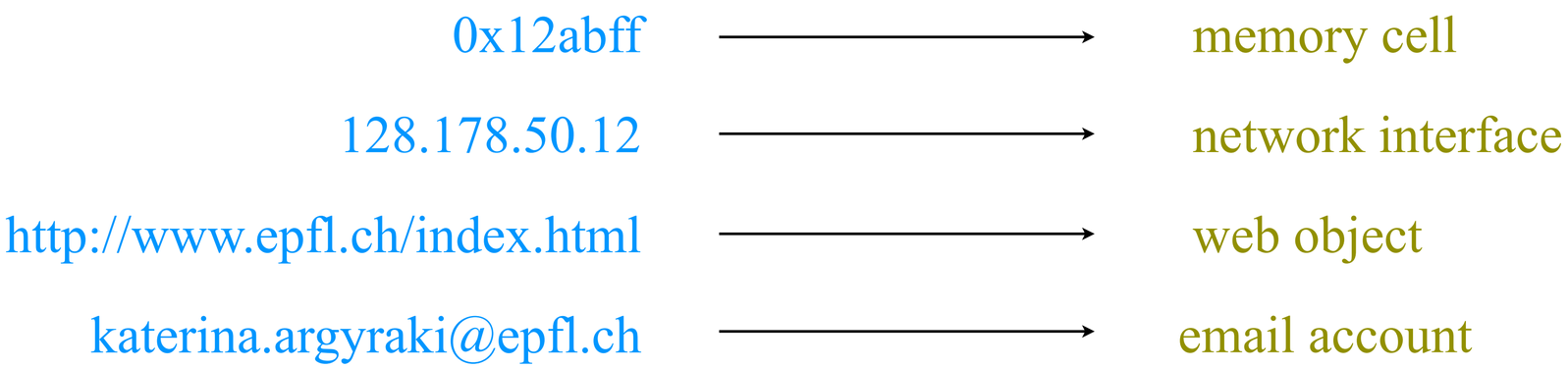
EPFL

Principles of Computer Systems: Names

Prof. Katerina Argyraki
School of Computer & Communication Sciences

Introduction



A name is a way to refer to a resource

network arch. lab

0x12aadd

0x1348ad

0x12aadd

Zeinab Shmeis

PhD student

enrolled 2018

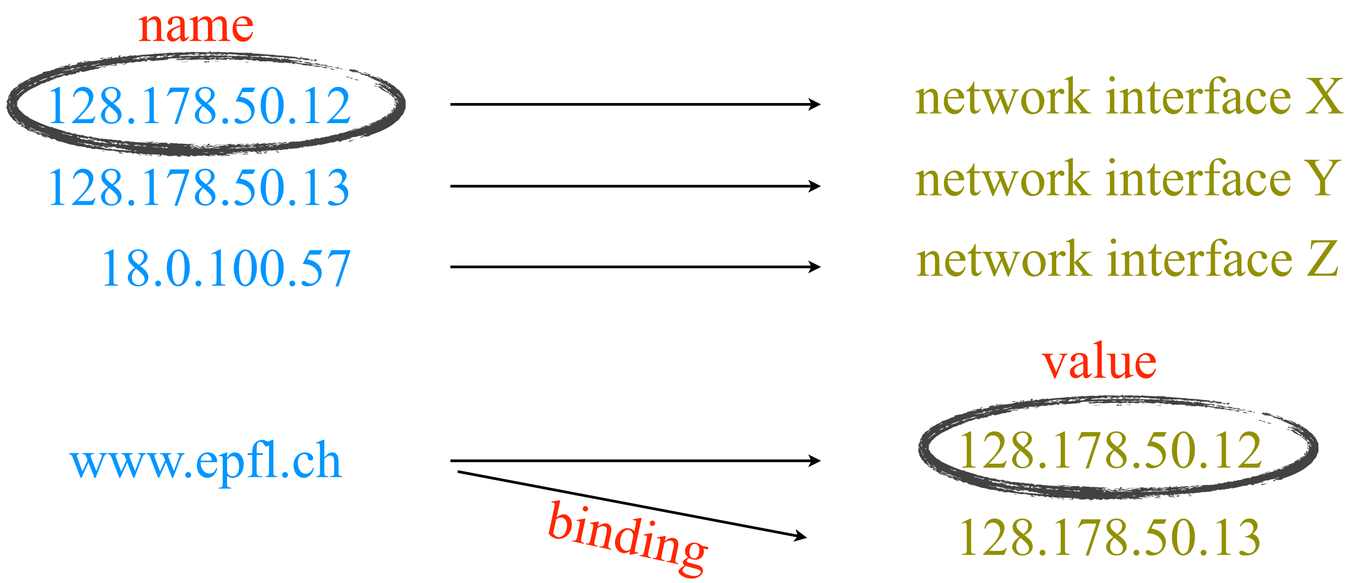
0x1348ad

Zhiyong Zhang

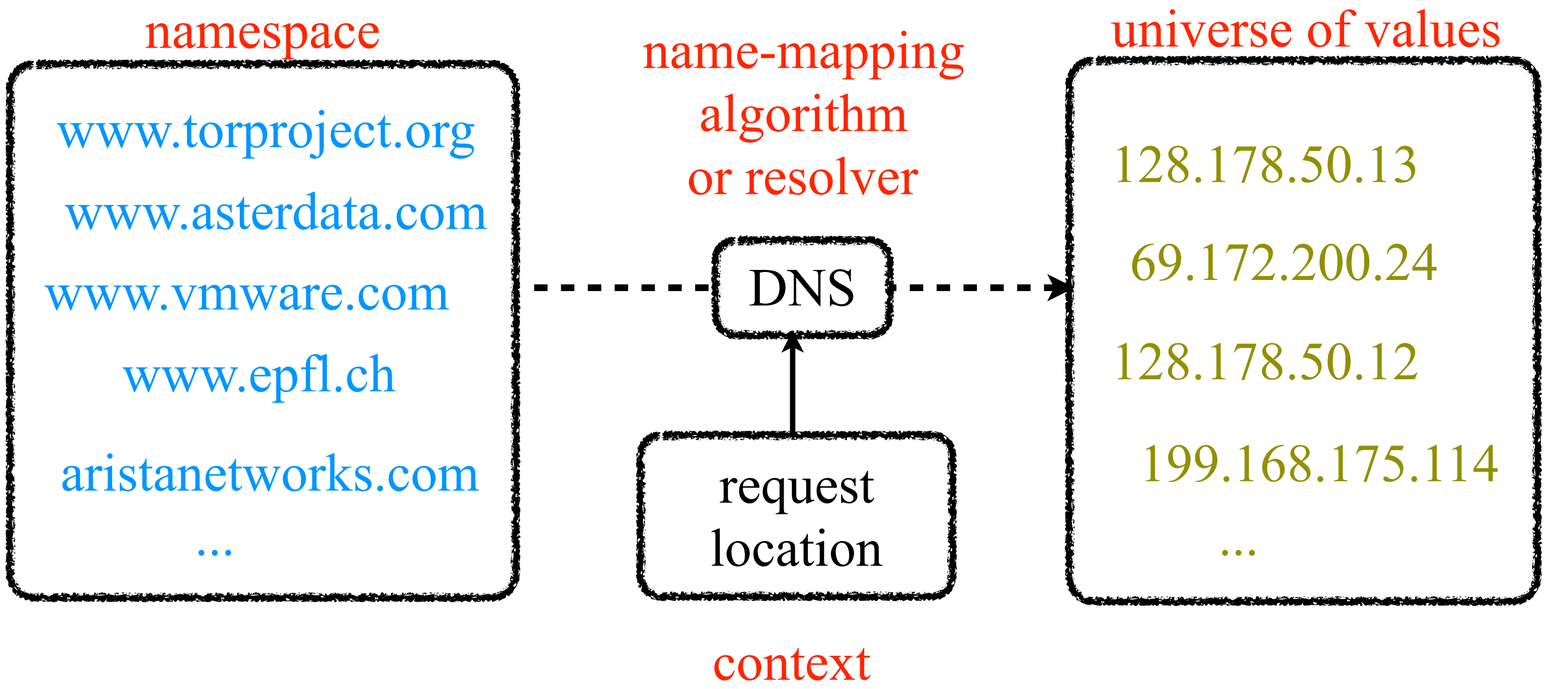
visiting student

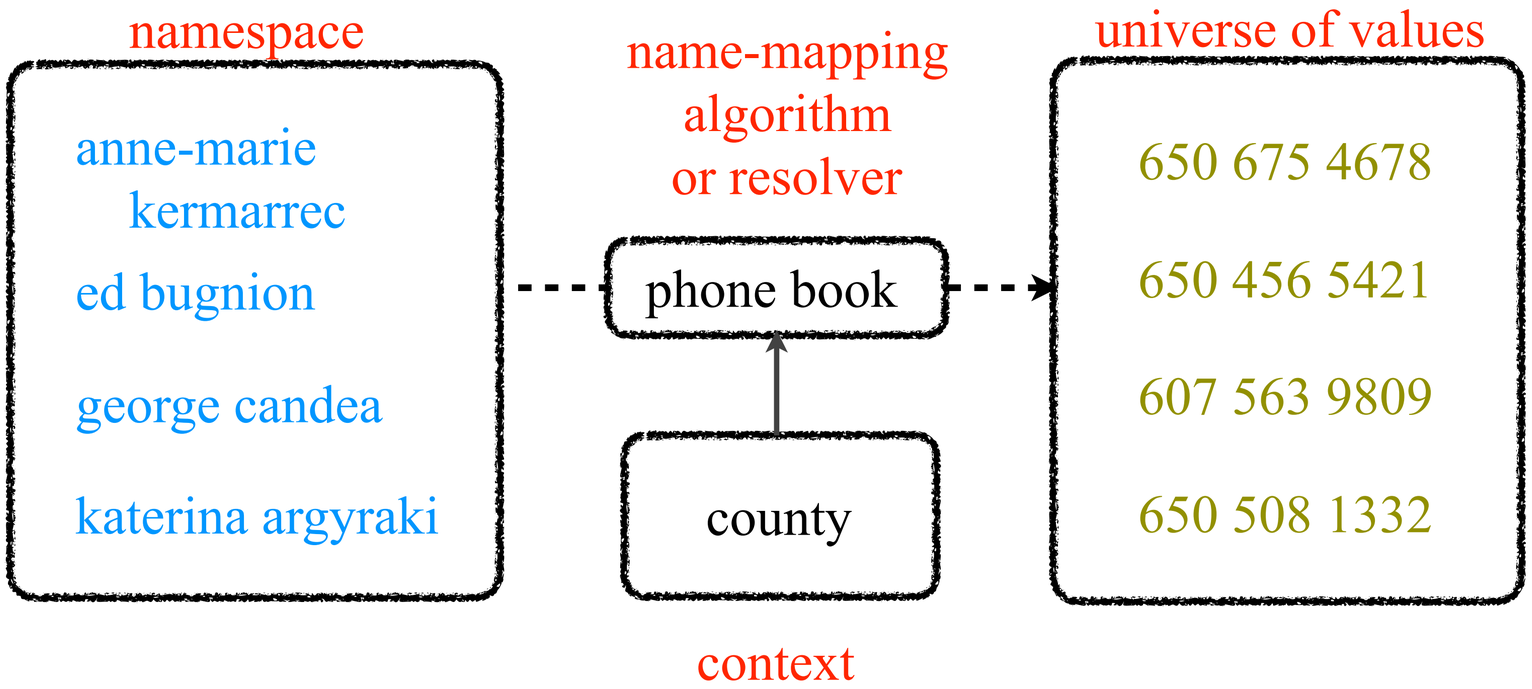
enrolled 2020

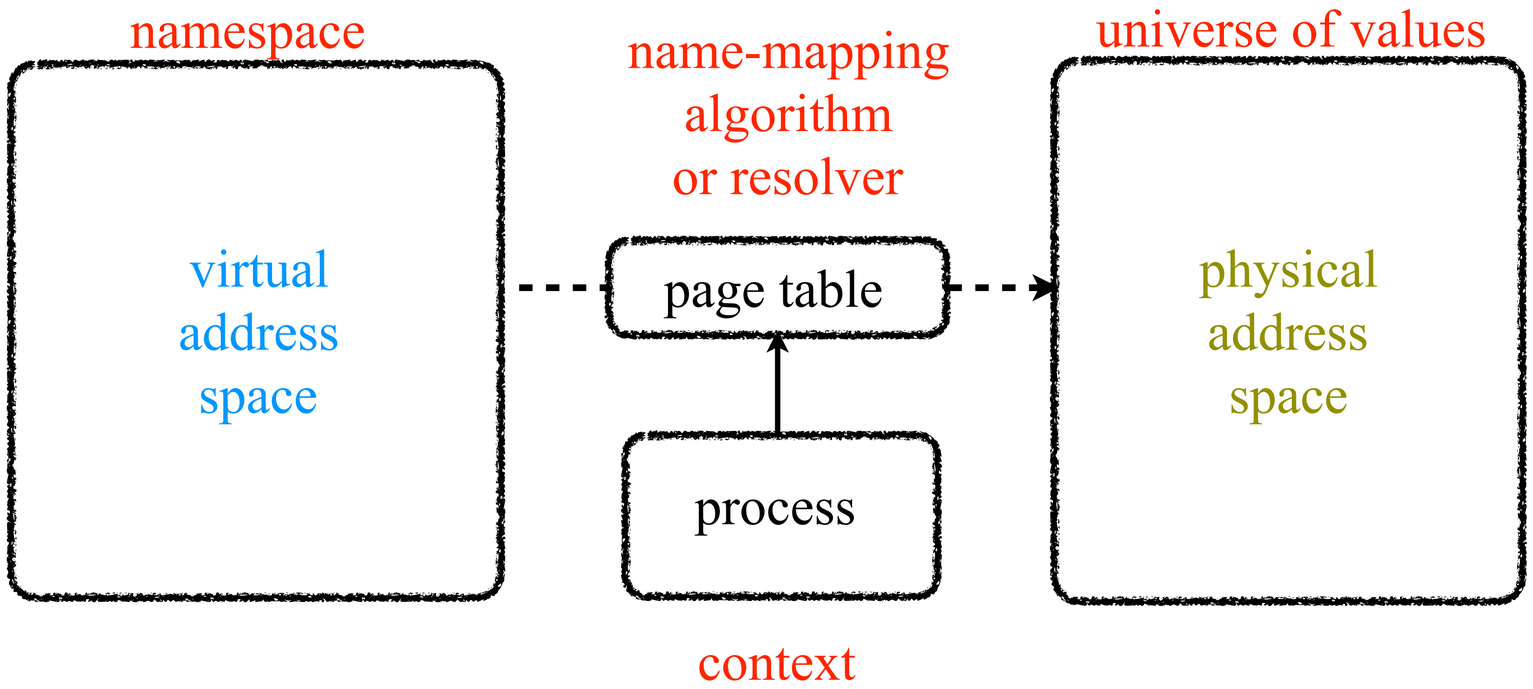
For efficient communication and organization



For indirection


namespace
www.torproject.org
www.asterdata.com
www.vmware.com
www.epfl.ch
aristanetworks.com
...
name-mapping
algorithm
or resolver
DNS
request
location
context
universe of values
128.178.50.13
69.172.200.24
128.178.50.12
199.168.175.114
...


namespace
anne-marie kermarrec
ed bugnion
george candea
katerina argyraki
name-mapping algorithm or resolver
phone book
county
context
universe of values
650 675 4678
650 456 5421
607 563 9809
650 508 1332

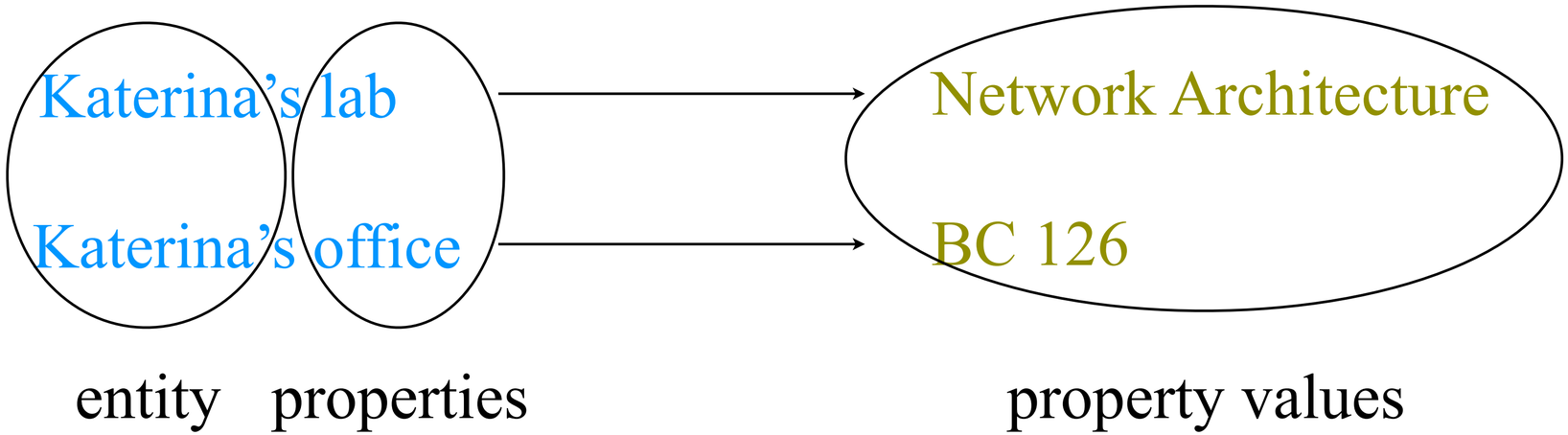

namespace
name-mapping
algorithm
or resolver
universe of values
virtual
address
space
page table
physical
address
space
process
context

Name types

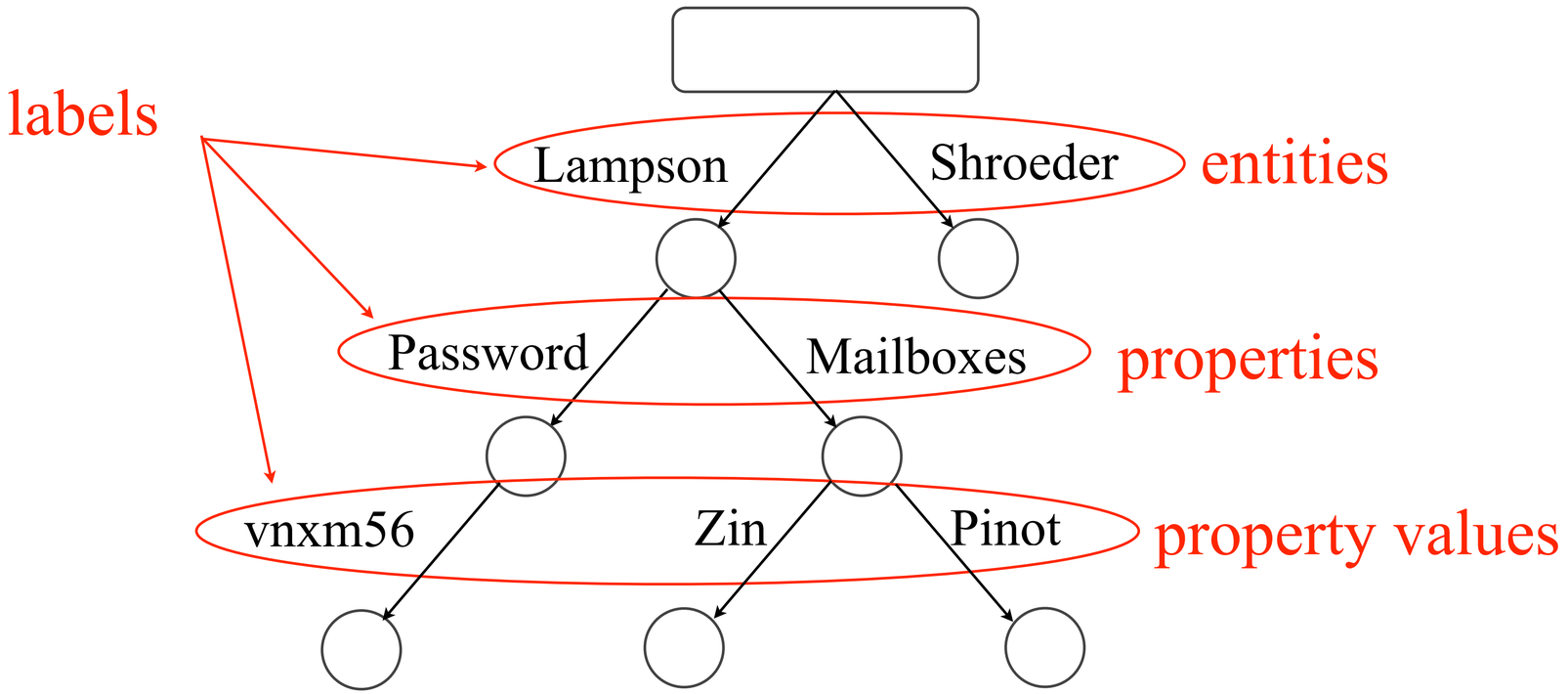
- Private: unique within a context
  - *e.g., a private IP address is unique within an organization*
- Global: unique across contexts
  - *e.g., a global IP address is unique within the Internet*

- Hierarchical: name relationship implies object relationship
  - *e.g., two IP addresses sharing the same prefix*
- Flat: name relationship implies nothing
  - *e.g., content IDs in Peer-to-Peer networks (*)*

Designing a Global Name Service



- A name service maps names to values
- In this paper, a name is hierarchical, typically consisting of an entity and a property

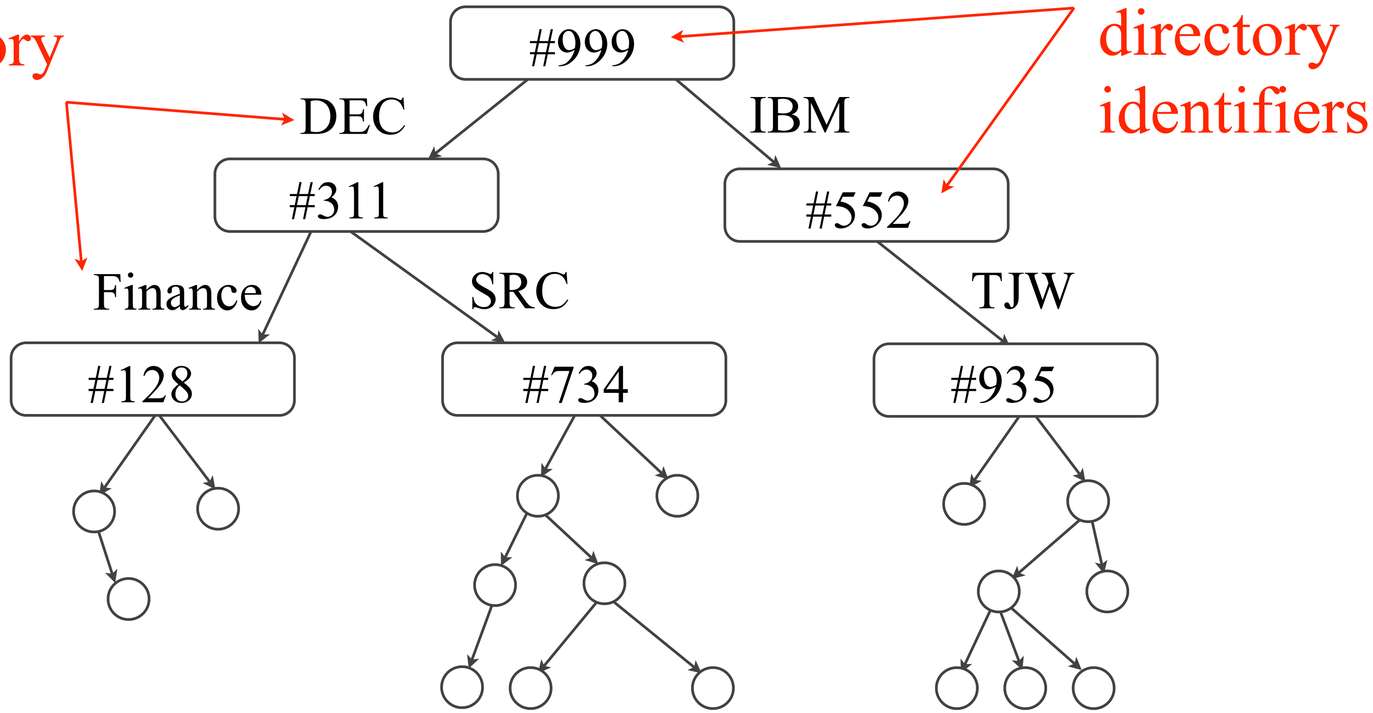

labels
Lampson
Shroeder
entities
Password
Mailboxes
properties
vnxm56
Zin
Pinot
property values

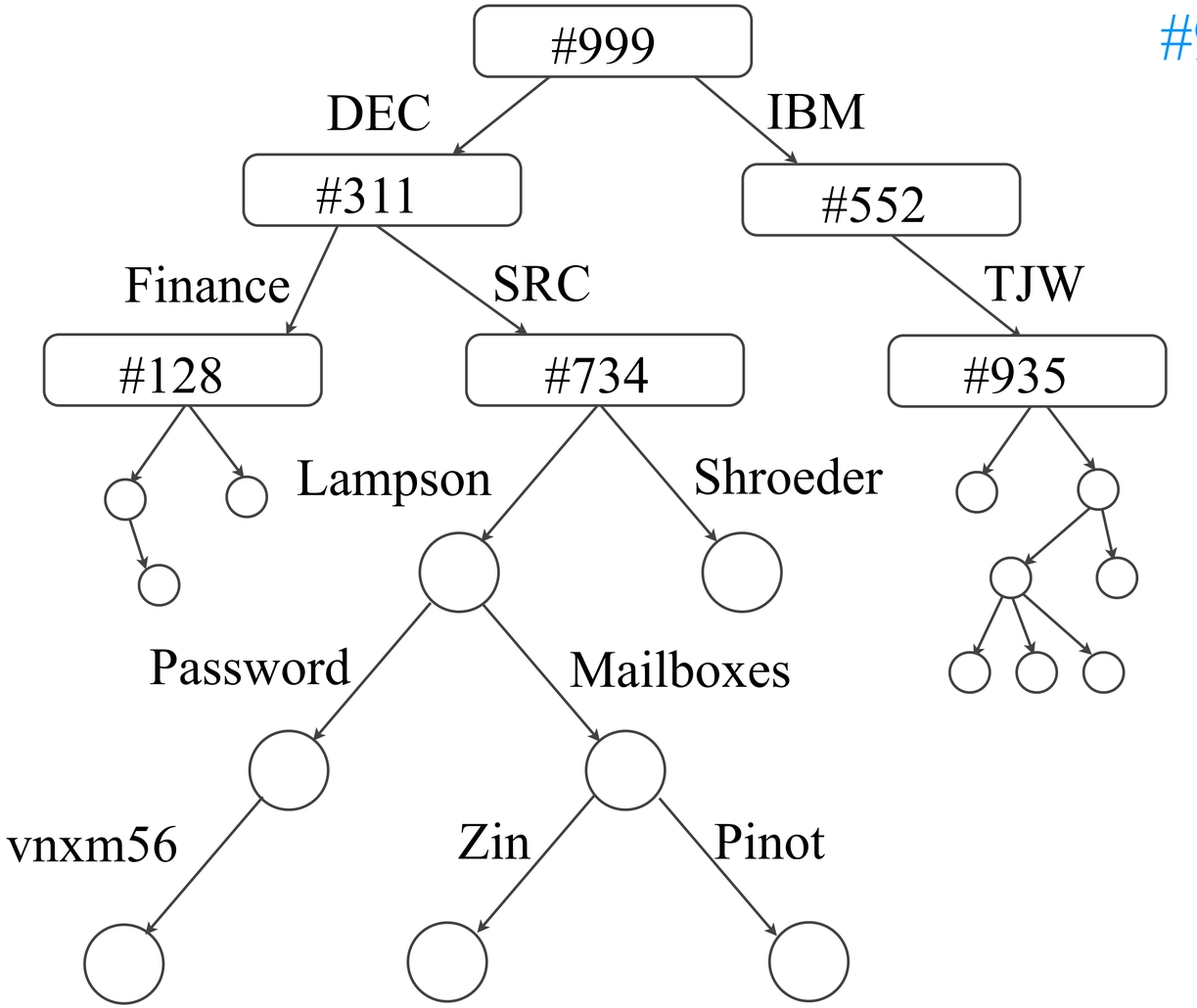
Design goal #1: scalability

- Must support an arbitrary number of names + administrative organizations

local directory names

global directory identifiers

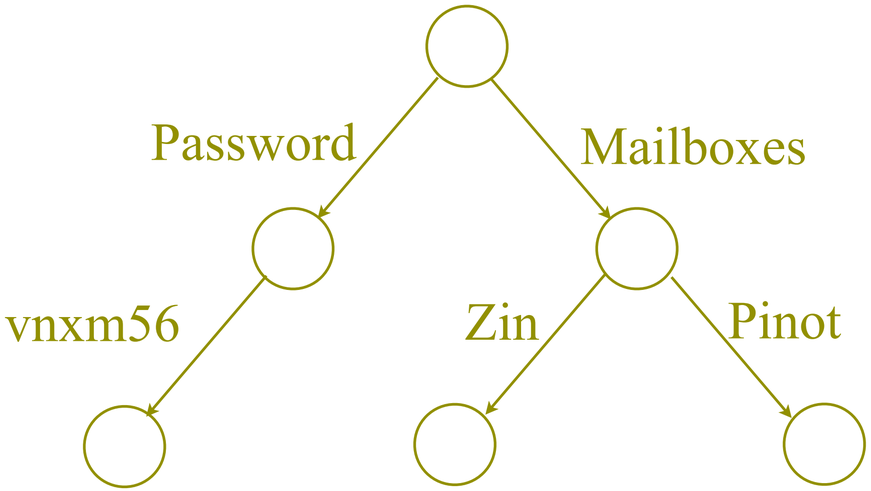




#999/DEC/SRC/Lampson/Password

vnxm56

#999/DEC/SRC/Lampson

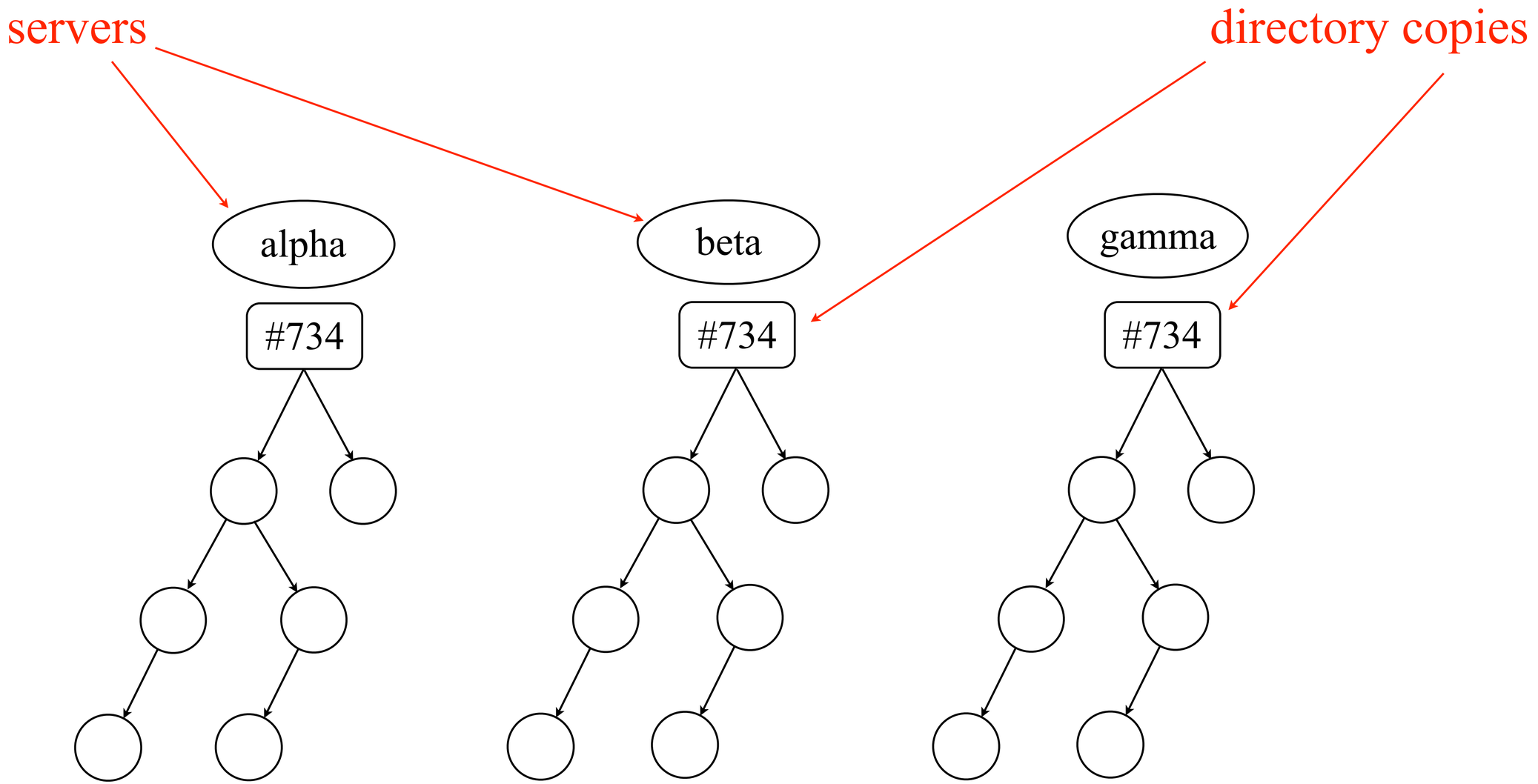


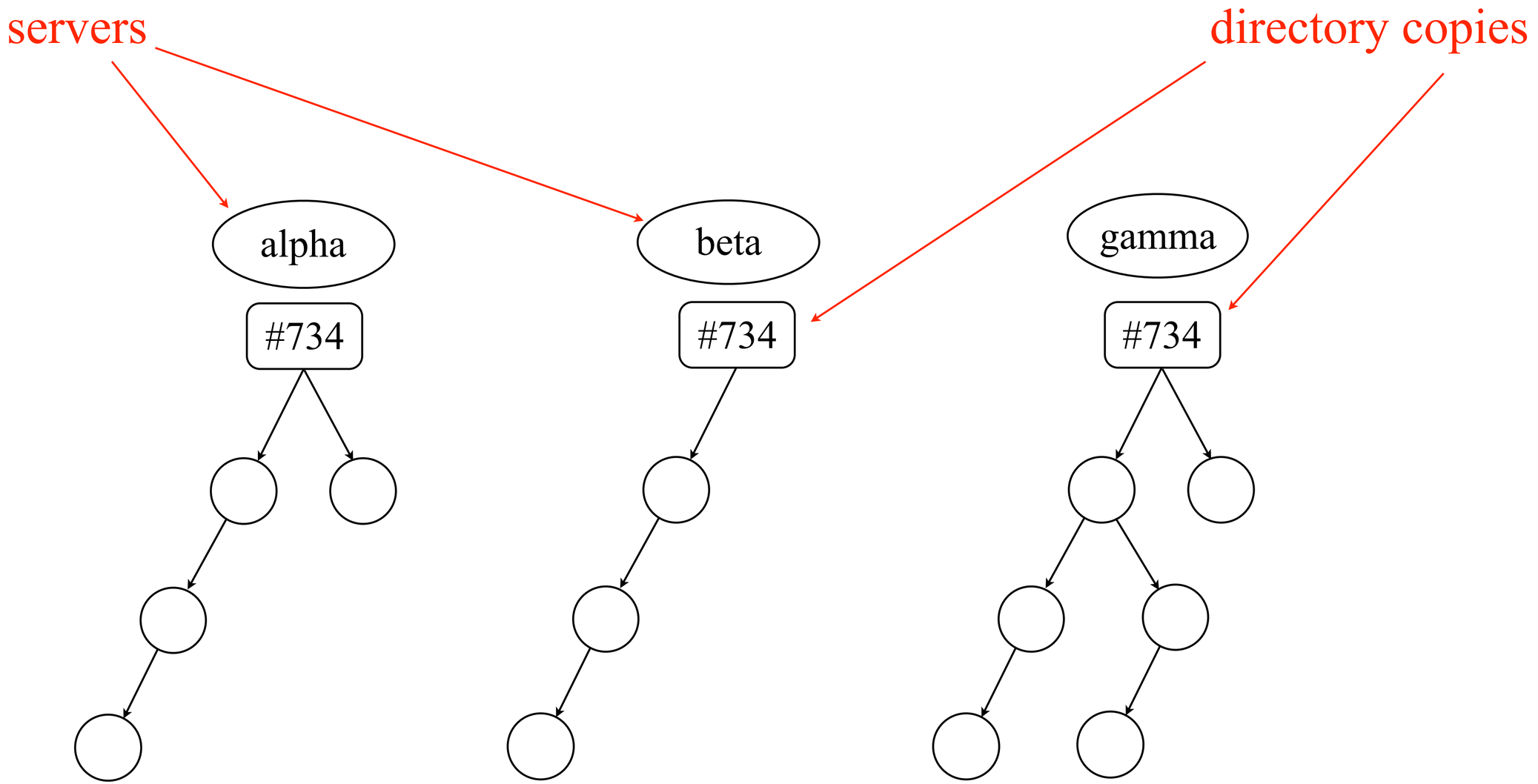
Design goal #1: scalability

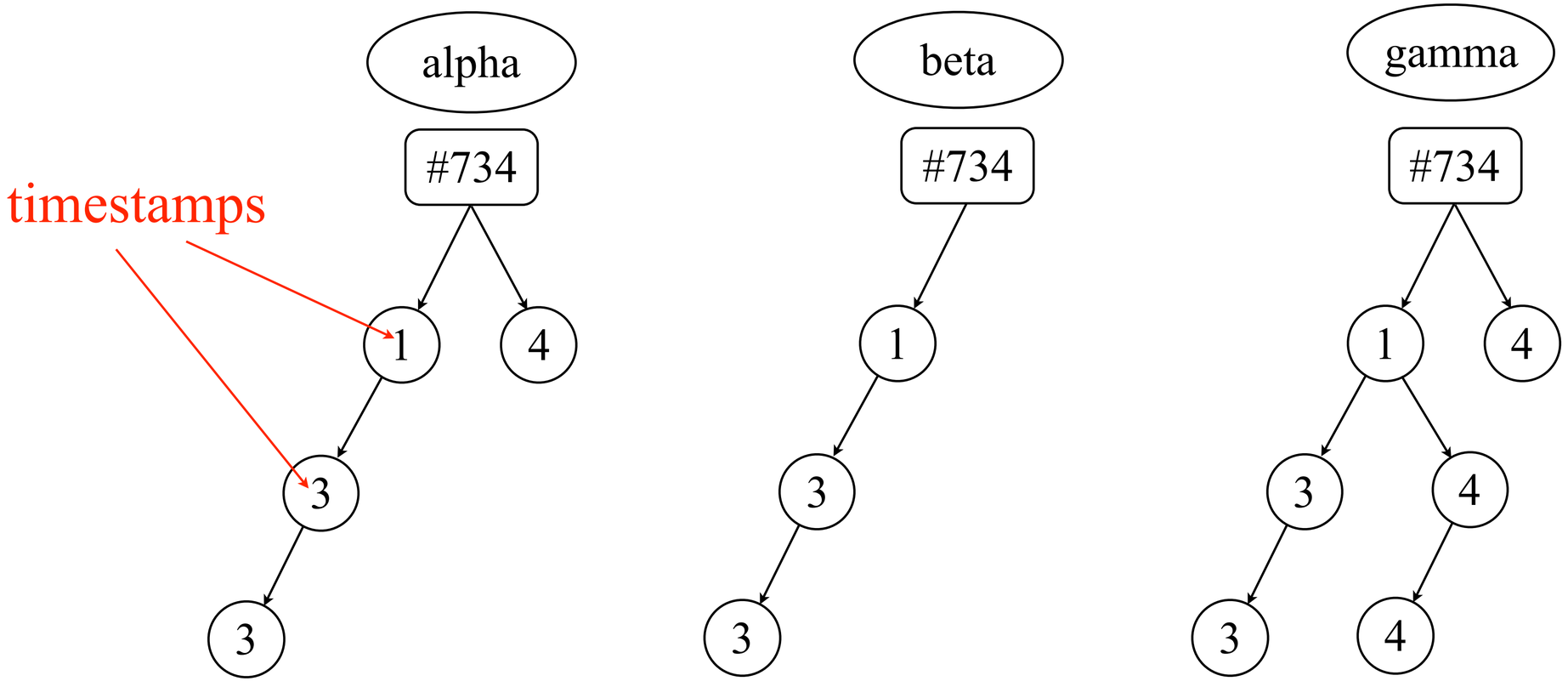
- Achieved through a hierarchy of directories, each potentially owned by a different entity, each with a private namespace

Design goal #2: fault tolerance

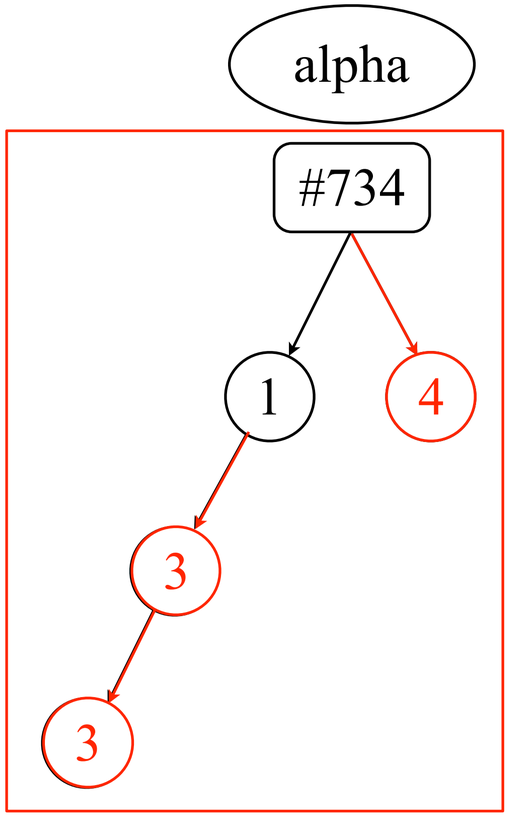
- The service should operate successfully even if N of its servers fail


servers
directory copies
alpha
beta
gamma
#734
#734
#734


servers
directory copies
alpha
beta
gamma
#734
#734
#734

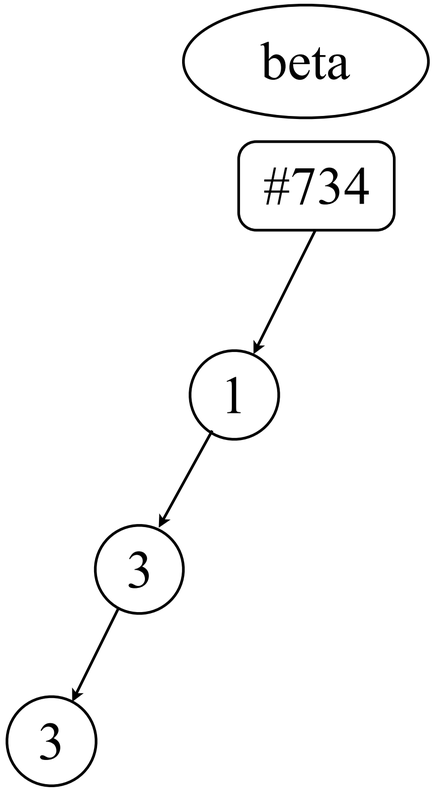

timestamps
alpha
#734
1
4
3
3
beta
#734
1
3
3
gamma
#734
1
4
3
4
3
4

lastSweep: 2

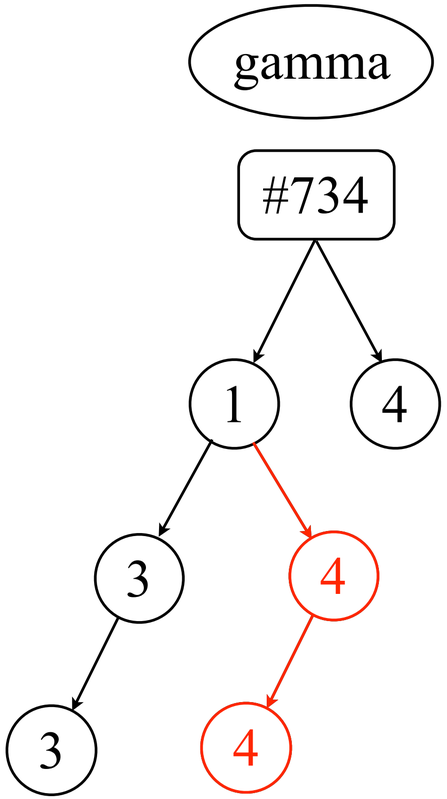


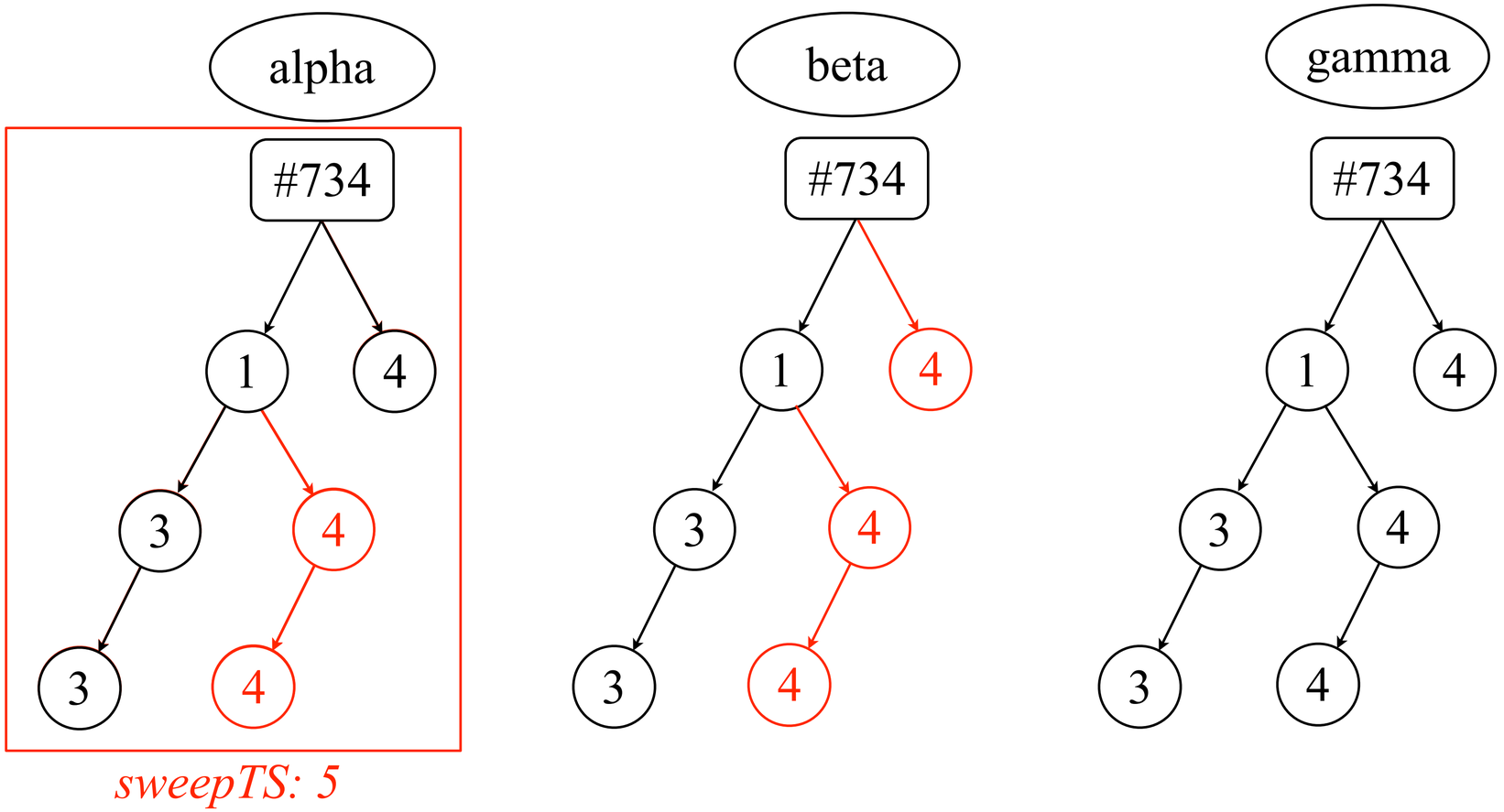
sweepTS: 5

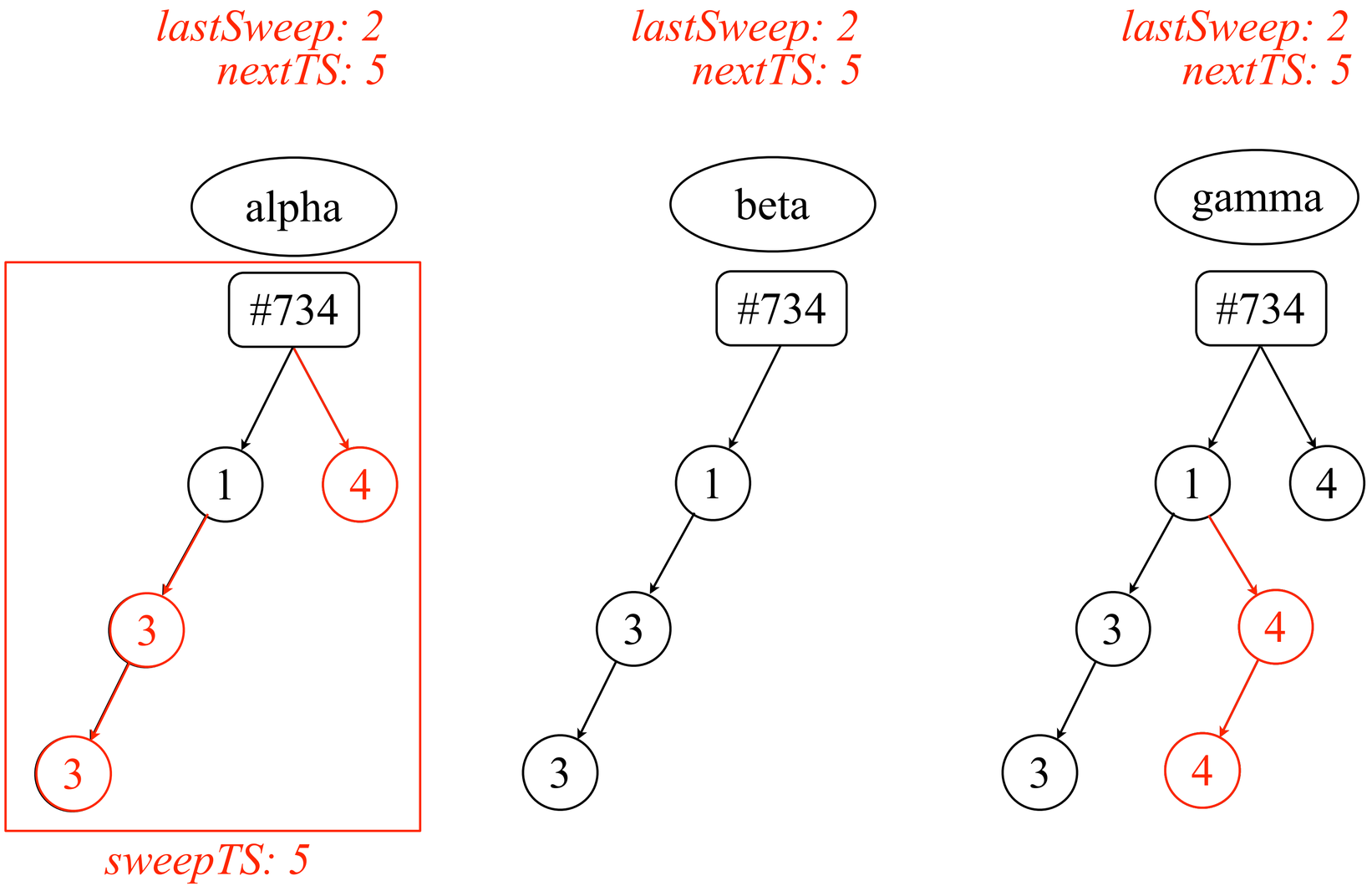
lastSweep: 2

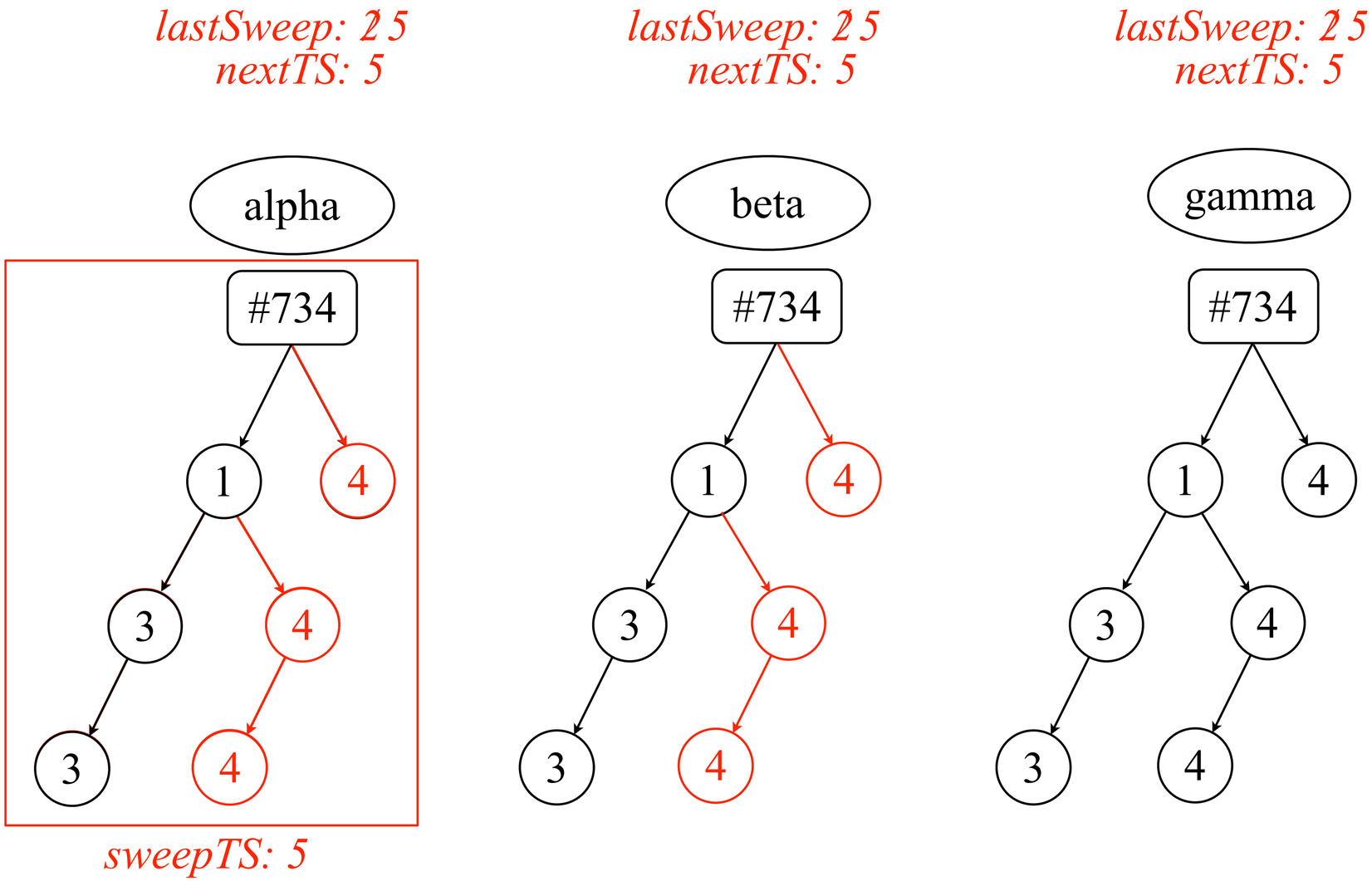


lastSweep: 2




lastSweep: 2 5
alpha
#734
1
4
3
4
3
4
sweepTS: 5
lastSweep: 2 5
beta
#734
1
4
3
4
3
4
lastSweep: 2 5
gamma
#734
1
4
3
4
3
4

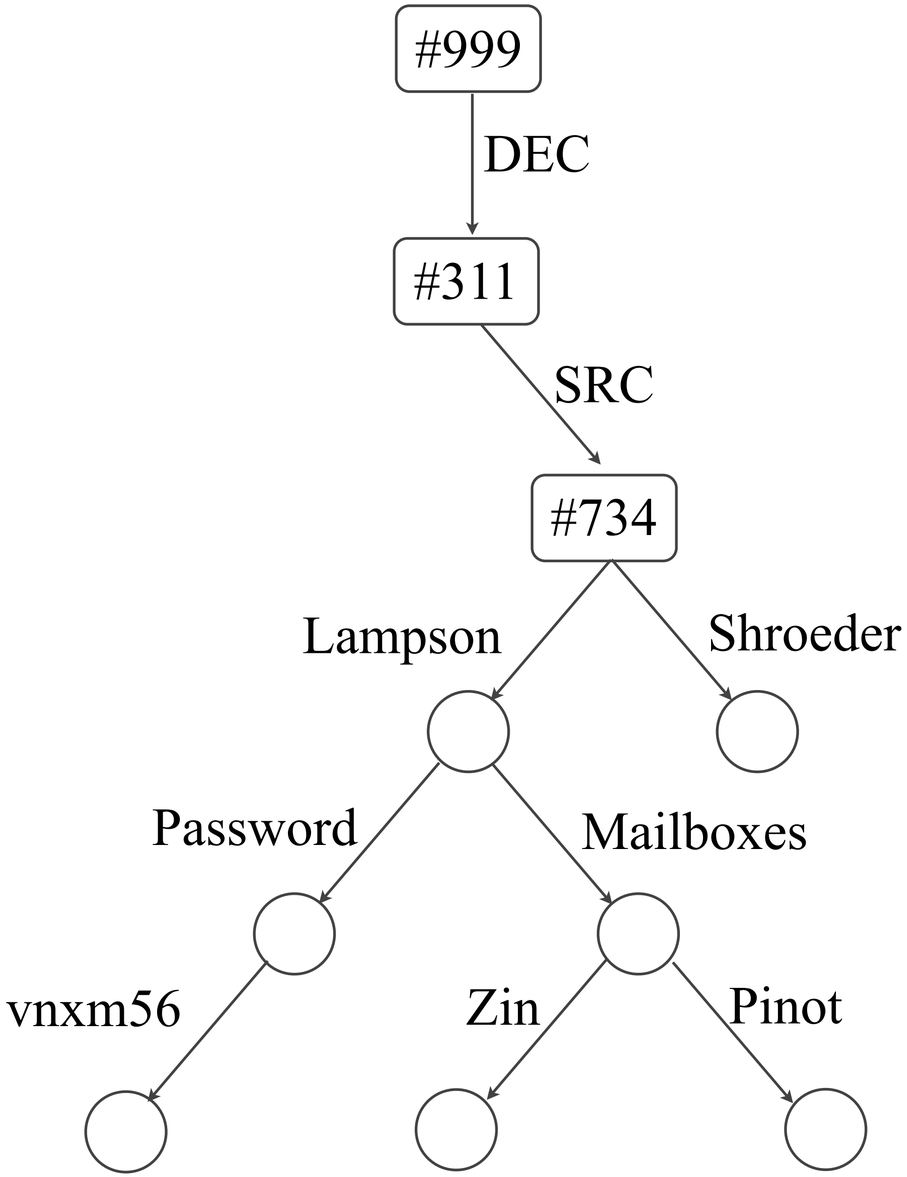

lastSweep: 2
nextTS: 5
alpha
#734
1
4
3
3
sweepTS: 5
lastSweep: 2
nextTS: 5
beta
#734
1
3
3
lastSweep: 2
nextTS: 5
gamma
#734
1
4
3
4
3
4


lastSweep: 2 5
nextTS: 5
alpha
#734
1
4
3
4
3
4
sweepTS: 5
lastSweep: 2 5
nextTS: 5
beta
#734
1
4
3
4
3
4
lastSweep: 2 5
nextTS: 5
gamma
#734
1
4
3
4
3
4

Design goal #2: fault tolerance

- Achieved through redundancy
  - *redundant directory copies on different servers*
- Combined with eventual consistency
  - *copies periodically synchronized through sweep*

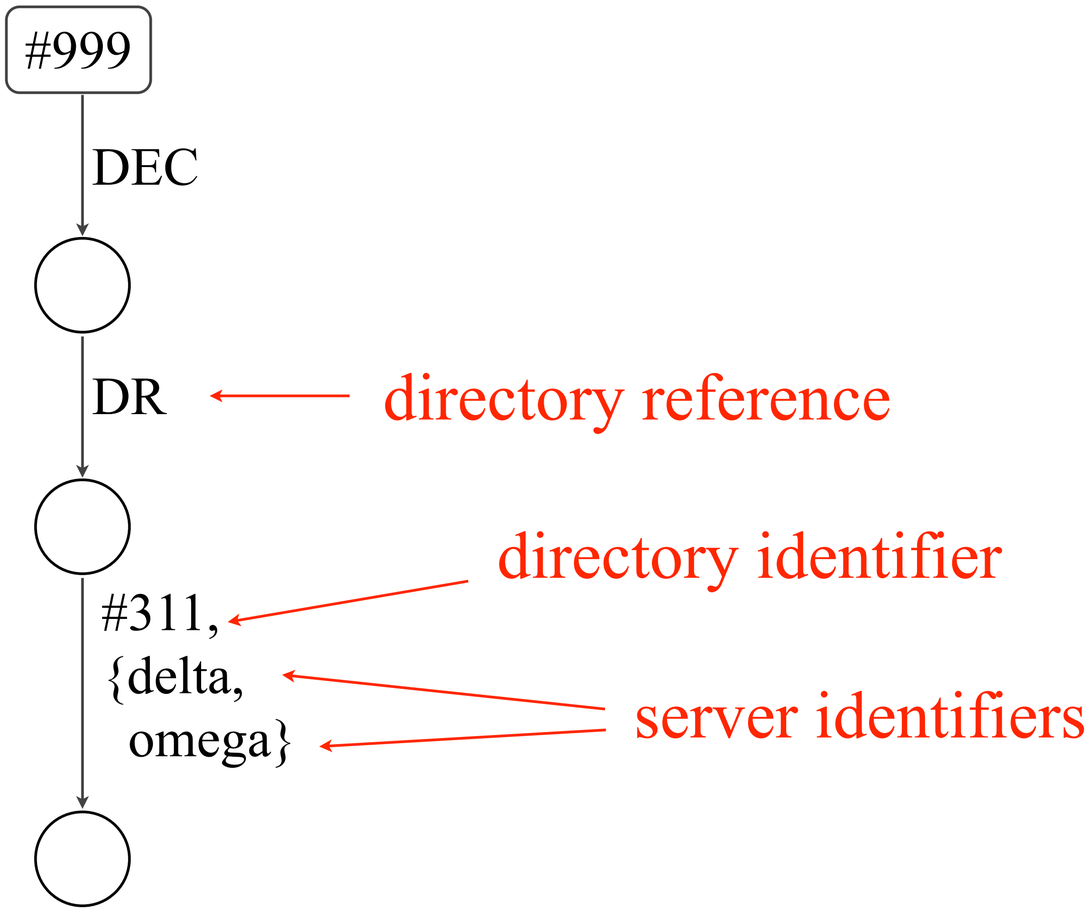
Name lookup



Name Value

#999/DEC/SRC/Lampson/Password

vnxm56

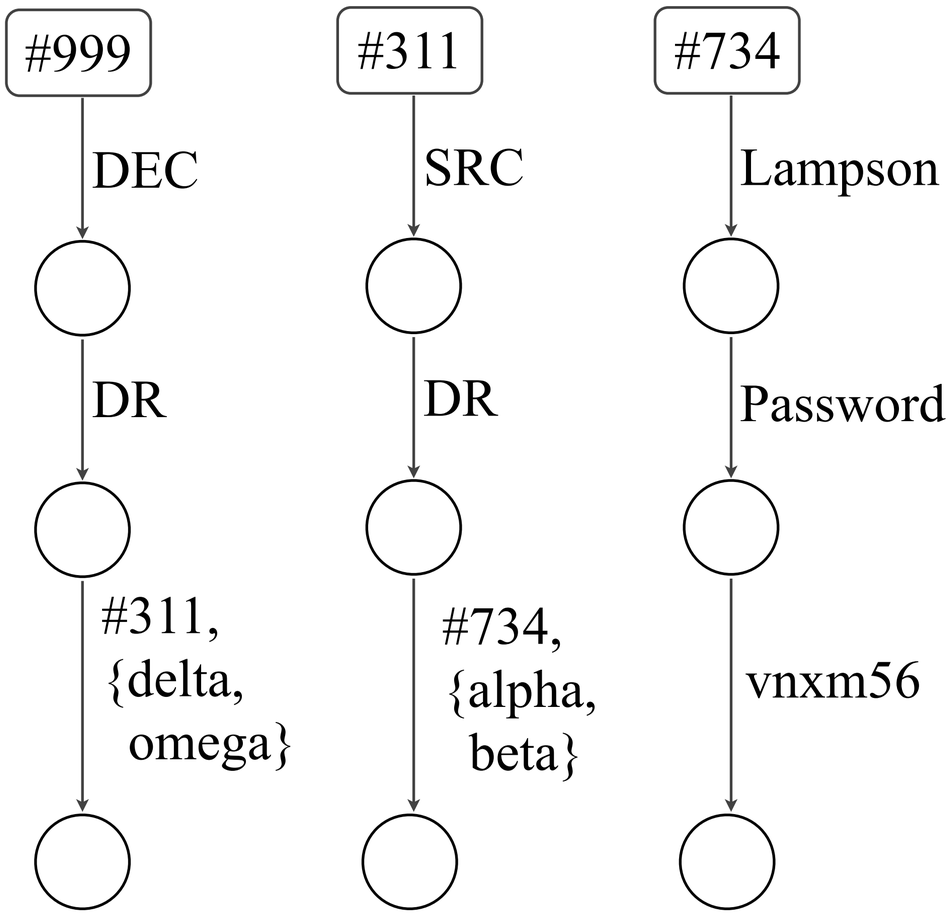


Names Values

#999/DEC/SRC/Lampson/Password

#999/DEC/DR

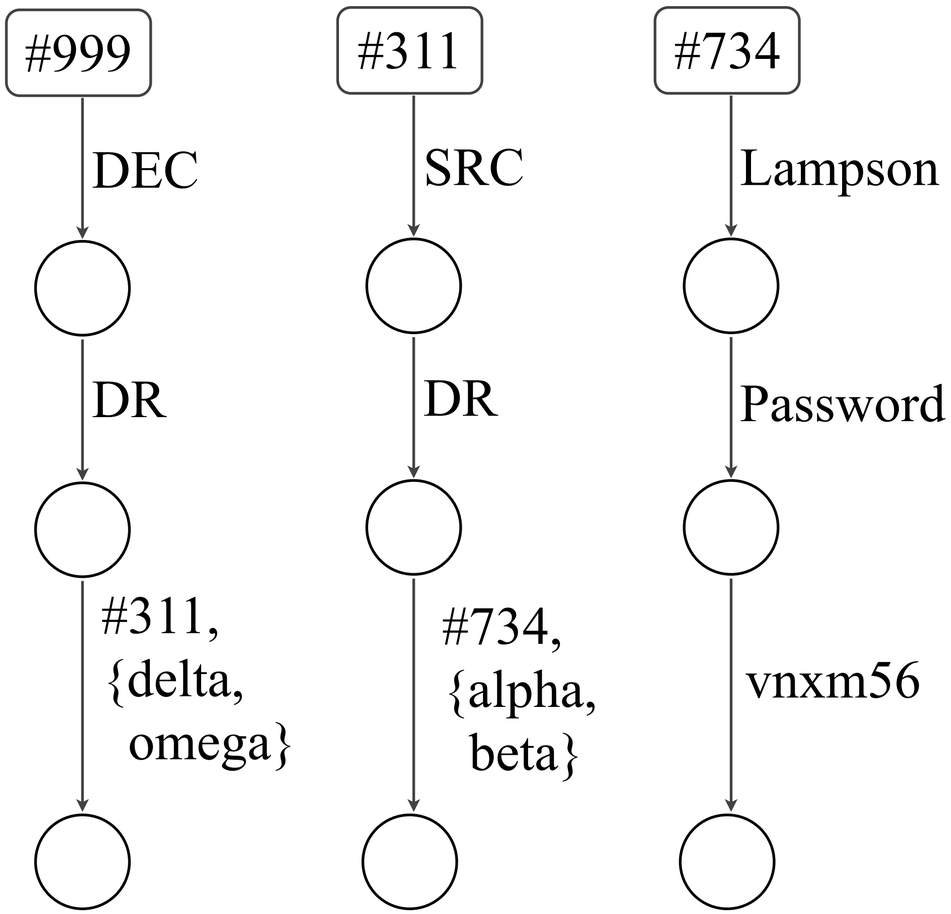
#311, {delta, omega}



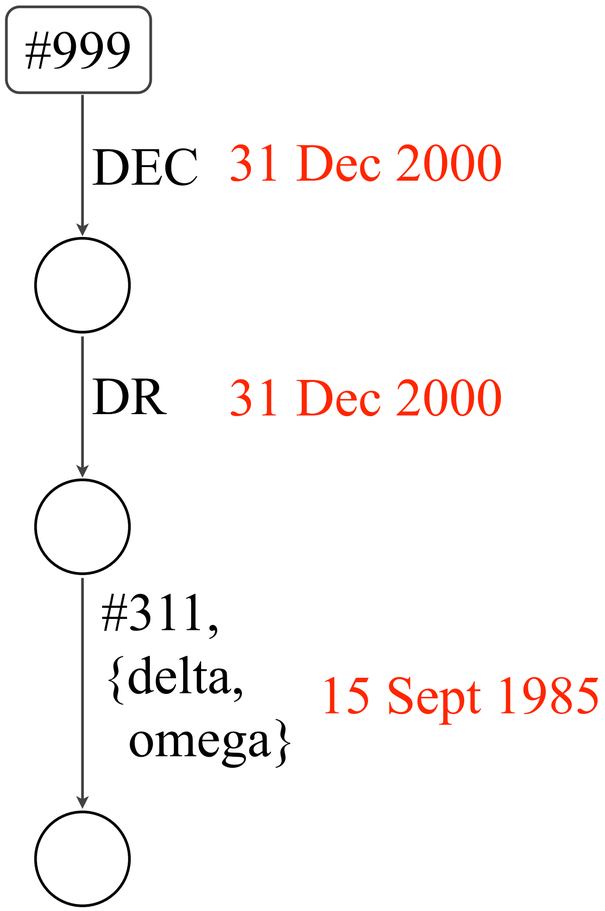
Names	Values
#999/DEC/SRC/Lampson/Password	
#999/DEC/DR	#311, {delta, omega}
#311/SRC/DR	#734, {alpha, beta}
#734/Lampson/Password	vnxm56

Name lookup

- The name service uses itself as a name service



Names	Values
#999/DEC/SRC/Lampson/Password	
#999/DEC/DR	#311, {delta, omega}
#311/SRC/DR	#734, {alpha, beta}
#734/Lampson/Password	vnxm56



Names Values

#999/DEC/SRC/Lampson/Password

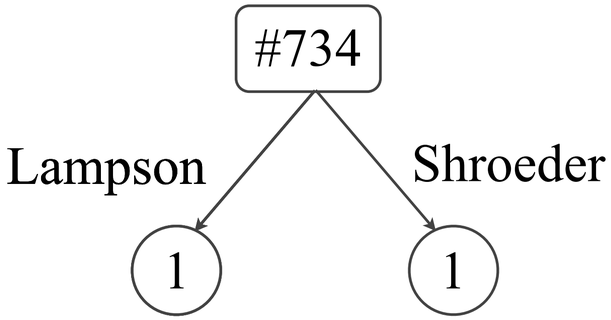
#999/DEC/DR

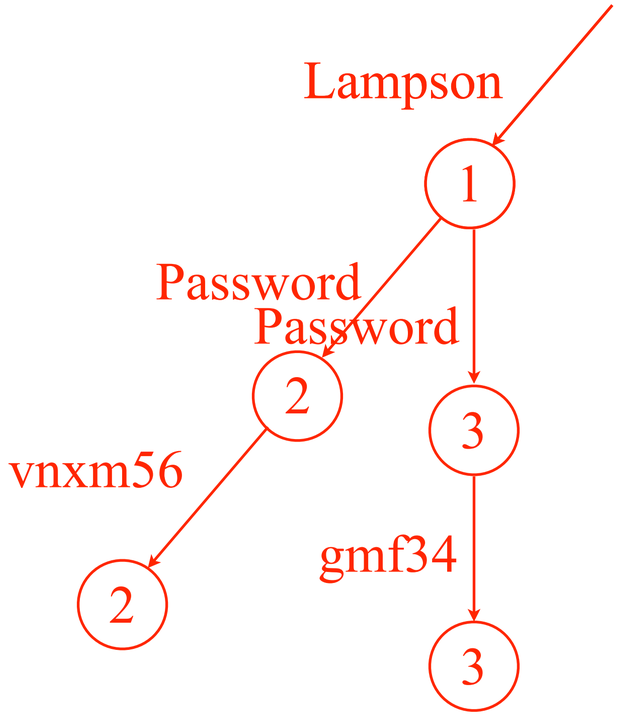
#311, {delta, omega}

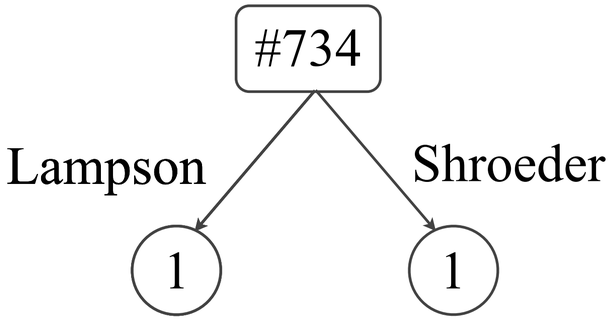
Name lookup

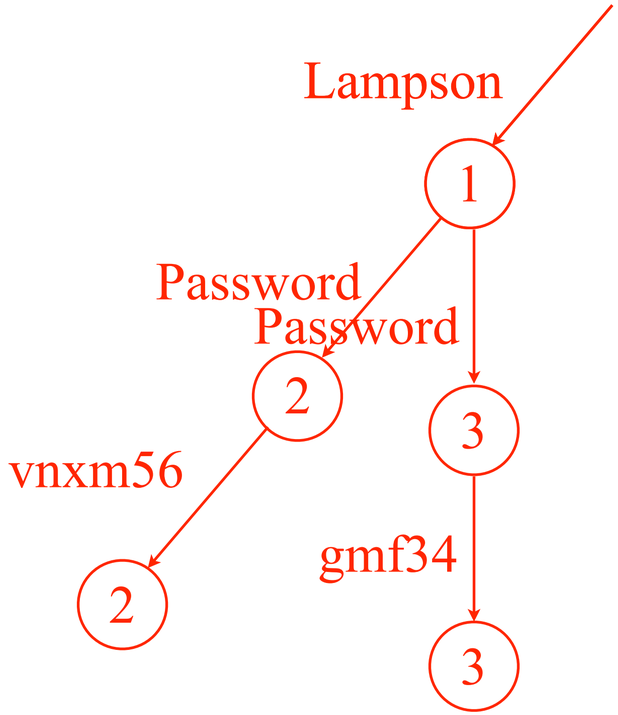
- The name service uses itself as a name service
- Clients can cache mappings to reduce latency
- Stale mappings avoided through expiration times

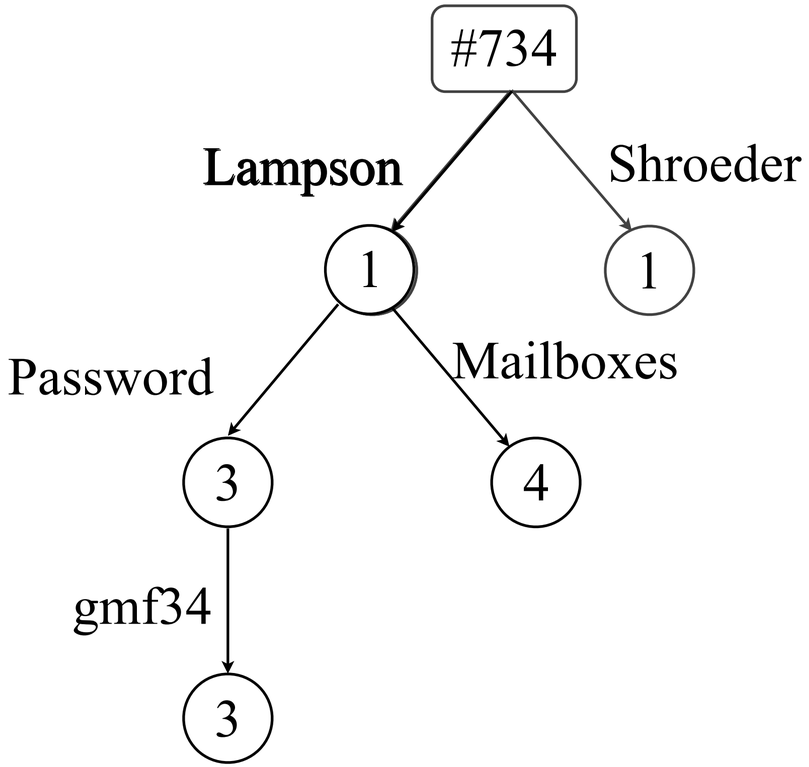
Updates

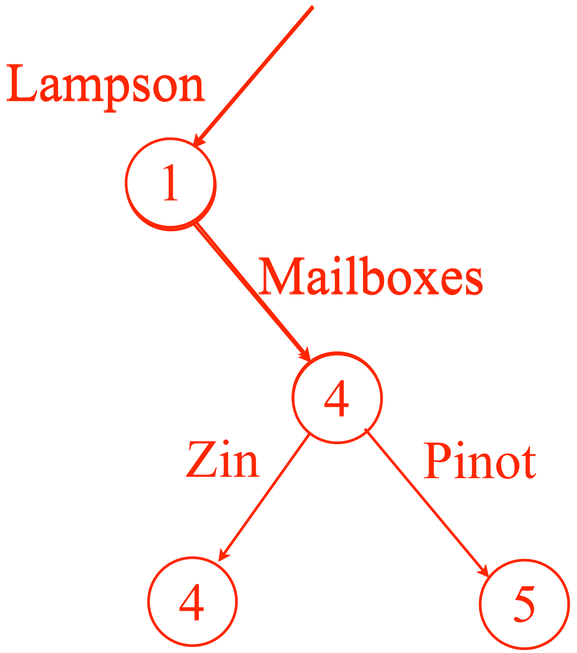

#734
Lampson
Shroeder
1
1

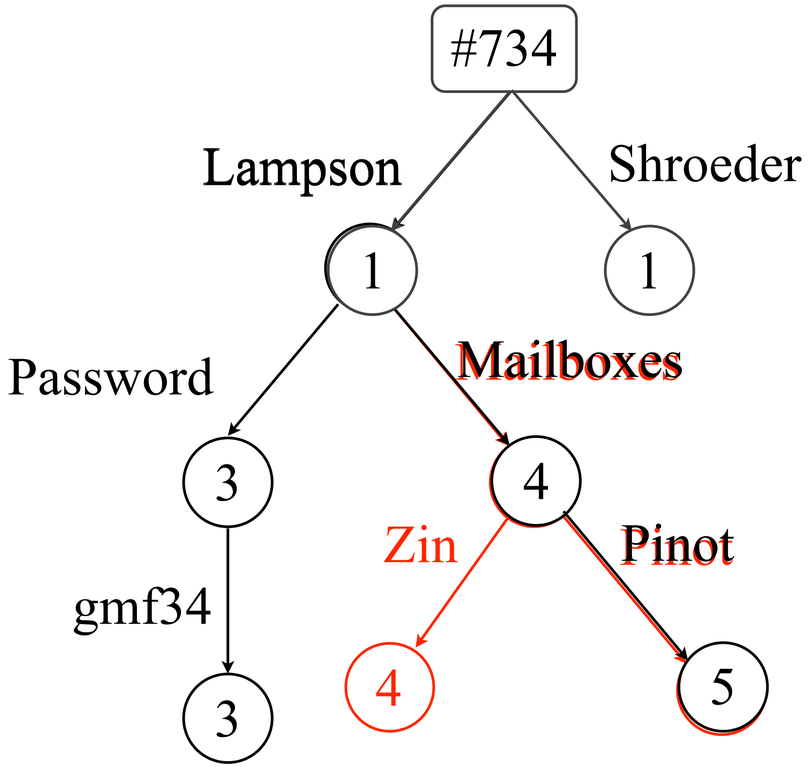

Lampson
1
Password
Password
2
3
vnxm56
gmf34
2
3

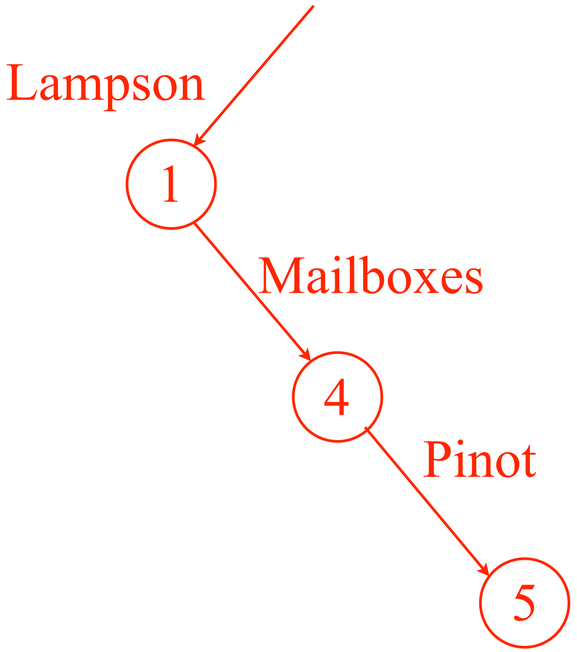

#734
Lampson
Shroeder
1
1


Lampson
1
Password
Password
2
3
vnxm56
gmf34
2
3


#734
Lampson
Shroeder
1
1
Password
Mailboxes
3
4
gmf34
3


Lampson
1
Mailboxes
4
Zin
Pinot
4
5


#734
Lampson
Shroeder
1
1
Password
Mailboxes
3
4
Zin
Pinot
gmf34
4
5
3


Lampson
1
Mailboxes
4
Pinot
5

Updates

- Commutative: reordering updates does not affect the outcome
- Idempotent: reapplying updates does not affect the outcome
- Both achieved through timestamp ordering

Designing a Global Name Service

- Scalability: supports an arbitrary number of names and organizations
  - *achieved through hierarchy and private name spaces*
- Fault-tolerance: operates even when N servers fail
  - *achieved through redundancy + eventual consistency*
- The system uses itself as a name service
- Clients and servers cache mappings
- Stale mappings avoided through expiration dates
- Updates are commutative and idempotent