Comparison of Dynamic Routing Protocols: RIPv2 vs. OSPFv2

IB Extended Essay

Word Count: 3856

February 27, 2024

Contents

1 Introduction

Dynamic routing is a corner piece of the Internet today, providing essential connectivity between Layer 3 networks and enabling the existence of the modern web. However, the development of dynamic routing has been strenuous and complicated like all long-term development projects, with multiple iterations and different proposals to solve long-standing problems regarding redundancy, efficiency, bandwidth, and speed. Numerous different attempts have been made to solve the problem of routing packets between destinations, and this paper is an attempt to quantify the differences between the solutions. It will compare different dynamic routing protocols to give an understanding of their respective utility. This paper will focus on the RIP (Routing Information Protocol) and OSPF (Open Shortest Path First) routing protocols due to their similarities in function and application, answering the research question: what are the best use cases for different dynamic routing protocols?

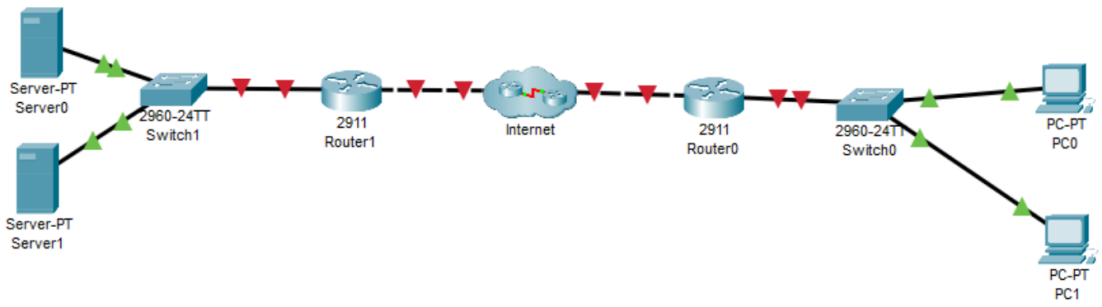

Figure 1: A standard inter-LAN network topology. LAN boundaries are set by routers. In this case, one can see two LANs defined, connected by the Internet. The first LAN includes Server0, Server1, Switch1, and Router1, while the second LAN includes Router0, Switch0, PC0, and PC1. The router routes messages out of the LAN if the destination IP address is not of the local LAN and resides in the other LAN or the Internet itself. This image was created with Cisco Packet Tracer[2].

2 Background

2.1 Brief History of the Internet

The Internet was originally known as ARPANET and came about as a research project meant for military communications that only later evolved into a publicly accessible network of networks. One of the original design ideas behind ARPANET was that of decentralization leading to redundancy[11]. The notion was that if a single node or computer in the network went down, other computers should still generally be able to communicate with one another. This central idea of connecting individual nodes or computers without a defined center or hierarchy continues in the modern existence of the Internet. While this decentralized model means that Internet data transmission is generally redundant with no central point of failure, it also necessitates the existence of many devices whose sole purpose is to decide what paths

information should go, as now there can be multiple paths that get from any point A to any B[11].

These special computers are known as network devices, and include routers and switches. They serve as couriers, forwarding information to a destination, as opposed to acting like a client, requesting information, or a server, offering a service. Note that network devices like routers can also fulfill server or client functions like, for example, a VPN service.

Companies and early pioneers sought to find ways to connect users with their requested services such as with their mail or with their files, hosted on servers. To do this, devices on the Internet had to be located, and the solution was with IP (Internet Protocol) addresses[11]. However, the path between devices still had to be decided, and this was done generally with routers. While routers had to decide where to send packets within their network, they also had to route outside their LAN (local area network), crossing what is known as the logical LAN boundary.

For different computers to communicate with one another, protocols or agreed-upon rules had to be developed to standardize communications. The different technologies required for transmitting information necessitated a protocol framework, a model for how different protocols serve different purposes. Any protocol of a particular layer in this framework can work with any protocol in another layer in the framework. Essentially, each layer can be customized without significantly modifying other layers.

The OSI model of the Internet was introduced in 1983 as a way of conceptually understanding these different protocol layers[11]. While the TCP/IP protocol stack is a more accurate representation of what technologies and protocols are involved, the OSI model remains a teaching model. The OSI model is composed of 7 layers as follows:

1. Physical Layer: The Physical Layer includes the hardware and binary data of all devices.

2. Data Link Layer: This layer refers to the physical addressing of transmission devices, checks for corruption, and the Ethernet standard. This comprises the traditional role of the switch and associated protocols like STP (Spanning-Tree Protocol). Switches work at this layer.

3. Network Layer: This layer locates the destination and source of a message through IP addressing and routing. NAT (Network Address Translation) is an example of a protocol that works at this layer. Routing works at this layer.

4. Transport Layer: This layer uses TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) to ensure reliable data transmission by splitting up data into segments.

5. Session Layer: This layer is responsible for ensuring the connection stays up for the duration of information transmission. Your web browser creates a session whenever you connect with a website using HTTPS (HyperText Transfer Protocol).

6. Presentation Layer: This layer is responsible for converting between different visual display systems so that the Application Layer can use the data.

7. Application Layer: This layer gives networking options for programs on an end device to communicate with devices besides itself.

This model is important to understand because all Internet communications are encapsulated and de-encapsulated through the different protocol layers[11]. In essence, the data is being repackaged to be transmitted physically across some form of cable or through radio waves and later unpackaged to be understood as coherent information. A common analogy is the act of putting a letter into an envelope for easy delivery by a postman. On the envelope, the recipient address is written, as well as a stamp to verify the letter (to some extent). The receiver, after receiving the letter, needs to open the envelope to read the contents of the letter.

For this paper, it is also important to know that routers handle Layer 3 packets, which contain IP address information. Every time a packet needs to cross from one network device to another, it must be encapsulated into a Layer 2 frame using the Ethernet standard, to be physically transmitted across the network cables to the receiving device[11]. The receiving device de-encapsulates the frame to the original Layer 3 packet to read Layer 3 information like the source and destination IP addresses. Encapsulation is generally done by adding headers, which are simply more information that describes specific information needed for that layer of encapsulation. Headers might include IP addresses that define the source and destination of a packet or extra sequences to check if the information is corrupted at the other end. De-encapsulation is the exact opposite; header information is removed. In other words, a router is a specific kind of network device that operates on Layer 3 of the OSI (Open Systems Interconnection) model, and performs both encapsulation and de-encapsulation.

2.2 Overview of Routing

Routing is the process conducted by routers whenever the source and destination network address of a message (or packet) do not match. When this is the case, devices on a network have to send the packet along to other devices, like a complicated game of hot potato. Every router has a routing table that contains the addresses of networks the router knows how to get to, and the next-hop address. The next hop is the next device a packet needs to go to to reach a destination. However, the device does not suddenly know what the best next-hop for a particular destination is. In fact, in many instances, it does not know.

When a router first boots up and you connect it to a network, the router has an empty routing table. Before sending out any of what is called discovery packets, it knows nothing of its surroundings except for whatever devices are directly connected to it. The solution to this problem of a lack of information about the network topology is to send discovery packets, which solicit responses from routers configured with the same routing protocol. These responses contain information about the topology but exactly what information is shared differs between different routing protocols, and will be elaborated upon later[12].

At first, routing tables were configured exclusively by network administrators, as network topologies remained small. However, soon many grew tired of configuring routing tables manually (static routing), especially for large networks, developing and transitioning to using algorithms that automatically find routes (dynamic routing). These protocols serve to automatically update routing tables through the aforementioned discovery packets

and thus allow the router to make routing decisions without human intervention. As engineers continually improved and refined these algorithms, internet connectivity was further enhanced and data transmission speed increased.

Dynamic routing protocol algorithms involve sending out discovery packets to other devices in a network to share information about connected hosts, filling in collective gaps about the topology of the network. Thus, all the routers in a network configured with the same dynamic routing protocol will receive the same information and be updated. That information, at minimum, includes what the nearest neighbors of a router are, and what routes a router should use for particular destination networks[11].

It is pedantic but important to note that while dynamic routing protocols provide the information for routing (finding the optimal routes for a certain source and destination), they do not do the actual process of routing. That is a task still reserved for and built-in to the actual router device.

2.3 Routing Behavior on a Router

When a router receives a message, it de-encapsulates or strips off the outer Data Link Layer header to see the destination IP (Internet Protocol) address of the packet. If the network portion of the destination IP address of the packet matches with an entry in the router's routing table, the router re-encapsulates the frame and sends it out to the associated exit interface to the next hop. If the network portion of the destination IP address does not match any routing table entries, the router will drop the packet. The frame will now have the MAC (Media Access Control) address of the directly connected destination port as the new destination MAC address. The router will have procured the MAC address from the ARP (Address Resolution Protocol) table or through making an ARP request. When the router re-encapsulates the packet into a frame, it adds an FCS (frame check sequence) that ensures data integrity i.e. that the data is not changed during transmission[11]. Then, the frame is forwarded out of the correct port, onto the next router or switch or end device to receive the message.

2.4 Justification for Selecting RIP and OSPF

RIP and OSPF are examples of interior gateway protocols[6], or dynamic routing protocols that run completely within a single autonomous system. An autonomous system is defined as a group of devices that share the same system for routing[13]. Many autonomous systems make up the Internet, each being in charge of a certain subset of all the allowed IP addresses and administered by a single entity like an ISP (Internet Service Provider). While RIP, OSPF, and other interior gateway routing protocols operate within an autonomous system, autonomous systems themselves are connected with other routing protocols like BGP (Border Gateway Protocol) or EGP (Exterior Gateway Protocol), which are known as inter-AS routing protocols.

RIP and OSPF thus play similar roles within all the possible dynamic routing protocols that can be used by network administrators, being able to be used in similar use cases. Both went through design iterations leading to their respective version 2's and extending support for IPv6 through RIPng (Routing Information Protocol Next Generation) and OSPFv3

(Open Shortest Path First version 3) respectively. However, they are fairly different in implementation, utilizing different algorithms and focusing on different design features dictated by the time frame when they were developed, making for a meaningful comparison[5][9].

2.5 Metrics Used in this Paper

The purpose of dynamic routing protocols is to allow for quick delivery of packets efficiently. While efficiency can be defined through different metrics, network designers and RFC writers have, over time, decided to optimize for different metrics depending on shifting needs and the realization of new technologies. The choice of what metric to optimize influences the use case of each routing protocol. However, all routing protocols optimize the following metrics:

1. Time Delay - Time delay refers to how long a message takes to reach a destination. In the case presented in this paper, it refers to whether the routing protocol had chosen the optimal route.

2. Time to Convergence - Convergence is the state where all routers in the same autonomous system share the same topological information. This is important because only after achieving convergence can all routers in a particular network share the same unambiguous routes that are not loops. This is achieved through the sending of routing protocol-specific broadcast packets that notify all recipient routers of the sending router's routing table. Time to convergence is thus how fast this process takes, and equally important, how often this process occurs. Every topology change in a network breaks the state of convergence and requires an update sent to neighboring routers. A topology change could include the addition of a new node (router or end device) or the breaking or addition of a new connection[12].

Using these metrics and other more qualitative and implementation-specific details, we can then perform a proper comparison.

3 RIP (Routing Information Protocol)

RIP is an example of a distance vector protocol, referring to it using a proxy for the distance to a network to find the optimal routes. RIP in particular uses the Bellman-Ford algorithm (sometimes known as the Bellman-Ford-Moore algorithm). It is also based on hop count, which is the number of devices between the source and the destination. Each device in between source and destination is known as a hop. At the time of its introduction, this was a relatively effective way of routing packets[5].

However, nowadays, this is somewhat rudimentary as the number of hops in a route is not directly how fast that route is. Using only hop count does not keep track of bandwidth, among other factors. Bandwidth is the maximum amount of data that can be transmitted through a connection. In comparison, throughput is the actual amount of data that is transferred. Connections with higher bandwidth can transfer more information at a time, leading to faster speeds. A route with a higher hop count but greater bandwidth for every connection may be faster than one with a lower hop count.

RIP is also somewhat limited by its implementation. Its use is strictly confined to small networks as there is a hard limit on hop count to 15[5]. Any destination that has to be reached that is at or beyond 16 hops is unreachable based on a routing table defined by RIP. Such a route would have an infinite hop count in the routing table. Routers that receive a packet with such a destination usually drop the packet. In most cases, the original sender will have sent multiple copies of their packet, which should find their destination through other paths.

In addition, RIP requires the broadcast of network update messages containing all known routes every 30 seconds[5]. That means every 30 seconds in a topology using RIP, there will be high bandwidth usage as all routers exchange the contents of their routing table. While this ensures that topology changes are detected and fixed quickly, it is rather rudimentary. When compared to OSPF, which only updates if there is a topology change, this is extremely inefficient and negatively impacts total network bandwidth[8].

3.1 Bellman-Ford Algorithm

RIP uses an algorithm known as Bellman-Ford, which is a single-source shortest-path algorithm, used to find locally optimal paths to a destination. The algorithm works as follows:

1. Assign all paths an over-weighted cost.
2. Choose a starting vertex/node and assign infinity path values to all other vertices.
3. Visit the edge and decrease (relax) the path value if inaccurate.
4. Continue until all paths have been updated. Check if a negative cycle is present.

In particular, one benefit of the Bellman-Ford algorithm for other applications is its ability to recognize and account for negative weight edges. Negative weight/cost edges suggest that a node should not be traversed as the destination cannot be reached through that edge. In reality, negative weight edges do not exist in routing and real-life network topologies.

One aspect of Bellman-Ford that has not been mentioned is that the functionality of checking for negative cost paths becomes an even larger problem when there is a large number of nodes in the topology. The check for negative cost paths must be done for every path, increasing the amount of computation required to reach convergence. This will be further analyzed in the algorithm comparison section of this paper.

4 OSPF (Open Shortest Path First)

OSPF is a link-state protocol, meaning that routers using this protocol exchange topology information with their nearest neighbor. Link-state protocols emphasize rapid convergence, a state where all the routers in a network are aware of everything that is in their network and all the routes necessary for them. OSPF, in particular, uses Dijkstra's Algorithm with a metric known as cost for its routes. Cost is used to measure which routes are more preferred over others if multiple routes are possible.

```
Gateway of last resort is 160.2.3.5 to network 0.0.0.0

     160.2.0.0/16 is variably subnetted, 6 subnets, 2 masks
C       160.2.3.0/30 is directly connected, Serial0/1/1
L       160.2.3.2/32 is directly connected, Serial0/1/1
C       160.2.3.4/30 is directly connected, Serial0/1/0
L       160.2.3.6/32 is directly connected, Serial0/1/0
O       160.2.3.8/30 [110/128] via 160.2.3.5, 00:00:32, Serial0/1/0
O       160.2.3.12/30 [110/128] via 160.2.3.5, 00:00:32, Serial0/1/0
     172.18.0.0/25 is subnetted, 1 subnets
O       172.18.9.0/25 [110/129] via 160.2.3.5, 00:00:32, Serial0/1/0
O*E2 0.0.0.0/0 [110/1] via 160.2.3.5, 00:00:32, Serial0/1/0
```

Figure 2: The numbers boxed in red are the costs for specific paths to networks (indicated by numbers like 160.2.3.4/30, etc). As seen, none of the numbers are negative. In practice, no network path has a negative cost. This is a snapshot taken from a topology created in Cisco Packet Tracer[2] by this paper's author.

A router configured with OSPF sends out an LSA (link state advertisement) to other routers every 30 seconds, which is generally only a synchronization check. It only sends a network update message if there is a known network topology change[10]. In practice, this means that OSPF sends out fewer messages and consumes less of the network bandwidth than RIP, conserving the space for more user traffic. In addition, when OSPF sends a network update message, it only sends the delta of the network, or whatever part of the network that has changed, saving valuable bandwidth.

4.1 Dijkstra's Algorithm

A short explanation of Dijkstra's Algorithm is warranted. It is used for finding the shortest paths between nodes in a graph network. It does this with an algorithm that grows like a circular wavefront across all the nodes connected to the starting node, essentially a breadth-first search. In Dijkstra's Algorithm, this refers to visiting all closest nodes first and making no leap of exploration. This makes OSPF relatively slow in certain topologies, but extremely certain of the optimal path once the network is fully explored[9].

Dijkstra's Algorithm works as follows:

1. First, all the nodes or network devices are marked as unvisited. Each node has an infinite cost.
2. Each path is proceeded down and relabeled with the cost at every intersection (a node connected with greater than two other nodes). Each visited node is marked as visited.
3. Step 2 is continued until the destination is found.
4. The way back to the source is traced by following each node's parents to the starting node.

From step 4, the router would add the calculated cost and associated route to its routing table for later use. By doing this, it can populate its routing table automatically by repeating this process.

5 Algorithm Comparison

Since RIP is based on Bellman-Ford and OSPf is based on Dijkstra's Algorithm, comparisons can be made between the algorithms to compare the protocols overall. Testing the algorithms in practice, other authors have found that while Bellman-Ford performs better when there are few nodes (smaller network radius), Dijkstra's Algorithm excels at staying relatively fast even as the number of nodes scales.

Table 1: Algorithm Simulation Results[1]

No of nodes	Dijkstra (in ms)	Bellman-Ford (in ms)
5	1351	513
10	3724	756
50	2072	9577

To understand this further, for this paper on dynamic routing protocol comparison, a program was written in Python to simulate the algorithms for even larger numbers of nodes, similar to an actual network topology[3] Dijkstra's and Bellman-Ford algorithms were run on randomly generated topologies, using an increasing number of nodes from 100 to 10000 nodes. The time spent to reach convergence (the algorithm finishing) was recorded using the Python *time* module. Graphing the data suggested some patterns that this paper has previously explained and thus one would come to expect.

It can be seen from the graph that Dijkstra's Algorithm generally outcompetes Bellman-Ford starting at topologies of at least 100 nodes, consistently reaching convergence faster. In addition, Dijkstra's Algorithm's time to convergence grows slower than Bellman-Ford's, meaning that the difference in times to convergence grows as network sizes grow bigger. This showcases why Dijkstra's Algorithm was chosen for OSPF, a link-state routing protocol meant for larger networks.

5.1 Big O Values

While a deeper dive into the precise mathematical properties of either algorithm is beyond the scope of this paper, one can consider the time complexity or Big O value to explain the results found previously. The time complexity of the Bellman-Ford Algorithm is $O(|V| \cdot |E|)$, where $|E|$ signifies the number of edges (connections) and $|V|$ is the number of nodes in the topology. The time complexity of Dijkstra's Algorithm is $O(|E| + |V| \log |V|)$[1].

The number of edges should increase with the number of nodes in a topology. For this reasoning, we'll make the naive assumption that the number of edges is equal to the number of nodes in a topology. In reality, it is likely that the number of edges grows faster than the number of nodes in a topology.

Using this approximation $|V| \approx |E|$, the two Big O equations can be simplified to $O(|V| \cdot |V|)$ for the Bellman-Ford Algorithm and $O(|V| + |V| \log |V|)$ for Dijkstra's Algorithm. This suggests that the Big O of Bellman-Ford is $O(|V|^2)$, while for Dijkstra it is $O(|V|+|V| \log |V|)$.

The constant in the Dijkstra Big O is irrelevant to the total Big O as it is relatively small to $|V| \log |V|$, so the whole expression can be simplified to $|V| \log |V|$. Knowing graphically

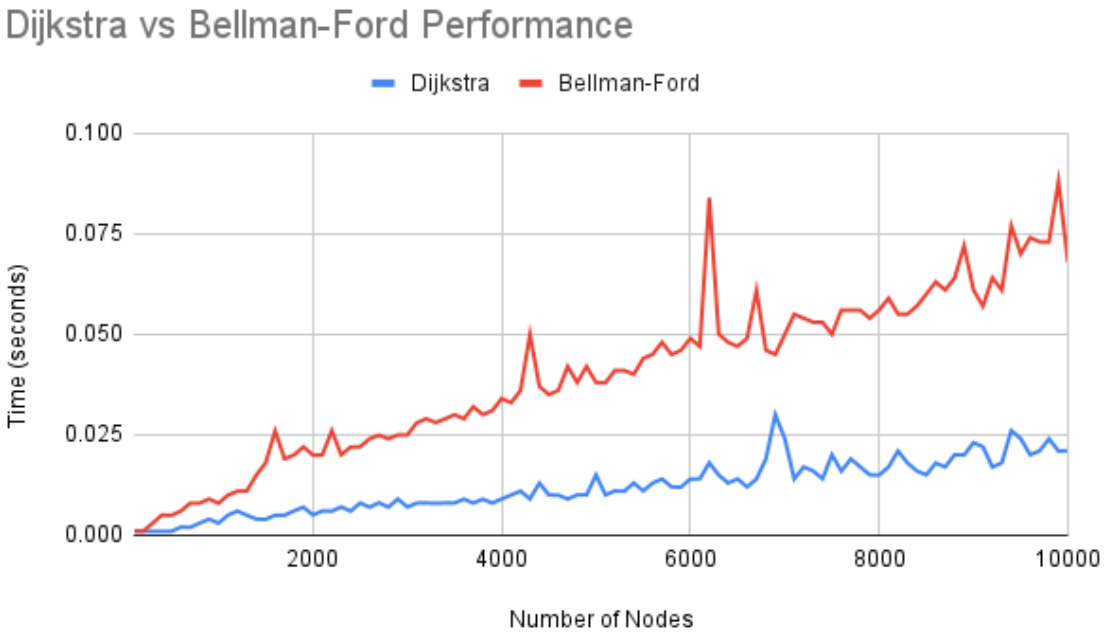

Figure 3: This graph depicts the time spent to reach convergence for both Dijkstra and Bellman-Ford algorithms on the same randomly generated topology. Topology sizes from 100 to 10000 nodes were tested. The spikes are where the randomly generated topology happened to contain paths of infinite cost (nodes were not connected) or were generated in a way to contain relatively more suboptimal paths.

that a square relation (x^2) grows faster than a $x \log x$ relation, one can thus conclude that the time complexity of the Bellman-Ford Algorithm (with the square relation) grows faster than that of Dijkstra's Algorithm, verifying the results in the graph.

6 Real-Life Comparison

Based on performance tests using Opnet Simulator, another paper found that OSPF generally selects better routes in actual testing[4]. This can be explained from OSPF being a link-state protocol. The use of hop count as the metric in RIP limits its ability to accurately optimize routing in a network. While the number of devices between one node to another may make sense as a good measure of data transfer speed, real network traffic is dependent on additional factors like bandwidth, which is not necessarily related to hop count and is instead related to the available technologies used at each point-to-point connection[7].

In addition, it was found that OSPF responds faster to network changes and has faster convergence. OSPF can calculate routes beyond a hop count of 15. In addition, as all routers in an OSPF network know the entire topology, each router can calculate its routes irrespective of what surrounding routers are down, an ability that RIP-configured routers do not have[11].

In actual applications, RIP is a simpler protocol to use and troubleshoot for network

administrators[11]. RIP is also slightly older than OSPF, meaning that it can be used for certain legacy networks that network administrators are reluctant to modify[5][9].

To summarize the conclusions in a table:

RIP	**OSPF**
RIP uses UDP	OSPF uses TCP
RIP uses the Bellman-Ford Algorithm	OSPF uses the Dijkstra Algorithm
RIP is a distance-vector protocol	OSPF is a link-state protocol
RIP can only be used for small network radii (less than 15 in hop count)	OSPF can be for larger network radii (can be up to around 300 in enterprise use cases[3])
RIP consumes more bandwidth	OSPF consumes less bandwidth generally

7 Conclusions

From the data, one can conclude that in most situations, OSPF is more effective in delivering traffic efficiently and effectively than RIP. OSPF has generally lower delay times and faster times to convergence through the use of a faster algorithm that is more efficient for the use case of routing. While individual routers need to retain more routing information, the bandwidth of the entire network benefits from the link-state protocol design, as there are no frequent network updates like the full network topology update sent every 30 seconds with RIP.

While most of the results and data shown in the algorithm comparison and real-life application comparison sections show that OSPF outperforms RIP in most metrics, RIP is still a good choice for network administrators working with small networks that want to use a simpler protocol. Its use of Bellman-Ford also suggests that future implementations of RIP could take advantage of its ability to recognize and make decisions off of negative edge weights, something the Dijkstra Algorithm of OSPF cannot do.

Works Cited

[1] Samah W.G. AbuSalim et al. "Comparative Analysis between Dijkstra and Bellman-Ford Algorithms in Shortest Path Optimization". In: *2020 International Conference on Technology, Engineering and Sciences (ICTES)* (2020).

[2] *Cisco Packet Tracer*. Available at `https://www.netacad.com/courses/packet-tracer` (2024/02/27).

[3] Pearson Education. *Designing Cisco Network Service Architectures (ARCH): Developing an Optimum Design for Layer 3 (CCDP)*. Available at `https://www.ciscopress.com/articles/article.asp?p=1763921&seqNum=6` (2024/02/27).

[4] Ioan Fiţigău and Gavril Toderean. "Network Performance Evaluation for RIP, OSPF and Eigrp Routing Protocols". In: *2013 International Conference on Electronics, Computers and Artificial Intelligence (ECAI)* (2013).

[5] C.L. Hedrick. *Routing Information Protocol*. Available at `https://www.rfc-editor.org/info/rfc1058` (2024/02/27).

[6] *Interior gateway protocols*. Available at `https://www.ibm.com/docs/en/zos/2.1.0?topic=terminology-interior-gateway-protocols` (2024/02/27).

[7] Megha Jayakumar, N Ramya Shanthi Rekha, and B. Bharathi. "A comparative study on RIP and OSPF protocols". In: *2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)* (2015).

[8] G. Malkin. *RIP Version 2*. Available at `https://www.rfc-editor.org/info/rfc2453` (2024/02/27).

[9] J. Moy. *OSPF specification*. Available at `https://www.rfc-editor.org/info/rfc1131` (2024/02/27).

[10] J. Moy. *OSPF Version 2*. Available at `https://www.rfc-editor.org/info/rfc1583` (2024/02/27).

[11] Wendell Odom. *Official Cert Guide: CCENT/CCNA ICND1 100-105*. 3rd ed. Pearson Education, 2006.

[12] Wendell Odom. *Official Cert Guide: CCNA Routing and Switching ICND2 200-105*. 2nd ed. Pearson Education, 2017.

[13] *What is an autonomous system?* Available at `https://www.cloudflare.com/learning/network-layer/what-is-an-autonomous-system/` (2024/02/27).