15-441 15-641 Computer Networking

Lecture 9 – IPv6, Translations

Peter Steenkiste

Fall 2014

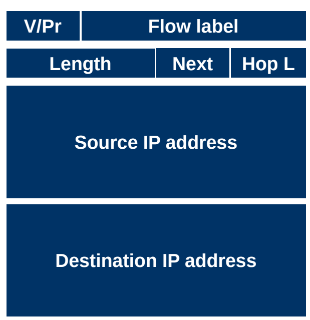
www.cs.cmu.edu/~prs/15-441-F14

Outline

- IPv6
- Translation: too many names and addresses!

IPv6

- "Next generation" IP.
- Most urgent issue: increasing address space.
  - 128 bit addresses
- Simplified header for faster processing:
  - No checksum (why not?)
  - No fragmentation (?)
- Support for guaranteed services: priority and flow id
- Options handled as "next header"
  - reduces overhead of handling options



IPv6 Addressing

- Do we need more addresses? Probably, long term
  - Big panic in 90s: "We're running out of addresses!"
  - Big worry: Devices. Small devices. Cell phones, toasters, everything.
- 128 bit addresses provide space for structure (good!)
  - Hierarchical addressing is much easier
  - Assign an entire 48-bit sized chunk per LAN – use Ethernet addresses
  - Different chunks for geographical addressing, the IPv4 address space,
  - Perhaps help clean up the routing tables - just use one huge chunk per ISP and one huge chunk per customer.

IPv6 Autoconfiguration

- Serverless ("Stateless"). No manual config at all.
  - Only configures addressing items, NOT other host things
    - If you want that, use DHCP.
- Link-local address
  - 1111 1110 10 :: 64 bit interface ID (usually from Ethernet addr)
    - (fe80::/64 prefix)
  - Uniqueness test ("anyone using this address?")
  - Router contact (solicit, or wait for announcement)
    - Contains globally unique prefix
    - Usually: Concatenate this prefix with local ID → globally unique IPv6 ID
- DHCP took some of the wind out of this, but nice for "zero-conf" (many OSes now do this for both v4 and v6)

Fast Path versus Slow Path

- Common case: Switched in silicon ("fast path")
  - Almost everything
- Weird cases: Handed to CPU ("slow path", or "process switched")
  - Fragmentation
  - TTL expiration (traceroute)
  - IP option handling
- Slow path is evil in today's environment
  - "Christmas Tree" attack sets weird IP options, bits, and overloads router.
  - Developers cannot (really) use things on the slow path
    - Slows down their traffic – not good for business
    - If it became popular, they'd be in the soup!

IPv6 Header Cleanup: Options

- 32 IPv4 options → variable length header
  - Rarely used
  - No development / many hosts/routers do not support
    - Worse than useless: Packets w/options often even get dropped!
  - Processed in "slow path".
- IPv6 options: "Next header" pointer
  - Combines "protocol" and "options" handling
    - Next header: "TCP", "UDP", etc.
  - Extensions header: Chained together
  - Makes it easy to implement host-based options
  - One value "hop-by-hop" examined by intermediate routers
    - E.g., "source route" implemented only at intermediate hops

IPv6 Header Cleanup: "no"

- No checksum
  - Motivation was efficiency: If packet corrupted at hop 1, don't waste b/w transmitting on hops 2..N.
  - Useful when corruption frequent, b/w expensive
  - Today: corruption is rare, bandwidth is cheap
- No fragmentation
  - Router discard packets, send ICMP "Packet Too Big" → host does MTU discovery and fragments
  - Reduced packet processing and network complexity.
  - Increased MTU a boon to application writers
  - Hosts can still fragment - using fragmentation header. Routers don't deal with it any more.

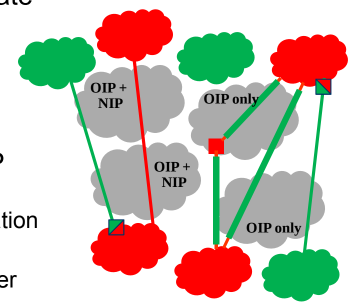
Migration from IPv4 to IPv6

- Interoperability with IP v4 is necessary for incremental deployment.
- Combination of mechanisms:
  - Dual stack operation: IP v6 nodes support both address types
  - Tunnel IP v6 packets through IP v4 clouds
  - IPv4-IPv6 translation at edge of network
    - NAT must not only translate addresses but also translate between IPv4 and IPv6 protocols
  - IPv6 addresses based on IPv4 – no benefit!
  - More on NATs and tunnels in the next lecture

9

Examples of Transition Models

- Green Old Networks and red New Networks communicate through grey ISPs
- ISPs only support OIP
  - NN-NN can use tunnel, or reflector or NIP backbone
  - NN-ON requires translation
- ISPs support OIP and NIP
  - NN-NN can be native e-e
  - NN-ON still requires translation
- Translation is unattractive
  - Complex, no benefit of larger address space
  - Incentive for deploying NIP?



10

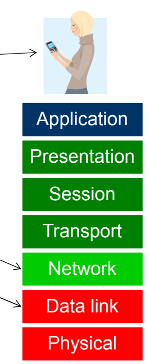
Outline

- IPv6
- Translation: too many names and addresses!
  - ARP
  - DNS

11

Too Much of a Good Thing?

- Hosts have a
  - host name
  - IP address
  - MAC address
- There is a reason ..
  - Remember?
- But how do we translate?



12

IP to MAC Address Translation

- How does one find the Ethernet address of a IP host?
- Address Resolution Protocol - ARP
  - Broadcast search for IP address
    - E.g., “who-has 128.2.184.45 tell 128.2.206.138” sent to Ethernet broadcast (all FF address)
  - Destination responds (only to requester using unicast) with appropriate 48-bit Ethernet address
    - E.g, “reply 128.2.184.45 is-at 0:d0:bc:f2:18:58” sent to 0:c0:4f:d:ed:c6

13

Caching ARP Entries

- Efficiency Concern
  - Would be very inefficient to use ARP request/reply every time need to send IP message to machine
- Each Host Maintains Cache of ARP Entries
  - Add entry to cache whenever get ARP response
  - “Soft state”: set timeout of ~20 minutes

14

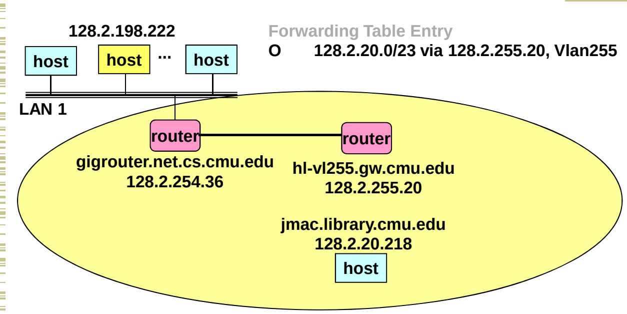
ARP Cache Example

- Show using command “arp -a”

```
Interface: 128.2.222.198 on Interface 0x1000003
  Internet Address      Physical Address      Type
  128.2.20.218          00-b0-8e-83-df-50     dynamic
  128.2.102.129         00-b0-8e-83-df-50     dynamic
  128.2.194.66          00-02-b3-8a-35-bf     dynamic
  128.2.198.34          00-06-5b-f3-5f-42     dynamic
  128.2.203.3           00-90-27-3c-41-11     dynamic
  128.2.203.61          08-00-20-a6-ba-2b     dynamic
  128.2.205.192         00-60-08-1e-9b-fd     dynamic
  128.2.206.125         00-d0-b7-c5-b3-f3     dynamic
  128.2.206.139         00-a0-c9-98-2c-46     dynamic
  128.2.222.180         08-00-20-a6-ba-c3     dynamic
  128.2.242.182         08-00-20-a7-19-73     dynamic
  128.2.254.36          00-b0-8e-83-df-50     dynamic
```

15

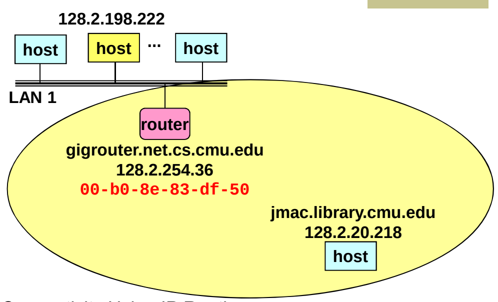
CMU’s Internal Network Structure



- CMU Uses Routing Internally
  - Maintains forwarding tables using OSPF
  - Most CMU hosts cannot be reached at link layer

16

Proxy ARP



- Provides Link-Layer Connectivity Using IP Routing
  - Local router (gigrouter) sees ARP request
  - Uses IP addressing to locate host, i.e., which subnet
  - Replies with its own MAC address - becomes “Proxy” for remote host
    - Must then forward packets for that destination
  - Requestor thinks that it is communicating directly with remote host

17

Outline

- IPv6
- Translation: too many names and addresses!
  - ARP
  - DNS

18

Naming

- How do we efficiently locate resources?
  - DNS: name → IP address
- Challenge
  - How do we scale this to the wide area?

19

Obvious Solutions (1)

Why not centralize DNS?

- Distant centralized database
  - Traffic volume
- Single point of failure
- Single point of update
- Single point of control

- Doesn’t *scale!*

20

Obvious Solutions (2)

Why not use /etc/hosts?

- Original Name to Address Mapping
  - Flat namespace
  - /etc/hosts keeps track of the mappings
  - SRI kept main copy
  - Downloaded regularly
- Count of hosts was increasing: machine per domain → machine per user
  - Many more downloads
  - Many more updates

Domain Name System Goals

- Basically a wide-area distributed database
- Scalability
- Decentralized maintenance
- Robustness
- Global scope
  - Names mean the same thing everywhere
- Don't need
  - Atomicity
  - Strong consistency

Programmer's View of DNS

- Conceptually, programmers can view the DNS database as a collection of millions of *host entry structures*:

```
/* DNS host entry structure */
struct addrinfo {
        int    ai_family;              /* host address type (AF_INET) */
        size_t ai_addrlen;             /* length of an address, in bytes */
        struct sockaddr *ai_addr;      /* address! */
        char   *ai_canonname;          /* official domain name of host */
        struct addrinfo *ai_next;      /* other entries for host */
};
```

- Functions for retrieving host entries from DNS:
  - `getaddrinfo`: query key is a DNS host name.
  - `getnameinfo`: query key is an IP address.

DNS Records

RR format: **(class, name, value, type, ttl)**

- DB contains tuples called resource records (RRs)
  - Classes = Internet (IN), Chaosnet (CH), etc.
  - Each class defines value associated with type

FOR IN class:

- Type=A
  - **name** is hostname
  - **value** is IP address
- Type=NS
  - **name** is domain (e.g. foo.com)
  - **value** is name of authoritative name server for this domain
- Type=CNAME
  - **name** is an alias name for some "canonical" (the real) name
  - **value** is canonical name
- Type=MX
  - **value** is hostname of mailserver associated with **name**

Properties of DNS Host Entries

- Different kinds of mappings are possible:
  - Simple case: 1-1 mapping between domain name and IP addr:
    - `kittyhawk.cmcl.cs.cmu.edu` maps to `128.2.194.242`
  - Multiple domain names maps to the same IP address:
    - `eecs.mit.edu` and `cs.mit.edu` both map to `18.62.1.6`
  - Single domain name maps to multiple IP addresses:
    - `aol.com` and `www.aol.com` map to multiple IP addrs.
  - Some valid domain names don't map to any IP address:
    - for example: `cmcl.cs.cmu.edu`

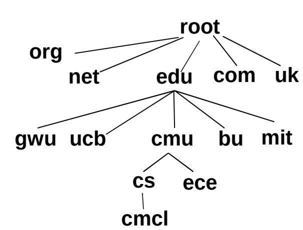
DNS Message Format

12 bytes	Identification	Flags
	No. of Questions	No. of Answer RRs
	No. of Authority RRs	No. of Additional RRs
Name, type fields for a query →	Questions (variable number of answers)	
RRs in response to query →	Answers (variable number of resource records)	
Records for authoritative servers →	Authority (variable number of resource records)	
Additional "helpful info that may be used →	Additional Info (variable number of resource records)	

DNS Header Fields

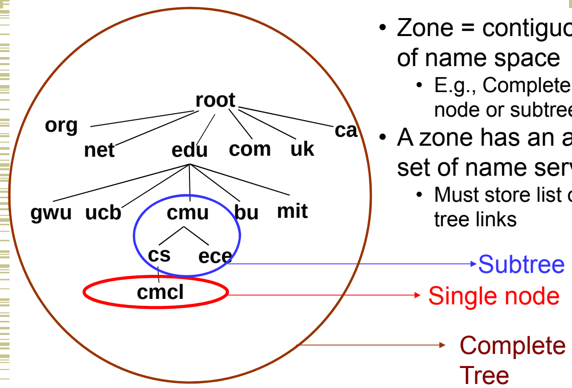
- Identification
  - Used to match up request/response
- Flags
  - 1-bit to mark query or response
  - 1-bit to mark authoritative or not
  - 1-bit to request recursive resolution
  - 1-bit to indicate support for recursive resolution

DNS Design: Hierarchy Definitions



- Each node in hierarchy stores a list of names that end with same suffix
  - Suffix = path up tree
- E.g., given this tree, where would following be stored:
  - Fred.com
  - Fred.edu
  - Fred.cmu.edu
  - Fred.cmcl.cs.cmu.edu
  - Fred.cs.mit.edu

DNS Design: Zone Definitions



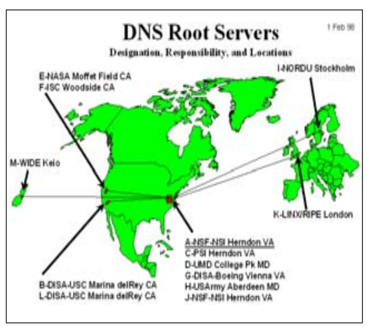
- Zone = contiguous section of name space
  - E.g., Complete tree, single node or subtree
- A zone has an associated set of name servers
  - Must store list of names and tree links

DNS Design: Management

- Zones are created by convincing owner node (parent) to create/delegate a subzone
  - Records within zone stored multiple redundant name servers
  - Primary/master name server updated manually
  - Secondary/redundant servers updated by zone transfer of name space
    - Zone transfer is a bulk transfer of the “configuration” of a DNS server – uses TCP to ensure reliability
- Example:
  - CS.CMU.EDU created by CMU.EDU administrators
  - Who creates CMU.EDU or .EDU?

DNS: Root Name Servers

- Responsible for “root” zone
- Approx. 13 root name servers worldwide
  - Currently {a-m}.root-servers.net
  - Very well protected
- Local name servers contact root servers when they cannot resolve a name
  - Configured with well-known root servers
  - Newer picture → www.root-servers.org



Root Zone

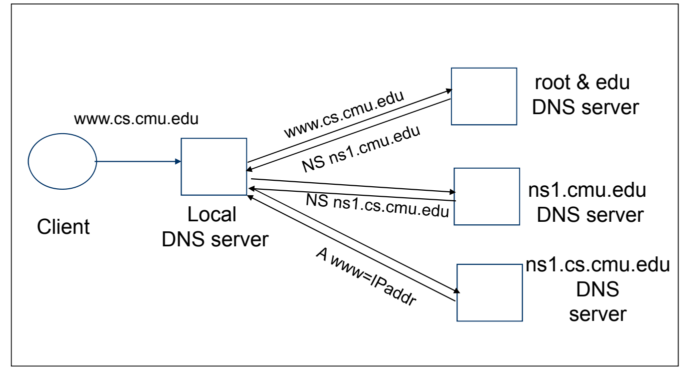
- Generic Top Level Domains (gTLD) = .com, .net, .org, etc…
- Country Code Top Level Domain (ccTLD) = .us, .ca, .fi, .uk, etc…
- Root server ({a-m}.root-servers.net) also used to cover gTLD domains
  - Load on root servers was growing quickly!
  - Moving .com, .net, .org off root servers was clearly necessary to reduce load → done Aug 2000

Servers/Resolvers

- Each host has a resolver
  - Typically a library that applications can link to
  - Local name servers hand-configured (e.g. /etc/resolv.conf)
- Name servers
  - Either responsible for some zone or…
  - Local servers
    - Do lookup of distant host names for local hosts
    - Typically answer queries about local zone

33

Typical Resolution



34

Typical Resolution: Steps

- Steps for resolving www.cmu.edu
  - Application calls gethostbyname() (RESOLVER)
  - Resolver contacts local name server (S_1)
  - S_1 queries root server (S_2) for (www.cmu.edu)
  - S_2 returns NS record for cmu.edu (S_3)
  - What about A record for S_3?
    - This is what the additional information section is for (PREFETCHING)
  - S_1 queries S_3 for www.cmu.edu
  - S_3 returns A record for www.cmu.edu

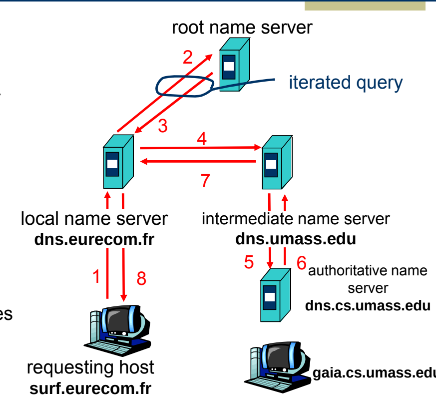
35

Lookup Methods

Recursive query:
- Server goes out and searches for more info (recursive)
- Only returns final answer or "not found"

Iterative query:
- Server responds with as much as it knows (iterative)
- "I don't know this name, but ask this server"

Workload impact on choice?
- Local server typically does recursive
- Root/distant server does iterative

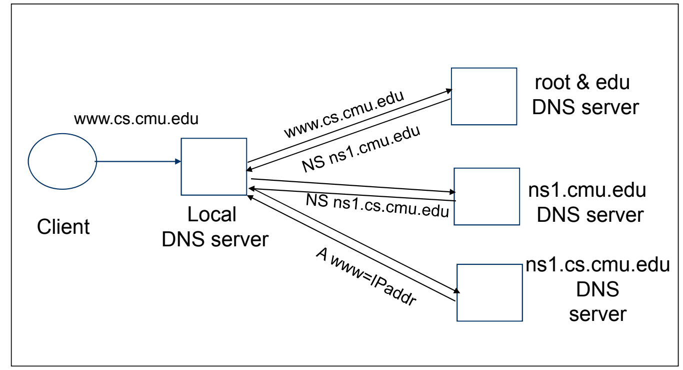


36

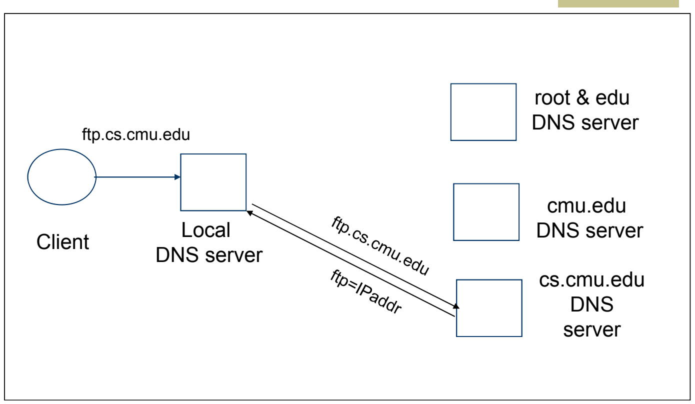
Workload and Caching

- Are all servers/names likely to be equally popular?
  - Why might this be a problem? How can we solve this problem?
- DNS responses are cached
  - Quick response for repeated translations
  - Other queries may reuse some parts of lookup
- DNS negative queries are cached
  - Don't have to repeat past mistakes, e.g., misspellings
- Cached data periodically times out
  - Lifetime (TTL) of data controlled by owner of data
  - TTL passed with every record
- Responses can include additional information
  - Often used for prefetching, e.g., CNAME/MX/NS records

Typical Resolution



Subsequent Lookup Example



Reliability

- DNS servers are replicated
  - Name service available if ≥ one replica is up
  - Queries can be load balanced between replicas
  - Queries return multiple A records
- UDP used for queries
  - Need reliability → must implement this on top of UDP!
  - Why not just use TCP?
- Try alternate servers on timeout
  - Exponential backoff when retrying same server
- Same identifier for all queries
  - Client does not care which server responds

Mail Addresses

- MX records point to mail exchanger for a name
  - E.g. mail.acm.org is MX for acm.org
- Addition of MX record type proved to be a challenge
  - How to get mail programs to lookup MX record for mail delivery?
  - Needed critical mass of such mailers

41

Tracing Hierarchy (1)

- Dig Program
  - Allows querying of DNS system
  - Use flags to find name server (NS)
  - Disable recursion so that operates one step at a time

```
unix> dig +norecurse @a.root-servers.net NS kittyhawk.cmcl.cs.cmu.edu

;; AUTHORITY SECTION:
edu.            172800  IN    NS    L3.NSTLD.COM.
edu.            172800  IN    NS    D3.NSTLD.COM.
edu.            172800  IN    NS    A3.NSTLD.COM.
edu.            172800  IN    NS    E3.NSTLD.COM.
edu.            172800  IN    NS    C3.NSTLD.COM.
edu.            172800  IN    NS    F3.NSTLD.COM.
edu.            172800  IN    NS    G3.NSTLD.COM.
edu.            172800  IN    NS    B3.NSTLD.COM.
edu.            172800  IN    NS    M3.NSTLD.COM.
```

42

DNS Summary

- Motivations → large distributed database
  - Scalability
  - Independent update
  - Robustness
- Hierarchical database structure
  - Zones
  - How is a lookup done
- Caching/prefetching and TTLs
- Reverse name lookup
- What are the steps to creating your own domain?

43