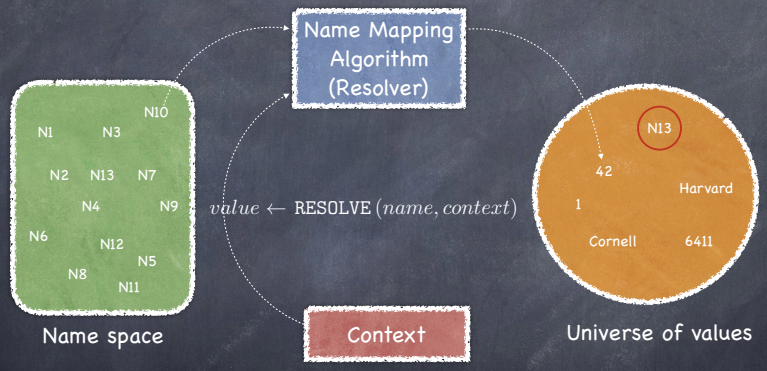
Naming

Systems manipulate objects

- Objects can use other objects as components
- Objects can use other objects by either
  - value, incorporating copy of used object
    - reduces propagation of effects
  - reference, incorporating name of used object
    - facilitates sharing and delays binding

Naming scheme



- Context helps distinguish the correct naming scheme
  - if admits only one context: universal
  - if name bound to a single object and never reused: unique identifier

Naming scheme: API

- $status \leftarrow \textsc{Bind}(name, value, context)$
- $status \leftarrow \textsc{Unbind}(name, context)$
- $list \leftarrow \textsc{Enumerate}(context)$
- $result \leftarrow \textsc{Compare}(name1, name2)$

Determining the context

- Supplied by the resolver
  - Hard coded in the resolver (name space is universal)
    - binding between registers and their names
  - Retrieved from an environment variable
    - page-map address register
- Supplied by the object
  - per object
    - closures in lexical scoping
  - per name (qualified name)
    - absolute path name

Name mapping algorithms

- Table lookup

Name	Value
N1	42
N2	Cornell
N3	N13
N4	Bach

  - Ex. page table
- Recursive lookup
  - Ex. /usr/bin/ls
- Multiple lookup
  - Ex. $PATH

Naming schemes: FAQ

- What is the names' syntax?
- What are the possible values?
- What context is used to resolve names?
- Who specifies the context?
- Are names qualified or unqualified?
- Does every value have a name?
- Does every name have a value?
- Can a single name have multiple values?
- Can a single value have multiple names?
- Can the value bound to a name change over time?
- How do you find the name of what you need?

Naming and layering in the Unix File system

A touch of history

- Time frame: late 60s
- A reaction to MIT Multics (hence the name)
- Initially developed by Thompson and Ritchie on a PDP-7 (1969)
- Published in SOSP '73
- Worse is better?

Unix File System API

- Open and close
  - open(name, flags,mode) opens file name. If file does not exist and flags is set, create file with permissions mode. Sets cursor to 0. Returns file descriptor
    - permissions checked at open() time (a capability!)
    - creates per-file data structure, referred to by file descriptor
      - file ID, R/W permission, cursor
  - close() deletes descriptor. If last reference to a file that has been unlinked, delete the file
- Creating and deleting files
  - link() creates a hard link–a new name for the same underlying file
  - unlink() removes a name for a file from its directory. If last link, file itself and resources it held are deleted
- File access
  - read(fd,buf,n), write(fd,buf,n), seek(fd,offset,whence)
    - but can use mmap() to create a mapping between region of file and region of memory
  - fsync() does not return until data is written to persistent storage

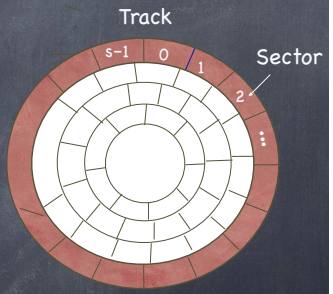
Unix naming layers

Layer	Purpose	
Symbolic link layer	Integrate multiple file systems with symbolic links.	↑
Absolute path name layer	Provide a root for the naming hierarchies.	user-oriented names
Path name layer	Organize files into naming hierarchies.	↓
File name layer	Provide human-oriented names for files.	machine-user interface
Inode number layer	Provide machine-oriented names for files.	↑
File layer	Organize blocks into files.	machine-oriented names
Block layer	Identify disk blocks.	↓

L1: The block layer

machine oriented names

- Purpose: bind block numbers to physical blocks
  - block: a few disk sectors
  - storage device is the context!




```
procedure BLOCK_NUMBER_TO_BLOCK (integer b) returns block
    return device[b]
```

The Superblock

- One superblock per file system, at a well-known location. It contains
  - size of FS
  - list of free blocks (today, a bitmap)
  - number of free blocks and index of next free block
  - size of inode list
  - number of free nodes and index of next free node
  - locks for free block and free node lists
  - flag to indicate superblock has been modified

L2: File layer

machine oriented names

- Naming layers for items (files) that are larger than one block and whose size changes over time
- A file's metadata is stored in an inode

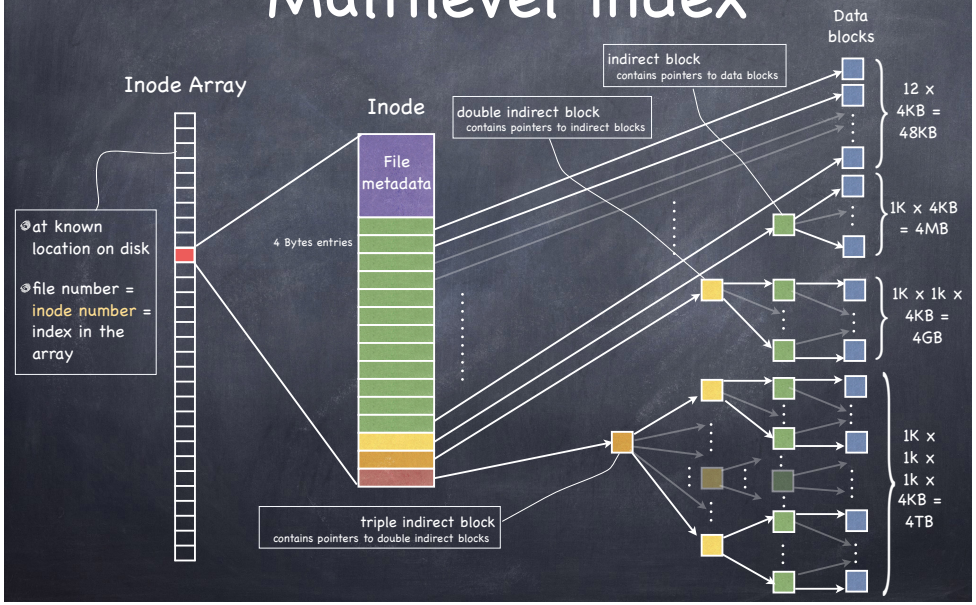
```
structure inode
  integer block_numbers[N]
  integer size
```

that can be seen as a context where the file's blocks are named by an integer block number:

```
procedure INDEX_TO_BLOCK_NUMBER(inode instance i, integer index) returns block
  return i.block_nubers[index]
```

Inodes are actually more complicated...

Multilevel index



L2: File layer

machine oriented names

- Access an arbitrary byte by layering BLOCK_NUMBER_TO_BLOCK and INDEX_TO_BLOCK_NUMBER

```
procedure INDEX_TO_BLOCK(integer offset, inode instance i) returns block
    o ← offset / Blocksize
    b ← INDEX_TO_BLOCK_NUMBER(i, o)
    return BLOCK_NUMBER_TO_BLOCK(b)
```

L3: inode number layer

machine oriented names

- Instead of passing around inode, pass around its name

```
procedure INODE_NUMBER_TO_INODE(integer inode_number) returns inode
    return inode_table[inode_number]
```

where the inode table is stored at a well-known location

- Combining all layers:

```
procedure INODE_NUMBER_TO_BLOCK(integer offset, integer inode_number) returns block
    inode instance i ← INODE_NUMBER_TO_INODE(inode_number)
    o ← offset / Blocksize
    b ← INDEX_TO_BLOCK_NUMBER(i, o)
    return BLOCK_NUMBER_TO_BLOCK(b)
```

L4: File Name layer

machine-user interface

- Hides metadata of file management
- Key component: directory

File name	Inode number
Music	34
Project 1	18
Ducati.jpg	21
Documents	6

A context containing binding between character-string names and inodes

```
structure inode
    integer block_numbers[N]
    integer size
    integer type
```

L4: File Name Layer

machine-user interface

```
procedure NAME_TO_INODE_NUMBER(character string filename, integer dir) returns integer
    return LOOKUP(filename, dir)
```

dir: working directory

```
procedure LOOKUP(character string filename, integer dir) returns integer
    block instance b
    inode instance i ← INODE_NUMBER_TO_INODE(dir)
    if i.type ≠ Directory then return Failure
    for offset from 0 to i.size − 1 do
        b ← INODE_NUMBER_TO_BLOCK(offset, dir)
        if STRING_MATCH(filename, b) then
            return INODE_NUMBER(filename, b)
        offset ← offset + Blocksize
    return Failure
```

L5: Path Name Layer

user-oriented name

```
procedure PATH_TO_INODE_NUMBER(character string path, integer dir) returns integer
    if (PLAIN_NAME(path) return NAME_TO_INODE_NUMBER(path, dir)
    else
        dir ← LOOKUP(FIRST(path), dir)
        path ← REST(path)
        if path = "" then return dir
        else return PATH_TO_INODE_NUMBER(path, dir)
```

context is the working directory

Links: Hard Links

- Bindings in different contexts that map different file names to the same inode
- Turn strict hierarchy into a directed graph
  - to keep graph acyclic, cannot create hard links to directory
- inodes can bind multiple file names

```
structure inode
    integer block_numbers[N]
    integer size
    integer type
    integer refcnt
```

Renaming

- Implementing rename($from\_name, to\_name$)

File name	Inode number
to_name	34
from_name	18
Ducati.jpg	21
Documents	6

1 UNLINK(*to_name*)

File name	Inode number
from_name	18
Ducati.jpg	21
Documents	6

2 LINK(*from_name*, *to_name*)

File name	Inode number
to_name	18
from_name	18
Ducati.jpg	21
Documents	6

3 UNLINK(*from_name*)

File name	Inode number
to_name	18
Ducati.jpg	21
Documents	6

Renaming

- Implementing rename($from\_name, to\_name$)

File name	Inode number
to_name	34
from_name	18
Ducati.jpg	21
Documents	6

1 UNLINK(*to_name*)

File name	Inode number
from_name	18
Ducati.jpg	21
Documents	6

to_name is lost!

Renaming

- Implementing rename($from\_name, to\_name$)

File name	Inode number
to_name	34
from_name	18
Ducati.jpg	21
Documents	6

1 LINK(*from_name*, *to_name*)

File name	Inode number
to_name	18
from_name	18
Ducati.jpg	21
Documents	6

3 UNLINK(*from_name*)

File name	Inode number
to_name	18
Ducati.jpg	21
Documents	6

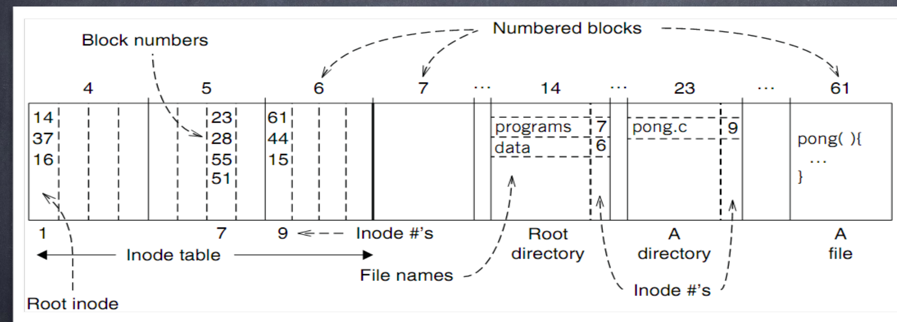
L6: Absolute Path Name Layer

user-oriented name

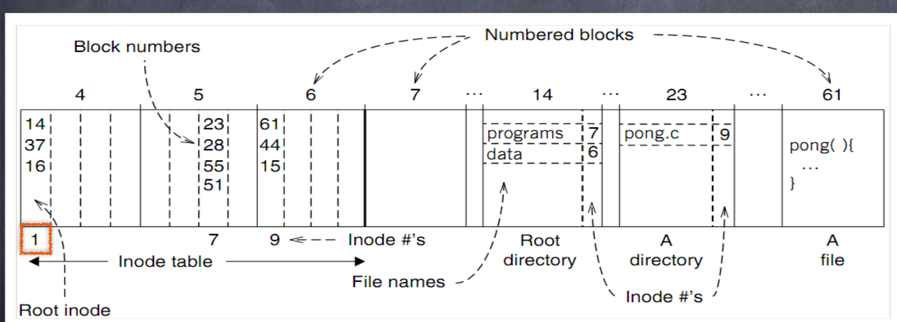
- Home directory is the default context (working directory)
- Root is a shared context for all users
  - well known name "/" and location on disk

```
procedure GENERALPATH_TO_INODE_NUMBER(character string path) returns integer
  if ( path[0] = "/") then return PATH_TO_INODE_NUMBER(STRIP( path, "/"), 1)
  else return PATH_TO_INODE_NUMBER(path, wd)
```

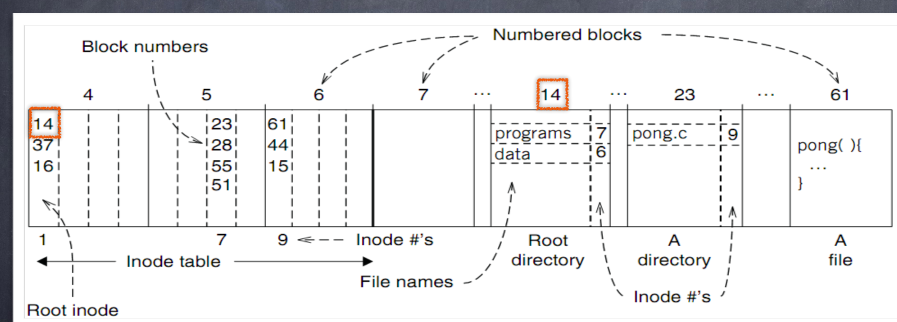
Finding the blocks of /programs/pong.c



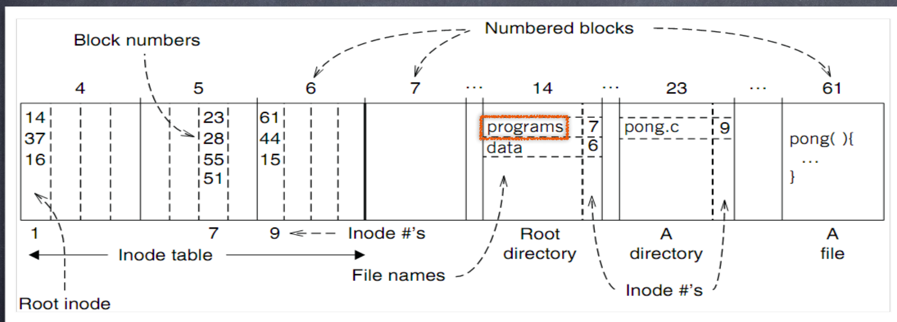
Finding the blocks of /programs/pong.c



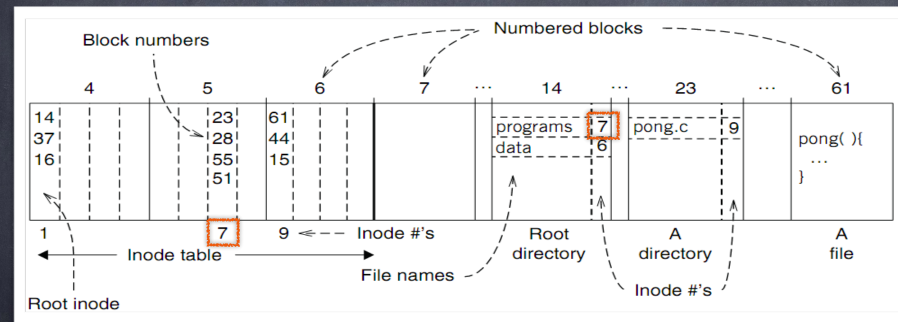
Finding the blocks of /programs/pong.c



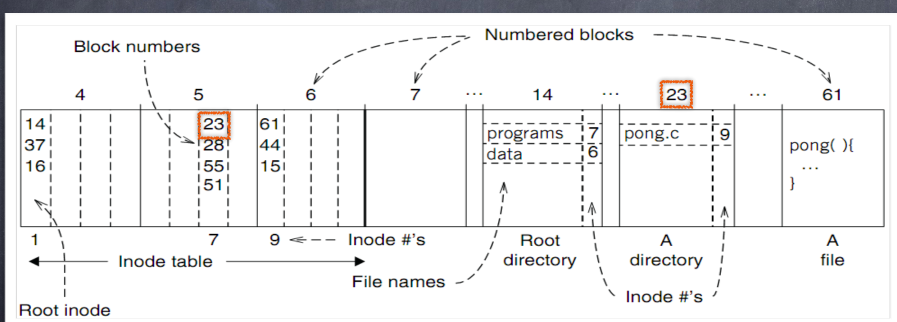
Finding the blocks of /programs/pong.c



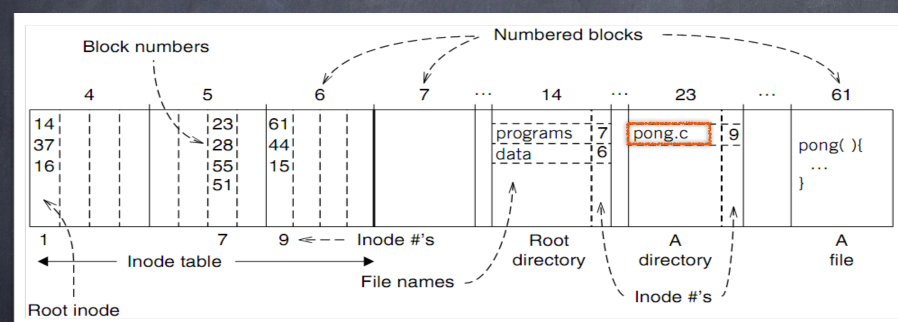
Finding the blocks of /programs/pong.c



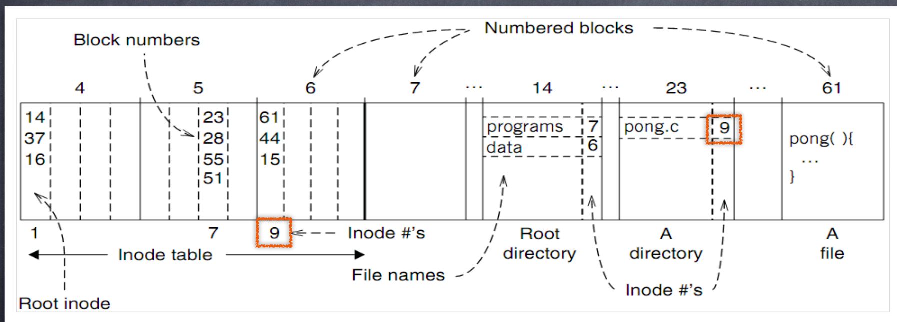
Finding the blocks of /programs/pong.c



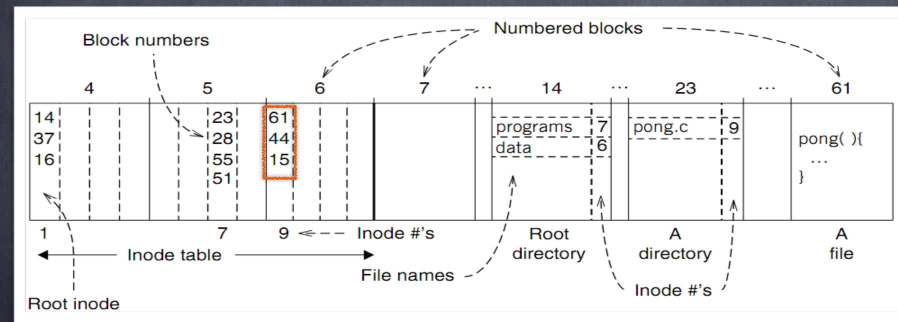
Finding the blocks of /programs/pong.c



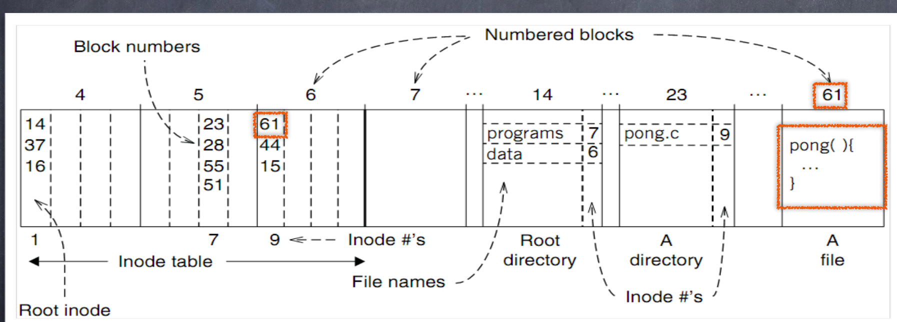
Finding the blocks of /programs/pong.c



Finding the blocks of /programs/pong.c



Finding the blocks of /programs/pong.c



L7: Symbolic Link Layer

user-oriented name

MOUNT("dev/fd1", "/flash")

grafts directory tree stored on '/dev/fd1' to directory '/flash'

- inode for '/flash' is kept in memory until UNMOUNT
- so are the root inode number and the device of the mounted file system
- inode for '/dev/fd1' stores parent's inode

if it encounters an inode on which a file system is mounted, LOOKUP proceeds to use the root of that file system

L7: Symbolic Link Layer

user-oriented name

- Symbolic link: a binding from a name to another name in the same name space
- Hard link: a binding from a name to another name in a lower-layer name space
- An inode of type symlink records in $blocks[i]$ the characters of a path name instead of a sequence of block numbers

```
procedure PATHNAME_TO_INODE(character string filename) returns inode
    inode instance i
    inode_number ← GENERALPATH_TO_INODE_NUMBER(filename)
    i ← INODE_NUMBER_TO_INODE(inode_number)
    if i.type = SYMLINK then
        i ← PATHNAME_TO_INODE(COERCE_TO_STRING(i.block_numbers))
    return i
```

Domain Name Service

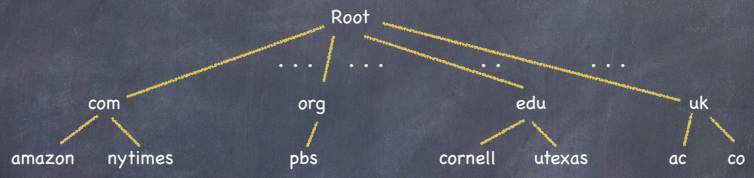
DNS Mockapetris '87

- A name service built above a hierarchical, distributed, autonomous, reliable database
  - Every TCP/IP traffic flow begins with at least one DNS transaction!
- Introduced to replace the original Internet naming scheme
  - a single central master file downloaded everywhere by FTP
  - ...not that there is anything wrong with that (?)

DNS as a Naming Scheme

- Originally designed to support multiple name spaces, but in practice mostly for...
  - Hostname resolution - www.cs.cornell.edu to its IP address
  - Mail host location - la13@cornell.edu to an ordered list of mail hosts that can accept mail for cornell.edu
- ... but also
  - Reverse resolution: IP -> hostname
  - Host information: hostname -> architectural attributes of the hostname
    - too much information?

A distributed, hierarchical, database

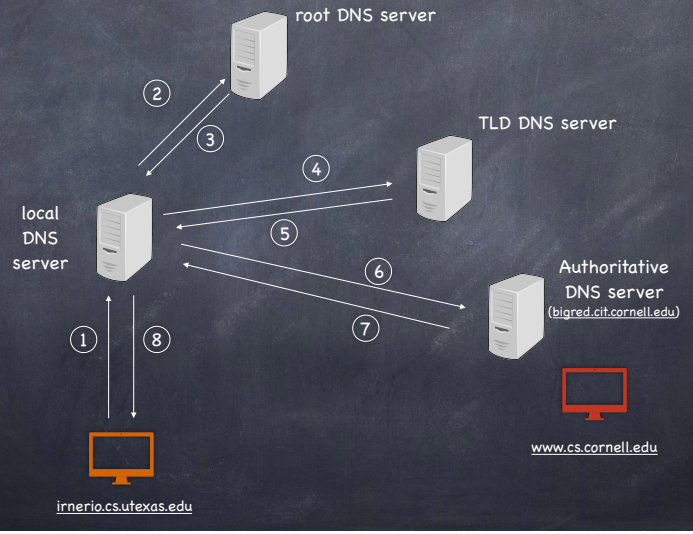


DNS names are global: they mean the same everywhere in the DNS

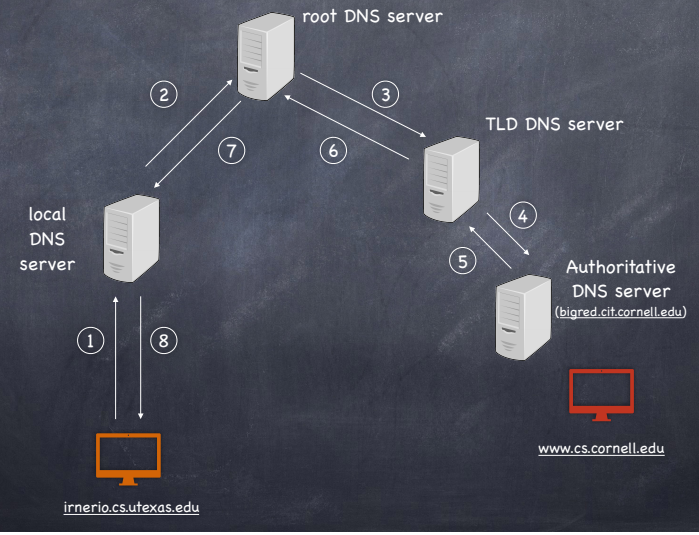
To find IP address of www.cs.cornell.edu (basic protocol)

- Query root server to find IP of DNS server for edu...
- which, when queried next, will provide IP of DNS server for cornell.edu
- which, when queried next, will provide IP of cs.cornell.edu
- which, when queried next, will provide IP of www.cs.cornell.edu

DNS Name Resolution: Iterative



DNS Name Resolution: Recursive



Caching

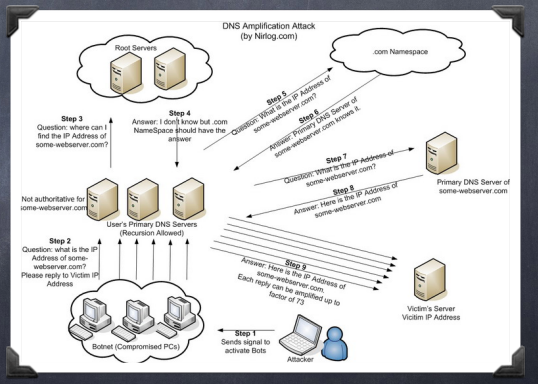
- Cache entries may be out of date
  - bindings are forgotten after TTL
  - TLD servers typically cached in local name servers
- Servers cache new bindings they learn
  - only eventual consistency
  - if binding changes, it may not be learned until TTL
  - update notify proposed IETF standard

Attacking DNS

- DDos attacks on DNS
  - target: root servers
    - to date, unsuccessful
    - traffic filtering / root server bypass via TLD, cashed at local servers
  - target: TLD servers
    - potentially more dangerous
- Redirect attacks
  - Man-in-the-middle
  - DNS poisoning

Leveraging DNS for DDoS

- DDOS Amplification
  - asymmetric attack
  - send query with spoofed source address
    - IP of the victim



(Stupid) DNS tricks

- Using lookups as mapping requests
  - CDNs respond to DNS lookup with IP of server for requested URI domain closest to requester
    - to work, must curb caching
    - IP source of DNS request may have little to do with network-location of end system web server
      - lookup request come from cache misses at DNS servers
- Remapping NXDOMAIN
  - returned whenever looking for a name that has no binding
  - remapped to advertisers sites
  - they get to see content of cookies, can potentially capture email (if request was to an MX)
- Domain guessing
  - why everyone wants .com