Routing

G54ACC – IP and Up

Lecture 2

Recap

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

Contents

- Overview
- Link-state routing
- Distance-vector routing
- Inter-network routing
- Alternatives & Summary

Contents

- Overview
  - What is routing?
  - IP routing
- Link-state routing
- Distance-vector routing
- Inter-network routing
- Alternatives & Summary

What is Routing?

- The process of building up information to enable forwarding
  - How does a router figure out the correct port on which to forward a packet?
  - 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

- Two basic techniques for wireline IP:
  - *Link-state* routing
  - vs. *Distance-vector* routing
- Both equivalent but make different tradeoffs
- Both require some degree of coordination
- Many other techniques in general
  - Particularly in ad-hoc wireless and other networks
  - Geographical and map-based are common
  - What information is reliably available?

Ok, *three* basic techniques

- Static routing
  - Entries in routing table independent of network state
  - Entered manually, or via DHCP
  - Common in hosts and very *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
```

Contents

- Overview
- Link-state routing
  - Determining link-states
  - Propagating link-states
  - Computing shortest paths
  - Dijkstra’s Algorithm
- Distance-vector routing
- Inter-network routing
- Alternatives & Summary

Link-state Routing

- Two common implementations
  - Open Shortest Path First, OSPF, RFC2328
  - Intermediate-System Intermediate-System, IS-IS, RFC1142, RFC1195
- Three phases:
  - Determine link states (HELLO)
  - Broadcast link states (UPDATE)
  - Compute and install shortest paths

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, reordering
  - Summarization to make reliable
  - Number space wrapping for long uptimes
  - 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 recomputation
  - 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 and paths
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
```

Contents

- Overview
- Link-state routing
- Distance-vector routing
  - Bellman-Ford Algorithm
  - Comparison
- Inter-network routing
- Alternatives & Summary

Distance-vector Routing

- Alternative is distance-vector
  - Routers co-operate in the computation itself
  - Should (eventually) converge
  - ...*if network is stable!*
- Protocol
  - Broadcast lowest cost to all known destinations
  - Forward using port via which best advert heard
- Common implementations:
  - Routing Information Protocol v2, RFC1723

Bellman-Ford (CLR, p.532)

```
# given graph G = (V,E), and
# positive weight function w,
# initialise costs and paths
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/router failure
  - Timer and timeout management
  - DV: *count-to-infinity*
  - LS: incremental recomputation
- Management, configuration overheads
  - Easier to see what's happening with LS
  - Need to explicitly configure link weights
  - Example default: proportional to bandwidth and latency

Contents

- Overview
- Link-state routing
- Distance-vector routing
- Inter-network routing
  - BGP v4 – it’s the only choice!
  - Route distribution and selection
  - Operations
  - Network interconnection
- Alternatives & Summary

Inter-network Routing

- An important distinction: local vs. global
  - Interior *vs*. Exterior Gateway Protocol (IGP, EGP)
  - Why is this important? Two reasons:
- Dynamics
  - Need to scope information propagation
- Protection (Information hiding)
  - Competition: your goals are not your neighbours’

There Can Be Only One

- Border Gateway Protocol, v4 (BGPv4)
  - Essentially distance-vector with knobs on
  - Another layer: the Autonomous System (AS)
  - Purely administrative: not relevant to data-plane
- Distance is defined as the ASPATH
  - So-called *path vector*
  - But there are many other attributes to consider
- Purpose is to enable *policy* to be applied
  - No universal (trusted) metric available

BGPv4

- Protocol for exchanging prefixes with attributes
  - Uses TCP as transport (for recursion, see recursion?)
  - OPEN, UPDATE, KEEPALIVE, (NOTIFICATION)
- OPEN sets up *sessions* between *peers*
  - Perform simple capability negotiation
  - iBGP *vs*. eBGP: do src and dst ASNs differ?
- UPDATEs indicate
  - Withdrawn routes
  - (Shared) attributes
  - Advertised routes (Network Layer Reachability Information)

Tables, Tables, Tables

- BGP speaker typically has many sessions
  - 10? 20? 400?
- Logically maintains *Adj-RIB-In*, *-Out* for each
  - Advertisements received and to be sent
- Selection process generates *Loc-RIB*
  - Based on reachability, attributes (local-pref, aspath)
  - Resolved into per-port forwarding tables

Operations

- Scalability is a *vital* consideration
  - 300,000 prefixes, x2 per session
  - Bind to lo0 to avoid dropping all tables on link failure
  - *Default-free*: every router can handle every prefix
- Distribute internally via iBGP rather than IGP
  - Can control the dynamics much better
  - But a large network has 100s of routers!
- Route reflectors, AS confederations
  - Tweak route selection rules somewhat
- *Anycast* (1:1-of-N)
  - Advertise same *prefix* in many places. Carefully.

Network Interconnection

- How does this all fit together?
  - Roughly hierarchical (this is changing)
  - Tier-1/core/backbone *vs*. the rest
- Multi-homing is often desirable
  - Note that this is all *logical* though: physical diversity
- Networks interconnect via eBGP sessions
  - Points-of-Presence (Sprint, AT&T, ... customers)
  - Internet eXchanges (mutual peering)
- As ever, business and politics
  - E.g., Level3 *vs*. Cogent depeering

Contents

- Overview
- Link-state routing
- Distance-vector routing
- Inter-network routing
- Alternatives & Summary

Alternatives

- Source routing
  - But how does the host know the topology?
  - How can the network trust the host?
- Location-based routing
  - Extensive use of information embedded in environment
  - E.g., lat-long and Euclidean distance
- Map-based
  - Alternative aggregation technique for LS
- Alternative, more complex, metrics
  - Cf. QoS, later

Summary

- Routing is the process of building up information to enable efficient forwarding
- Networks are *dynamic* which makes it hard
- The two main algorithmic approaches are *link-state* and *distance-vector*
- Another operational distinction is interior *vs*. exterior
- BGP v4 is the only inter-domain routing protocol that counts