IP Addressing & Routing

G54ACC

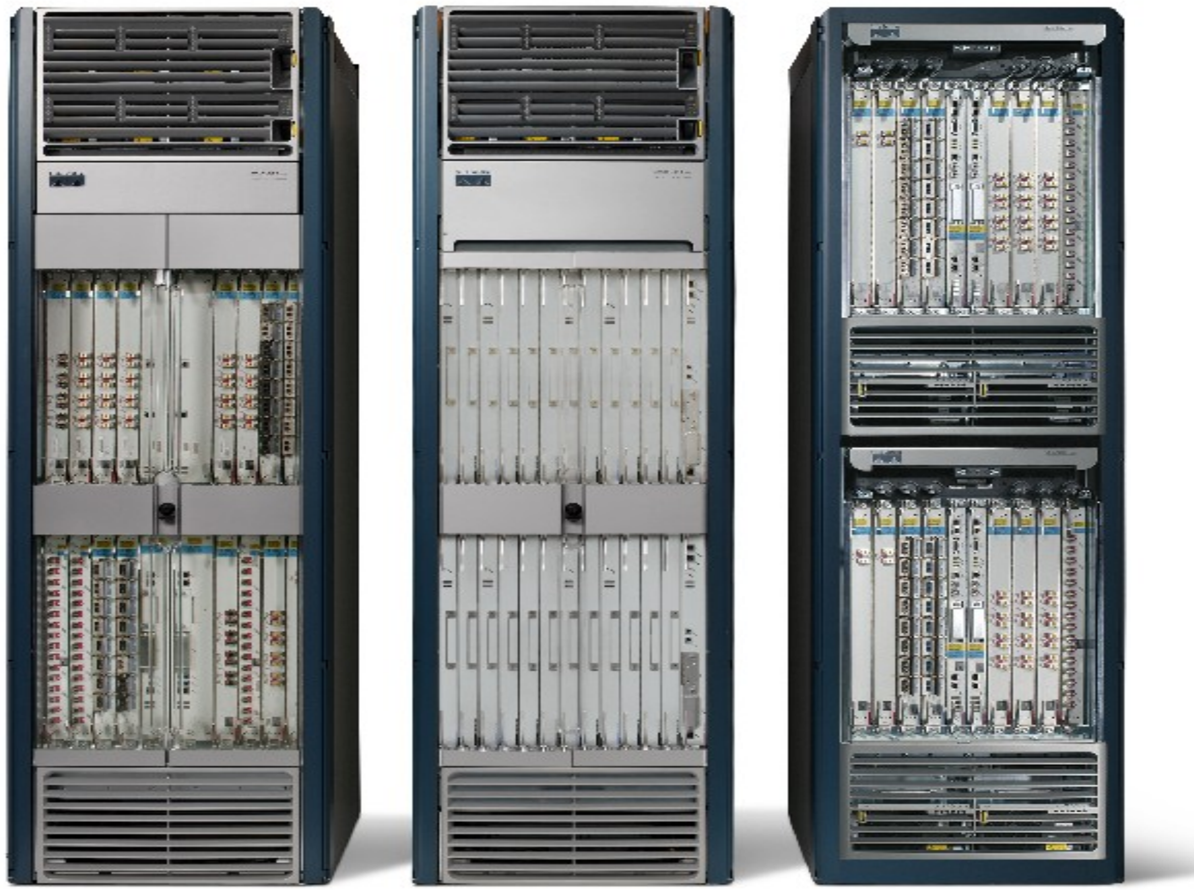
Lecture 3

richard.mortier@nottingham.ac.uk

Recap

- “The Internet” consists of connected routers
- Routers *store & forward* packets toward destinations
- Decisions are based on IP destination address and contents of *routing tables*

Internet Routers



Cisco CRS-1
Multi-shelf system

Contents

- Addressing
- Routing *vs*. Forwarding
- IP Routing

Contents

- Addressing
  - Address Management
  - Address Shortages
  - IPv4 Issues and IPv6
- Routing *vs*. Forwarding
- IP Routing

Internet Addressing

- Host addresses in IPv4 are 32 bits long
  - Intended to be globally unique
  - Really *interfaces* get allocated addresses
  - ...and 32 bits is no longer enough
- Commonly depicted in 4 parts
  - E.g., `128.243.35.39` == `0x80f32327` == `2163417895`
- So, how do addresses get allocated?

Address Management: Macro

- Internet Assigned Numbers Authority (IANA)
  - Co-ordinates number spaces
- Delegates *netblocks* to Regional Internet Registries (RIRs)
  - Africa, Asia/Pacific, North America, Latin America, EMEA
  - ...and down to National and Local registries (NIRs, LIRs)
- Approach appropriate registry for an allocation
  - Must provide suitable justification
  - Much harder to get a large allocation (== short prefix)
- Deeply manual process involving much politics
  - http://www.iana.org/numbers/ and http://www.iana.org/

Address Management: Micro

- Used to also be a very manual process
  - E.g., a big file containing (IP, Ethernet) address map
  - Also needed to maintain subnet allocations for routing protocols
- Nowadays, ARP and DHCP [RFC 2131]
  - “Can anyone give me an address?”
  - Usually provides a pile of other configuration information
- Uses IP broadcast address, 255.255.255.255

Address Shortages

- IPv4 supports only 32 bit addresses
  - Advertised as nearly 400,000 netblocks
  - IANA pool exhausted 3/Feb/2011
  - First RIR pool exhausted 15/Apr/2011 (APNIC)
- Complete exhaustion expected in 2012—2013
  - Virtualization (cloud!) is accelerating this
- IPv6 supports 128 bit addresses
  - So not a problem?
  - ...except for the routing protocols
  - ...and all the associated services needing to move

Other Issues with IPv4

- Multicast
  - Considered *unicast* (1:1) and *broadcast* (1:all)
  - IP natively supports *multicast* (M:N)
  - Need to manage *group membership*, *routing*
- Mobility (RFC3344)
  - Assign *Care-of-Address* in addition to home address
  - *Home-agent* and *foreign-agent* manage nodes
  - What about multicast? (ARP? Broadcast? ...!)
- Everything becomes more complex...

IPv6

- IPv4 is dead! Long live IPv6!
  - Well, sort of: in transition since 1998
- Primary benefit: larger address space (128b)
  - But what about NAT
- Other benefits:
  - Integrated security, back-ported as IPSec
  - Various auto-configuration mechanisms
  - Mobility support: avoid *triangular routing*
  - Jumbograms, extended options space, ...
- Costs: all the other protocols need to change too!
  - *Network layer is **much more** than just an encapsulation*

Contents

- Addressing
- Routing *vs*. Forwarding
  - Forwarding
  - Longest Prefix Match
- IP Routing

Routing *vs.* Forwarding

- Router receives an IP packet: what to do?
  - Drop or forward?
  - If forward, which interface to transmit it on?
- Making this decision is **forwarding**
  - Which interface is closest to the destination?
  - IP bases this decision (almost) solely on the destination IP address
  - Also decrement *time-to-live* (TTL) field to ensure looping packets eventually die
- Building up the information to do so is **routing**
  - Where are all the addresses at the moment?

Forwarding

- Need to map packet's destination to interface
  - Group addresses for efficiency
  - Use *prefixes* with explicit *prefix lengths*
  - E.g., 172.16/12; 10/8; 192.168/16; 128.243/16
- Routing generates (prefix, interface) mapping
  - The *forwarding table*
- Map *address* to *prefix* via *longest prefix match*
  - Means the *most specific* entry is used
  - If no match, then use *default route*; else drop

Longest Prefix Matching

192 168 10 12 **/32** – Host

1100 0000 . 1010 1000 . 0000 1010 . 0000 1100

192 168 0 0 **/16**

1100 0000 . 1010 1000 . 0000 0000 . 0000 0000

192 168 8 0 **/21**

1100 0000 . 1010 1000 . 0000 1000 . 0000 0000

192 168 10 0 **/23**

1100 0000 . 1010 1000 . 0000 1010 . 0000 0000

192 168 10 0 **/24**

1100 0000 . 1010 1000 . 0000 1010 . 0000 0000

192 168 4 0 **/24**

1100 0000 . 1010 1000 . 0000 0100 . 0000 0000

Contents

- Addressing
- Routing *vs*. Forwarding
- IP Routing
  - Static
  - Link State
  - Distance Vector
  - Comparison

What is Routing?

- Disseminating information to build the *forwarding table*
  - Need to minimise lookup latency
  - Need to handle variable and substantial update dynamics
- How to decide correct interface to use?
  - Implicit: “correct” means “most efficient”
  - ...subject to other constraints
- Why is it a problem?
  - Scalability: networks may become large
  - Dynamics: need to handle host and link failures

IP Routing

- Three basic techniques for wireline IP:
  - *Static* routing
  - vs. *Link-state* routing
  - vs. *Distance-vector* routing
- Many other techniques in general
  - Particularly in ad-hoc wireless and other networks
  - Geographical, landmark, map-based are common
  - What information is reliably available?

Static Routing

- Static routing
  - Entries in routing table independent of network state
  - Entered manually, or via DHCP
  - Common in hosts and very small networks
- For example:

```
[Linux 2.6] $ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.8.35.0       *               255.255.255.0   U     0      0        0 eth2
192.168.9.0     *               255.255.255.0   U     0      0        0 br0
default         10.8.35.1       0.0.0.0         UG    100    0        0 eth2
```

A More Complex Example

```
[Mac OSX] $ netstat -rlf inet
Routing tables

Internet:
Destination        Gateway            Flags        Refs      Use    Mtu   Netif Expire
default            tfa1-gw-v-35-35-0. UGSc           38        0   1500     en0
default            link#7             UCSI            0        0   1500     en3
127                localhost          UCS             0        0  16384     lo0
localhost          localhost          UH              5    56699  16384     lo0
128.243.35/24      link#4             UCS             5        0   1500     en0
tfa1-gw-v-35-35-0. 0:18:74:1e:bd:40   UHLWI          38       13   1500     en0    345
ppshorizon316.nott 0:25:64:9c:d9:61   UHLWI           0      665   1500     en0   1183
xpshorizoncanon.no 0:1e:8f:2e:de:8f   UHLWI           0        0   1500     en0   1022
puidhcp-035-215.is 0:23:18:c0:67:7e   UHLWI           0        0   1500     en0   1196
puidhcp-035-223.is localhost          UHS             0        0  16384     lo0
128.243.35.255     ff:ff:ff:ff:ff:ff  UHLWbI          0        8   1500     en0
169.254            link#4             UCS             1        0   1500     en0
greyjay.local      localhost          UHS             0        0  16384     lo0
169.254.255.255    link#4             UHLW            1      113   1500     en0
```

Link-State Routing

- Two common implementations
  - OSPF [RFC2328]
  - IS-IS [RFC1142, RFC1195]
- Three phases:
  - Determine link states (HELLO)
  - Broadcast link states (UPDATE)
  - Compute and install shortest paths (Dijkstra)

Determining Link States

- Use a *three-way handshake* across each link
  - "Is anyone there?"
  - "I can see you; can you see me?"
  - "I can see you"
- Runs periodically to ensure link remains alive
  - Sometimes shortcut to alert "link down"
- Result?
  - Each router knows to whom it's connected

Broadcast Link States

- Each router summarises and forwards
  - Each prefix represented as a link
- Simple? In principle, but in practice...
  - Versioning in case of delay or reordering
  - Implies number space wrapping for long uptimes
  - Need to provide reliability
  - Handle flapping, convergence, loop detection, &c
- Result?
  - Each router eventually knows to whom others are connected, approximately

Compute & Install Shortest Paths

- Each router now has a representation of the network's current state
  - < *originating-at, connected-to, metric* >
- Can run a standard shortest-path computation
  - Typically some form of Dijkstra's algorithm
  - Possibly optimize to minimize re-computation
  - Generates best *next hop* for each prefix
  - Mapped to specific interface

Dijkstra’s Algorithm (CLR, p.527)

```
# given graph G = (V,E), and
# positive weight function w,
# initialise costs d, and paths p
initialise-single-source G s =
foreach vertex v in V[G] do
    d[v] = infty
    p[v] = NIL
d[s] = 0

# given subpaths s-u and s-v,
# try s-u-v as alternative to s-v
relax u v w =
if d[v] > d[u] + w(u,v) then
    d[v] = d[u] + w(u,v)
    p[v] = u
```

```
# consider each node in turn from
# a priority queue, until all
# nodes tried
dijkstra G w s =
initialize-single-source G s
S = {}
Q = V[G]
while Q != {} do
    u = extract-min Q
    S += {u}
foreach vertex v in Adj[u] do
    relax u v w
```

Distance-Vector Routing

- Alternative is *distance-vector*
  - Routers co-operate in the computation itself
  - Should (eventually) converge
  - ...*if the network is stable!*
- Protocol
  - Broadcast lowest cost to all known destinations
  - Forward using interface on which best advert received
- Common implementations:
  - RIP v2 [RFC1723]

Bellman-Ford (CLR, p.532)

```
# given graph G = (V,E), and
# positive weight function w,
# initialise costs d, and paths p
initialise-single-source G s =
foreach vertex v in V[G] do
    d[v] = infty
    p[v] = NIL
d[s] = 0
```

```
# given subpaths s-u and s-v,
# try s-u-v as alternative to s-v
relax u v w =
if d[v] > d[u] + w(u,v) then
    d[v] = d[u] + w(u,v)
    p[v] = u
```

```
# repeatedly pass over the graph,
# relaxing each edge per node.
# distributed version has
# nodes doing this in parallel.
bellman-ford G w s =
initialize-single-source G s
for i = 1 .. |V[G]|-1 do
  foreach edge (u,v) in E[G] do
    relax u v w
```

Comparison

- Centralized vs. distributed computation
- State scales with #links *vs*. #nodes (dests)
- Network dynamics, e.g., link or router failure
  - Timer and timeout management
  - DV: *count-to-infinity*
  - LS: incremental re-computation
- Management, configuration overheads
  - Easier to see what's happening with LS
  - Need to explicitly configure link weights
  - E.g., proportional to bandwidth and latency

Contents

- Addressing
  - Address Management
  - Address Shortages
  - IPv4 Issues and IPv6
- Routing *vs*. Forwarding
  - Forwarding
  - Longest Prefix Match
- IP Routing
  - Static
  - Link State
  - Distance Vector
  - Comparison

Summary

- *Routing* is the process of building up information to enable efficient *forwarding*
- This is hard because networks are *dynamic* and can be *very large*
- The two main algorithmic approaches are
  - *link-state*
  - and *distance-vector*

Quiz (1)

1. Show, with an example, why a short IP prefix corresponds to a large address allocation.
2. Give three reasons why it has taken so long to move to IPv6 (and we're not completely there yet).
3. Treating all the prefixes on sl.15 as if they were a forwarding table, say which prefix would be used to forward packets destined for the following, and why:
   a. 192.168.9.2
   b. 192.168.11.3
   c. 192.160.4.12
   d. 192.168.10.12
4. In the simple network shown overleaf, say what the route taken by a packet from node A to node D is, and why.
5. In the same network, simulate operation of a link-state routing protocol and a distance vector routing protocol. Rerun the simulation assuming that the link B-D has failed.

Quiz (2)