7 Scaling the Network Layer

7.1 Goal of this section

The goal of this section is for readers to understand how routing protocols work to automatically create and share routing information to allow routing across large networks. We review IP routing in this context and cover the two main paradigms for routing protocols – Distance Vector and Link State – and look at and understand the limits of these protocols.

7.2 IP Routing

In the last section we covered how IP routing works "hop by hop", with each layer 3 node (router or host) having a table of destinations and next hops. Creating these tables by hand would take a very long time – and these would need constant updating as the network changes. How do we populate the IP routing tables of our layer 3 nodes? Doing this by hand would take a long time and need constant updating. We typically do this by hand for local subnets – but want to automate everything else.

Note that our routers therefore have two separate jobs within a network:

- Route Calculation. Routers calculate routes in order to populate their routing tables to use when forwarding IP packets. This is potentially slow and complicated and can take up a lot of processing time. It also needs communication between routers, via routing protocols, to understand the network.
- Packet Forwarding. Routers receive IP packets, look up where to send them in their routing tables, and forward them on. This must be very fast, and often uses specialist hardware in order to get more throughput and to do this faster.

These two pieces of work (route calculation to build up the rules to use for forwarding, and then using those rules to forward data) are logically distinct. In this section, we're focussing on the Route Calculation work.

For an arbitrary network, we (as administrators of a network) want to achieve three goals via our routing protocols.

- When in a stable situation, we want every router in the network to calculate a routing table that is consistent with every other router in the network and together provide the best route across the network to every destination from everywhere.
- When there are topology changes in the network, we want to minimise the disruption and for the routers to be able to recover and update their routing tables quickly, and again maintain consistency of routing tables across the network.
- As administrators, we want our routers to have a minimum of setup and configuration, and to then run autonomously to handle the above two goals without further input from us.

There are two "standard" paradigms for how routers do this. We'll examine each in turn, using a specific protocol as an example.

7.3 Distance Vector Protocols

7.3.1 Concept

In distance vector protocols, the key information being passed around between routers contains the routes themselves. These are passed around with a cost associated, so that a router receiving multiple routes to the same destination can compare and use the better one.

1. Each router starts with its own initial routing table (the set of nearby hosts) that it knows how to get to. It assigns a cost to these routes (typically 0).

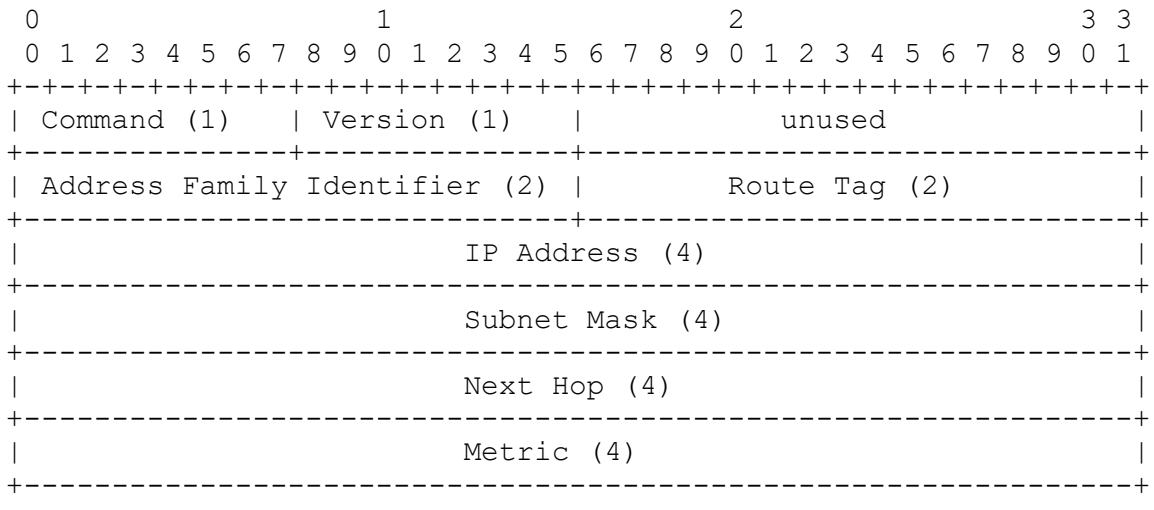
2. Each router then advertises to its neighbouring routers a set of routes, which are all of the routes that it knows about, but now with itself as the next hop.
3. When a router receives a set of routes from a neighbour, to get the true cost of the routes, it adds the cost of the hop to its neighbour to the cost of each of the routes its neighbour has advertised. (This could be a simple +1 for one hop, or different hops could have different costs.)
4. For each destination, a router compares all the routes that it has learnt about chooses the best (lowest cost one), and ignores the rest. It adds the chosen route to its routing table *and advertises it on with the new cost.*
5. If a link goes down, routers inform all their neighbours about the loss of any routes via that link. (Either explicitly indicating withdrawal of the route or by updating the route cost to be "infinity".)

7.3.2 Example Distance Vector Protocol: RIP (RFC 1723)

Distance Vector protocols can be very simple to implement. The RIP protocol (which we'll come across in the second lab practical) is a good example of a distance vector protocol being simple but effective for small networks.

Note that technically RFC 1723 covers RIPv2 (there is no v1 any more) which is for IPv4 routes. There is also an IPv6 version of the RIP protocol called Ring (RIP next generation) – which works in a very similar way.

RIP has only one protocol message, which (after the IPv4 header) looks like this:



As we unpack these fields, we see that these do indeed cover everything needed for RIP to implement the abstract above.

- **Command** (1 byte) This is set to 1 for a request "Please send me all of your routes", or 2 for a response "Here is a route"
- **Version** (1 byte) This is the version of RIP being used. Set to 2 (for RIPv2).
- **Address Family Identifier** (2 bytes) This is the type of addresses being used. Set to 2 (for IPv4.
- **Route Tag** (4 bytes). An arbitrary two-byte value associated with a route that is passed around unchanged. This allows routers to piggy-back a little bit of extra info about each route should they wish, but is not relevant to the protocol.
- **IP Address** (4 bytes) and **Subnet Mask** (4 bytes). Between them, these give the destination of the route.

- **Next Hop** (4 bytes). The next hop for the route (which is set to the IP address of the sending router.
- **Metric** (4 bytes)**.** The cost of the route.

The RIP protocol defines the cost of every hop in the network to be 1 (so all routers just add 1 to the cost of all incoming routes, requiring no additional per-hop configuration). The protocol is soft state. Routers are required to advertise all their routes to their peers every 30 seconds, and if a router does not hear of a route from a peer for 180 seconds, the route is assumed to no longer exist. RIP withdraws routes by advertising a route with cost "infinity". By default RIP uses 16 to mean infinity, and any route of cost 16 (or greater) means "there is no route to this destination".

7.3.3 Example flow

07_RIPng.pcap shows the RIPng protocol running between some RIP routers. RIPng is the IPv6 version of RIP. Look carefully at the Wireshark trace, and you'll be able to see the differences (and very strong similarities) between this and RIPv2 protocol flow and messages described above.

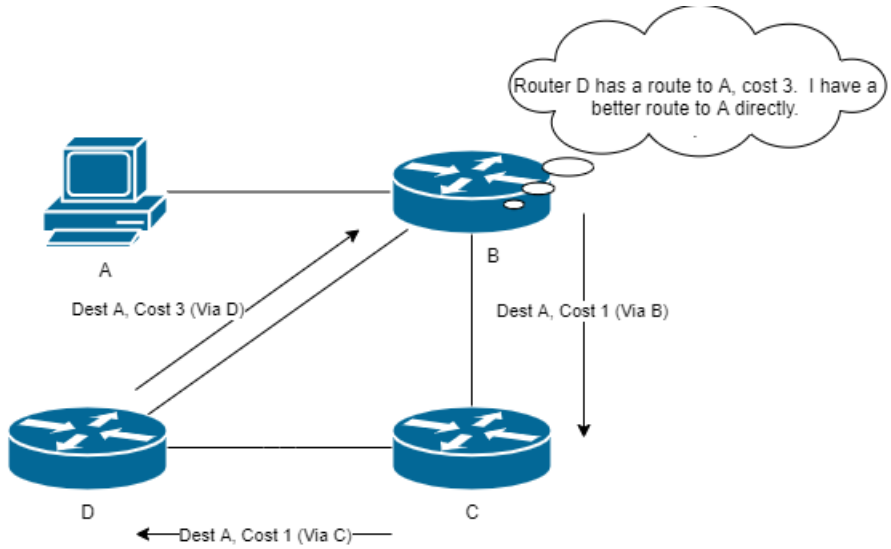
7.3.4 Pros and Cons of Distance Vector Protocols

Pros

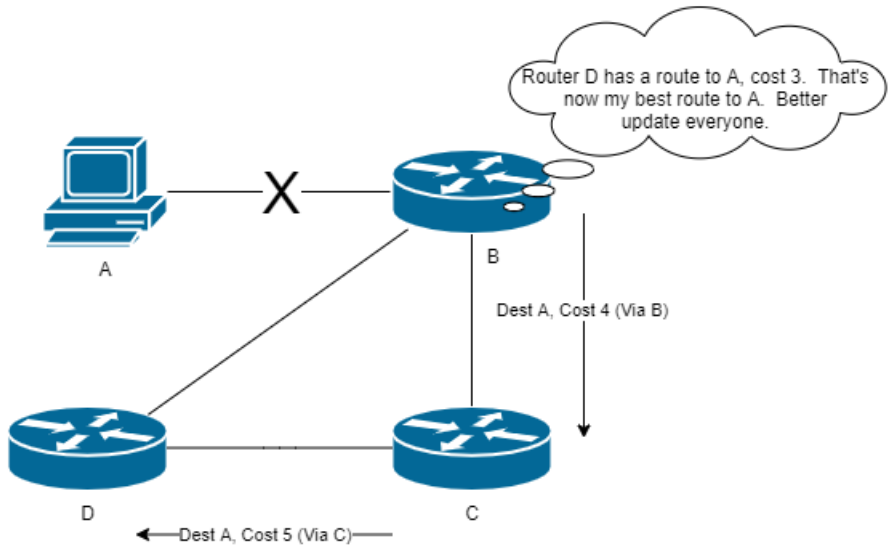
As you can see from the example above, Distance Vector protocols are simple to implement, and don't require a lot configuration to set up or a lot of processing power on the routers running them. Also, when a new or better route is added to a router, it is very quickly flooded through the network.

Cons

Distance Vector protocols do not cope well with network changes that involve routes to a destination getting worse, or indeed going away completely. The classic problem here is called counting to infinity, as shown with this simple network.



At first, everything is fine. The route to destination A is advertised around the network. However, notice that B is being advertised a second route to A, and it does not know that the second route to A travels through it or not (there could be a valid alternative route to A via D).



The link to A goes down, and B checks the set of possible routes, and spots the backup route from D. Good news, B, still has a route to A – just a bit more expensive. It updates its route. Each subsequent router round the loop updates its route.

Follow this on, and we can see what happens. Even though humans looking at the whole network, we can see there is no route to A, each router in turn tells the next one that it really does still have a route to A – just a bit more expensive than a moment ago. And they keep counting.

7.3.5 Split Horizon, Poison Reverse, Triggered Updates and Small Infinity

There are several ways to mitigate the count to infinity – basically cheap ways to attempt to stop loops like this, and indeed RIP uses these.

- Split Horizon. A router does not advertise a route to a destination back to the router that is the next hop. (This prevents two routers mutually supporting each other in a route that doesn't exist.)
- Poison Reverse. Even stronger than Split Horizon. A router advertises an infinite cost route to any router that is the next hop for any of its chosen routes.
- Triggered updates. The issue above is a timing window. If B's withdrawal of the route to A arrived at D before D sent its next regular update, then everything would converge. By immediately advertising on routing changes, you can reduce the risk here.
- Small infinity. Ultimately you can't cover every possible situation here, so distance vector protocols use a small value for infinity. (RIP uses 16.) This limits the size of network they can run in, but (especially with triggered updates) causes the network routing tables to re-converge quickly.

7.4 Link State Protocols

7.4.1 Concept

In link state protocols, the key information passed around between routers is network topology information (i.e. what the network graph actually looks like) – *not routing information.*

1. Each router advertises to its neighbours all of the network topology information that it knows of (which initially is just the interfaces, local subnet destinations and links to immediate neighbour routers that it has).
2. Routers pass on topology information that they learn to other routers too, and all the routers gradually build up the entire picture of the network.
3. Once all nodes have the complete picture of the network graph, they calculate routes to every destination based on that graph (for example using Dijkstra's algorithm) – and program their routing tables with those routes.
4. When a link changes, routers pass around the updated topology information, and recalculate their routes.

7.4.2 Example Link State Protocol: OSPF (RFC 2328)

Link state protocols are significantly more complex than distance vector protocols. We will not cover OSPF down to the level of detail that we covered RIP.

Note (again) that technically RFC 2328 covers OSPFv2 – which is the IPv4 version. OSPFv3, which handles IPv6 is covered by RFC 5340 (and we won't cover that here).

The OSPF protocol has 3 parts – neighbour discovery, information transfer, and routing calculation.

7.4.3 Neighbour discovery

OSPF broadcasts "Hello" messages out of every interface to a well-known multicast address (Allocated by IANA: 224.0.0.5). These are broadcast at layer 2 so are delivered to every IP endpoint on their local layer 2 subnets. These Hellos include the IP address of the router, and a list of IP addresses of all other routers from which they have seen a Hello. To avoid creating n-way links, one router on each subnet is elected to be the Designated Router (OSPF configuration covers exactly how routers elect the DR) and it creates OSPF neighbour connections to the other routers – and is responsible for advertising the network information for the subnet.

OSPF is soft-state. The OSPF links time out if they stop receiving hellos from the neighbour. By default hellos are sent every 30 seconds, and a link times out after 40 seconds.

7.4.4 Information Transfer

OSPF uses Link State Advertisements (LSAs) to advertise information about local subnets, links, etc. These are sent immediately on any topology change (when new routers are discovered, or lost) – but they're not sent more often than every 5 seconds, to prevent an unstable network link flapping up and down quickly causing masses of topology refreshes.

All LSAs are refreshed every 30 mins (with an additional small random timer, to flatten the sudden spike of additional network traffic every 30 mins) and LSA information is aged out after 60 minutes. (To actively withdraw an LSA, a router advertises it again with the age set to 60 minutes.)

LSAs include destinations that are reachable on a link, the OSPF links themselves, and the cost of those links (configurable within OSPF) in the graph. Because the cost of each link is configurable by an administrator, the administrator of the network can use those costs to shape the network as they chose, rather than cost necessarily being mapped to the shortest link.

7.4.5 Route calculation

Each OSPF router runs a Dijkstra calculation to calculate the shortest path through the graph to every destination, and stores those routes and next hops in its routing table. This is rerun every time the topology is updated.

As a side note: There is no *requirement* that IP routing protocols run over layer 3 themselves. For example, IS-IS (yes it's called that – it was invented ~1990) is a link state IP routing protocol that works much like OSPF, but itself runs at layer 2 (IS-IS packets are directly MAC addressed).

7.4.6 Example flow

07_OSPF.pcap shows some OSPF routers setting up a link between themselves. This shows the initial multicast neighbour discovery and the exchange of some topology information. (The routing calculations are performed independently on the nodes and are not part of the protocol flows on the network!)

7.4.7 Pros and Cons of Link State protocols

The key benefit of a link state protocol is that because all the routers have complete information about the network, they can correctly calculate routes – and don't suffer from the count-to-infinity type problems of Distance Vector protocols.

An additional advantage is that by design link state protocols pass around networking information. It is easy to piggy-back additional networking information onto these and use them as a transport method for passing around network information for use by other protocols. (We'll see some of that later in the course.)

Problems occur when routers have inconsistent (or out of date) topology information. Until the network has reconverted and routes recalculated, traffic will be black holed. By default, OSPF itself only detects network failures after 40 seconds (hellos time out) which is pretty slow. In reality, there are other protocols that check for link liveness that and inform OSPF that the link is down faster. We won't explicitly cover those in this course.

7.5 What are the limits?

Both distance vector and link state protocols are limited in scale by two key problems.

- Firstly, as the network grows, both the set of information that must be passed around per change and the number of network changes grows. In practice the "normal" limit here seems to be around 300 routers in a network, before the network becomes unusable.
- Secondly, the networks require trust. The routers must provide detailed information about the internals of the network and must work together to route packets. In practice, the limit here is almost by definition one company/one group/one organisation.

We cover how we solve these problems and scale even wider in the next section.