

Naming and Directories

Announcements

- Midterm is tonight (7:30-8:45 pm, Hewlett 201)
- Prof Cheriton's lectures from last year on this topic are online

Amazon Cloud Outage

- What happened to Amazon EC2?
- <http://aws.amazon.com/message/65648/>

Directory

- Module used to lookup objects by name
 - `obj = lookup(name)`
 - Can add/remove names

Implementing Directories

Implementing Directories

- Easy to implement on a single machine

Implementing Directories

- Easy to implement on a single machine
- Much more difficult in a distributed system

Implementing Directories

- Easy to implement on a single machine
- Much more difficult in a distributed system
 - No pointers to objects

Implementing Directories

- Easy to implement on a single machine
- Much more difficult in a distributed system
 - No pointers to objects
 - How to scale?

Implementing Directories

- Easy to implement on a single machine
- Much more difficult in a distributed system
 - No pointers to objects
 - How to scale?
 - Trust Issues

Names

- Value used to identify an object
- A name can be relative to a context
 - e.g. relative paths

Why use names?

Why use names?

- Correctness:
 - Descriptions not enough

Why use names?

- Correctness:
 - Descriptions not enough
- Efficiency:
 - Easier to manipulate and pass around

Why use names?

- Correctness:
 - Descriptions not enough
- Efficiency:
 - Easier to manipulate and pass around
- Flexibility:
 - Symbolic references and indirection

Naming Theory

Naming Theory

- Proper Name:
 - Denotes but does not connote
 - Name does not convey meaning, e.g. “Plato”

Naming Theory

- Proper Name:
 - Denotes but does not connote
 - Name does not convey meaning, e.g. “Plato”
- Description:
 - Set of properties that object satisfies
 - Definite description: Describes exactly one object
 - Ambiguous Description: Can describe multiple objects

Naming Theory

- Proper Name:
 - Denotes but does not connote
 - Name does not convey meaning, e.g. “Plato”
- Description:
 - Set of properties that object satisfies
 - Definite description: Describes exactly one object
 - Ambiguous Description: Can describe multiple objects
- Common name: Reference to a description
 - (e.g. “chair”->desc(“furniture to sit on”))

Naming Theory

- Proper Name:
 - Denotes but does not connote
 - Name does not convey meaning, e.g. “Plato”
- Description:
 - Set of properties that object satisfies
 - Definite description: Describes exactly one object
 - Ambiguous Description: Can describe multiple objects
- Common name: Reference to a description
 - (e.g. “chair” -> desc(“furniture to sit on”))
- Dictionary: maps common names to descriptions

More naming theory

- Relevance to computing?
 - Object name: Proper name
 - Type name: Common Name

Pure and Impure Names

Pure and Impure Names

- Pure names: Contains no descriptive info

Pure and Impure Names

- Pure names: Contains no descriptive info
- Impure name: e.g. Lawrence of Arabia, /dev/tty

Advantages of Impure Names

Advantages of Impure Names

- Facilitate lookup: e.g. telephone numbers

Advantages of Impure Names

- Facilitate lookup: e.g. telephone numbers
- Facilitate allocation

Advantages of Impure Names

- Facilitate lookup: e.g. telephone numbers
- Facilitate allocation
- Informative for holder of name

Disadvantages of Impure Names

Disadvantages of Impure Names

- Mapping may have to change often, e.g. if you move

Disadvantages of Impure Names

- Mapping may have to change often, e.g. if you move
- Smaller limit on available names, e.g. with fixed length phone numbers

Disadvantages of Impure Names

- Mapping may have to change often, e.g. if you move
- Smaller limit on available names, e.g. with fixed length phone numbers
- Info in name may be incorrect, e.g. jpg may not really be a picture

Some More Naming Theory

- Spectrum of Trade-offs:

Pure Names \leftrightarrow Impure Names \leftrightarrow Descriptions

Some More Naming Theory

- Spectrum of Trade-offs:

Pure Names \leftrightarrow Impure Names \leftrightarrow Descriptions

Some More Naming Theory

- Spectrum of Trade-offs:
Pure Names \leftrightarrow Impure Names \leftrightarrow Descriptions
- Structured Names
 - Names with well-defined components, e.g. URLs or UNIX pathnames
 - Different from impure names - impure could be unstructured or structured could be pure

Naming

- Uniform naming across objects
- Use hierarchical character strings - as in file systems
- Two types of object names

External Names

External Names

- Chosen by the user

External Names

- Chosen by the user
- Should be human-readable => a char string

External Names

- Chosen by the user
- Should be human-readable => a char string
- Why external names?
 - Users need a readable name to reference objects
 - Error reporting
 - Scaling: Internet only had IP addresses - but hostnames needed for scaling

Internal Names

Internal Names

- System-generated

Internal Names

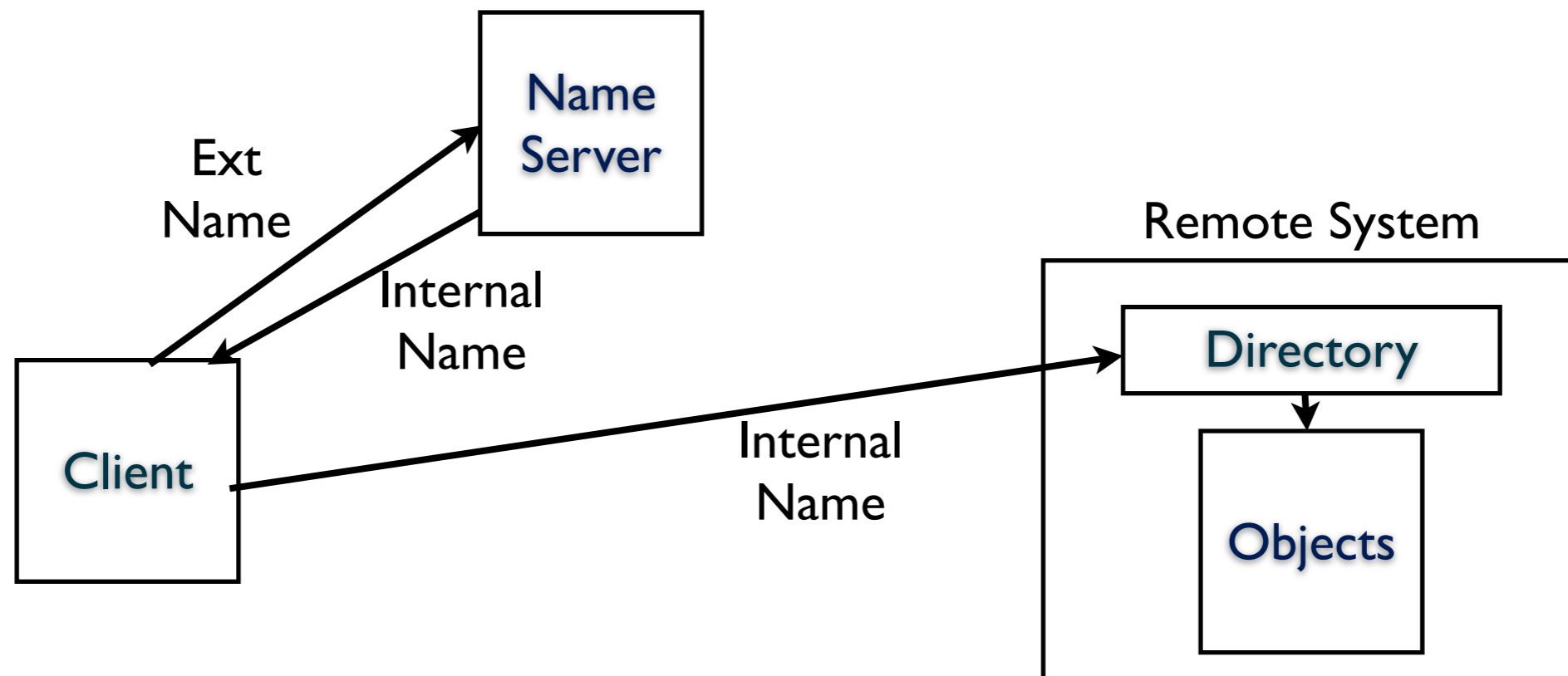
- System-generated
- Only an optimization!

Internal Names

- System-generated
- Only an optimization!
- Examples: file descriptors, IP addresses

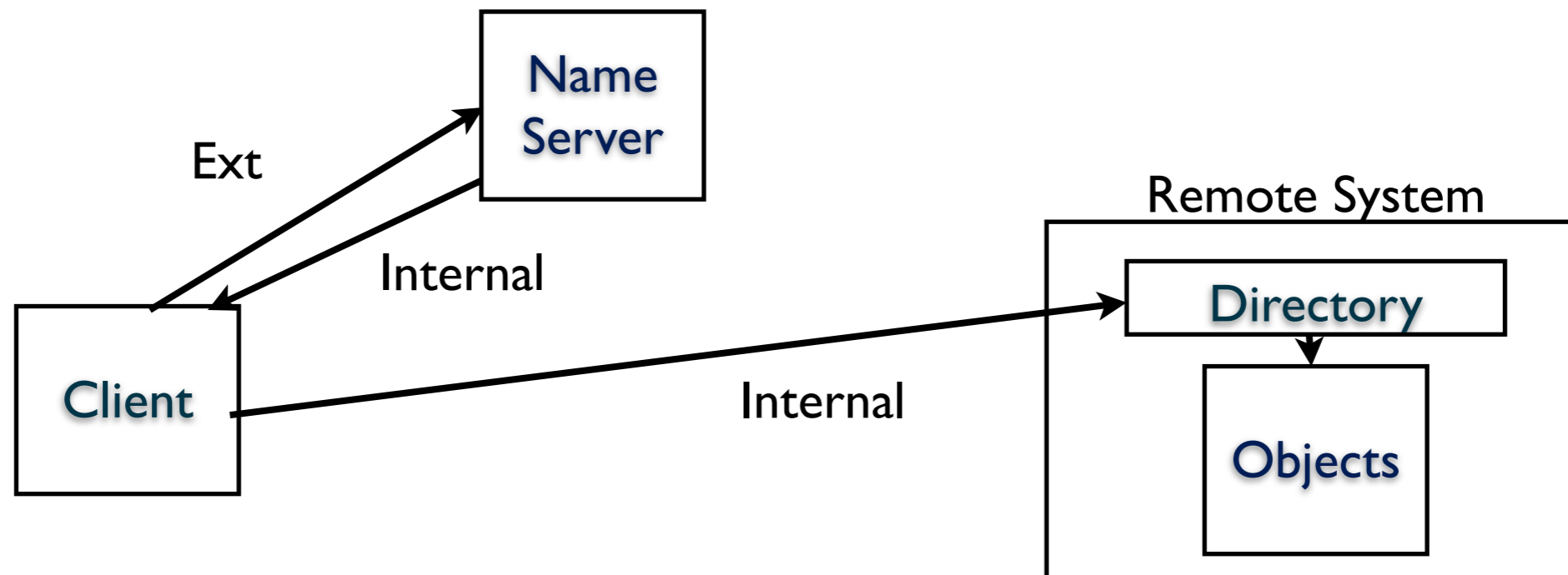
How to lookup names?

- Conventional thinking: Name servers
- Given an external name, returns an internal name



Cheriton's View

- Every module should implement the external names of objects it implements
- Better performance
- No inconsistency between module and nameserver
- No dependence on availability of nameserver



**UUIDs considered
harmful**

UUIDs considered harmful

- Size: Have to be unique forever: 128 to 256 bits.
- Add impurities for efficient allocation and mapping
- Some huge name server required
- Need extra reliability on the name server

UUIDs considered harmful

- Size: Have to be unique forever: 128 to 256 bits.
 - Add impurities for efficient allocation and mapping
 - Some huge name server required
 - Need extra reliability on the name server
- What about legacy objects?
 - Embed legacy names in UUID plus special prefix?

UUIDs considered harmful

- Size: Have to be unique forever: 128 to 256 bits.
 - Add impurities for efficient allocation and mapping
 - Some huge name server required
 - Need extra reliability on the name server
- What about legacy objects?
 - Embed legacy names in UUID plus special prefix?
- Semantic problem: When does an object get a new UUID?
 - How do you replace an object with a new one and keep the old copy?

Questions?

Naming and Directories

Chapter 5

Semantic Problem with UUIDs

Semantic Problem with UUIDs

- When does an object get a new UUID?

Semantic Problem with UUIDs

- When does an object get a new UUID?
 - How do you replace an object with a new one and keep the old copy?

Semantic Problem with UUIDs

- When does an object get a new UUID?
 - How do you replace an object with a new one and keep the old copy?
 - Example: Replace `/bin/emacs` with `/bin/emacs_v2`

Semantic Problem with UUIDs

- When does an object get a new UUID?
 - How do you replace an object with a new one and keep the old copy?
 - Example: Replace `/bin/emacs` with `/bin/emacs_v2`
 - Usual way:

Semantic Problem with UUIDs

- When does an object get a new UUID?
 - How do you replace an object with a new one and keep the old copy?
 - Example: Replace `/bin/emacs` with `/bin/emacs_v2`
 - Usual way:
 - Rename `/bin/emacs` to `/bin/emacs-old`

Semantic Problem with UUIDs

- When does an object get a new UUID?
 - How do you replace an object with a new one and keep the old copy?
 - Example: Replace `/bin/emacs` with `/bin/emacs_v2`
 - Usual way:
 - Rename `/bin/emacs` to `/bin/emacs-old`
 - Rename `/bin/emacs_v2` to `/bin/emacs`

Semantic Problem with UUIDs

- When does an object get a new UUID?
 - How do you replace an object with a new one and keep the old copy?
 - Example: Replace `/bin/emacs` with `/bin/emacs_v2`
 - Usual way:
 - Rename `/bin/emacs` to `/bin/emacs-old`
 - Rename `/bin/emacs_v2` to `/bin/emacs`
 - With UUIDs, each file's name looks like `a2341500-e29b-41d4-a716-446655310000`

Semantic Problem with UUIDs

- When does an object get a new UUID?
 - How do you replace an object with a new one and keep the old copy?
 - Example: Replace `/bin/emacs` with `/bin/emacs_v2`
 - Usual way:
 - Rename `/bin/emacs` to `/bin/emacs-old`
 - Rename `/bin/emacs_v2` to `/bin/emacs`
 - With UUIDs, each file's name looks like `a2341500-e29b-41d4-a716-446655310000`
 - Who gets a new UUID? The new version? The old version? Both?

Semantic Problem with UUIDs

- When does an object get a new UUID?
 - How do you replace an object with a new one and keep the old copy?
 - Example: Replace `/bin/emacs` with `/bin/emacs_v2`
 - Usual way:
 - Rename `/bin/emacs` to `/bin/emacs-old`
 - Rename `/bin/emacs_v2` to `/bin/emacs`
 - With UUIDs, each file's name looks like `a2341500-e29b-41d4-a716-446655310000`
 - Who gets a new UUID? The new version? The old version? Both?

Internal Names

Internal Names

- System-generated names/identifiers

Internal Names

- System-generated names/identifiers
- Lightweight names, important optimization

Internal Names

- System-generated names/identifiers
- Lightweight names, important optimization
- How small? 16-bit, 32-bit, 64-bit?

Internal Names

- System-generated names/identifiers
- Lightweight names, important optimization
- How small? 16-bit, 32-bit, 64-bit?
 - To avoid wrap-around, 64-bit.

Internal Names

- System-generated names/identifiers
- Lightweight names, important optimization
- How small? 16-bit, 32-bit, 64-bit?
 - To avoid wrap-around, 64-bit.
 - Google varInt another option: variable-length int

Internal Names

- System-generated names/identifiers
- Lightweight names, important optimization
- How small? 16-bit, 32-bit, 64-bit?
 - To avoid wrap-around, 64-bit.
 - Google varInt another option: variable-length int
- Can be transient and limited to lifetime of a connection or context

IP: Example of Internal Name

IP: Example of Internal Name

- How to map at scale?

IP: Example of Internal Name

- How to map at scale?
 - Structure using classes

IP: Example of Internal Name

- How to map at scale?
 - Structure using classes
 - Needed because memory was costly

IP: Example of Internal Name

- How to map at scale?
 - Structure using classes
 - Needed because memory was costly
 - IP a bit like external names (36.x.x.x was Stanford)

IP: Example of Internal Name

- How to map at scale?
 - Structure using classes
 - Needed because memory was costly
 - IP a bit like external names (36.x.x.x was Stanford)
 - Replaced by CIDR

IP: Example of Internal Name

- How to map at scale?
 - Structure using classes
 - Needed because memory was costly
 - IP a bit like external names (36.x.x.x was Stanford)
 - Replaced by CIDR
 - Then incorporated DHCP and NAT

IP: Example of Internal Name

- How to map at scale?
 - Structure using classes
 - Needed because memory was costly
 - IP a bit like external names (36.x.x.x was Stanford)
 - Replaced by CIDR
 - Then incorporated DHCP and NAT
 - Now reduced to a transient, routing hint

External Names

External Names

- Longer, human-readable character strings

External Names

- Longer, human-readable character strings
- An external naming service should have:
 - A uniform, global namespace
 - Stability/Durability
 - Efficient lookup, directory listing
 - Customizability (e.g. shortcuts, symlinks)
 - Locality support
 - High Availability

Proposed Directory Architecture

Proposed Directory Architecture

- Three levels of naming:

Proposed Directory Architecture

- Three levels of naming:
 - Global: Relatively stable

Proposed Directory Architecture

- Three levels of naming:
 - Global: Relatively stable
 - Caching very effective

Proposed Directory Architecture

- Three levels of naming:
 - Global: Relatively stable
 - Caching very effective
 - High availability required

Proposed Directory Architecture

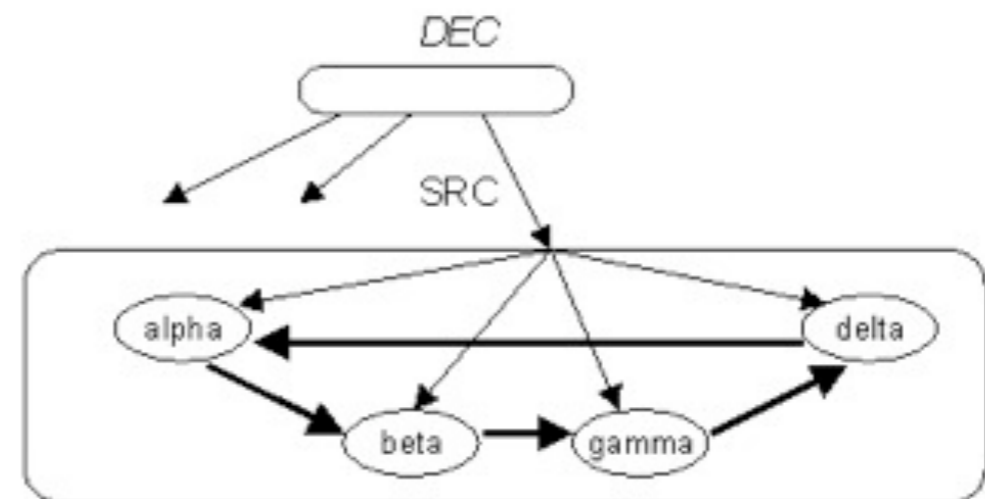
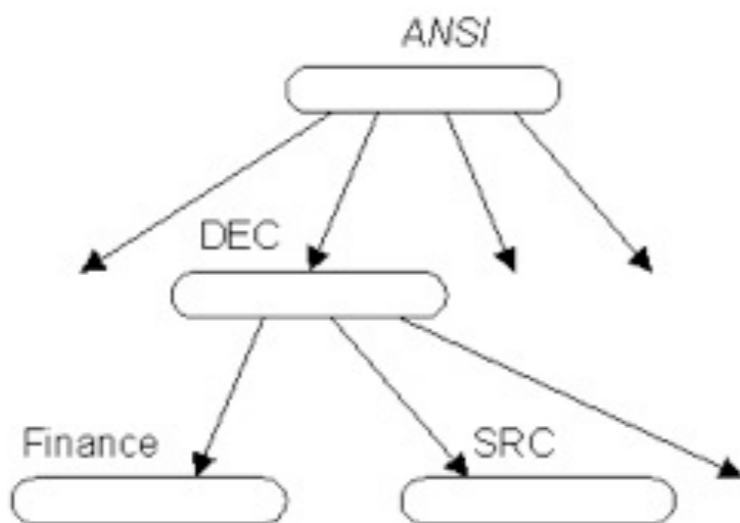
- Three levels of naming:
 - Global: Relatively stable
 - Caching very effective
 - High availability required
 - Administration: Higher rate of change

Proposed Directory Architecture

- Three levels of naming:
 - Global: Relatively stable
 - Caching very effective
 - High availability required
 - Administration: Higher rate of change
 - Module/Managerial: Frequent change

Lampson's Global Name Service

- Hierarchical directory structure
- Availability vs Consistency:
 - Favor availability and reconciliation in case of duplicates



Problems with Lampson's Name Service

Problems with Lampson's Name Service

- Too inefficient for module level

Problems with Lampson's Name Service

- Too inefficient for module level
- Limited support for locality

Problems with Lampson's Name Service

- Too inefficient for module level
- Limited support for locality
- Limited support for customizability

Epidemic Approach

- Use an unstructured approach
 - Direct mail
 - Anti-entropy
 - Rumor-mongering
- Deletes represented as anti-directory entries

Problems with Epidemic Approach

Problems with Epidemic Approach

- Performance too unpredictable/inefficient

Problems with Epidemic Approach

- Performance too unpredictable/inefficient
- Getting to 100% coverage difficult

Problems with Epidemic Approach

- Performance too unpredictable/inefficient
- Getting to 100% coverage difficult
- Solution: Build in some topology info?
 - Make it structured?

Decentralized Naming

Decentralized Naming

- Global directories highly replicated nameservers

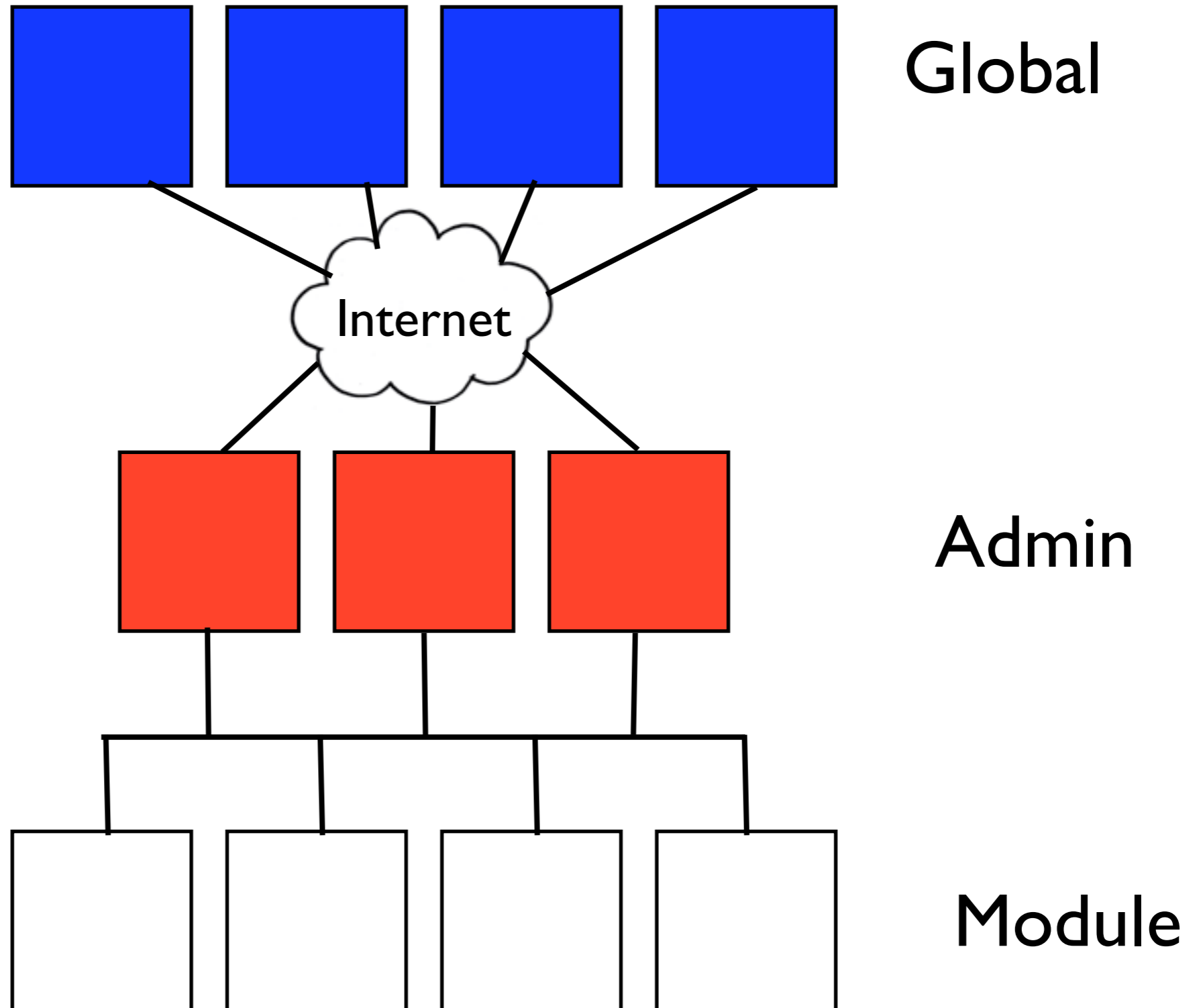
Decentralized Naming

- Global directories highly replicated nameservers
- Decentralized approach for module level
 - Every module implements naming for its objects

Decentralized Naming

- Global directories highly replicated nameservers
- Decentralized approach for module level
 - Every module implements naming for its objects
- Admin directories: Caching servers

Decentralized Naming Architecture



Caching for Decentralized Naming

- Prefix name cache: 99.7% cache hits.
- Multicast for cache misses.
- Cache misses were mostly for non-existent entries.

Security with Decentralized Naming

Security with Decentralized Naming

- Authenticator to maintain namespace partitioning

Security with Decentralized Naming

- Authenticator to maintain namespace partitioning
- Multicast namespace to everyone. Object if incorrect namespace advertised

URLs

`protocol://hostname:port/filepath`

Cons of URLs

- Why need a protocol?
- How to do object migration?

Pros of URLs

Pros of URLs

- Branding

Pros of URLs

- Branding
- Hostname can be virtualized to many servers

Pros of URLs

- Branding
- Hostname can be virtualized to many servers
- Cost of virtualization only paid if needed

Questions

V System Hierarchy

