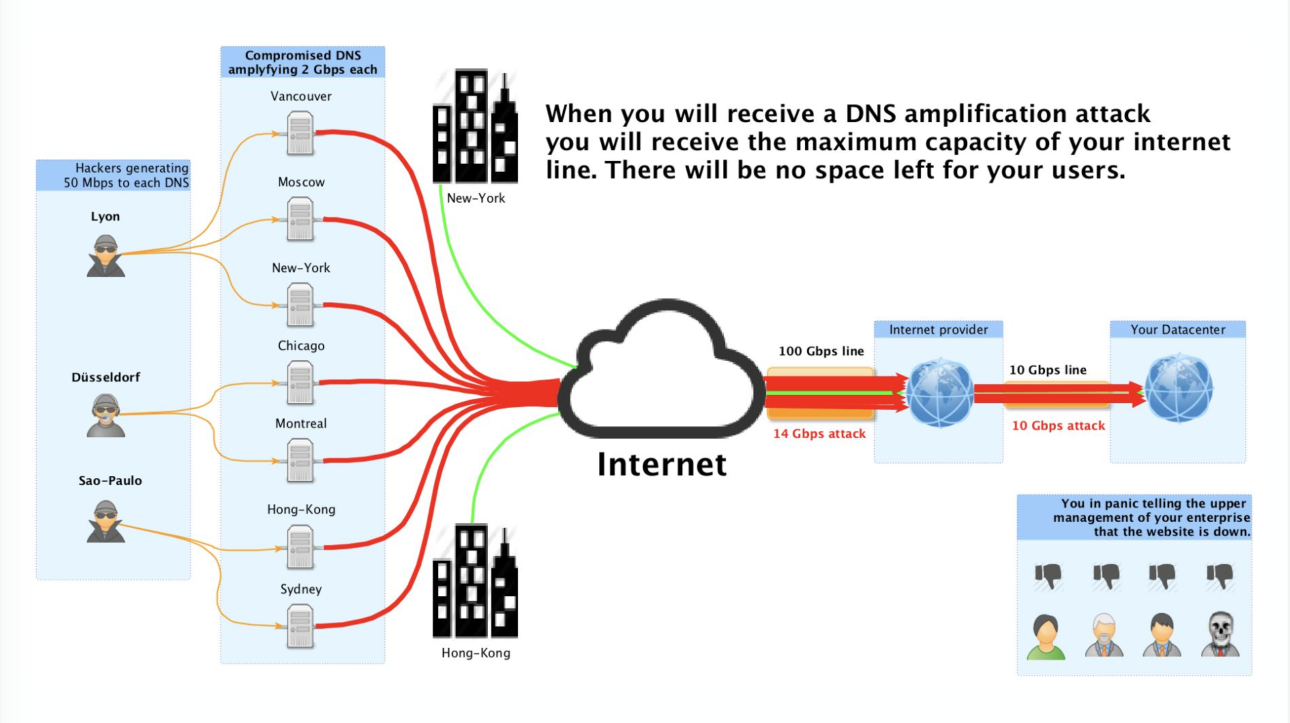
DNS & Security

- DNS Spoofing



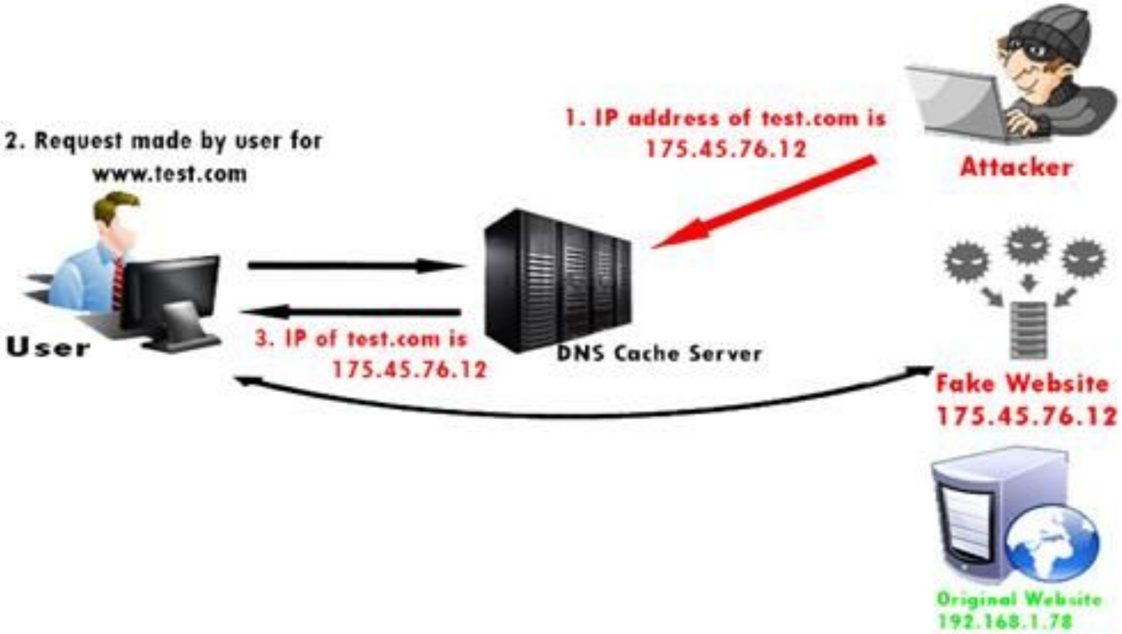
DNS & Security

- DNS amplification for DDoS attacks



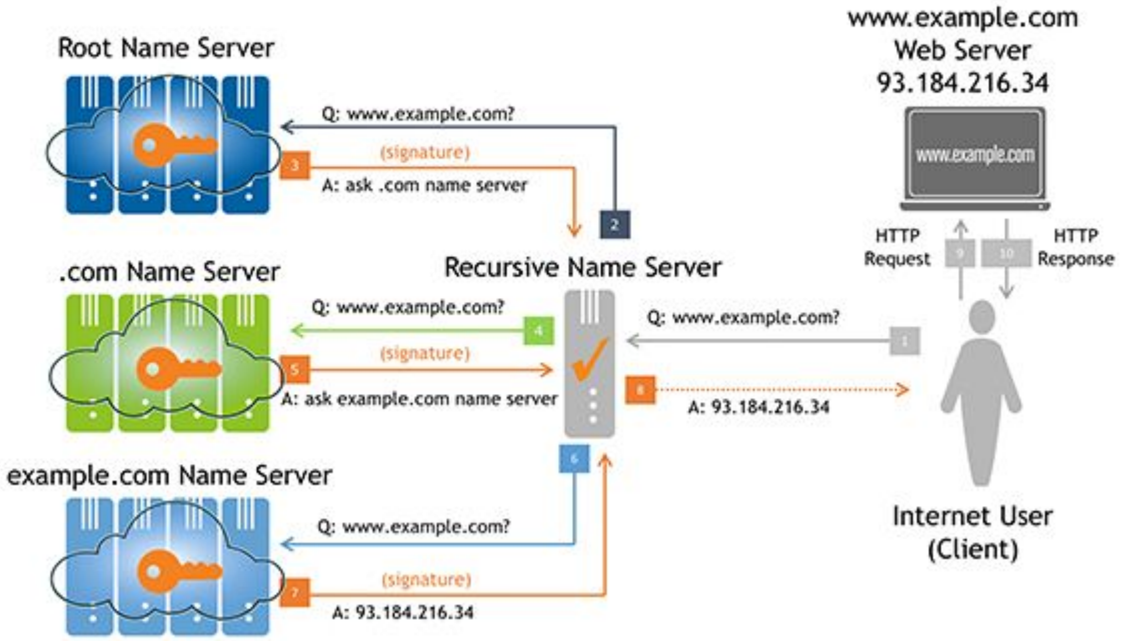
DNS & Security

- DNS cache poisoning



DNS & Security

- DNSSEC



All the Info

Standards[edit]

The Domain Name System is defined by Request for Comments (RFC) documents published by the Internet Engineering Task Force (Internet standards). The following is a list of RFCs that define the DNS protocol.

- RFC 1034, *Domain Names - Concepts and Facilities*
- RFC 1035, *Domain Names - Implementation and Specification*
- RFC 1123, *Requirements for Internet Hosts—Application and Support*
- RFC 1995, *Incremental Zone Transfer in DNS*
- RFC 1996, *A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)*
- RFC 2136, *Dynamic Updates in the domain name system (DNS UPDATE)*
- RFC 2181, *Clarifications to the DNS Specification*
- RFC 2308, *Negative Caching of DNS Queries (DNS NCACHE)*
- RFC 2672, *Non-Terminal DNS Name Redirection*
- RFC 2845, *Secret Key Transaction Authentication for DNS (TSIG)*
- RFC 3225, *Indicating Resolver Support of DNSSEC*
- RFC 3226, *DNSSEC and IPv6 A6 aware server/resolver message size requirements*
- RFC 3596, *DNS Extensions to Support IP Version 6*
- RFC 3597, *Handling of Unknown DNS Resource Record (RR) Types*

All the Info

Standards[edit]

- ...
- RFC 4343, *Domain Name System (DNS) Case Insensitivity Clarification*
- RFC 4592, *The Role of Wildcards in the Domain Name System*
- RFC 4635, *HMAC SHA TSIG Algorithm Identifiers*
- RFC 5001, *DNS Name Server Identifier (NSID) Option*
- RFC 5452, *Measures for Making DNS More Resilient against Forged Answers*
- RFC 5890, *Internationalized Domain Names for Applications (IDNA):Definitions and Document Framework*
- RFC 5891, *Internationalized Domain Names in Applications (IDNA): Protocol*
- RFC 5892, *The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)*
- RFC 5893, *Right-to-Left Scripts for Internationalized Domain Names for Applications (IDNA)*
- RFC 6891, *Extension Mechanisms for DNS (EDNS0)*
- RFC 7766, *DNS Transport over TCP - Implementation Requirements*

All the Info

Standards

Security

- RFC 4033, *DNS Security Introduction and Requirements*
- RFC 4034, *Resource Records for the DNS Security Extensions*
- RFC 4035, *Protocol Modifications for the DNS Security Extensions*
- RFC 4509, *Use of SHA-256 in DNSSEC Delegation Signer (DS) Resource Records*
- RFC 4470, *Minimally Covering NSEC Records and DNSSEC On-line Signing*
- RFC 5011, *Automated Updates of DNS Security (DNSSEC) Trust Anchors*
- RFC 5155, *DNS Security (DNSSEC) Hashed Authenticated Denial of Existence*
- RFC 5702, *Use of SHA-2 Algorithms with RSA in DNSKEY and RRSIG Resource Records for DNSSEC*
- RFC 5910, *Domain Name System (DNS) Security Extensions Mapping for the Extensible Provisioning Protocol (EPP)*
- RFC 5933, *Use of GOST Signature Algorithms in DNSKEY and RRSIG Resource Records for DNSSEC*
- RFC 7858, *Specification for DNS over Transport Layer Security (TLS)*

All the Info

Experimental

- RFC 1183, *New DNS RR Definitions*

Best Current Practices

- RFC 2182, *Selection and Operation of Secondary DNS Servers* (BCP 16)
- RFC 2317, *Classless IN-ADDR.ARPA delegation* (BCP 20)
- RFC 5625, *DNS Proxy Implementation Guidelines* (BCP 152)
- RFC 6895, *Domain Name System (DNS) IANA Considerations* (BCP 42)
- RFC 7720, *DNS Root Name Service Protocol and Deployment Requirements* (BCP 40)

Informational

These RFCs are advisory in nature, but may provide useful information despite defining neither a standard or BCP. (RFC 1796)

- RFC 1178, *Choosing a Name for Your Computer* (FYI 5)
- RFC 1591, *Domain Name System Structure and Delegation*
- RFC 1912, *Common DNS Operational and Configuration Errors*
- RFC 2100, *The Naming of Hosts*

All the Info

Informational...

- RFC 3696, *Application Techniques for Checking and Transformation of Names*
- RFC 4892, *Requirements for a Mechanism Identifying a Name Server Instance*
- RFC 5894, *Internationalized Domain Names for Applications (IDNA):Background, Explanation, and Rationale*
- RFC 5895, *Mapping Characters for Internationalized Domain Names in Applications (IDNA) 2008*
- RFC 7626, *DNS Privacy Considerations*
- RFC 7706, *Decreasing Access Time to Root Servers by Running One on Loopback*

Unknown

These RFCs have an official status of Unknown, but due to their age are not clearly labeled as such.

- RFC 920, *Domain Requirements* – Specified original top-level domains
- RFC 1032, *Domain Administrators Guide*
- RFC 1033, *Domain Administrators Operations Guide*
- RFC 1101, *DNS Encodings of Network Names and Other Types*

Designing a Global Name Service

Butler W. Lampson

Published in the proceedings of the 1986 Conference on Principles of Distributed Computing

Presented by
Ramakrishnan Sundara Raman

What is the problem?

In a large distributed computing system which continuously evolves and grows, there needs to be an efficient decentralized system for managing resource location and authentication.

Name Service

- Think about the old telephone system
- In simple terms, a name service maps names to (a set of) values
- Most prominent example today: DNS
- The name service design in this work focuses on mapping names to a set of labeled properties in a distributed system.

Design Considerations

- Large size
- Long lifetime
- High availability
- Fault isolation
- Tolerance of mistrust

Best Design: Hierarchical System

Two Levels

Client level:

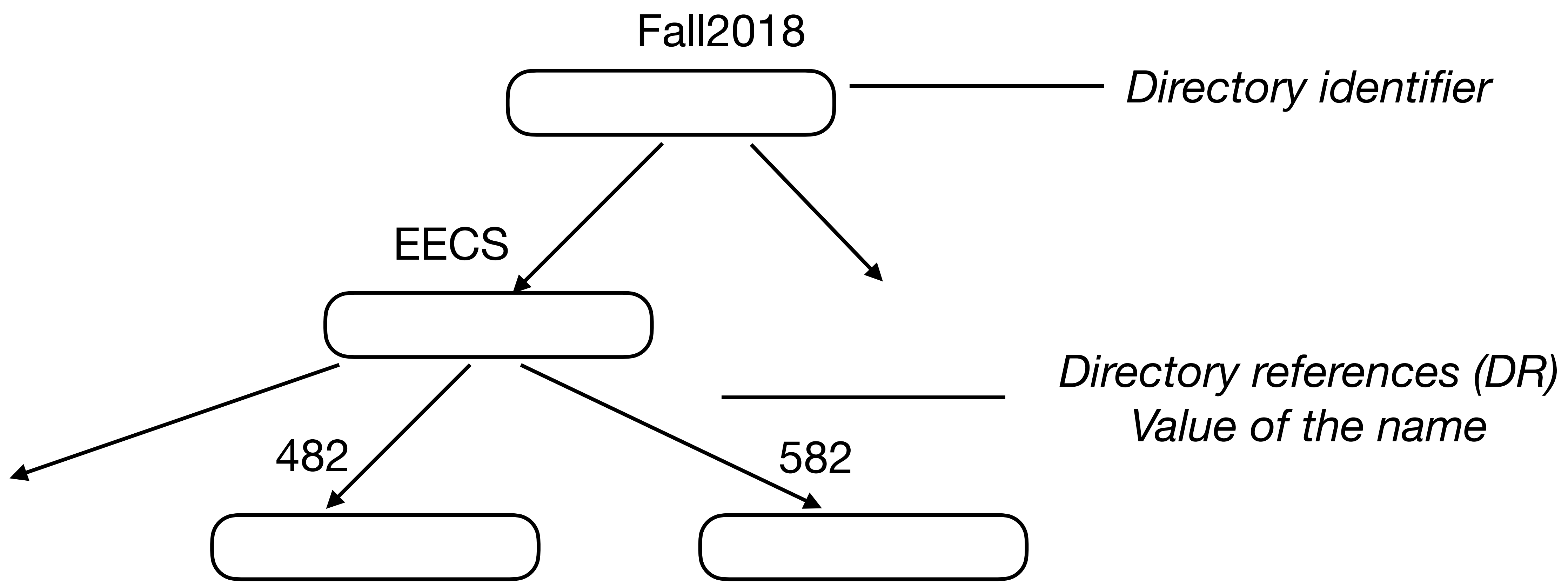
- Hierarchical names and values
- Operations for reading and updating
- No idea about replication

Administrative level:

- All copies are visible
- Mechanisms for synchronizing
- copies

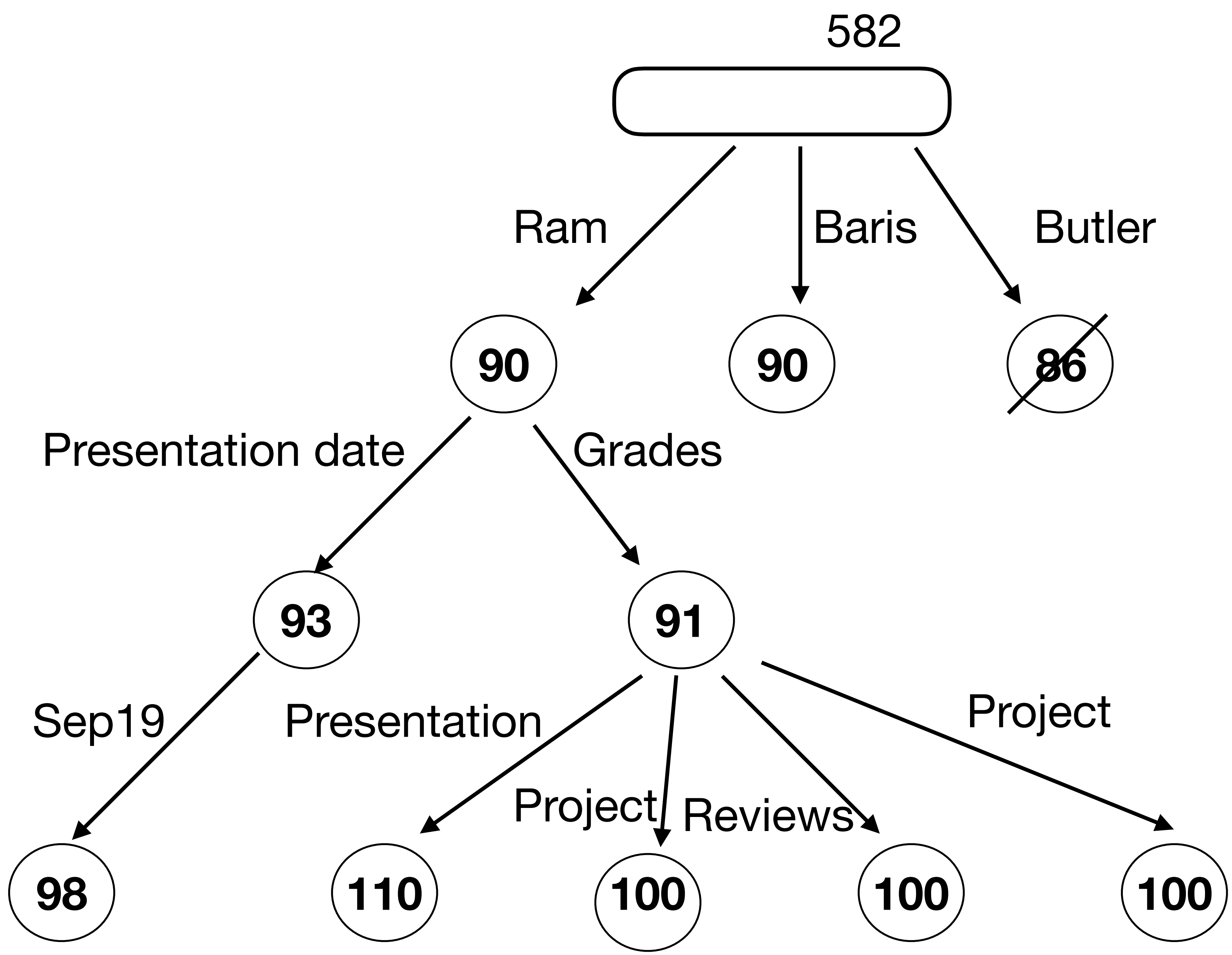
Client Level

Directory Tree



- Directory structure much like filesystems
- Full Name (FN) of directory 582 - Fall2018/EECS/582

Values in a directory



Access Control and Authentication

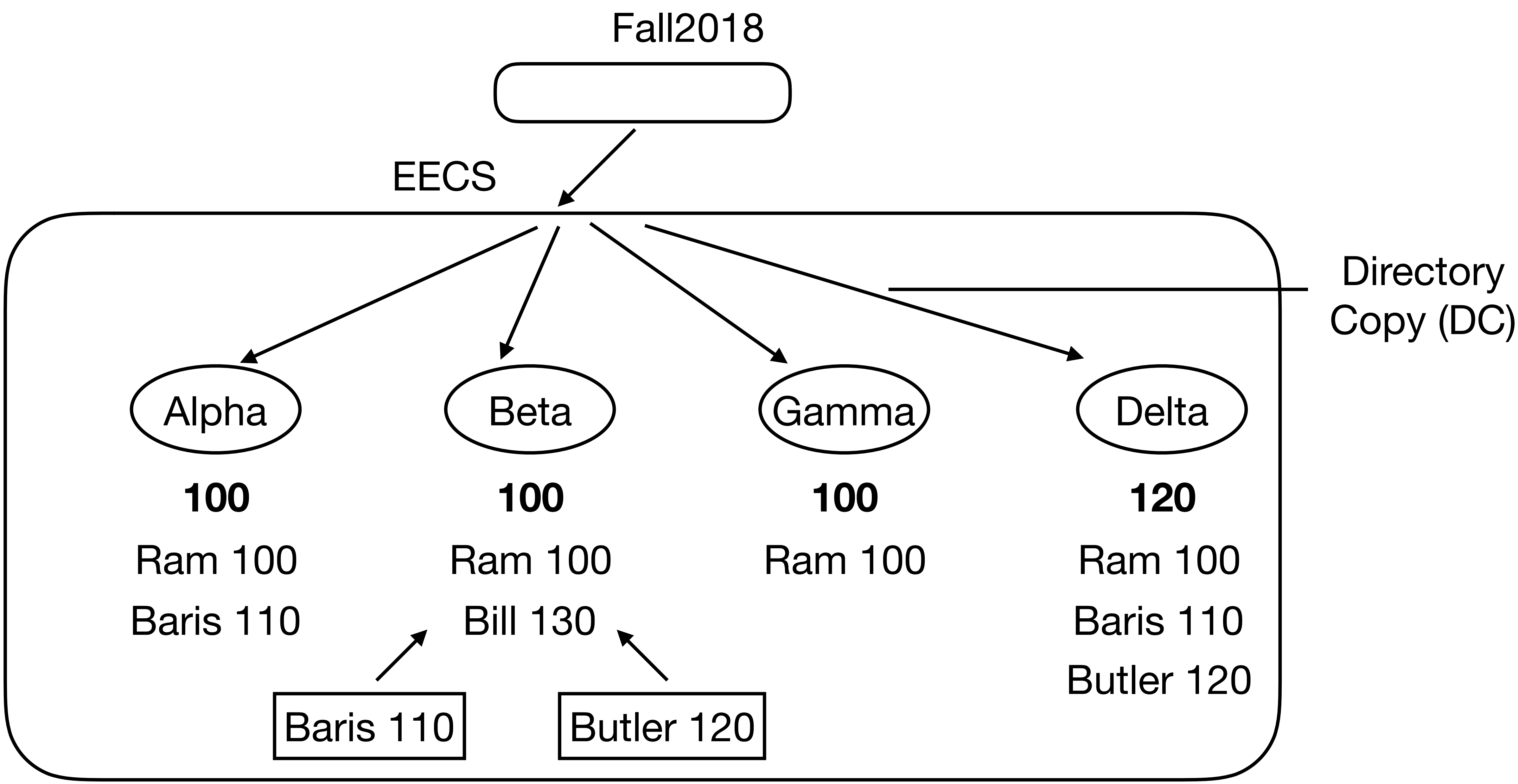
- Triplet (*Fall2018/EECS/582/Baris*, *Ram/**, write) gives principal Baris write access to the subtree which is the value of *Ram*

- Authentication is based on secure channel between caller of operation and implementor

- Authentication can be extended with a certificate

Administrative Level

Directory copies

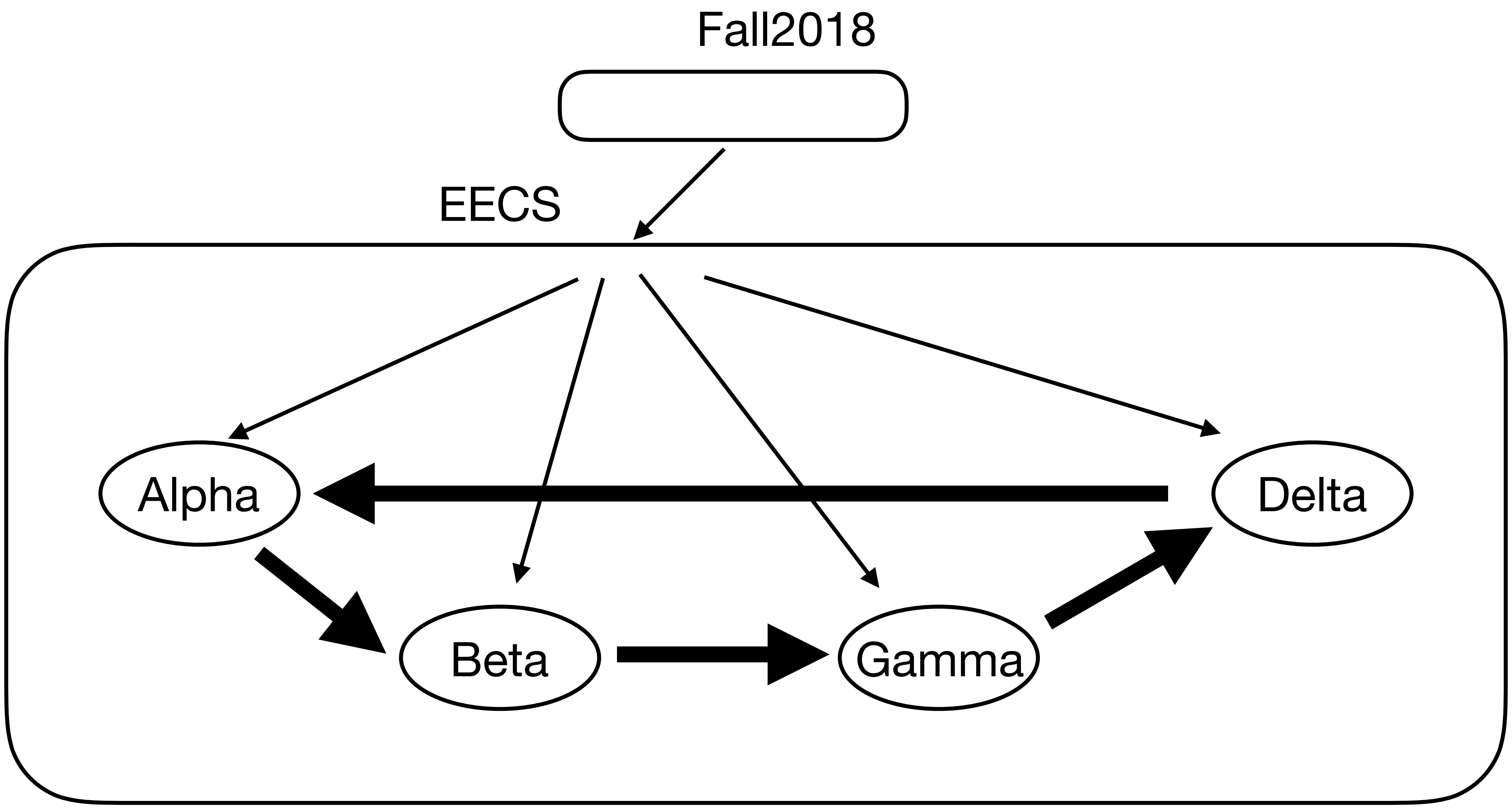


- DR now includes DI and DC.

Sweep updates

- Four updates with timestamps 100, 110, 120 and 130
- lastSweep of alpha, beta and gamma is 100, and lastSweep of delta is 120
- Update originates at one DC at sweepTS -> visits every other DC and collects all updates -> Write updates back to each DC
- All updates collected till sweepTS, sets lastSweep of all DCs to sweepTS
- Updates also possible through direct messages from one DC to another but this is not reliable

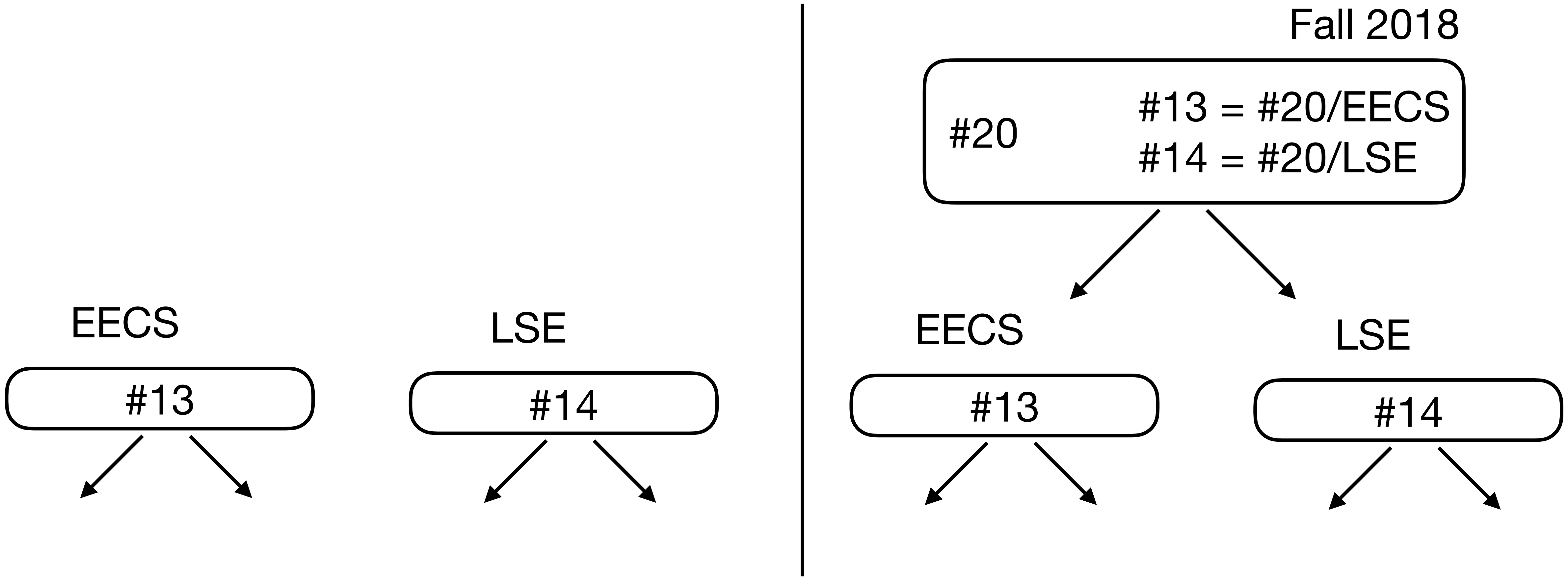
Sweep Ring



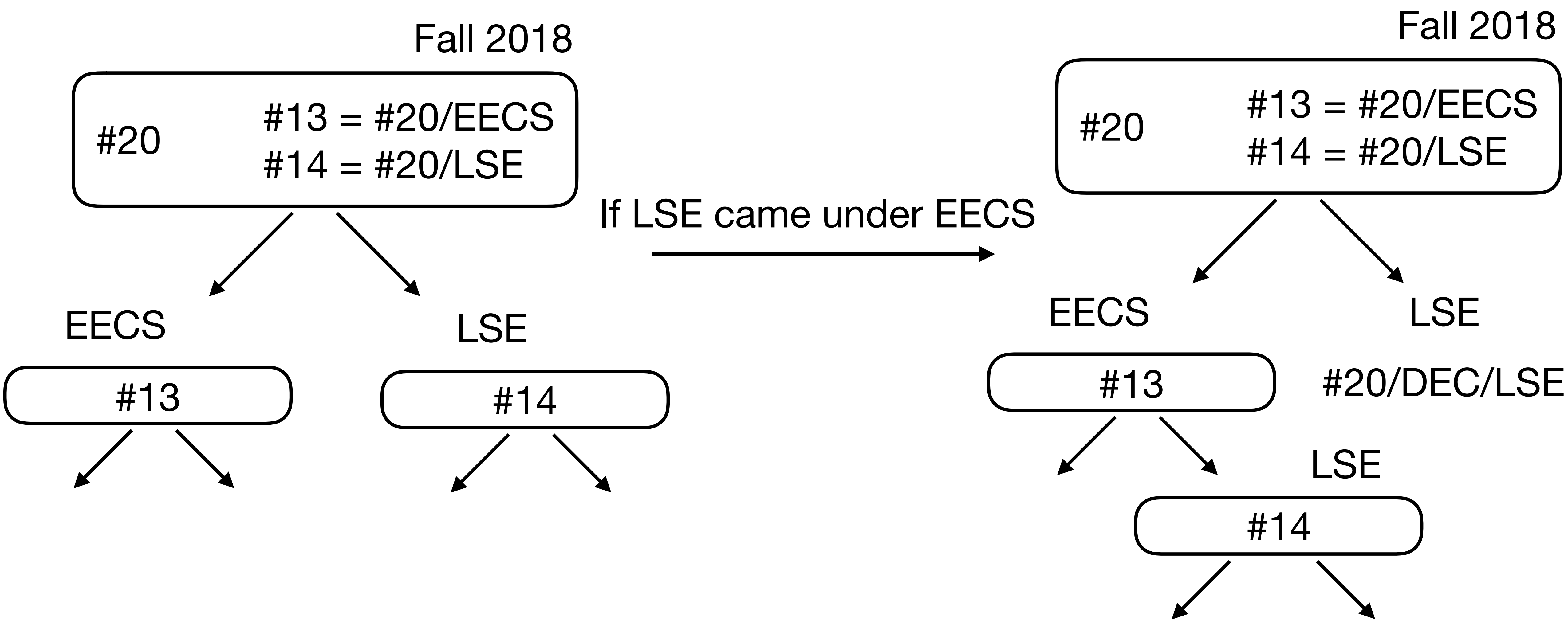
The Name Space

Growth of name space

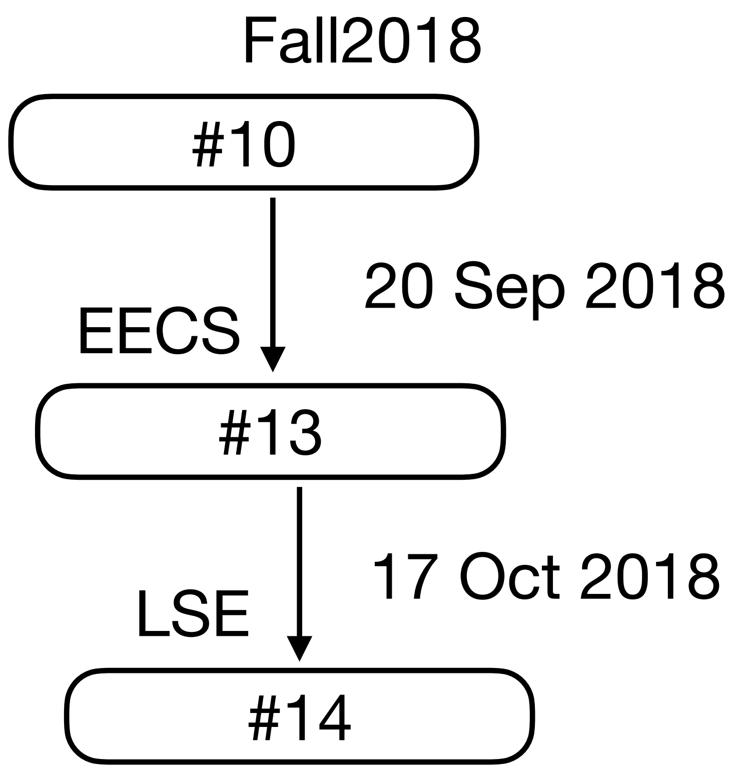
- Inside each directory, names can be generated without considering any other directory.
- Combining exiting name services?



Restructuring



Caching



#10/EECS = #13
Valid until 20 Sep 2018

#13/LSE = #14
Valid until 17 Oct 2018

#10/EECS/LSE = #14
Valid until 20 Sep 2018

The name service itself and authentication are also candidates for caching

The name service interface

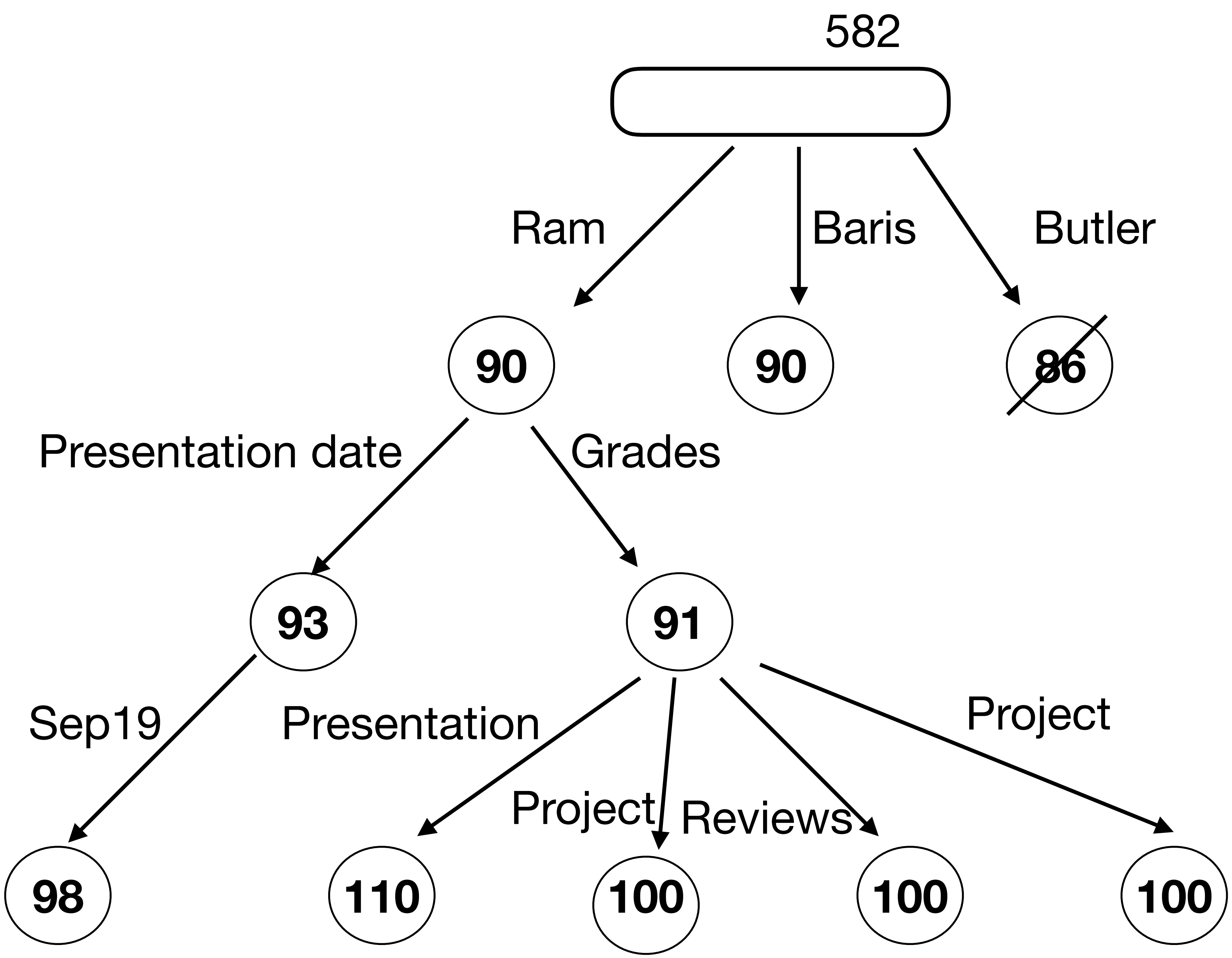
- The interface is based on remote procedure calls
- Example interfaces for
    - Reading values - GetValue(VD) -> Tree
    - Directories - NewD(SN,FN) -> DI
    - Directory Copies - Sweep(FN,SN) -> time-stamp
    - Servers - NewServer() -> Server Address

Specifications

Name Service Specifications

- Model of computation - Interleaving of atomic actions in separate processes
- Collection of predicates
- Two kinds - Invariants and post-conditions
- Predicates can be both variables in the program and auxiliary variables.
- DB - Function DI -> directory - Defines current content of database

Values and Updates



Values and Updates

- A value v is modified if an update operation names a node by giving path p (including TS) and says whether it is present or absent

- Update operation:

  1. Find arcs and nodes in v for longest prefix of p - v'

  2. If prefix is all of p and operation = present, then do nothing. Otherwise, replace v' with absent leaf node

  3. Otherwise, next element of p = (l,ts)

  4. If there is an arc labeled l from v' to a node with timestamp > ts, do nothing. Otherwise, remove subtree from v' labeled by l, and add subtree defined by p

- This specification is total, commutative and idompotent, so the order of applying updates doesn't matter.

Directories (D)

- D1 - The result of a read operation on directory d with identifier d.di is determined by the state of d after S set of updates are applied on D where S includes

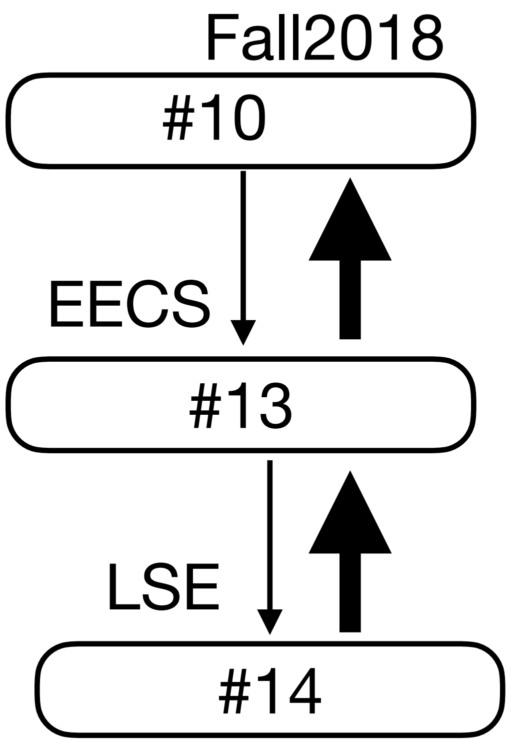
    1. All updates with timestamps less than d.lastSweep

    2. Arbitrary set of updates with timestamp greater than d.lastSweep

- D2 - Looking up FN di/n1/…/nk yields a directory if either di is the root and each nj is defined in di/n1/..n(j-1) or di is in the root’s well-known table and di’/…/ni’/n1/…nk satisfies the previous condition

Directories (D)

- Back-references (BRs) are child-parent arcs in the tree and is the primary invariant in ensuring the structure of the tree, while the DRs are secondary



- D3 - The D's defined in DB form a tree rooted in root whose arcs are DRs that are the reverse of the BR backpointers.

- D4 - Each DR is pointed to by a BR with a longer TX.

Directory Copies (DC)

- DC1 - The database value of a directory DB(di) = Union(Updates in all DCs)
- DC2 - There is always at least one complete ring in the set of DCs for a directory
- DC3 - There are never two complete non-intersecting rings

Servers (S)

- Can reference server names through directory references. This might create infinite loops. Server invariants solve this problem by using a boolean inSN in each directory. A directory cannot be part of a server name unless its inSN is true.

- S1 - All d's from root to an entry that stores a server address has d.inSN = true

- S2 - If d.inSN is true, then either d is a root, or a copy of d is is stored on a server with a name shorter than any direct name of d

- S3 - Every server s either stores the root, or s.up is a shorter name of another server, and s stores a copy of the directory for s.up

Conclusion

Formal specifications for functional requirements of a name service that supports growth and evolution of the name space.

Discussion

Strengths

- Decentralized management -> High replication -> large size and high availability
- Hierarchical design supports growth, change in structure and fault isolation
- Authentication and access control mechanisms are easy to implement
- Simple interface
- Exact specifications for the name service

Weaknesses

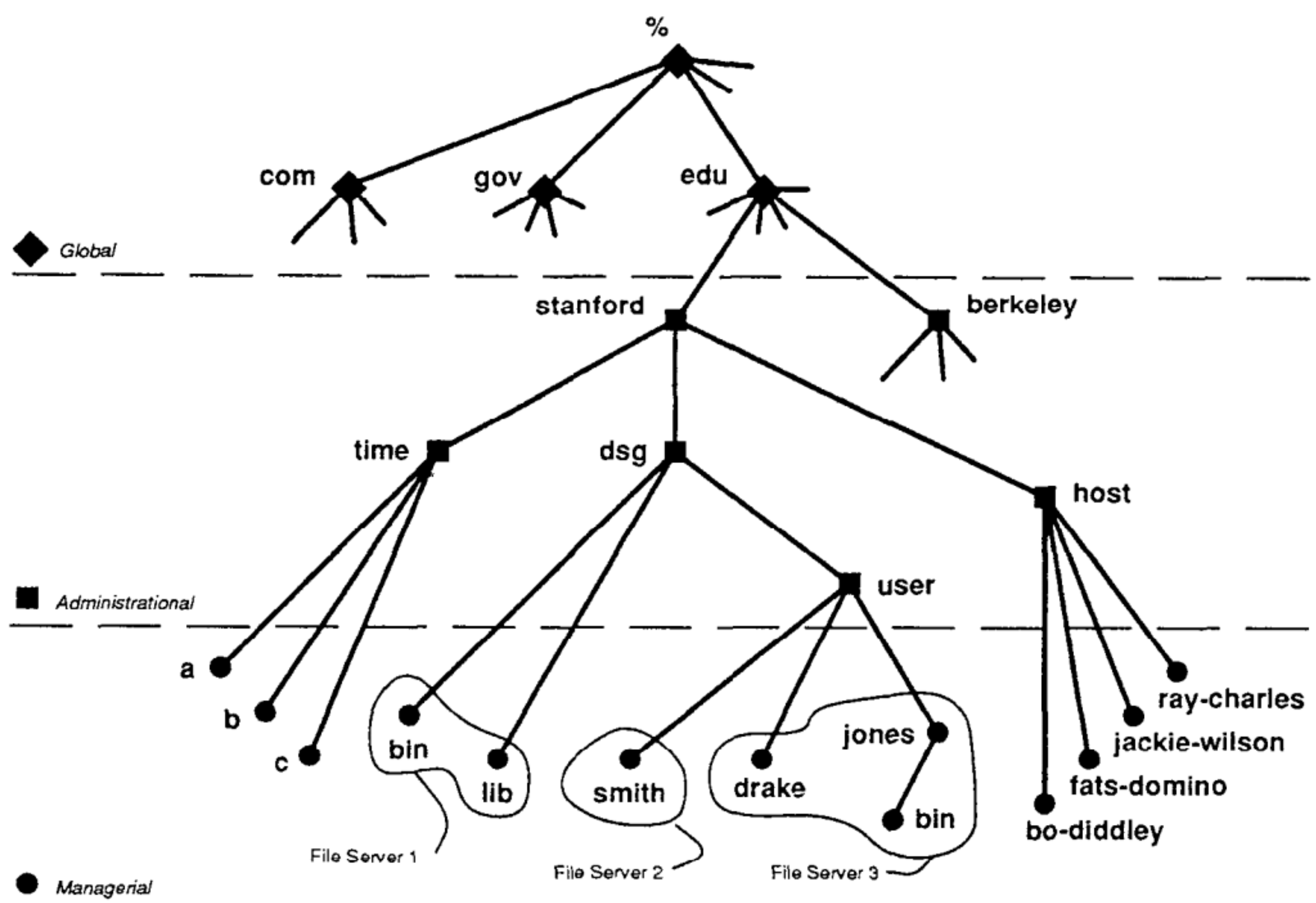
- Frequent changes may cause database to be outdated
- Cost of update propagation might go up as size increases
- Security issues
- Reforming rings (starting a new epoch) needs an administrator, so cannot be fully automated

Open questions

- Why can't direct message updates be made reliable?
- How do current name services like the DNS compare to this design?
- Can this design be practically implemented for efficient name lookup?
- Space and computation requirements on the servers? How many must there be to enable fast lookup?

More on name services

"Decentralizing a Global Naming Services for Improved Performance and Fault Tolerance," (Cheriton & Mann,1989)



More on name services

- P. Mockapetris and K. J. Dunlap. 1988. **Development of the domain name system**. In Symposium proceedings on Communications architectures and protocols (SIGCOMM '88), Vinton Cerf (Ed.). ACM, New York, NY, USA, 123-133. DOI=http://dx.doi.org/10.1145/52324.5233

- More recently, there has been work on developing name services designed for high mobility and security.

- A. Venkataramani *et al*., "**Design requirements of a global name service for a mobility-centric, trustworthy internetwork**," *2013 Fifth International Conference on Communication Systems and Networks (COMSNETS)*, Bangalore, 2013, pp. 1-9.