Naming and Addressing

Digital Communications II

Michaelmas Term 2007
Based on Prof. Jon Crowcroft's notes, and thus transitively on
S. Keshav's "An Engineering Approach to Computer Networking"

Outline

- What you saw in Digital Communications I
- Names and addresses
- Name resolution
- Hierarchical naming
- Examining DNS
- Addressing
- Addressing in the telephone network
- Addressing in the Internet
- ATM addresses
- Finding datalink layer addresses

What you saw in Digital Communications I

- Terminology
  - **Names** denote something
  - **Addresses** denote where something is
  - A **route** tells you how to get to an address
- (Another perspective: they're all names – often "impure" ones)
- **Binding** is the key process of linking, e.g.
  - names to addresses,
  - addresses to routes,
  - addresses to hosts, etc.
- Overview of IP names and addresses

Names and addresses (for IP here)

- Names and addresses are both ways to uniquely identify a host (or an interface on the host)
-
```
: dme26@pip:510:0$; nslookup www.srcf.ucam.org
Server:         127.0.0.1
Address:        127.0.0.1#53
www.srcf.ucam.org  canonical name = kern.srcf.ucam.org.
Name:   kern.srcf.ucam.org
Address: 131.111.179.82
```
- Other similar tools: **dig** and **host**.
- *Resolution*: the process of determining an address from a name

Names and addresses on various layers

- Application – your web browser:
  (N/A) … http://www.cl.cam.ac.uk/DeptInfo/CST06/node62.html
- DNS:
  pip.srcf.ucam.org … www.cl.cam.ac.uk
- Transport:
  131.111.179.83:44127 … 128.232.0.20:80
- Network:
  131.111.179.83 … 128.232.0.20
- Datalink:
  00:04:23:D9:91:6C … (unknown by me)

Why do we need both?

- Names tend to be for human use
  - Often 'long' and frequently variable length
  - 'Difficult' for computers to parse
  - Wastes space to carry them in packet headers
- Addresses are shorter and machine understandable
  - If fixed size, easier to carry in headers and to parse
  - Probably amenable to making routing decisions efficiently
- Indirection
  - The usual story: abstraction benefits + dereferencing costs
  - Multiple names may point to same address
  - Can move a machine and just update the resolution table

Name resolution

- Done by name servers
  - essentially look up a name and return an address

- Centralized design
  - consistent
  - single point of failure
  - concentrates load

Naming

- Goal: our addressable 'things' have sufficiently unique names
  - Often things are hosts. (e.g. IP's perspective)
  - Can also be services. (e.g. Appletalk, CNAMEs, …)

- What 'sufficient' means in the above depends on the context
  - Wide area or not?
  - Aiming to make the technology reasonably 'future-proof'?

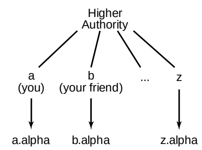
- Want the following operations to be scalable and efficient:
  - Creation of names (well, CRUD really)
  - Lookup of names

Hierarchical Naming

- Naïve approach: ask other naming authorities whether the name you're proposing is unique before using it
  - doesn't scale (why?)
  - not robust to network partitions

- Instead recursively decompose the *name space* (the set of all possible names) into mutually exclusive portions.

- As you know, such hierarchies:
  - Can scale arbitrarily
  - Guarantee naming uniqueness
  - Are fairly easy to comprehend



9

Before DNS

- Surely a naming within a hierarchy is crucial / prudent ?
- Not if you are only considering LAN-scope interconnections:
  - Appletalk (early versions), NetBIOS (NetBUI), …

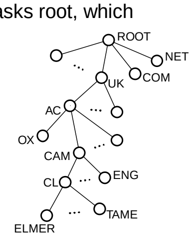
- Iterate a few years, and we have:
  - Appletalk over TCP/IP, NetBIOS over TCP/IP, …

- Explicit 'hosts' configuration
  - /etc/hosts or %windir%\System32\drivers\etc\hosts
  - Provided list of IP / hostname mappings

10

The Domain Name Service (DNS)

- Distributed name server
- A name server is responsible (an *authoritative server)* for a set of domains
- May delegate responsibility for part of a domain to a child
- Root servers are *replicated*
- If local server cannot answer a query, it asks root, which delegates reply
- Reply is *cached* and timed out



11

The Domain Name Service (DNS)

- The **Domain Name Service (DNS)** is ubiquitous Internet-wise
  - Scalable through hierarchy and distribution
  - Tree of names separated by periods
  - All names in the same domain share a unique *suffix*

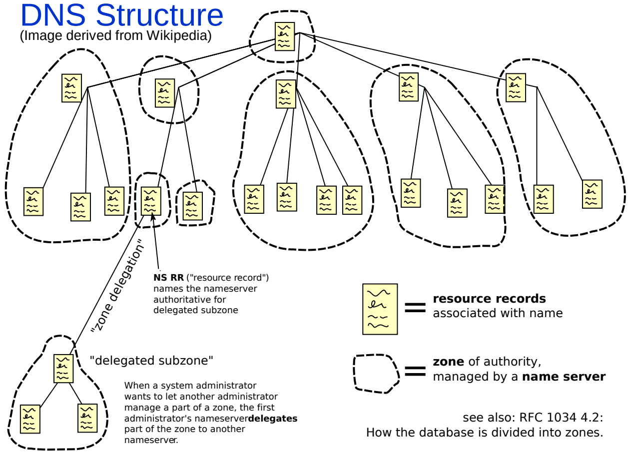
- Legally, ICANN (Internet Corporation for Assigned Names and Numbers) allocates Top Level Domains (TLD)

- Top Level Domains – we're all familiar with .com, .edu, .uk, …
  - Newer TLDs: .aero, .jobs, .museum, .travel, …

- Naming authorities can delegate the naming of subdomains recursively by defining **zones**.

12

DNS Structure
(Image derived from Wikipedia)



DNS steps (rather oversimplified)

- Lookup www.cl.cam.ac.uk.
- OS configured with root nameserver (**root hints**)
  - The "." suffix is essentially the root name-server
  - Assume we use 198.41.0.4 as root nameserver
- Ask 198.41.0.4 who knows about the ".uk" domain?
  - Receive IP address of a .uk TLD server (195.66.240.130)
- Ask 195.66.240.130 about the ".ac" sub-domain (of ".uk.")?
- Ask 128.86.1.20 about ".cam" sub-domain (of ".ac.uk.")
- 131.111.8.42 knows who to ask about .cl.cam.ac.uk.
- Eventually 128.232.1.1 will tell us 128.232.0.20 is what we want

DNS records

- Does much more than just DNS name to IP mapping.
- Some common DNS record types (from RFC 1035):
  - **A** – An IPv4 address
  - **AAAA** – An IPv6 address
  - **NS** – Name server record (effects zone delegation)
  - **CNAME** – an alias mapping a name to an **A** record
  - **SOA** – start of authority
  - **PTR** – used in reverse DNS
  - **MX** – email handling
  - **TXT** – free-form text… but now used heavily for programmatic functions too…
- New records can be added:
  - **NAPTR** – regular expression rewriting of URIs (!)

DNS steps (closer to what happens)

- Your computer does not pound the DNS root servers
  - It is configured to use a closer DNS server
  - It also caches DNS responses
- Your closer DNS server will also cache responses
- Simple **resolvers** may expect the DNS server to perform recursive queries on their behalf
  - Recursing name servers accumulate useful cache data
- Note that NS (zone delegation) records use DNS names
  - Sometimes need **glue records** to add IP information
- Negative caching is possible if SOA record included
  - Helps solve problems caused by 68 year TTL fields…

Common DNS usage

- CNAMEs allow aliases to refer to a canonical name.
  - Frequently used to map service names to specific hosts
  - E.g. ftp.csx.cam.ac.uk → zircon.csx.cam.ac.uk.
- Reverse DNS – getting a DNS name for an IP address

$ host 128.232.100.4
4.100.232.128.in-addr.arpa domain name pointer heathrow.net.cl.cam.ac.uk.

  - Clearly .in-addr.arpa collects IPv4 information (in byte chunks)
  - ip6.arpa collects IPv6 information (in nibble chunks)
- The reverse octet order hints at IPv4 *address* hierarchy…
- Caching issues
  - Some badly behaved web browsers maintain their own cache

17

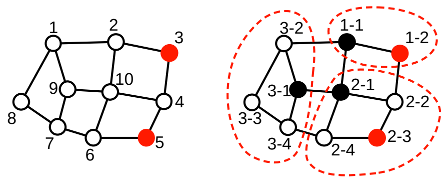
DNS extensions

- Original DNS was not focused on security
  - TSIG allows a shared secret to sign DNS conversations
  - DNSSEC is an overall effort to secure DNS
    - DNS validity is critical
- TXT records are being used in spam combat:
  - Sender Policy Framework (SPF)
    - Indicate hosts in a domain that are allowed to send email
  - DomainKeys (DKIM)
    - Check for forged email origination

18

Addressing

- Addresses need to be globally unique, so again we use hierarchical schemes
- Another reason for hierarchy: *aggregation*
  - reduces size of routing tables
  - at the expense of longer routes



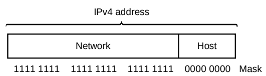
19

Addressing in the telephone network

- Telephone network has only addresses and no names (why?)
- E.164 specifications
  - Up to 15 digits "+" prefix
- ITU assigns each country a unique *country code*
- Naming authority in each country chooses unique prefixes (e.g. area)
- Telephone numbers are variable length
  - this is OK since they are only used in call establishment
- Optimisation to help dialling – use lower level name space directly
  - Pointers address higher level domains ('0' in this case)
  - UK Ofcom's National Numbering Plan
    - 01 geographic area (01223…)
    - 08 premium rate (they know you'll pay)
    - and 00 gets us to the international scope…

20

Addressing in the Internet



- You looked at IPv4 in DigiComms I
- IP addresses are 4 bytes long, two part hierarchy
  - network number and host number
  - boundary identified with a *subnet* mask
  - can aggregate addresses within subnets
- Originally every host interface had its own IP address
- Routers have multiple interfaces, each with its own IP address
- First addressing scheme: 8-bits of network number only.
  - Only 256 networks on the Internet? Well…

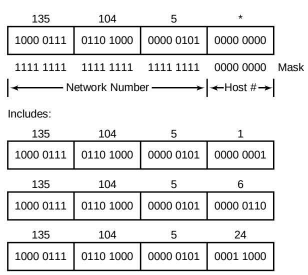
21

Address classes

- Second addressing scheme:
  - Class A addresses have 8 bits of network number
  - Class B addresses have 16 bits of network number
  - Class C addresses have 24 bits of network number
- Distinguished by leading bits of address
  - leading 0 → class A (first byte < 128)
  - leading 10 → class B (first byte in the range 128-191)
  - leading 110 → class C (first byte in the range 192-223)
  - (also class D, class E, but they are not relevant here)
- As you know – class A is too big, class C is too small.
  - Why would this have been tried anyway?

22

Address evolution



- Extending IPv4's functional lifetime
  - Subnetting
  - Classless Inter-Domain Routing
  - Dynamic host configuration
  - Network Address Translation
- Subnetting
  - Allows administrator to cluster IP addresses *within* a network
  - E.g. CL's 128.232.0.0 class B now carved up within UCam
  - King's admin network: 131.111.199.128 / 255.255.255.128

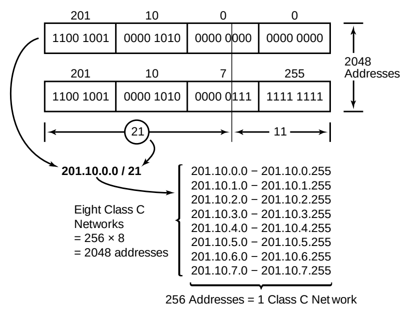
23

CIDR

- 'Classful' scheme forced medium sized nets to choose class B addresses: wasteful and risked address space exhaustion
- The CIDR solution (around 1993)
  - allow contiguous class C blocks to be referred to as a larger address block
  - use a CIDR mask,
  - idea is very similar to subnet masks,
    - except that all routers must agree to use it:
    - subnet masks are not visible outside the network (why?)

24

CIDR (contd.)



25

Dynamic host configuration

- Allows a set of hosts to share a pool of IP addresses
- Dynamic Host Configuration Protocol (DHCP)
- Newly booted computer broadcasts **discover** to subnet
- DHCP servers reply with **offers** of IP addresses
- Host picks one and broadcasts a **request** to a particular server
- All other servers withdraw offers, and selected server sends an **ack**
- When done, host sends a **release**
- IP address has a **lease** which limits time it is valid
- Server *reuses* IP addresses if their lease is over
- Similar technique used in *Point-to-point* protocol (PPP)

26

Network Address Translation (NAT)

- DHCP breaks static one-interface per IP address association
- Go even further with NAT:
  - Aggregate multiple IP addresses 'behind' one IP address
  - This does not make sense at the network level
    - "Cheat": multiplex network level onto the transport level
    - I.e. use UDP and TCP ports.
- To avoid the 'hidden' addresses messing up Internet routing:
  - Reuse sets of non-routable addresses across organisations:
  - 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16, or 169.254.0.0/16
  - Of course such traffic does leak out occasionally…
- Problem is that reaching a host behind NAT can be difficult

27

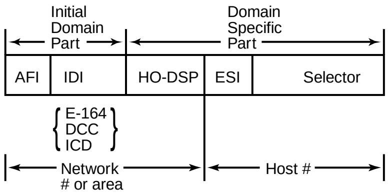
IPv6

- 32-bit address space will run out fairly soon (2010?).
- IPv6 extends address size to 128 bits
- Main features
  - classless addresses from the outset
  - multiple levels of aggregation are possible
    - registry
    - provider
    - subscriber
    - subnet
  - several flavours of multicast
  - anycast – route packets to one of a set of hosts
  - reasonable interoperability with IPv4

28

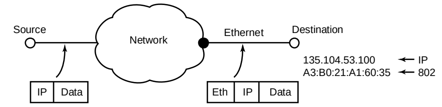
ATM network addressing

- Uses *Network Service Access Point (NSAP)* addresses
- Variable length (7-20 bytes)
- Several levels of hierarchy
  - national or international naming authority
  - addressing domain
  - subnet

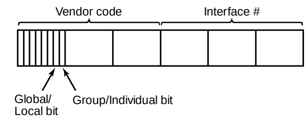


Finding datalink layer addresses

- So much for the network, but what about on a LAN?
- Need to know datalink layer address typically for the last hop



- In DigiComms I: most common datalink address is IEEE 802
- Media Access Control (MAC) or Ethernet addresses:
  - Hierarchically *allocated*, but not hierarchically *deployed*



ARP

- To get datalink layer address of a machine on the local subnet
  - Datalink broadcast a query with IP address onto local LAN
  - Host that owns that address (or proxy) replies with address
  - All hosts are required to listen for ARP requests and reply
- Reply stored in an ARP cache and timed out
- In point-to-point LANs, need an ARP server
  - register translation with server
  - ask ARP server instead of broadcasting
- ARP is susceptible to spoofing
  - Attacker with Ethernet access manipulates IP/MAC mapping:
    - Poses as the network gateway
    - ‘Poisons’ a victim’s ARP cache