**Chapter 11: Unicast Routing Protocols**

Table of Contents

[11.2 Intra and Inter-Domain Routing 5](#_Toc86833970)

[Intradomain Routing Protocols 6](#_Toc86833971)

[11.3 Distance Vector Routing 8](#_Toc86833972)

[Two-Node Instability 10](#_Toc86833973)

[Three Node Instability 12](#_Toc86833974)

[11.4 Routing Information Protocol 14](#_Toc86833975)

[RIP Message Format 14](#_Toc86833976)

[Requests and Responses 15](#_Toc86833977)

[Timers 16](#_Toc86833978)

[RIP Version 2 16](#_Toc86833979)

[Encapsulation 18](#_Toc86833980)

[11.5 Link State Routing 19](#_Toc86833981)

[Link State Packets 20](#_Toc86833982)

[Flooding 20](#_Toc86833983)

[Dijkstra’s Algorithm 21](#_Toc86833984)

[11.6 Open Shortest Path First 23](#_Toc86833985)

[Types of Links 24](#_Toc86833986)

[11.7 Path Vector Routing 27](#_Toc86833987)

[Loop Prevention 28](#_Toc86833988)

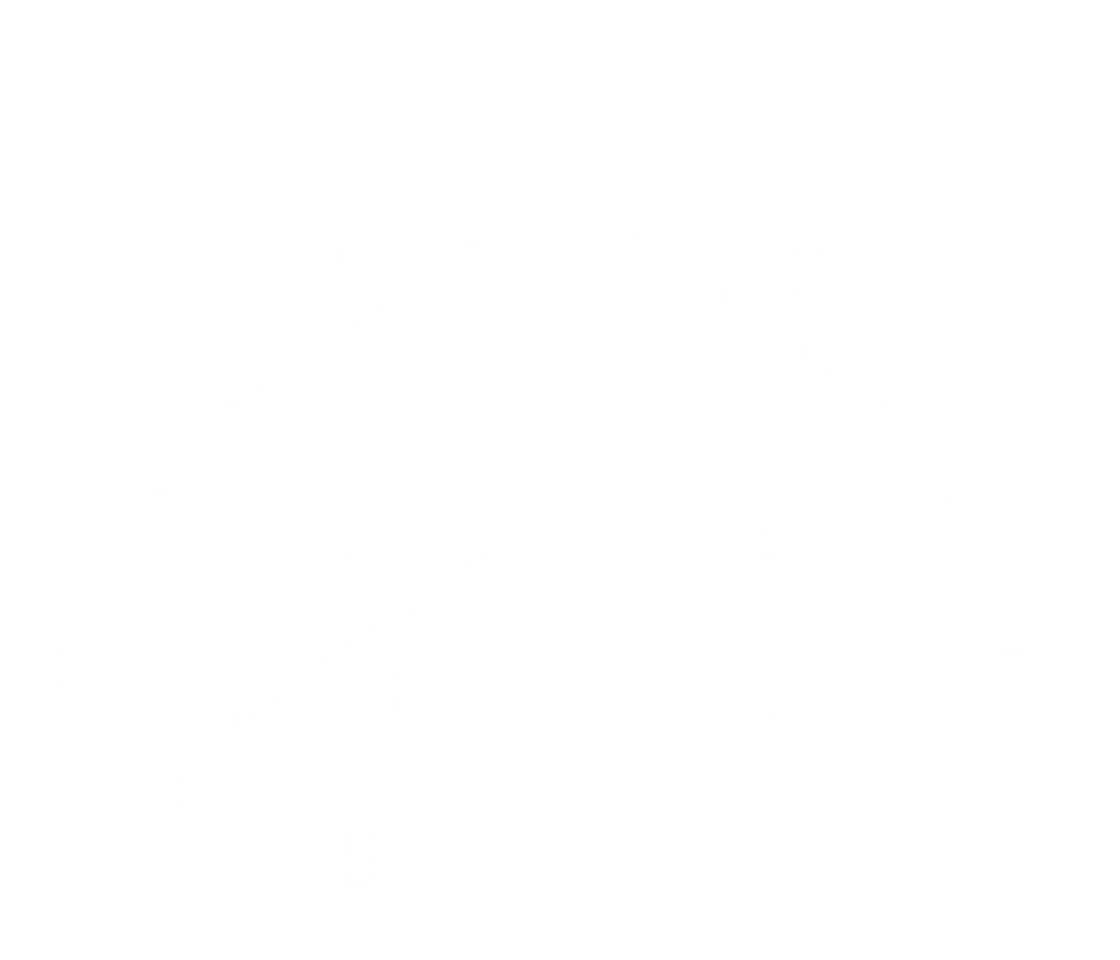
[Policy Routing 29](#_Toc86833989)

[11.8 Border Gateway Protocol 30](#_Toc86833990)

[BGP Sessions 31](#_Toc86833991)

Suppose we have multiple nodes connected to each other. Each of the nodes must keep a **routing table**, which has three columns at the minimum, Network, Cost and Next Hop. **Network** is the network we wish to reach and **Next Hop** is the node to which we must forward the packet in order to eventually reach the desired network. The **Cost**, also called the **Metric**, defines the cost of going to a particular hop. This helps us decide which path would be optimum, since it would be the one with the least cost.

A graph should make this image a little clearer:



For this graph, the routing table at the node might look like this:

|  |  |  |
| --- | --- | --- |
| Network | Cost | Next Hop |
| A | 6 | F |
| B | 10 | C |
| C | 5 | - |
| D | 4 | F |
| F | 1 | - |
| G | 6 | F |

The purpose of **Routing Protocols** is to create routing tables for routers by figuring out which path is the optimum one. What exactly we mean by optimum varies. It could be the path that is fastest, or cheapest, or a range of other things.

The routing protocols we will be looking into are all **unicast** routing protocols. We will not be dealing with multicast routing protocols at the moment.

It is possible to create routing tables **statically**, i.e. manually without the help of routing protocols, but that is not practical outside of a very limited scope. When considering the global Internet, which has millions of nodes with changing addresses, we need to take a **dynamic** approach. This is where routing protocols come in.

## 11.2 Intra and Inter-Domain Routing

The internet is too large for an individual router to find the optimum path between different nodes. To deal with this, the internet has been divided into **Autonomous Systems**, more commonly called **Domains**.

Based on this, routing protocols can be divided into two groups:

* **Intradomain Routing Protocols** - Some routing protocols work inside a particular domain. These are called Intradomain Routing Protocols. These can follow one of two methods, Distance Vector Routing, with RIP being the most famous protocol to follow this method, or Link State Routing, with OSPF being the most famous protocol to follow this method.
* **Interdomain Routing Protocols** – Some routing protocols work in between different domains, essentially routing from one domain to another. These are called Interdomain Routing Protocols. This only has one method, which is like Distance Vector Routing but improves upon it, called Path Vector Routing. The most famous protocol to use this method is BGP.

The router communicating between one domain and another is called the **boundary router**. This router has both Intradomain routing protocols and Interdomain routing protocols.

### Intradomain Routing Protocols

The **Distance Vector Routing** method uses the **Bellman-Ford** algorithm to calculate the shortest path between two nodes. This is opposed to the **Link State Routing** method, which uses the **Dijkstra** algorithm.

To fill up their routing tables, each router **shares** the information it has with other routers. This applies for all the routing protocols. What varies from one method to another is **what** information is shared, with **whom** it is shared and **when** it is shared.

For Distance Vector Routing, **all information** is shared. This means literally all the data in its own routing table will be shared. The information is shared with just the **immediate neighbours** of the router. This means that information is shared from one router to the next and from that router to the next, causing a **ripple effect**. Because of this, any changes to the routing table will take a long time to reach all the nodes in the network. This time is called the **convergence time**. The high convergence time was one of the biggest issues with the RIP method. However, there is **less network traffic**.

For Link State Routing, only the **neighbours’ information** is shared. This means just the entries that are for direct neighbours of the router will be shared. However, this information is shared with **every node** in the network. Because of this, any changes are conveyed almost immediately everywhere in the network. There is **more network traffic** because of this, but given that bandwidth has generally increased in the modern internet, OSPF has become more widely used.

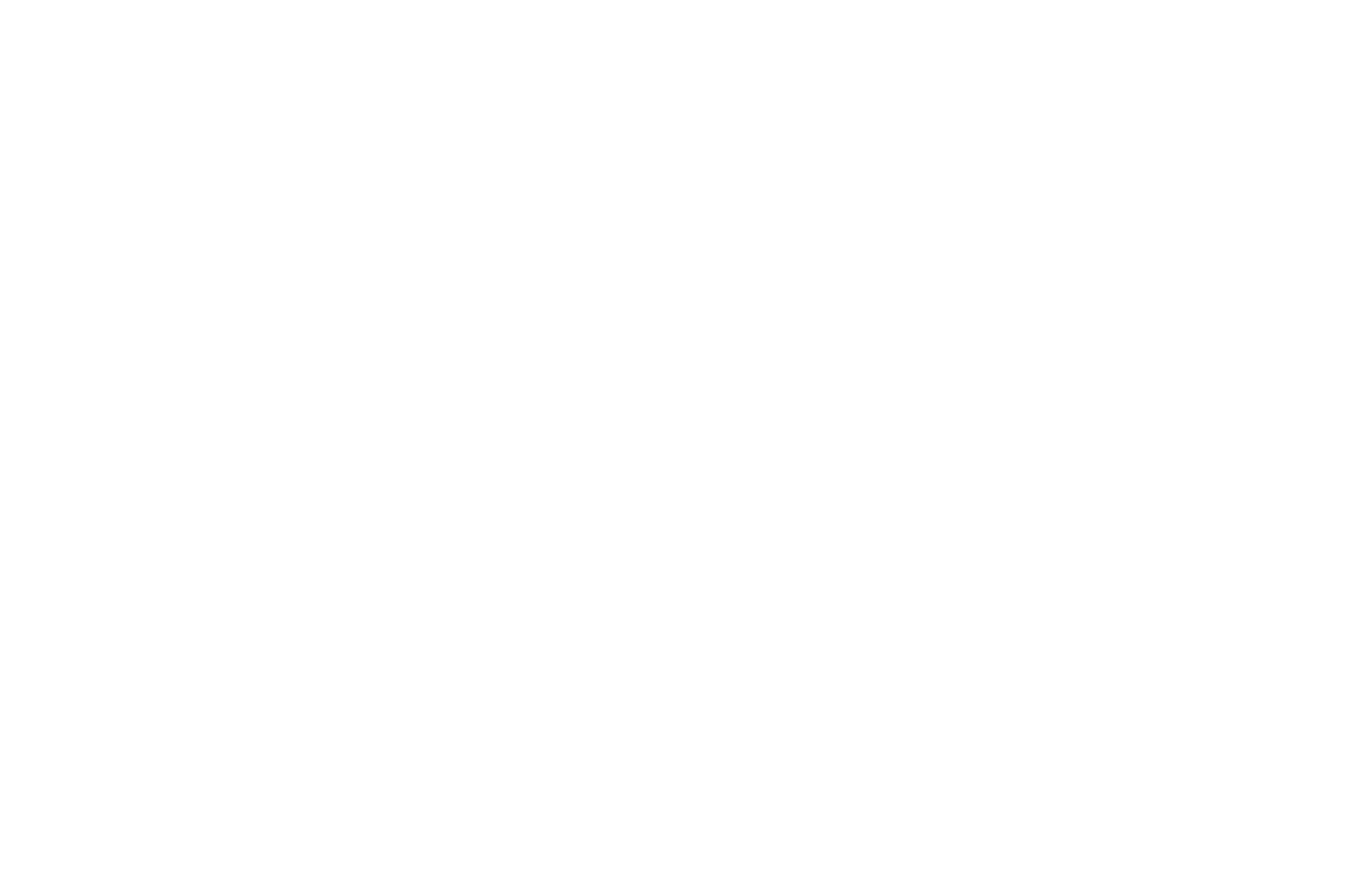
For both cases, the information is **shared repeatedly** after a **fixed period**. For Distance Vector Routing, this period is roughly **30 seconds**, while for Link State Routing, it is much higher, **over an hour**. However, any changes cause an **immediate update**.

In summary:

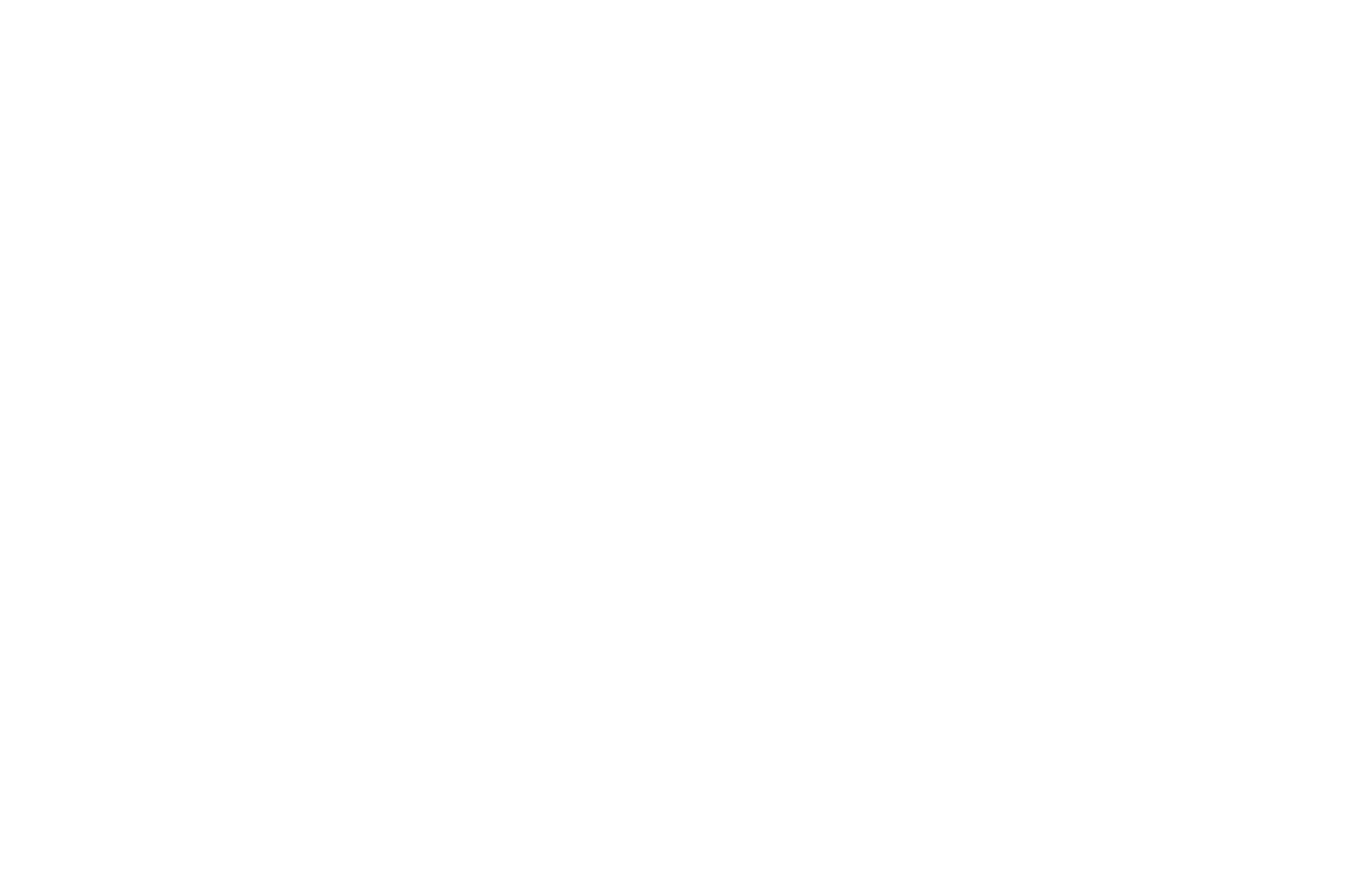
|  |  |  |
| --- | --- | --- |
|  | Distance Vector Routing | Link State Routing |
| Shares | All Information | Neighbours’ Information |
| Shares With | Neighbours | All Nodes |
| Change Speed | Slow | Fast |
| Network Traffic | Low | High |
| Share Time | 30 Seconds | Over 1 Hour |
| Famous Method | RIP | OSPF |
| Algorithm | Bellman-Ford | Dijkstra |

## 11.3 Distance Vector Routing

For **Distance Vector Routing**, the minimum cost is the **minimum distance**.

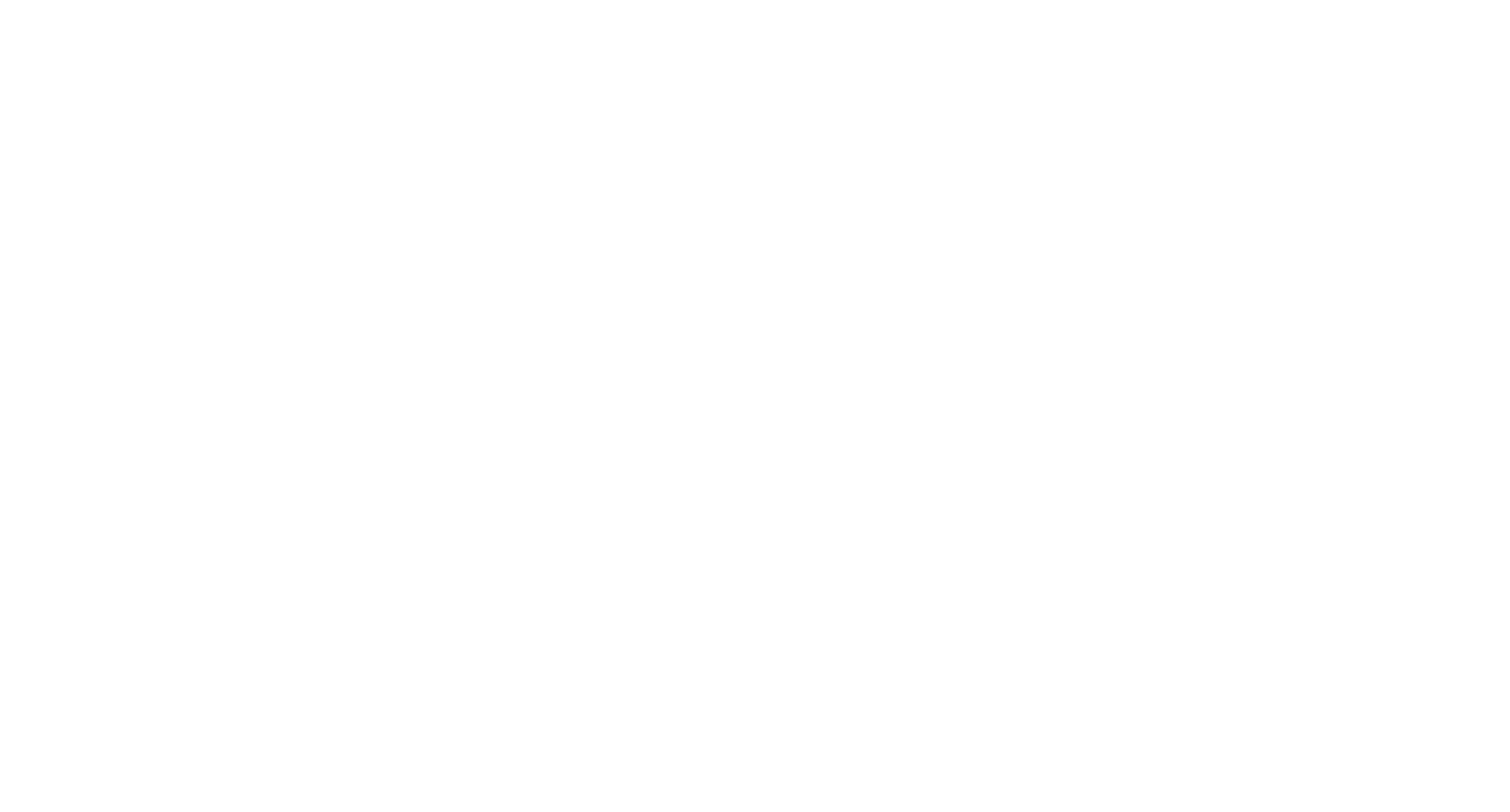


Initially, each node only knows about the distance between itself and its **immediate neighbours**.



Next, the nodes **share** their complete tables with their neighbours. More specifically, the **first and second columns** are shared, since the third column is redundant. For example, if node C is sending the table to node A, all the information it gives applies if node C is the next hop. When node A gets the table, it **adds** the third column and sets the next hop as C for all the rows. Additionally, the **cost** of going from node A to node C is also added to the **existing costs** for each row.

Each node **compares** the modified table with the old one and updates any rows that can be improved upon. This comparison stage is where the **Bellman-Ford algorithm** is used.



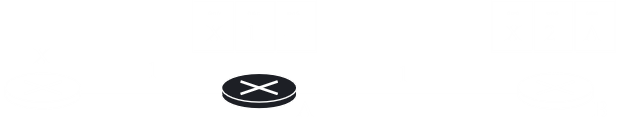
One tricky thing is that if the **next-node** entry is the **same** for a record, the **new row** is taken blindly, **without comparison**. Consider that the connection between node C and node E is **broken**, so the cost becomes **infinite**. When node C gives an update to node A about this, node A will not compare the exiting value (6) with the new one. If it did, it would not change the cost and the cost would be **incorrect**.

Instead, since the next-node, node C, is still the same, the row is **blindly updated**. Because it does this, the entry in node A’s table will also become **infinite**. This will allow the table to be updated correctly later on when a **different node** (say node B, with a cost of 8) advertises a path that is larger than node C’s previously working path (with a cost of 6) but is smaller than its current one (with an infinite cost).

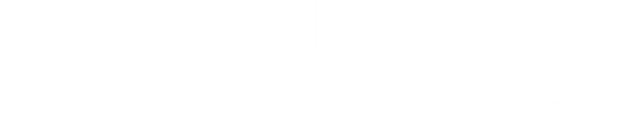
### Two-Node Instability

We mentioned that nodes will update their neighbours as soon as there is a change in their own routing table. This is done to deal with the **two-node instability** problem, also called the **counting to infinity** (C2I) problem.

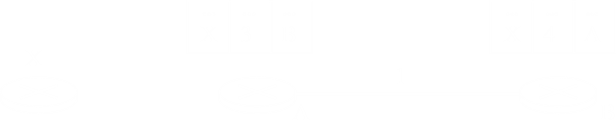
Consider that we have two nodes, node A and node B, connected as shown below:



For some reason, if the connection between node A and node X is **broken**, the cost to node X in **node A’s routing table** will become **infinite**. But say node A **does not inform** node B about this change. Soon after, node A will see that node B is able to reach node X with a cost of 2. This entry is obviously **incorrect**, but node A **does not know** that. Because of this, node A will also **erroneously update** its own table.



Node B will go on to make the same erroneous update.



And this will **continue indefinitely**.

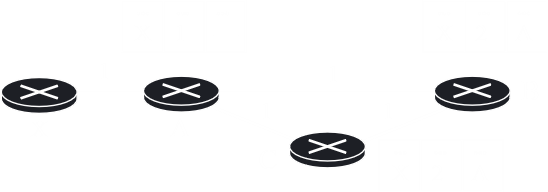
If node A had simply informed node B immediately after the initial change, this problem could have been avoided.

There are several solutions to the Counting to Infinity problem:

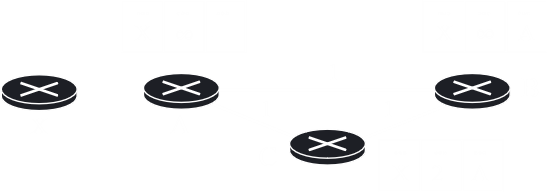
1. In the RIP protocol, infinite is set to **16**. Once this value is reached, the path is deemed unreachable. This is to prevent repeated forwarding of packets between nodes.
2. The **triggered update** solution we already know.
3. **Hold-Down**, which is when a router detects a problem with one of its connections and stops updating its routing table for that connection. In the above example, node A’s routing table would not update the row for node X.
4. **Split Horizon**, where a router does not share information about another router with that router. For the above example, the row in node B’s table where node A is the next hop is not shared with node A.
5. **Split Horizon with Poison Reverse**, where a router which is the next hop is actively told that cost is infinite. In the example above, node B would tell node A that the cost of going to node X through node B is infinite, even though this is a lie.

### Three Node Instability

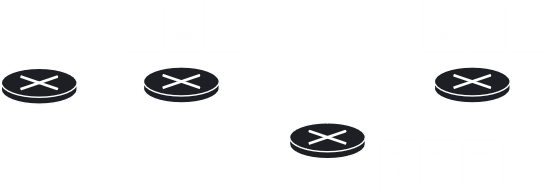
Consider that we have three nodes connected to each other, with node A connected to a fourth node, node X.



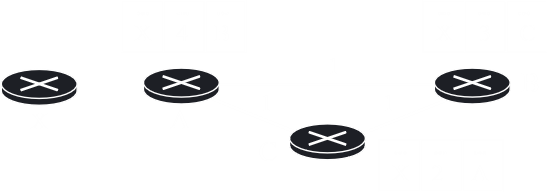
Say node A **loses its connection** to node X, and immediately **updates** both node B and node C. Unfortunately, the update to node C does not reach due to a **packet drop**.



This results in node B thinking that node X can be **reached via node C**, so node B updates its table.



Node A will now think that node X can be **reached via node B** and will update its table.

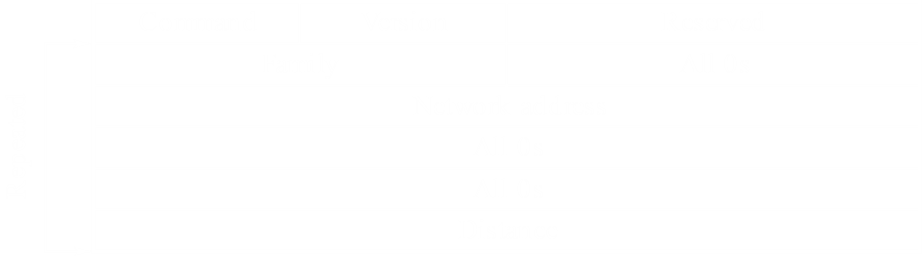


Once we have reached this point, node C will **blindly update**, which will cause node B to blindly update which will cause node A to blindly update. We have entered a loop.

## 11.4 Routing Information Protocol

The **Routing Information Protocol** is simply Distance Vector Routing, except that the cost is defined as the **hop count**. The maximum hop count is **15**, since infinity is defined as 16 hops.

### RIP Message Format



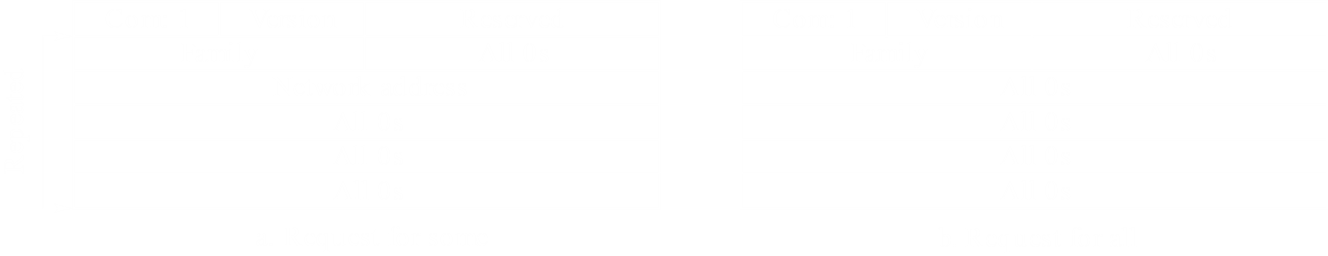
* **Command** – This is the message type, for request and for response.
* **Version** – This is the RIP version, which can be or . We are currently discussing version .
* **Family** – This refers to the protocol family. TCP/IP has the value .
* **Network Address** – This is a 14-byte field for the destination network. IP only uses 4 bytes. The rest are s.
* **Distance** – This 32-bit field defines the hop count from the router to the destination network.

The bottom 5 rows are repeated for each destination network. These rows form an **entry**.

### Requests and Responses

RIP has two types of messages, requests and responses.

A **request** is sent by a router that has just come up or that has some time-out entries. A request can ask for specific entries or for all entries.



A **response** can be either solicited or unsolicited. A **solicited** response is one that is sent in response to a request, with the destination information specified in the request. An **unsolicited** response is sent periodically or when there is a change in the routing table. The response is also called an **update packet**. The RIP message format shown earlier when discussing the different fields was actually a response message.

### Timers

The RIP protocol has three types of timers, periodic timers, expiration timers and garbage collection timers.

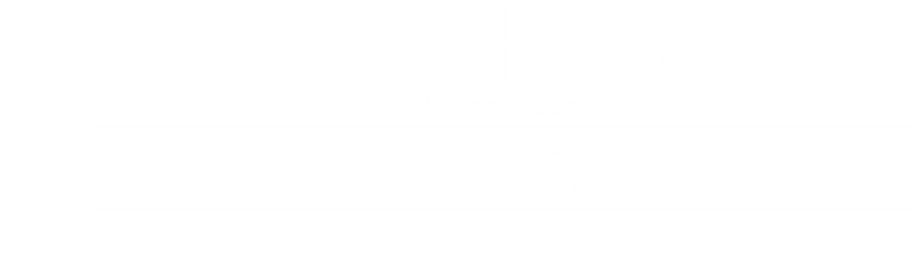
A **periodic timer** keeps track of how long has passed since the last update. By default, this is **30 seconds**. After this time has passed, a router sends its routing table to all its neighbours.

An **expiration timer** exists for **each entry** of the routing table. If the expiration timer reaches 0 and there have been no updates for this entry within this time, it is assumed that the entry is **invalid**. By default, the expiration time is **180 seconds**.

Once the expiration timer has expired, a third timer starts called the **garbage collection timer**, which is set to **120 seconds** by default. Once this time expires, the path is **removed** from the routing table.

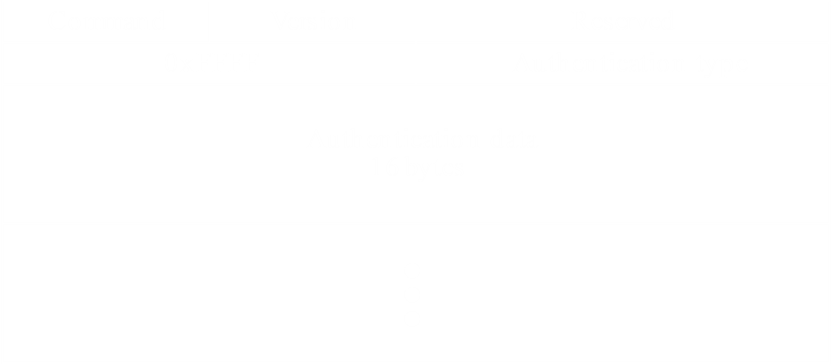
### RIP Version 2

Version 2 was created to overcome the **shortcomings** of version 1. The fields filled with s have been replaced.



* **Router Tag** – This contains the autonomous system number, which allows RIP to receive information from an interdomain routing protocol.
* **Subnet Mask** – This 4-byte field allows support for classless addressing and CIDR.
* **Next-Hop Address** – This is especially useful if two autonomous systems share a backbone network. The router can then be in either the same autonomous system or a different one.

The first entry of the message is also set aside for **authentication**. Note that this is just an entry in the message, not a separate message itself. The family field for this entry has the value . The second field is **authentication type**, which defines the authentication protocol used. The third field has the actual **authentication data**.



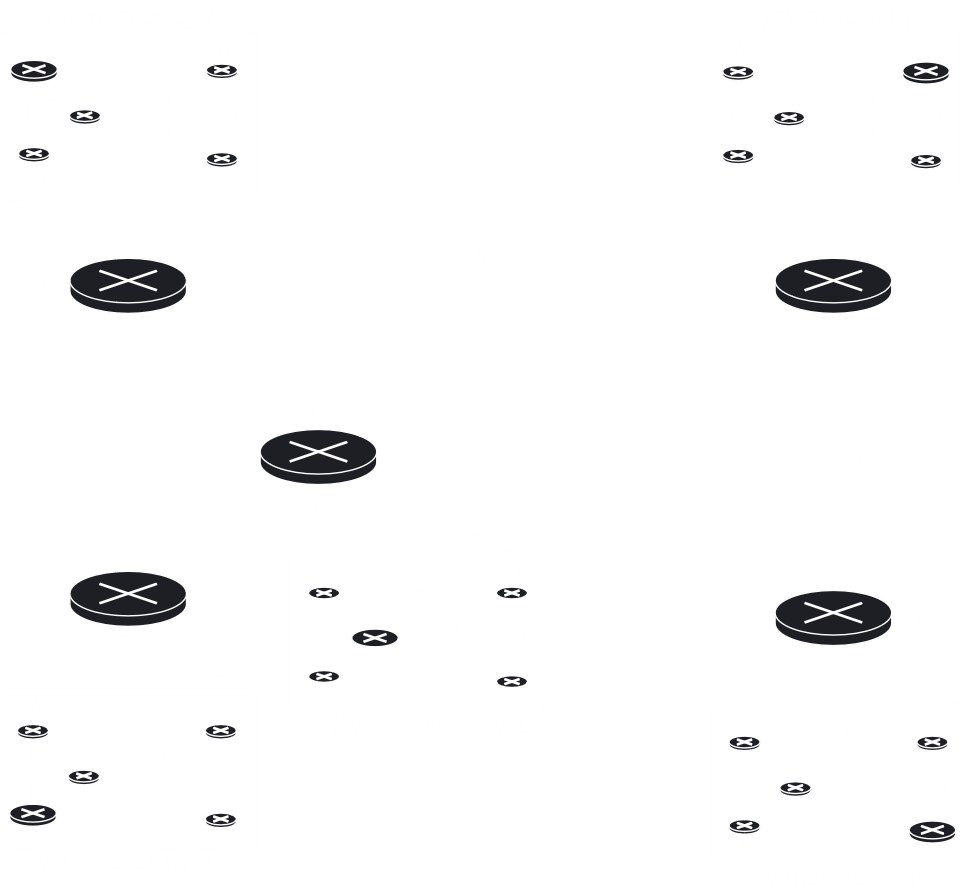
A further difference is that RIP version 2 uses **all-router multicast**, which sends RIP messages to all RIP routers in a network, as opposed to the normal broadcasting used in version 1 to send messages to every neighbour.

### Encapsulation

RIP messages are encapsulated in **UDP** user datagrams. RIP messages do not have a **length** field and rely on the length field of the UDP packet. The well-known port assigned to RIP in UDP is **port 520**.

## 11.5 Link State Routing

In **Link State Routing**, each node has the **entire topology** of the domain, the list of nodes and links, how they are connected and even the type, cost and condition of the links. The nodes can then use **Dijkstra’s Algorithm** to generate a routing table.



Each node only ever shares the details of its **immediate neighbours**, but it shares this knowledge with the **entire domain**. For the above diagram, the node A will send each of its neighbours information about the other neighbours, i.e. node B will get information about the links between node A and node C and node A and node D. Each of the neighbours in turn will **blindly forward** the information to their other neighbours. This causes a significant amount of duplicate information. Because of the nature of the sharing, this process is called **flooding**.

### Link State Packets

To send the information about the links with their neighbours, each node makes a **Link State Packet**. This packet contains each neighbour of the node and the **cost** of reaching the neighbour. Unlike RIP, this cost is not the **hop count**, but instead relies on some pre-assigned metric, such as throughput.

There is some more information as well, such as a **sequence number**. This helps with the flooding process, to avoid duplication. Since the entire network redistributes packets, a single node might get the same packet from multiple sources. It can look at the sequence number to determine that the packets are duplicates and drop all but one of them. Another piece of information is **age**, which allows routers to remove older packets.

### Flooding

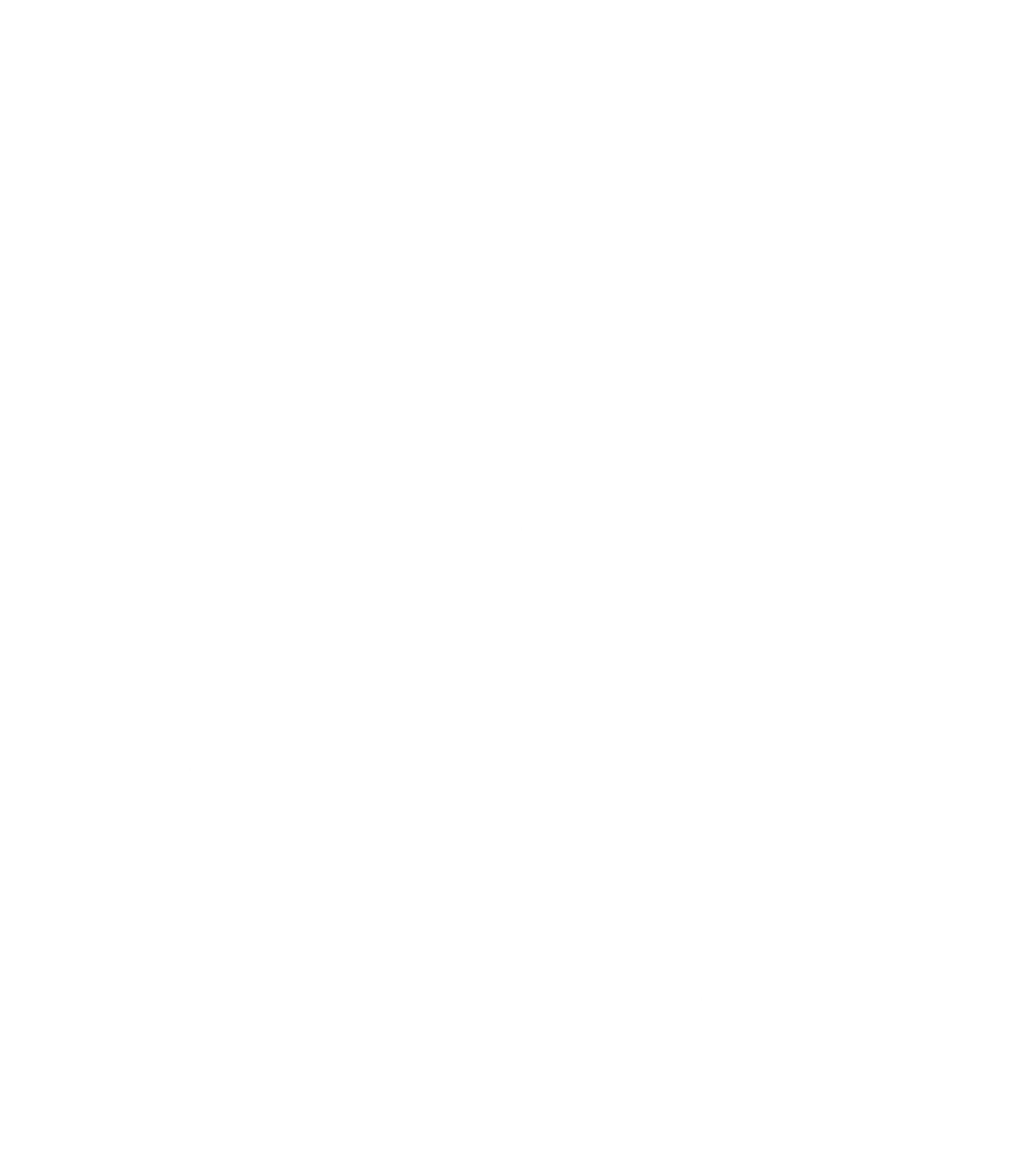
In the process of **flooding**, the creating node sends a copy of the LSP out of **each interface**. Each receiving node compares the LSP with the copy that it already has.

If the newly arrived LSP is **older** than the copy it has, it **discards** it.

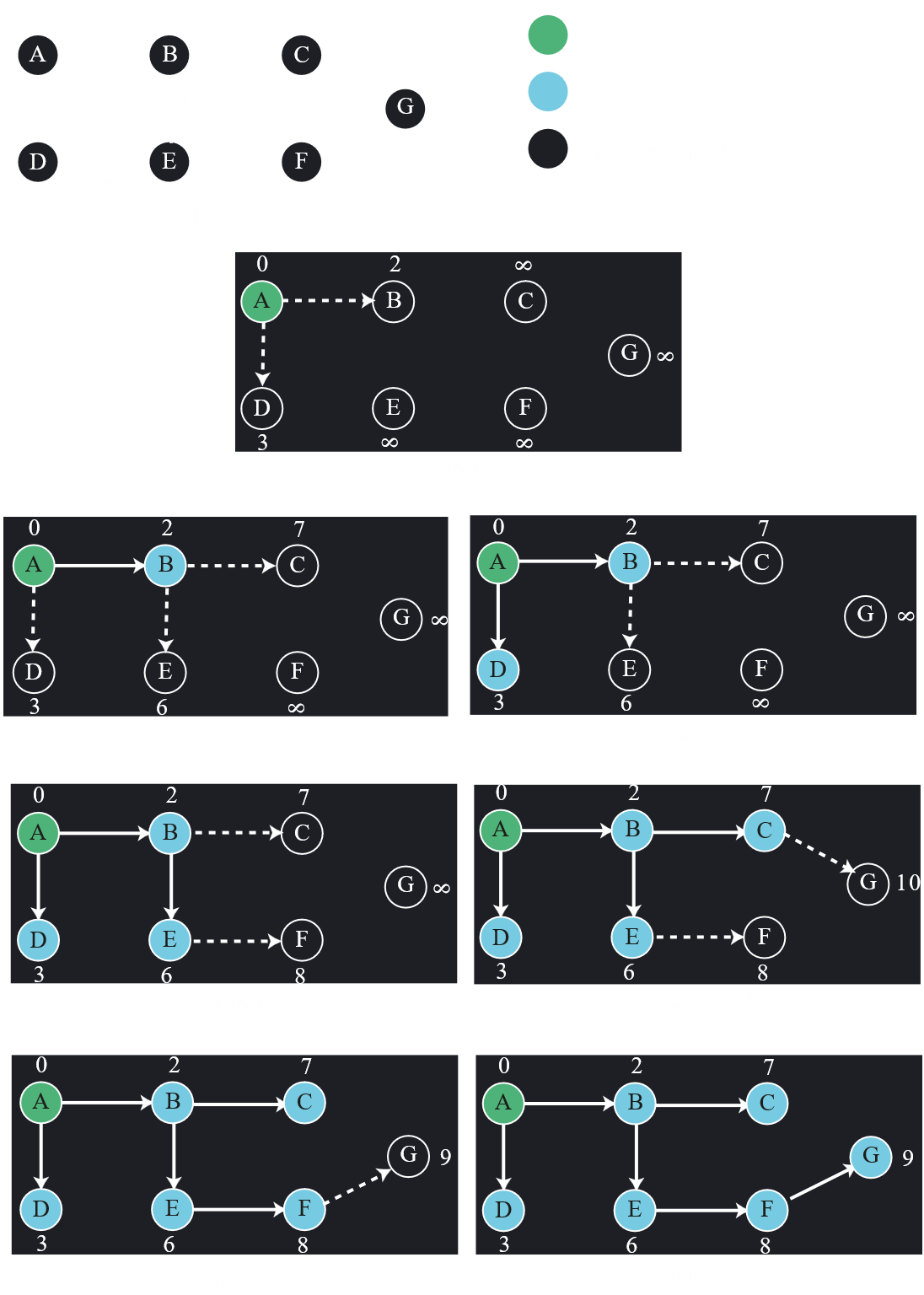
If it is newer, it discards the **old copy** and keeps the new one. Additionally, it also **sends out** a copy of the LSP through each interface except the one through which it received the LSP.

### Dijkstra’s Algorithm

Once **all the LSPs** have been received by a node, it essentially has a copy of the **whole topology**. From here, the node needs to find the **shortest path** to every other node. For this, the **Dijkstra Algorithm** is used.



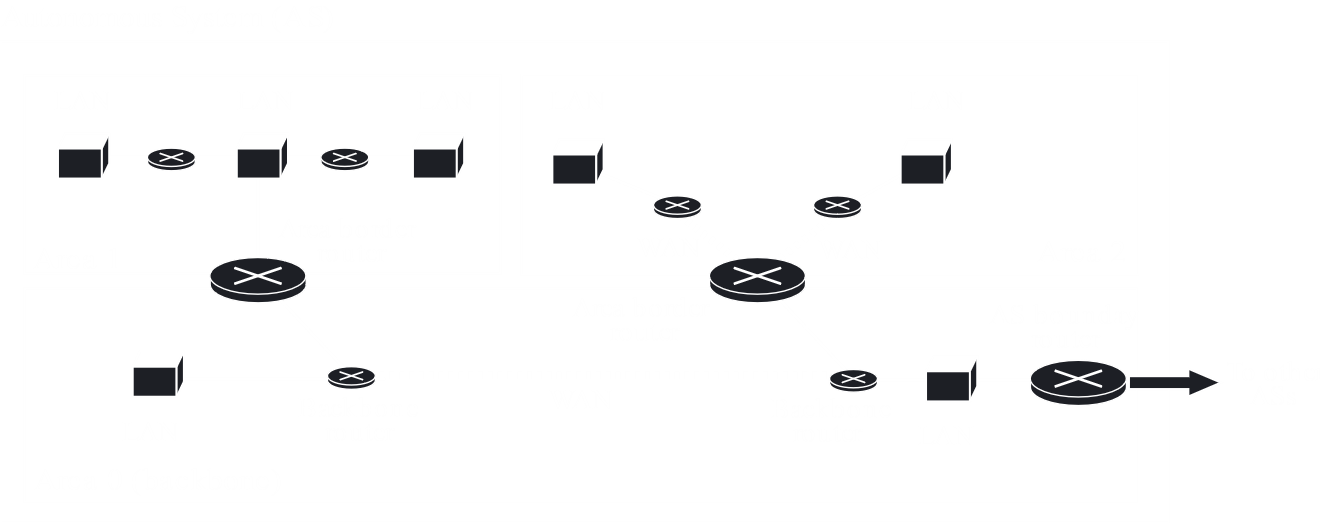
This process creates a **tree**, which has **no cycles** in it.



## 11.6 Open Shortest Path First

The **Open Shortest Path First** (OSPF) protocol is an interdomain routing protocol based on **Link State Routing**.

In OSPF, a single **autonomous system** is divided into smaller **areas**, just to make the process of routing easier.



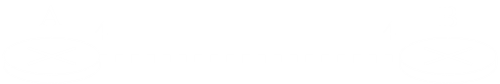
There is always an **Area 0**, which acts as the backbone. All the other areas connect to Area 0, which in turn connects to a different Autonomous System through an **Autonomous System (AS) Boundary Router**, also called the **Speaker Node**.

At the border of an area, we have an **Area Border Router**, which connects the area to Area 0. This makes the Flooding process easier. Nodes are usually flooding only inside their own areas, while the backbone is only being flooded from the Area Border Routers. Thus, the number of **broadcast packets** is reduced.

### Types of Links

OSPF allows several **types of links**:

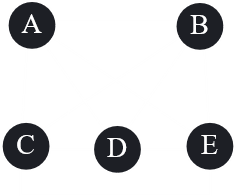
1. **Point-to-Point** – This is a direct link between two routers with no hosts or other routers in between.



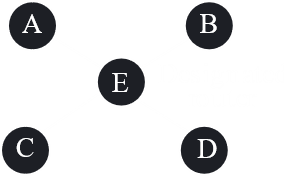
1. **Transient Link** – This is a network with multiple routers attached.



This kind of looks like a **bus topology**, which seems to imply that every router is connected to all the others. This would end up looking something like this in real life:

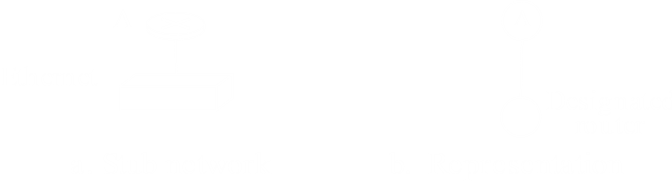


Clearly this is inconvenient, since there are packets for each flooding session, so what we end up doing is using one of the routers as a **designated router**, which acts as the ‘bus’.



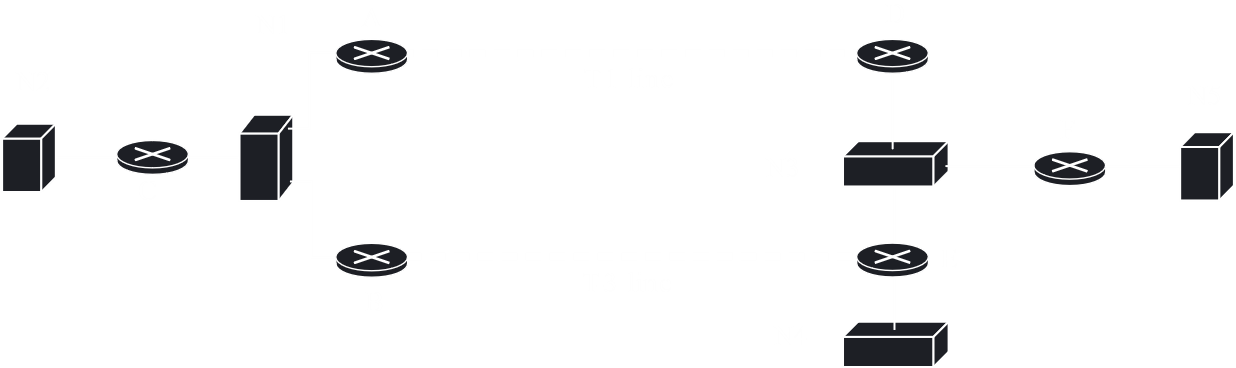
This diagram reduces the number of packets in the flooding session to .

1. **Stub Link** – This is a network that is connected to just one router.



1. **Virtual Link** – This is a link created by the administrator in case the link between two networks is broken.

Example



In the diagram above:

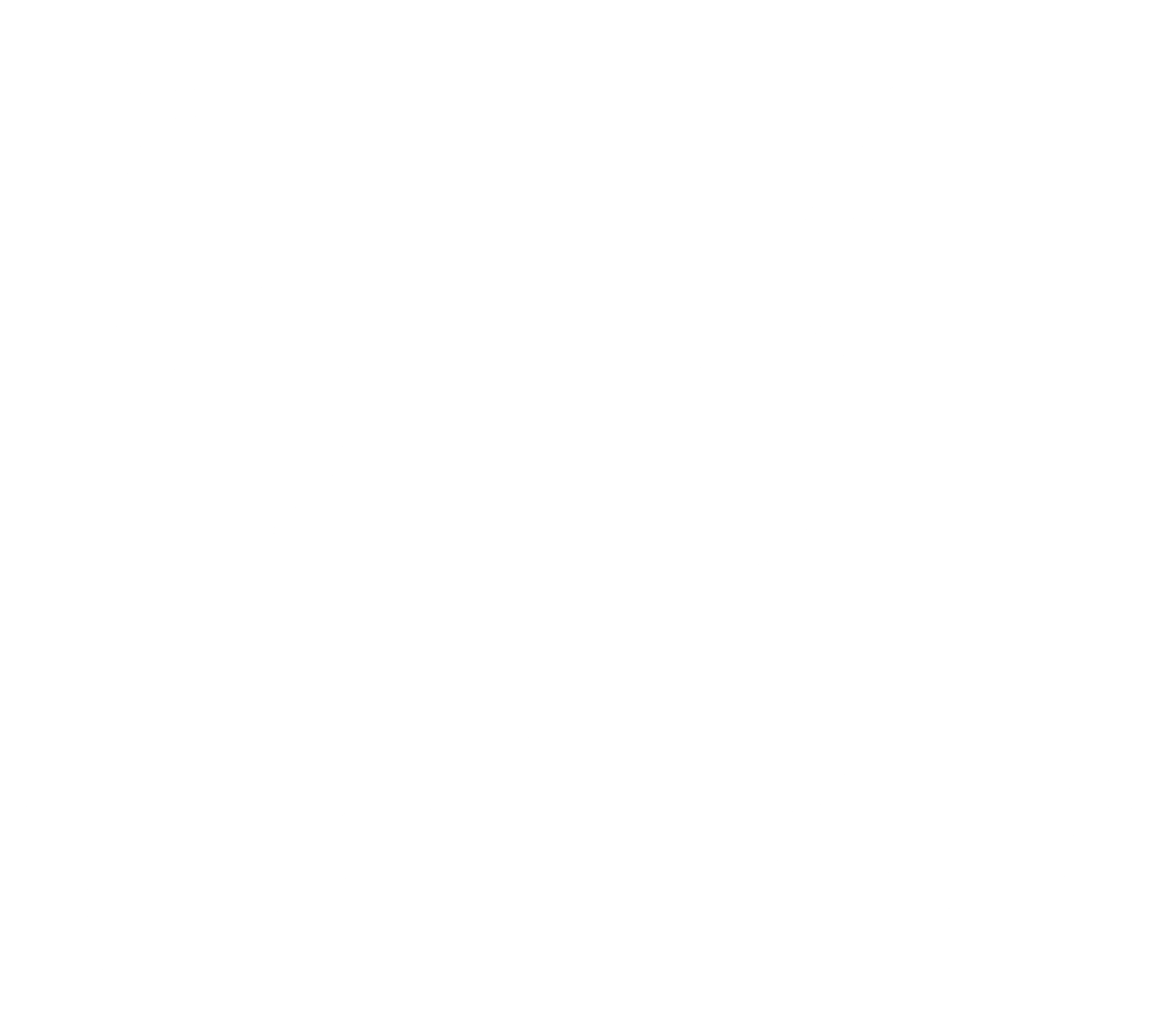
* N2, N4 N5 are Stub Links.
* N1 and N3 are Transient Links.
* The links between A and D and B and E are Point-to-Point Links. These are not given an identity, which is why they are dotted.

## 11.7 Path Vector Routing

We will now be moving on to **interdomain routing**, from which we will be studying the **Path Vector Routing** protocol. This is a huge topic and requires dedicated lectures, so for now, we will just be touching on the main parts of it.

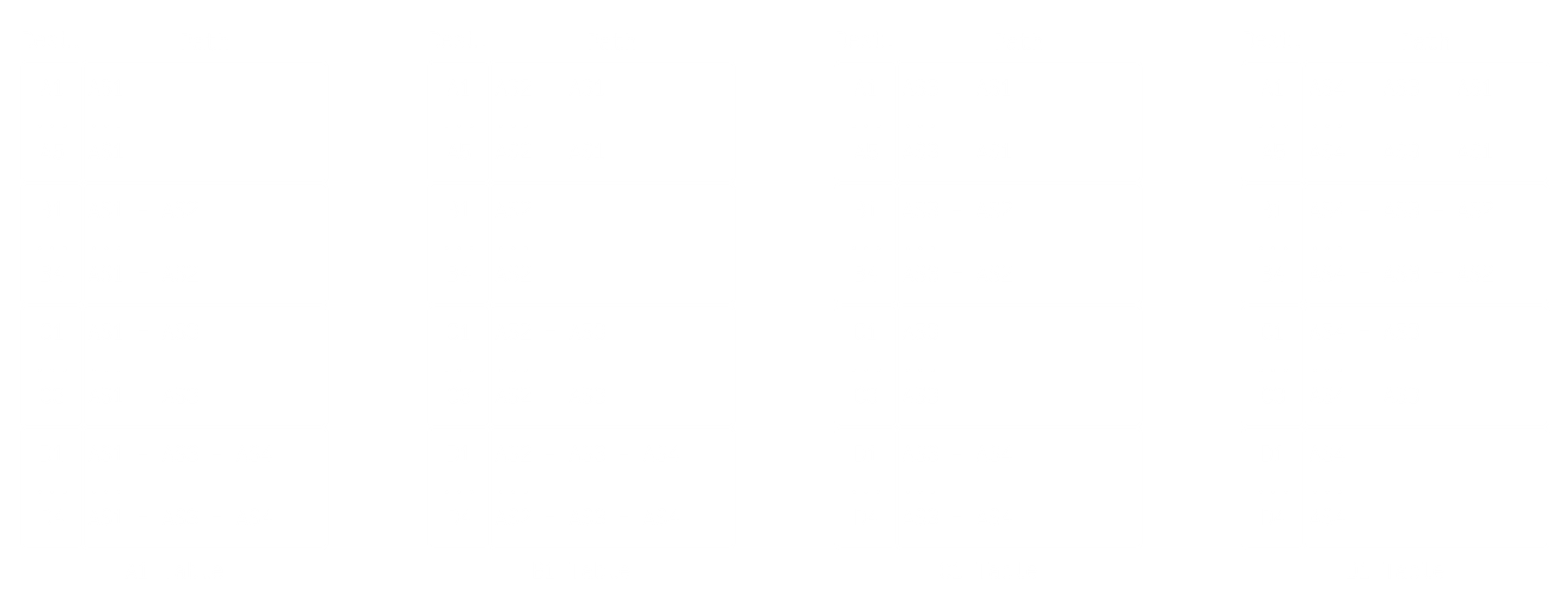
We cannot directly use Distance Vector Routing, since it will be instable, mainly due to the Count-to-Infinity problem. We also cannot use Link State Routing directly, since it is costly to calculate and to flood.

Path Vector Routing is like **Distance Vector Routing**, except that it is only the **speaker node** which creates a routing table and advertises it to other speaker nodes in neighbouring autonomous systems. This sharing takes place over a **TCP connection**, which is very stable.



The above diagram shows the **initialization** state. Each speaker node has created a table which has the different possible destination routers within its autonomous system and provided the ‘path’ to that destination in the form of the autonomous system it must go through.

After the tables are **shared**, they become like this:



Thus, for a router in AS1 to reach a destination in AS4, it must go through AS1-AS3-AS4. This is the **path** that must be followed, which is why this process is called Path Vector Routing.

### Loop Prevention

When sharing routing tables, since the entire path is visible, an autonomous system will immediately know if its own name is already in the path. If it is, then this packet is a repetitive packet, meaning the autonomous system has already passed it. This packet can be **dropped**. In this way, **loops** can be **prevented**.

### Policy Routing

It is possible that, e.g, AS1 is not accepting any packets from AS3 as a policy. In these cases, any packets that advertise paths containing AS3 will be ignored by AS1. This ability to maintain a **policy** is why the most widely used Path Vector Routing protocol, **BGP**, is also said to be **Policy Based Routing**.

## 11.8 Border Gateway Protocol

The **Border Gateway Protocol** (BGP) is basically the only Path Vector Routing protocol.

BGP can maintain certain **attributes** related to the path. These attributes are divided into two categories:

1. **Well-Known Attributes** - These are attributes that must be supported by every router. These can be of two types:
   1. **Well-Known Mandatory Attributes** – When advertising any path, these attributes must be present. If they are not, then it will be considered an error. Well-Known Mandatory Attributes include ORIGIN (which tells us what the original source of the information is), AS\_PATH (which is the list of ASs) and NEXT\_HOP.
   2. **Well-Known Discretionary Attributes** – These do not necessarily need to be included in every packet, but if they are, every router needs to be able to handle it.
2. **Optional Attributes** – These are attributes that are not necessarily supported by all routers. Depending on what a router does when it does not support an attribute, the optional attributes can be further divided into two sub-categories:
   1. **Optional Transitive Attributes** – These attributes are forwarded by the router that does not support the attribute.
   2. **Optional Non-Transitive Attributes** – These attributes are dropped by the router that does not support the attribute.

### BGP Sessions

A **session** is a connection between BGP routers for the exchange of routing information. These sessions can be of two types:

1. **External BGP** (E-BGP) – This is used to exchange information between two speaker nodes in two different ASs.
2. **Internal BGP** (I-BGP) – This is used to exchange information between two routers in the same AS. This can be used instead of relying on RIP or OSPF.