IPv6 and DNS

Chapters 22,29

CSA 442

IPv6 – The Future of IP

- Current version of IP - version *4* - is over 20 years old
- IPv4 has shown remarkable ability to move to new technologies
  - IP has accommodated dramatic changes since original design
  - Basic principles still appropriate today
  - Many new types of hardware
  - Scaling - from a few tens to a few tens of millions of computers
- But, as with any old technology, it has some problems
- IETF has proposed entirely new version to address some specific problems

Motivation for Change

- Address space
  - 32 bit address space allows for over a million networks
  - But…all that is left is Class C and too small for many organizations
  - Predictions we would have run out of IP addresses by now
  - Besides, how will we network all our toasters and cell phones to the Internet?
- Type of service
  - Different applications have different requirements for delivery reliability and speed ; i.e. real time data, quality of service
  - Current IP has type of service that's not often implemented
- Multicast

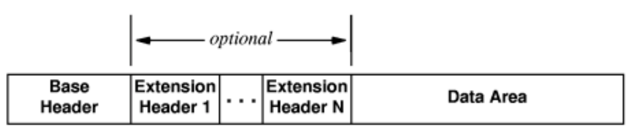
Name and Version Number

- Preliminary versions called *IP - Next Generation* (IPng)
- Several proposals all called IPng
- One was selected and uses next available version number (6)
  - Version 5 was already assigned to an experimental protocol, *ST, Streams Protocol*
- Result is *IP version 6* (IPv6)
- In the works since 1990

New Features of IPv6

- Address size - IPv6 addresses are 128bits
  - ~$3*10^{38}$ possible addresses in theory
- Header format - entirely different
- Extension headers - Additional information stored in optional extension headers, followed by data
  - Makes the protocol extensible – new features can be added more easily
- Support for audio and video - flow labels and quality of service allow audio and video applications to establish appropriate connections

IPv6 Datagram Format

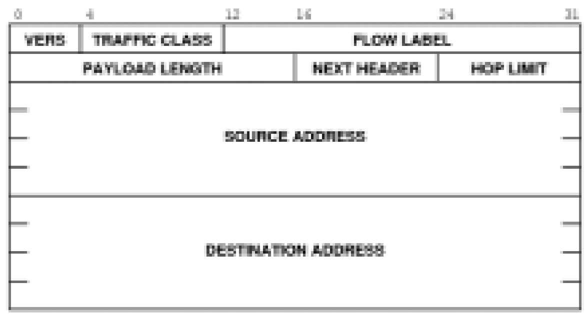


Base header is followed by some optional number of Extension headers followed by the data

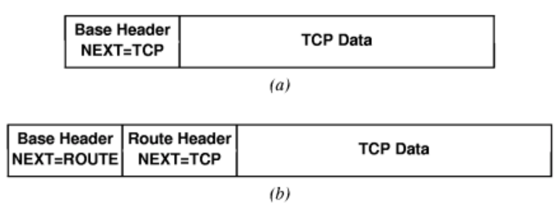
Base header is a fixed 40 byte length

IPv6 Base Header

- Despite larger size, contains less information than IPv4 header
- NEXT HEADER points to first extension header
- FLOW LABEL used to associate datagrams belonging to a *flow* or communication between two applications
  - *Traffic class* used to establish priorities for the packet
  - Routers use FLOW LABEL to forward datagrams along prearranged path



IPv6 Next Header



Next Header field indicates what comes next
If no extension headers, identifies type of payload

If extension header is added on, Next Header identifies type
Extension header must specify length of header, following header

Why Multiple Headers?

- Efficiency - header only as large as necessary
- Flexibility - can add new headers for new features
- Incremental development - can add processing for new features to testbed; other routers will skip those headers

Other Changes from IPv6

- Checksum: removed entirely to reduce processing time at each hop
  - Depends on checksum for Ethernet, TCP
- Fragmentation only allowed at source
  - No fragmentation at intermediate routers
  - Router will drop, send error message to source telling it to send a smaller packet, source must find smallest MTU of intermediate networks (*path MTU discovery*)
- ICMPv6: new version of ICMP
  - additional message types, e.g. “Packet Too Big”
  - Multicast group management functions

Addressing

- 128-bit addresses
- Includes network prefix and host suffix, just like IPv6 but bigger address space
- No address classes - prefix/suffix boundary can fall anywhere as in CIDR
- Special types of addresses:
  - *unicast* - single destination computer
  - *multicast* - multiple destinations; possibly not at same site
  - *cluster* - collection of computers with same prefix; datagram is delivered to one out of cluster
- Cluster addressing allows for duplication of services, e.g. specify a cluster of servers providing the same service, but we just want at least one of them to work

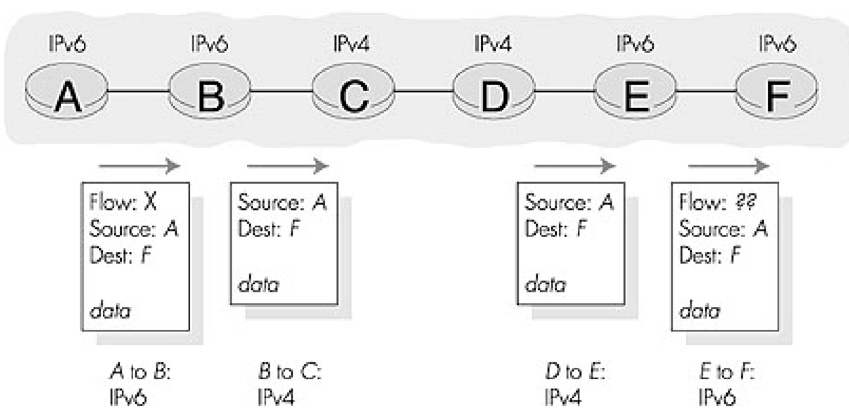
Address Notation

- 128-bit addresses unwieldy in dotted decimal; requires 16 numbers
    - 105.220.136.100.255.255.255.255.0.0.18.128.140.10.255.255
- Groups of 16-bit numbers in hex separated by colons - *colon hexadecimal* (or *colon hex*)
    - 69DC:8864:FFFF:FFFF:0:1280:8C0A:FFFF
- Zero-compression - series of zeroes indicated by two colons
    - FF0C:0:0:0:0:0:0:B1
    - FF0C::B1
- IPv6 address with 96 leading zeros is interpreted to hold an IPv4 address

Transition From IPv4 To IPv6

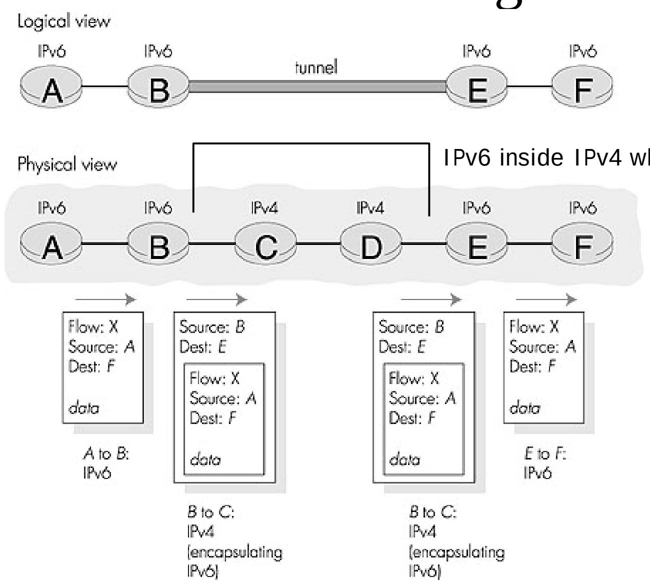
- Not all routers can be upgraded simultaneously
  – no “flag days”
  – How will the network operate with mixed IPv4 and IPv6 routers?
- Two proposed approaches:
  – Dual Stack: some routers with dual stack (v6, v4) can “translate” between formats
  – Tunneling: IPv6 carried as payload in IPv4 datagram among IPv4 routers

Dual Stack Approach



some IPv6 stuff may be lost (e.g. flow control) in conversion

Tunneling



IPv6 inside IPv4 where needed

Only lose IPv6 stuff inside the tunnel, not outside

IP v6 Lessons

- Will IPv6 arrive in the near future?
  - Used in a few places, definitely not widespread though
  - Some North American ISP’s have said they don’t plan to buy IPv6 enabled equipment
    - Cite little demand
    - Expensive
    - Can “hack” IPv4
      - CIDR
      - NAT – Network Address Translator
        - » Set up a private IP address space, translate with a gateway
  - Lesson: Enormously difficult to change a fundamental protocol, best to anticipate as much as possible and plan for growth

DNS – Domain Name System

Introduction to DNS

- IP assigns 32-bit addresses to hosts (interfaces)
- Binary addresses easy for computers to manage
- All applications use IP addresses through the TCP/IP protocol software
- But difficult for humans to remember:
    - % telnet 137.229.114.139
- The *Domain Name System* (DNS) provides translation between symbolic names and IP addresses

Structure of DNS Names

- Each name consists of a sequence of alphanumeric components separated by periods
- Examples:
  - www.math.uaa.alaska.edu
  - thanatos.uaa.alaska.edu
  - www.alaska.edu
- Names are hierarchical, with most-significant component on the right
- Middle is the organization
- Left-most component is computer name

DNS Naming Structure

- *Top level domains* (right-most components; also known as *TLD*s) defined by the global authority ICANN
  - com   Commercial organization
  - edu   Educational institution
  - gov   Government organization
  - mil   Military organization
- Organizations apply for names in a top-level domain:
  - alaska.edu
  - mcdonalds.com
- Organizations determine own internal structure
  - E.g. www.alaska.edu, bigmac.mcdonalds.com

Geographic Structure

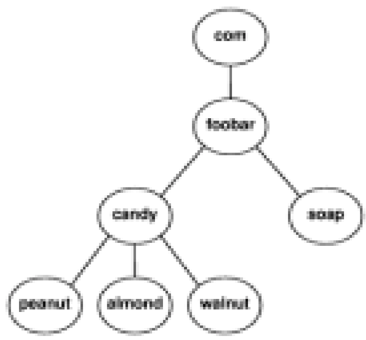
- Top-level domains are US-centric
- Geographic TLDs used for organizations in other countries:
  - .uk United Kingdom
  - .fr France
  - .ch Switzerland
  - .to Togo
- Countries define their own internal hierarchy: ac.uk and .edu.au are used for academic organizations in the United Kingdom and Australia

Domain Names within an Org

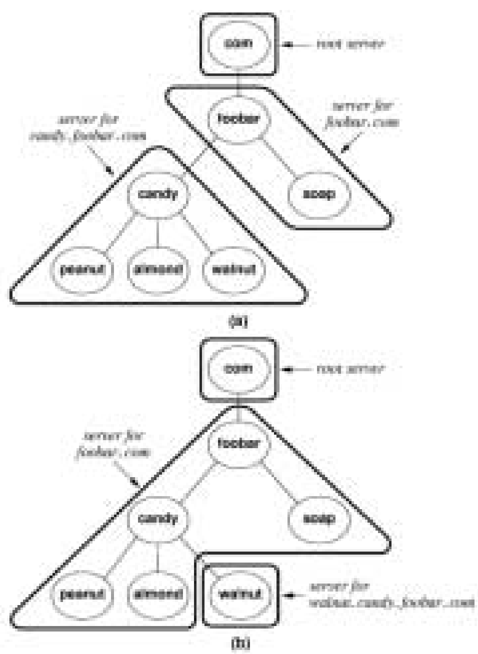
- Organizations can create any internal DNS hierarchy
- Uniqueness of TLD and organization name guarantee uniqueness of any internal name (much like file names in your directories)
- All but the left-most component of a domain name is called the *domain* for that name:
- Authority for creating new *subdomains* is delegated to each domain
  - E.g. Name = www.cs.ucdavis.edu
  - Domain = cs.ucdavis.edu
- Administrator of cs.ucdavis.edu could create krunk.cs.ucdavis.edu

Example Hierarchy

- Domain: foobar.com
- Subdomain: soap.foobar.com,   candy.foobar.com
- Machine: liquid.soap.foobar.com, peanut.candy.foobar.com



DNS Client-Server



- DNS names are managed by a hierarchy of DNS servers
- Hierarchy is related to DNS domain hierarchy
- Root server at top of tree knows about next level servers
- Next level servers, in turn, know about lower level servers

Choosing a DNS Architecture

- Small organizations can use a single server
  - Easy to administer
  - Inexpensive
- Large organizations often use multiple servers
  - Reliability through redundancy
  - Improved response time through load-sharing
  - Delegation of naming authority
- Locality of reference applies - users will most often look up names of computers within same organization

Name Resolution

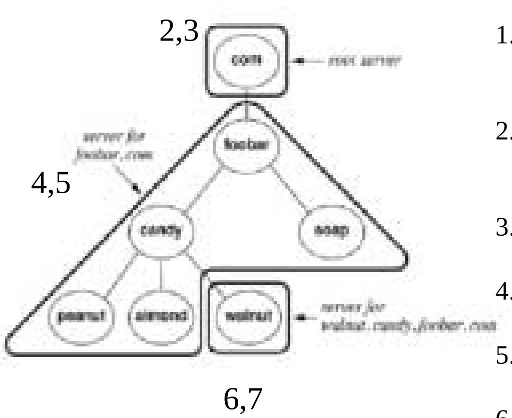
- Resolver software typically available as library procedures
  - Implement DNS application protocol
  - Software configured for local servers
  - Example - UNIX gethostbyname or built into the OS
- Calling program is *client*
  - Constructs DNS protocol message - a *DNS request*
  - Sends message to local DNS server, “What is the IP address of machine <blah>?”
- DNS *server* resolves name
  - Constructs DNS protocol message - a *DNS reply* containging the IP address of the requested name
  - Sends message to client program and waits for next request

DNS Servers

- Each DNS server is the *authoritative server* for the names it manages
- If request contains name managed by receiving server, that server replies directly
- Otherwise, request must be forwarded to the appropriate authoritative server

- Process:
  - Client contacts local DNS server, L
  - If L knows the requested IP or is the authority, return the IP
  - Otherwise, contact the root server
    - Root server returns to L the authoritative server for the domain
    - L contacts this server
    - Process may repeat until we find the authoritative server

DNS Lookup Example



Iterative Resolution

1. Computer requests IP for comp.walnut.candy.foobar.com from local DNS
2. Not found in local DNS, local DNS becomes a client and contacts root server
3. Root server returns server below, for foobar.com
4. Local DNS contacts server at foobar.com
5. Foobar.com server returns server below, for walnut.foobar.com
6. Local DNS contacts server at walnut.foobar.com
7. This is the authority, returns IP
8. Local DNS returns IP to Computer

DNS Efficiencies

- DNS resolution can be **very** inefficient
  - Every host referenced by name triggers a DNS request
  - Every DNS request for the address of a host in a different organization goes through the root server
- Servers and hosts use *caching* to reduce the number of DNS requests
  - Cache is a list of recently resolved names and IP addresses
  - Authoritative server include *time-to-live* with each reply
- Servers use *replication* to decrease the load on root servers
- DNS servers use UDP for efficiency
  - Port 53 UDP, Port 53 TCP for long messages
  - Often running Berkeley Internet Name Domain (BIND) s/w

Types of DNS Entries

- DNS can hold several types of records
- Each record includes
  - Domain name, Record type, Data value
- *"A"* Type records map from domain name to IP address
  - Domain name - mazzy
  - Record type - A
  - Data value – 137.229.134.207
- Other types:
  - *MX* (Mail eXchanger) - maps domain name used as e-mail destination to IP address
  - *CNAME* - alias from one domain name to another
- Result - name that works with one application may not work with another! (e.g. could email to a domain but not ping it)

Abbreviations

- May be convenient to use abbreviations for local computers; e.g. mazzy for mazzy.math.uaa.alaska.edu
- Abbreviations are handled in the *resolver*; DNS servers only know *full-qualified domain names* (FQDNs)
- Local resolver is configured with list of suffixes to append
- Suffixes are tried sequentially until match found
- Other heuristics may be tried (e.g. add .com)

Summary

- Domain Name System maps from computer names and IP addresses
- Important to hide 32-bit IP addresses from humans
- DNS names are hierarchical and allocated locally
- Replication and caching are important performance enhancements
- DNS provides several types of records