Interior Gateway Protocol

Patrik Valach, Stephan Günther*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: patrik.valach@tum.de, guenther@tum.de

***Abstract*—This paper explains the basics of Interior Gateway Protocols. The purpose is to understand where we use them and what topologies are considered suitable. To understand what the protocols are based on. The types and strategies that are used to make them work and their algorithmic foundation from which the basic principles of the protocols work for use in a network. To give a concrete example of an interior gateway protocol, RIP and RIPv2 are described in more detail, especially their struggles and solutions to the complications. Keeping in mind that these are old protocols and not very effective nowadays, a look into other protocols is given to guide the search for other readings.**

***Index Terms*—distactance vector, bellman ford, interior gateway protocol, routing information protocol, convergence, counting to infinity, split horizon with poison reverse, triggered update**

1. Autonomous Systems

To be able to discuss the importance and purpose of creating Interior Gateway Protocols (IGP), it is required to understand the systems it is intended to run in. An Autonomous System(AS) is defined as "a set of routers under a single technical administration, using an interior gateway protocol". [1] From the outside, an AS must look like a coherent entity that can route packets to its hosts. [1] Founding a route has to be able to be done without a direct connection, as the level 2 layer cannot reach beyond. We must look into layer three and find a way to resolve this. Anyhow, navigating complexity is reduced to mere knowledge of the next hop from the packet's point of view. [2] That is what is needed to navigate any topology that we can create. The only thing needed to figure out is how to provide this information to the node and, through that, the packet. By addressing any subnetwork of a functional AS, an arbitrary packet should be able to find its route to its destination through the AS. [3] Therefore, we are dealing with layer 3 of the ISO/OSI model since we are looking for a way to go beyond direct connections and know how to resolve IP addresses most efficiently. [4] The question remains: How do we decide which route is the best and which should we use?

2. Routing

As discussed in the previous section, routing is usually required within an AS to reach nodes outside of direct connection (reachable by layer 2). Therefore, we need to use a gateway to reach our destination. [4] Each node stores information about its routing in the "route cache". It compares the destination IP to the cache, and if it finds a match based on the subnet mask, it sends the packet to the next hop, where the process repeats. However, if no route is found, the node resolves this by sending the packet to the default gateway, hoping the next hop will know where our destination node is. Therefore, the accuracy of the routes is vital to the whole process, as mistakes or undetected changes will lead to packet loss. [4] Routing can be set up either statically or it can be done so dynamically.

2.1. Static Routing

A static route needs to be set up manually by the administrator of the AS for each node that a packet needs to pass to get to its destination. This is done by updating the routing table with the next hop address. Nevertheless, even the smallest of systems can get complicated quickly, and relying only on static routing can lead to errors as it must be manually set up. Also, it is essential to acknowledge that any change would not be responded to automatically, so if an interface is downed for any reason, we would need to update any affected routes, and as one can imagine, that would be pretty labour-intensive and unresponsive, as it first has to be manually noticed before the downing would be resolved. [4] Note that dynamic protocols can further propagate the static routes. [5] This can be useful if we want to divert traffic from a route that, for some reason, is decided to be the best by a dynamic protocol, but we do not actually want to use it.

2.2. Dynamic Routing

A dynamically routed system is a system that implements a dynamic routing protocol to update a node's routing table. The idea behind it is to give an AS a way of detecting and finding the best route to route packets through the system. Another advantage of a dynamic protocol is that it detects unresponsive nodes and updates the tables accordingly, either dropping the routes if they are no longer reachable or merely updating the next hop to another route that may be available in the system. Depending on the topology, different strategies are used to determine this. [5]

3. Routing protocol types

Over the history of computer networking, there have been two types of routing protocols distinguished. Dis-

tance vector and Link state. They are different in their strategies on how to implement the creation of routing tables, but all IGP fall into one of these two types.

3.1. Distance Vector

What makes a protocol a distance vector protocol? It is the way it finds the route; it is through short entries for each destination. Each entry contains information about which nodes are connected to one network, and with that, it knows where to send the next packet. To determine which network we should put the nodes to if we have more than one, a metric called distance is also stored and implemented in the protocols. Distance, per definition, is a metric measuring some value to the destination. The value can be whatever we choose to make it, but usually it is going to try to be a metric that will help determine the fastest route to the destination. Another example could be the cost of use. The distance metric is key to these protocols and is always shared with all nodes within the common network. [2] RIP, which is going to be a future subject in this paper, is implemented as a distance vector protocol. EIGRP is also a more modern protocol that uses this kind of structure. [6]

3.2. Link State

As this paper will not go into detail about a link state protocol, only a quick overview will be given. More information can be found in RFC 3626 [7]. A link state protocol will have each node contain knowledge of the whole topology of the network and will therefore calculate the next hop for it's packets for itself and not rely on information from other routers to determine which one is the fastest, as is the case in distance vector protocols. [7] Examples of protocols that use this type are OSPF or IS-IS. [6] [8] [9]

4. Algorithmic background for finding the shortest path

There is a mathematical basis for all protocols. The reason is simple: we want to be as effective and precise as possible. The best sources for IGP are algorithms for finding the shortest path. There are multiple, but the most useful have proven the following two: Bellman Ford and Dijkstra.

4.1. Bellman Ford

The algorithm starts by setting the starting point to a distance of 0 and the rest to infinity. (stages 1 and 2 of Figure 1) (In networking, we do not know that the nodes exist before reaching them. Therefore, as will be discussed later, there is no node until it is found.) Then the algorithm goes over all the edges |V|-1 (number of nodes - 1) times, and if any destination has been reached with a lower value than the one it was set to, then it sets a new value for the node. (stage 3 of Figure 1) This repeats over and over until finally, there are no more values to decrease. (stages 4 and 5 of Figure 1) This also works for negative connections, but in RIP, which we are going to be discussing, we only have positive distances. (stage 6 of Figure 1) [2] As we have seen or will see, this algorithm is very similar to the way distance algorithms are implemented in networking, with some practical changes due to the nature of communication. This algorithm is used in the RIP protocol, which will be described in Section 6. [2] [10] [11]

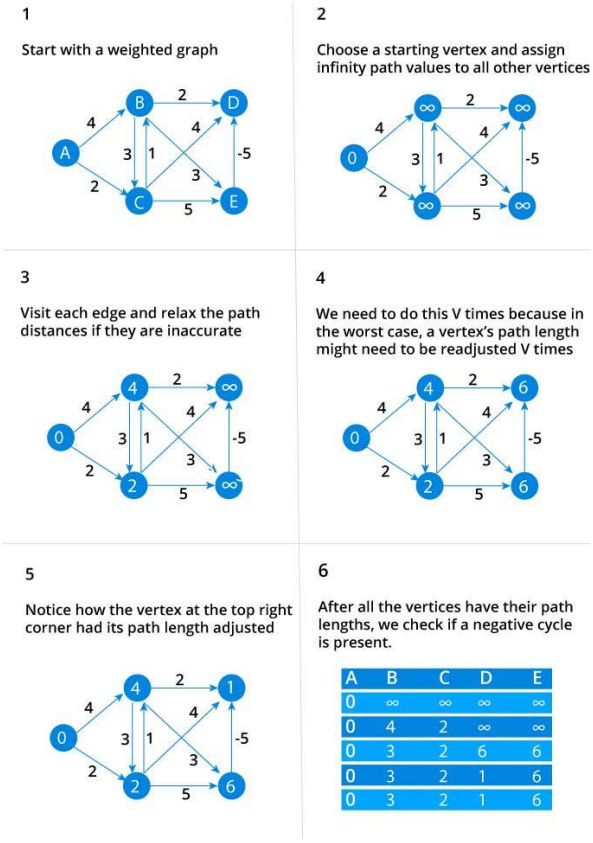

Figure 1 [12]

4.2. Dijkstra

The Dijkstra algorithm works a bit differently; it starts off at its starting node, and as before, the start node has a value of 0 and the other has a value of infinity for now. It always tries to use the connection with the lowest value from the node with the lowest value. When the algorithm figures out the network distances this way, we end up with the most efficient path. [13]

5. IGP

An IGP is responsible for taking care of routing within an AS. As it can be seen from the previous sections, different implementations are possible and needed for different use cases. This usually depends on the size and topology we have for the network, and depending on that, we choose an appropriate IGP and, if needed, even modify it to suit the specific needs that a network might require. We will be discussing RIP and RIPv2, but these protocols can be beaten in effectiveness and scale by modern protocols. A surface overview of current implementations is mentioned later (Section 7) [6], but to understand how an IGP might try to solve many issues, a deeper dive is provided into the issues and solutions that RIP encountered.

6. RIP

RIP, or Routing Information Protocol, is an IGP. To point out explicitly in this section how RIP was designed, RIP is a distance vetor protocol using the thinking of Bellman Ford's algorithm to determine the routing for all the nodes. From this, one can conclude what the RIP routing table will look like. There are four values for RIP stores for each destination. One needs to know the destination address. Then, for each destination, one needs to have a gateway to send it to to get one step closer. From that, one can deduce that information about the interface used to reach the gateway is also needed. And finally, it needs a stored metric about the distance to the destination. (More information is in Section Distance Vector.) [2] The official memos about RIP and RIPv2 are RFC1058 [2] and RFC2453 [14].

These parameters are propagated via datagram packages. They are sent through UDP-based protocols, so there is no confirmation of delivery by the other nodes. They are sent periodically by each gateway with some minor offset. However, there is also a request-response system in place. When a new node enters the network, it requests a datagram, and then the gateways on the subnetwork are responsible for sending theirs, even outside their scheduled update. The hosts and gateways propagate these datagrams depending on the relevance of the entries and the settings of RIP. Every gateway is responsible for taking into its datagram its direct connections and also taking care of the propagation of non-direct ones. Therefore, it sends out its datagrams with all the information to the other nodes. Then the receiving nodes take out relevant information. That could be either a new entry for a node it did not know about, or it could be an update of the entry it has already known of, but now it has found a more efficient route. Datagrams are sent out periodically (default: 30 sec) to ensure up-to-date routing tables. An issue arises when the network loses a node. It has to be somehow discarded from the routing tables and possibly rerouted packets through a different route. [2]

6.1. Counting to infinity

This is where so-called "counting to infinity" comes into play. A route is inaccessible when its distance value is set to infinity, which in the case of RIP is 16. Once a node is inacesible, the directly connected nodes realise this and set their metric to 16, but the gateways that are a hop away will just discard the gateway as its next hop to the inacesible node and find another, which, in the case of the implementation as discussed so far, could lead them to point at each other with the next hop. With each update cycle, the disctance value would increment by one in each of them as they slowly just kept on pointing at each other. This would stop as soon as either an actual route is found with the value they counted up to so far or if they reach infinity (16). At infinity, they would just drop the route since it would be regarded as inaccessible. This counting up can take a long time, up to a few minutes, and in computing, a long time is a no-go. This is the reason why distance in RIP is limited to 16 and why only medium-sized networks can use it. Because any higher number would cause more headaches than joy. It is still not ideal, so what does RIP do to take care of the long time it takes to converge? [2]

6.2. Convergence Tactics

Time to converge is the time it takes for all the nodes in the network to have up-to-date information about the routing. A state every protocol wants to reach as fast as possible. [15]

6.2.1. Split horizon with poison reverse. To prevent this long back and forth between gateways, a solution was devised for RIP. Imagine if the gateways omitted the destinations when sending updates to the gateway it had learned the destination from. That way, when a change occurs, gateways won't be pointing at each other anymore, thinking the other one knows an actual path. Poison reverse does this a step further and sends back the destination, but with a value of 16. Another advantage of not sending routes back where they came from is that within a subnetwork, we can send broadcast messages to everyone, and we do not need to tailor each individual datagram for each connected node. Only a single datagram per interface will need to be determined. Since only the source of the path will be advertising the path beyond the hop, nodes within the subnetwork can each get a datagram from that gateway and update their respective routing tables. [2]

But why is poison reverse needed? Why do we advertise the destination with the value of 16? Because in special cases, if two gateways were to point at each other, it would not solve the issue to just not advertise the destination, as it would still think the other gateway has a way. It would still take time to time out the connection (default 180s). So to speed it up a bit more, advertising the destinations with the distance 16 solves the issue as the gateways will realise this pointing at the moment of the first update (30s default). The issue with poison reverse is that even though it speeds up convergence as it is designed to, it comes with a cost. That is, every node needs to keep the value 16 for all the nodes it has learned about and not provide the route to it. This comes with big datagrams and a lot of bandwidth used just to keep the network up to date for what some may consider marginal improvements (keep in mind the limitations of the year 1988). Also, this only solves the issue of two gateways pointing at each other. If a loop of more than two gateways were to be created, this would not solve the issue, and we are back to counting to infinity. [2]

6.2.2. Triggered update. The way triggered updates work is basically in the name itself. As soon as a gateway detects a change, it immediately sends out an update message about it. Then the receivers propagate this change until it has reached the whole network. If a node has information about the route from the gateway it got the update from, it has to accept this change. It does not matter if the number is higher than the one stored; now the lowest first determination does not work; it has to store the new value for the destination. This is a very easy way to solve counting to infinity in theory, but once again, there are drawbacks. If these updates were propagated internally, it would work just fine, but there is a delay in sending

these messages and therefore cracks in the idea of the implementation. If, in between the triggered update, a regular update is sent, it might create a false entry in some of the gateways and keep the node available. This would once again be resolved with couting to infinity eventually. [2]

6.3. RIPv2 upgrades

RIPv2 has an additional entry for each routing entry, and that is the network submask to make the entry clearly understandable. Thus, the RIPv2 message structure is also a bit different. Instead of broadcasting, multicasting is used, which reduces the number of unintentioned receivers. In case there is RIP present on the network, nodes switch to communicate with it instead of RIPv2 to prevent confusion. Whenever possible, we try to authenticate RIPv2 messages, a feature the RIP did not have. Saving of resources on talking with subnetworks with another IGP in place. The other network needs merely to have one node connecting. In RIP, the other subnetwork would need to be incorporated into using RIP for them to reach the RIP-controlled part. [14]

7. Current IGP implementations

As mentioned before, there are other IGP protocols that one can use. They all vary in efficiency, hardware support, and use cases. If one wants to pick the best protocol to implement on a network, it is crucial to know the options and choose the best-suited one. These are some other IGP protocols that one can use instead of RIP:

7.1. EIGRP

EIGRP stands for Enhanced Interior Gateway Routing Protocol. It is a distance vector protocol. It is superior in terms of scalability to RIP and convergence time. The issue with EIGRP is that the hardware support is limited. Cisco has kept the protocol proprietary, and therefore many devices do not support it. Therefore, it is only usable within certain networks; one needs to be sure every device can support it, and one needs to be careful when implementing it. [6]

7.2. OSPF

OSPF stands for Open Shortest Path First. It is a link state protocol (section number). In terms of convergence, it is once again faster than RIP, and it supports bigger topologies. On top of that, it is an open standard protocol, and therefore it is supported by most routers out there. Therefore, it is one of the most used protocols in today's day and age. [6] [9]

8. Conclusion

In conclusion, autonomous systems are in need of a suitable interior gateway protocol. Within any AS, up-to-date or also referred to as converged routing tables for all the nodes are crucial for an AS to work. Protocols are devised based on certain tactics, and therefore we divide them into two types: Distance vector and Link state. This depends on the type of data each node has to hold. The protocols are also based on algorithmic principles, namely Bellman Ford or Dijkstra. From which the core logic of finding the shortest route comes. To understand the concept and issues of interior gateway protocol, understanding RIP is the easiest place to start. Nevertheless, it has its drawbacks, but it is simple and, for that time, in many ways, the best solution to dynamic routing. Nowadays, we have more effective IGPs like OSPF that take over many AS.

References

[1] J. Hawkinson, "Guidelines for creation, selection, and registration of an Autonomous System (AS)," https://www.rfc-editor.org/rfc/rfc1930.

[2] C. Hedrick, "Routing Information Protocol," https://www.rfc-editor.org/rfc/rfc1058.

[3] Y. Rekhterp, "A Border Gateway Protocol 4 (BGP-4)," https://www.rfc-editor.org/rfc/rfc1654.

[4] R. Braden, "Requirements for Internet Hosts – Communication Layers," https://www.rfc-editor.org/rfc/rfc1122.

[5] F. Baker, "Requirements for IP Version 4 Routers," https://www.rfc-editor.org/rfc/rfc1812.

[6] A. Froehlich, "Article: Comparing Dynamic Routing Protocols," https://www.networkcomputing.com/data-centers/comparing-dynamic-routing-protocols.

[7] P. J. T. Clausen, "Optimized Link State Routing Protocol (OLSR)," https://www.rfc-editor.org/rfc/rfc3626.

[8] R. Callon, "Use of OSI IS-IS for Routing in TCP/IP and Dual Environments," https://www.rfc-editor.org/rfc/rfc1195.

[9] J. Moy, "OSPF Version 2," https://www.rfc-editor.org/rfc/rfc2328.

[10] R. Bellman, ""On a routing problem"."

[11] L. Ford, ""Network Flow Theory"."

[12] dvmannan, https://dev.to/dvmannan/dijkstra-s-algorithm-bellman-ford-algorithm-3hfk.

[13] E. Dijkstra, ""A note on two problems in connexion with graphs"."

[14] G. Malkin, "RIP Version 2," https://www.rfc-editor.org/rfc/rfc2453.

[15] S. Poretsky, "Terminology for Benchmarking Link-State IGP Data-Plane Route Convergence," https://www.rfc-editor.org/rfc/rfc6412.