**3.1 Introduction**

* A routing table can he either static or dynamic.
* A static table is one with manual entries A dynamic table, on the other hand, is one that is updated automatically when there is a change somewhere on the Internet.
* Today, an internet needs dynamic routing tables. The tables need to be updated as soon as there is a change in the internet. For instance, they need to be updated when a router is down, and they need to be updated whenever a better route has been found.
* Routing protocols have been created in response to the demand for dynamic routing tables.
* A routing protocol is a combination of rules and procedures that lets routers on the internet inform each other of changes. It allows routers to share whatever they know about the internet or their neighbourhood. The sharing of information allows a router in San Francisco to know about the failure of a network in Texas. The routing protocols also include procedures for combining information received from other routers.

**3.1.1 Optimization**

* A router receives a packet from a network and passes it to another network.
* A router is usually attached to several networks.
* When it receives a packet, to which network should it pass the packet? The decision is based on optimization: Which of the available pathways is the optimum pathway? What is the definition of the term optimum?
* One approach is to assign a cost for passing through a network. We call this cost a metric.
* However, the metric assigned to each network depends on the type of protocol.
* Some simple protocols,
  + **Routing Information Protocol (RIP)**
    - It treats all networks as equals.
    - The cost of passing through a network is the same; it is one hop count. So if a packet passes through 10 networks to reach the destination, the total cost is 10 hop counts.
  + **Open Shortest Path First (OSPF)**
    - It allows the administrator to assign a cost for passing through a network based on the type of service required.
    - A route through a network can have different costs (metrics).
    - For example, if maximum throughput is the desired type of service, a satellite link has a lower metric than a fiber optic line. On the other hand, if minimum delay is the desired type of service, a fiber-optic line has a lower metric than a satellite link. Routers use routing tables to help decide the best route.
    - OSPF protocol allows each router to have several routing tables based on the required type of service.
  + **Border Gateway Protocol (BGP)**
    - It allows administrator to set the policy based on the different criterion.
    - The policy defines what paths should be chosen.

0

**3.2 Intra- and Interdomain Routing**

* Today, an internet can be so large that one routing protocol cannot handle the task of updating the routing tables of all routers. For this reason, an internet is divided into autonomous systems.
* An autonomous system (AS) is a group of networks and routers under the authority of a single administration.
* Routing inside an autonomous system is referred to as intradomain routing.
* Routing between autonomous systems is referred to as interdomain routing.
* Each autonomous system can choose one or more intradomain routing protocols to handle routing inside the autonomous system. However, only one interdomain routing protocol handles routing between autonomous systems as shown in the following figure 3.2.1.

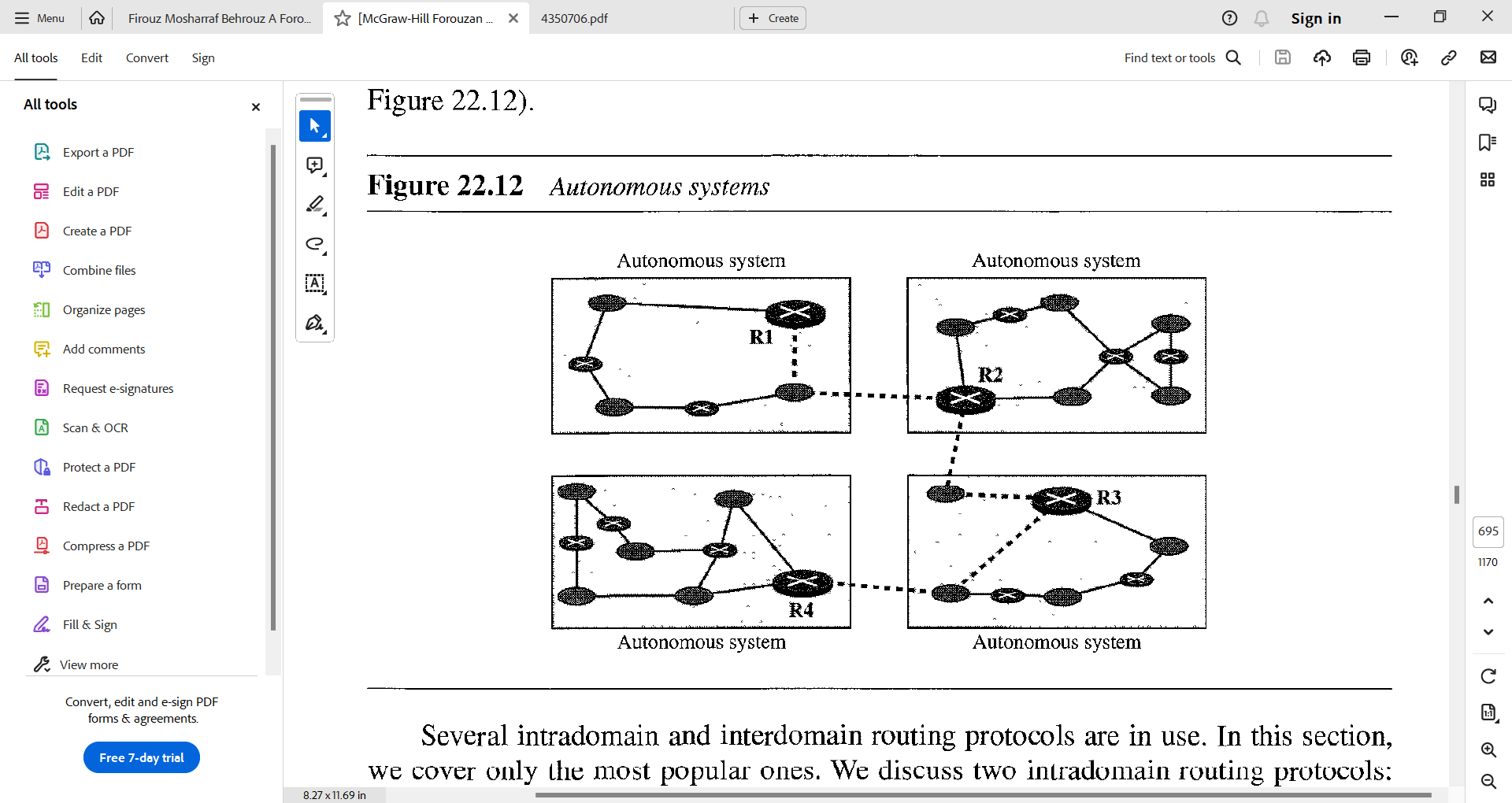


Figure 3.2.1 Autonomous Systems

* Several intradomain and interdomain routing protocols are in use.
* In this section, we cover only the most popular ones. We discuss two intradomain routing protocols: **distance vector** and **link state.** We also introduce one interdomain routing protocol: **path vector**. As shown in the figure 3.2.2.
* **Routing Information Protocol (RIP)** is an implementation of the distance vector protocol.
* **Open Shortest Path First (OSPF)** is an implementation of the link state protocol.
* **Border Gateway Protocol (BGP)** is an implementation of the path vector protocol.

A screenshot of a computer

Description automatically generated

Figure 3.2.2 Popular Routing Protocols.

**3.3 General Idea of Unicast Routing**

* Unicast routing on the Internet, with a large number of routers and a huge number of hosts, can be done only by using hierarchical routing: routing in several steps using different routing algorithms.
* In unicast routing, a packet is routed, hop by hop, from its source to its destination by the help of forwarding tables.
* The source host needs no forwarding table because it delivers its packet to the default router in its local network.
* The destination host needs no forwarding table either because it receives the packet from its default router in its local network.
* This means that only the routers that glue together the networks in the internet need forwarding tables.
* With the above explanation, routing a packet from its source to its destination means routing the packet from a source router (the default router of the source host) to a destination router (the router connected to the destination network).
* Although a packet needs to visit the source and the destination routers, the question is what other routers the packet should visit. In other words, there are several routes that a packet can travel from the source to the destination; what must be determined is which route the packet should take.

***An internet as a graph***

* To find the best route, an internet can be modelled as a graph. A graph in computer science is a set of nodes and edges (lines) that connect the nodes.
* To model an internet as a graph, we can think of each router as a node and each network between a pair of routers as an edge.
* An internet is, in fact, modelled as a weighted graph, in which each edge is associated with a cost. If a weighted graph is used to represent a geographical area, the nodes can be cities and the edges can be roads connecting the cities; the weights, in this case, are distances between cities.
* In routing, however, the cost of an edge has a different interpretation in different routing protocols, which we discuss in a later section. For the moment, we assume that there is a cost associated with each edge. If there is no edge between the nodes, the cost is infinity.
* Figure 3.1.2.1 shows how an internet can be modeled as a graph.

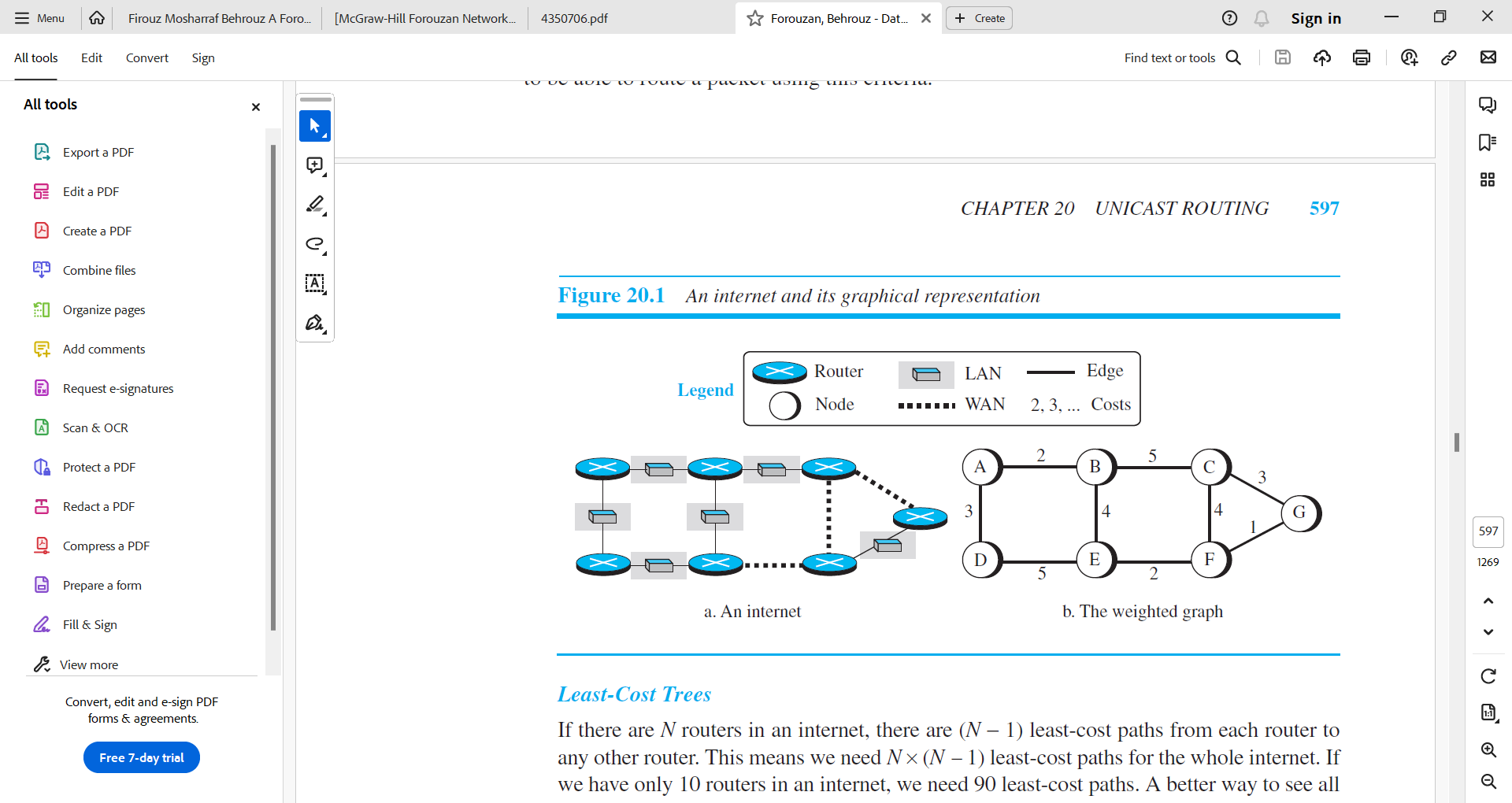


Figure 3.3.1 An internet and its graphical representation

**3.3.1 Least-cost Routing**

* When an internet is modelled as a weighted graph, one of the ways to interpret the best route from the source router to the destination router is to find the least cost between the two.
* In other words, the source router chooses a route to the destination router in such a way that the total cost for the route is the least cost among all possible routes.
* In Figure 3.3.1, the best route between A and E is A-B-E, with the cost of 6. This means that each router needs to find the least-cost route between itself and all the other routers to be able to route a packet using this criteria.

***Least-cost trees***

* If there are N routers in an internet, there are (N − 1) least-cost paths from each router to any other router. This means we need N × (N − 1) least-cost paths for the whole internet.
* If we have only 10 routers in an internet, we need 90 least-cost paths.
* A better way to see all of these paths is to combine them in a least-cost tree. A least-cost tree is a tree with the source router as the root that spans the whole graph (visits all other nodes) and in which the path between the root and any other node is the shortest.
* In this way, we can have only one shortest-path tree for each node; we have N least-cost trees for the whole internet.
* We show how to create a least-cost tree for each node later in this section; for the moment, Figure 3.3.1.1 shows the seven least-cost trees for the internet in Figure 3.3.1.

A screenshot of a computer

Description automatically generated

Figure 3.3.1.1 Least-cost trees for nodes in the internet of Figure 3.3.1

* The least-cost trees for a weighted graph can have several properties if they are created using consistent criteria.

1. The least-cost route from X to Y in X’s tree is the inverse of the least-cost route from Y to X in Y’s tree; the cost in both directions is the same. For example, in Figure 3.3.1.1, the route from A to F in A’s tree is (A → B → E → F), but the route from F to A in F’s tree is (F → E → B → A), which is the inverse of the first route. The cost is 8 in each case.
2. Instead of travelling from X to Z using X’s tree, we can travel from X to Y using X’s tree and continue from Y to Z using Y’s tree. For example, in Figure 3.3.1.1, we can go from A to G in A’s tree using the route (A → B → E → F → G). We can also go from A to E in A’s tree (A → B → E) and then continue in E’s tree using the route (E → F → G). The combination of the two routes in the second case is the same route as in the first case. The cost in the first case is 9; the cost in the second case is also 9 (6 + 3).

**3.4 Routing Algorithms**

* After discussing the general idea behind least-cost trees and the forwarding tables that can be made from them, now we concentrate on the routing algorithms.
* Several routing algorithms have been designed in the past. The differences between these methods are in the way they interpret the least cost and the way they create the least-cost tree for each node.
* In this section, we discuss the common algorithms; later we show how protocol on the Internet implements one of these algorithms.

**3.4.1 Distance-Vector Routing**

* In distance vector routing, the least-cost route between any two nodes is the route with minimum distance. In this protocol, as the name implies, each node maintains a vector (table) of minimum distances to every node. The table at each node also guides the packets to the desired node by showing the next stop in the route (next-hop routing). We can think of nodes as the cities in an area and the lines as the roads connecting them. ,A. table can show a tourist the minimum distance between cities.
* In Figure 3.4.1, we show a system of five nodes with their corresponding tables.

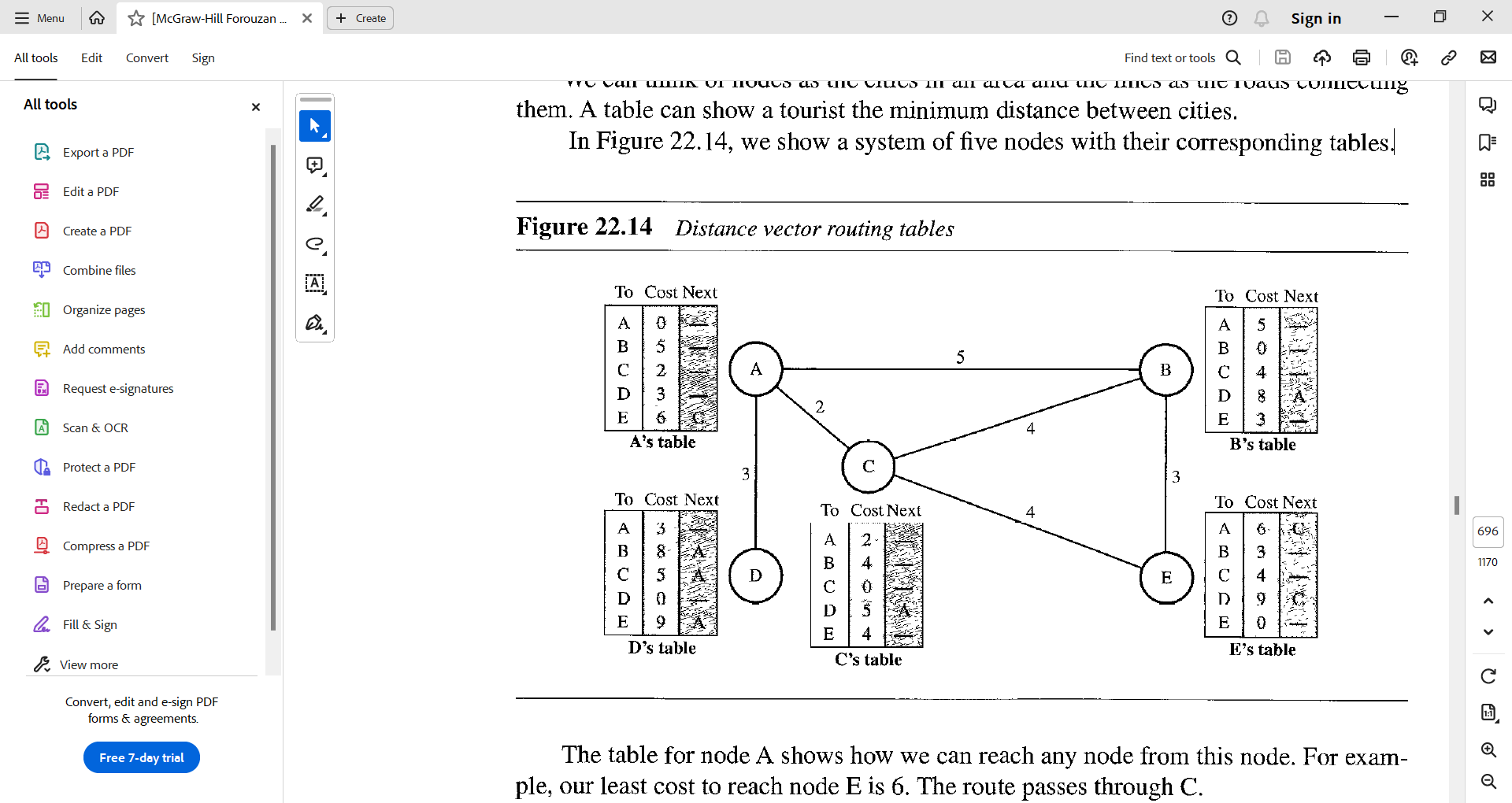


Figure 3.4.1 Distance Vector Routing Tables

* The table for node A shows how we can reach any node from this node. For example, our least cost to reach node E is 6. The route passes through C.

Initialization

* The tables in Figure 3.4.1 are stable; each node knows how to reach any other node and the cost.
* At the beginning, however, this is not the case. Each node can know only the distance between itself and its immediate neighbors, those directly connected to it.
* So, for the moment, we assume that each node can send a message to the immediate neighbors and find the distance between itself and these neighbors.
* Figure 3.4.2 shows the initial tables for each node.
* The distance for any entry that is not a neighbor is marked as infinite (unreachable).

A screenshot of a computer

Description automatically generated

Figure 3.4.2 Initialization of tables in distance vector routing

***Sharing***

* The whole idea of distance vector routing is the sharing of information between neighhors.
* Although node A does not know about node E, node C does. So, if node C shares its routing table with A, node A can also know how to reach node E.
* On the other hand, node C does not know how to reach node D, but node A does. If node A shares its routing table with node C, node C also knows how to reach node D. In other words, nodes A and C, as immediate neighbors, can improve their routing tables if they help each other.
* There is only one problem. How much of the table must be shared with each neighbor? A node is not aware of a neighbor's table.
* The best solution for each node is to send its entire table to the neighbor and let the neighbor decide what part to use and what part to discard. However, the third column of a table (next stop) is not useful for the neighbor. When the neighbor receives a table, this column needs to be replaced with the sender's name. If any of the rows can be used, the next node is the sender of the table. A node therefore can send only the first two columns of its table to any neighbor. In other words, sharing here means sharing only the first two columns.

Updating

* When a node receives a two-column table from a neighbor, it needs to update its routing table. Updating takes three steps:

1. The receiving node needs to add the cost between itself and the sending node to each value in the second column. The logic is clear. If node C claims that its distance to a destination is x mi, and the distance between A and C is y mi, then the distance between A and that destination, via C, is x + y mi.
2. The receiving node needs to add the name of the sending node to each row as the third column if the receiving node uses information from any row. The sending node is the next node in the route.
3. The receiving node needs to compare each row of its old table with the corresponding row of the modified version of the received table.
4. If the next-node entry is different, the receiving node chooses the row with the smaller cost. If there is tie, the old one is kept.
5. If the next-node entry is the same, the receiving node chooses the new row. For example, suppose node C has previously advertised a route to node X with distance 3. Suppose that now there is no path between C and X; node C now advertises this route with a distance of infinity. Node A must not ignore this value even though its old entry is smaller. The old route does not exist any more. The new route has a distance of infinity.

* Figure 3.4.3 shows how node A updates its routing table after receiving the partial table from node C.

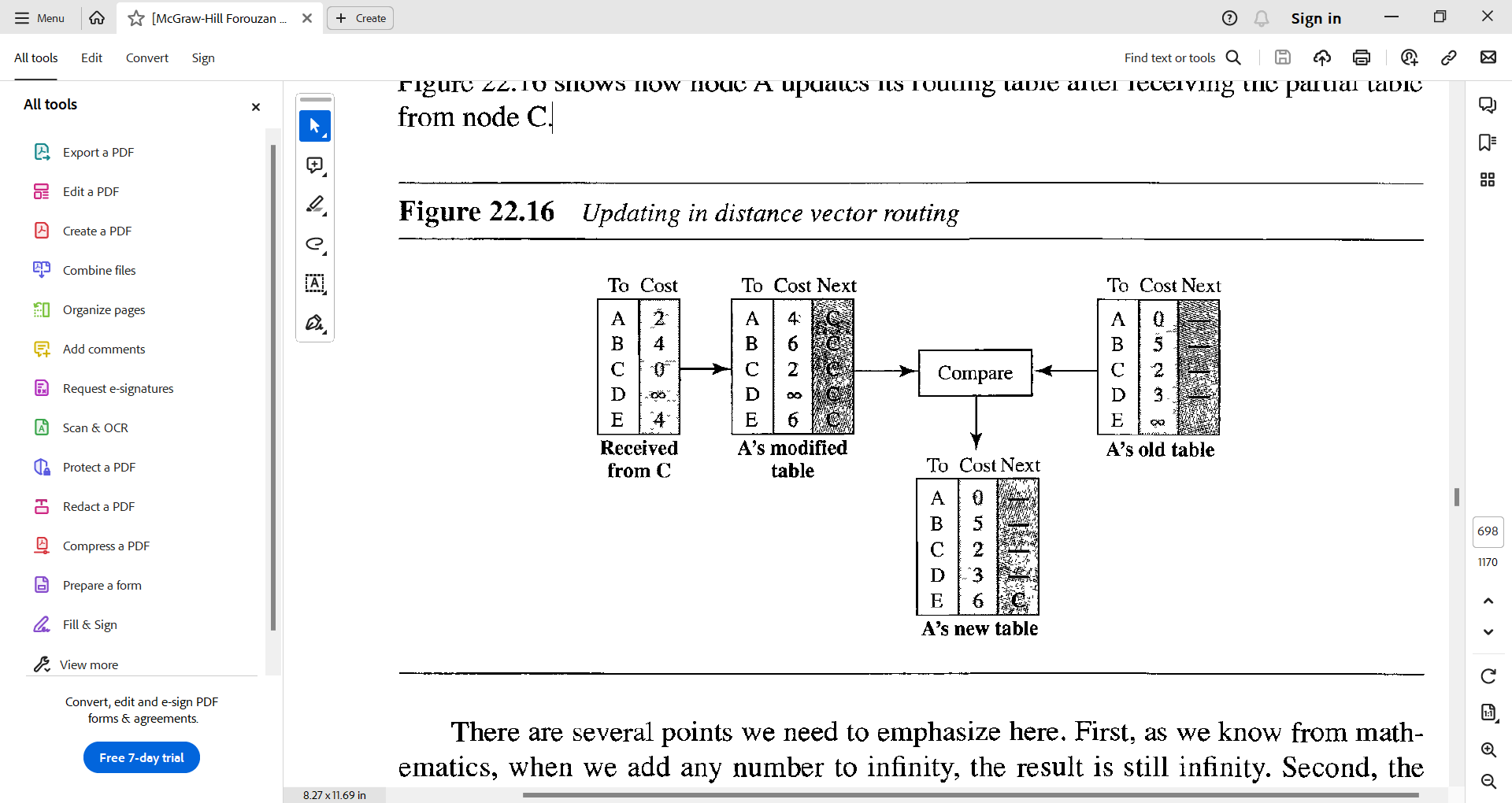


Figure 3.4.3 Updating in distance vector routing

* There are several points we need to emphasize here.
* First, as we know from mathematics, when we add any number to infinity, the result is still infinity.
* Second, the modified table shows how to reach A from A via C. If A needs to reach itself via C, it needs to go to C and come back, a distance of 4.
* Third, the only benefit from this updating of node A is the last entry, how to reach E. Previously, node A did not know how to reach E (distance of infinity); now it knows that the cost is 6 via C.
* Each node can update its table by using the tables received from other nodes. In a short time, if there is no change in the network itself, such as a failure in a link, each node reaches a stable condition in which the contents of its table remains the same.

***When to share***

* The question now is, When does a node send its partial routing table (only two columns) to all its immediate neighbors? The table is sent both periodically and when there is a change in the table.
* **Periodic Update** 
  + A node sends Its routing table, normally every 30 s, in a periodic update. The period depends on the protocol that is using distance vector routing.
* **Triggered Update** 
  + A node sends its two-column routing table to its neighoors any time there is a change in its routing table. This is called a triggered update.
  + The change can result from the following.
    - A node receives a table from a neighbor, resulting in changes in its own table after updating.
    - A node detects some failure in the neighboring links which results in a distance change to infinity.

***Count to Infinity***

* A problem with distance-vector routing is that any decrease in cost (good news) propagates quickly, but any increase in cost (bad news) will propagate slowly.
* For a routing protocol to work properly, if a link is broken (cost becomes infinity), every other router should be aware of it immediately, but in distance-vector routing, this takes some time. The problem is referred to as count to infinity. It sometimes takes several updates before the cost for a broken link is recorded as infinity by all routers.

**Two-Node loop**

* One example of count to infinity is the two-node loop problem. To understand the problem, let us look at the scenario depicted in Figure 3.4.4.

A screenshot of a computer

Description automatically generated

Figure 3.4.4 Two-node instability

* The figure shows a system with three nodes. We have shown only the portions of the forwarding table needed for our discussion.
* At the beginning,
  + both nodes A and B know how to reach node X.
  + But suddenly, the link between A and X fails.
  + Node A changes its table.
  + If A can send its table to B immediately, everything is fine. However, the system becomes unstable if B sends its forwarding table to A before receiving A’s forwarding table.
  + Node A receives the update and, assuming that B has found a way to reach X, immediately updates its forwarding table.
  + Now A sends its new update to B. Now B thinks that something has been changed around A and updates its forwarding table.
  + The cost of reaching X increases gradually until it reaches infinity.
  + At this moment, both A and B know that X cannot be reached.
  + However, during this time the system is not stable. Node A thinks that the route to X is via B; node B thinks that the route to X is via A. If A receives a packet destined for X, the packet goes to B and then comes back to A. Similarly, if B receives a packet destined for X, it goes to A and comes back to B. Packets bounce between A and B, creating a two-node loop problem.
* A few solutions have been proposed for instability of this kind.

***Split Horizon***

* One solution to instability is called split horizon.
* In this strategy, instead of flooding the table through each interface, each node sends only part of its table through each interface.
* If, according to its table, node B thinks that the optimum route to reach X is via A, it does not need to advertise this piece of information to A; the information has come from A (A already knows).
* Taking information from node A, modifying it, and sending it back to node A is what creates the confusion.
* In our scenario, node B eliminates the last line of its forwarding table before it sends it to A. In this case, node A keeps the value of infinity as the distance to X. Later, when node A sends its forwarding table to B, node B also corrects its forwarding table.
* The system becomes stable after the first update: both node A and node B know that X is not reachable.

***Split Horizon and Poison Reverse***

* Using the split-horizon strategy has one drawback. Normally, the corresponding protocol uses a timer, and if there is no news about a route, the node deletes the route from its table. When node B in the previous scenario eliminates the route to X from its advertisement to A, node A cannot guess whether this is due to the split-horizon strategy (the source of information was A) or because B has not received any news about X recently. In the poison reverse strategy B can still advertise the value for X, but if the source of information is A, it can replace the distance with infinity as a warning: “Do not use this value; what I know about this route comes from you.”

***Three Node Instability***

* The two-node instability can be avoided using split horizon combined with poison reverse. However, if the instability is between three nodes, stability cannot be guaranteed.

**3.4.2 Link State Routing**

* Link state routing has a different philosophy from that of distance vector routing.
* In link state routing, if each node in the domain has the entire topology of the domain the list of nodes and links, how they are connected including the type, cost (metric), and condition of the links (up or down)-the node can use Dijkstra's algorithm to build a routing table Figure 3.4.2.1 shows the concept

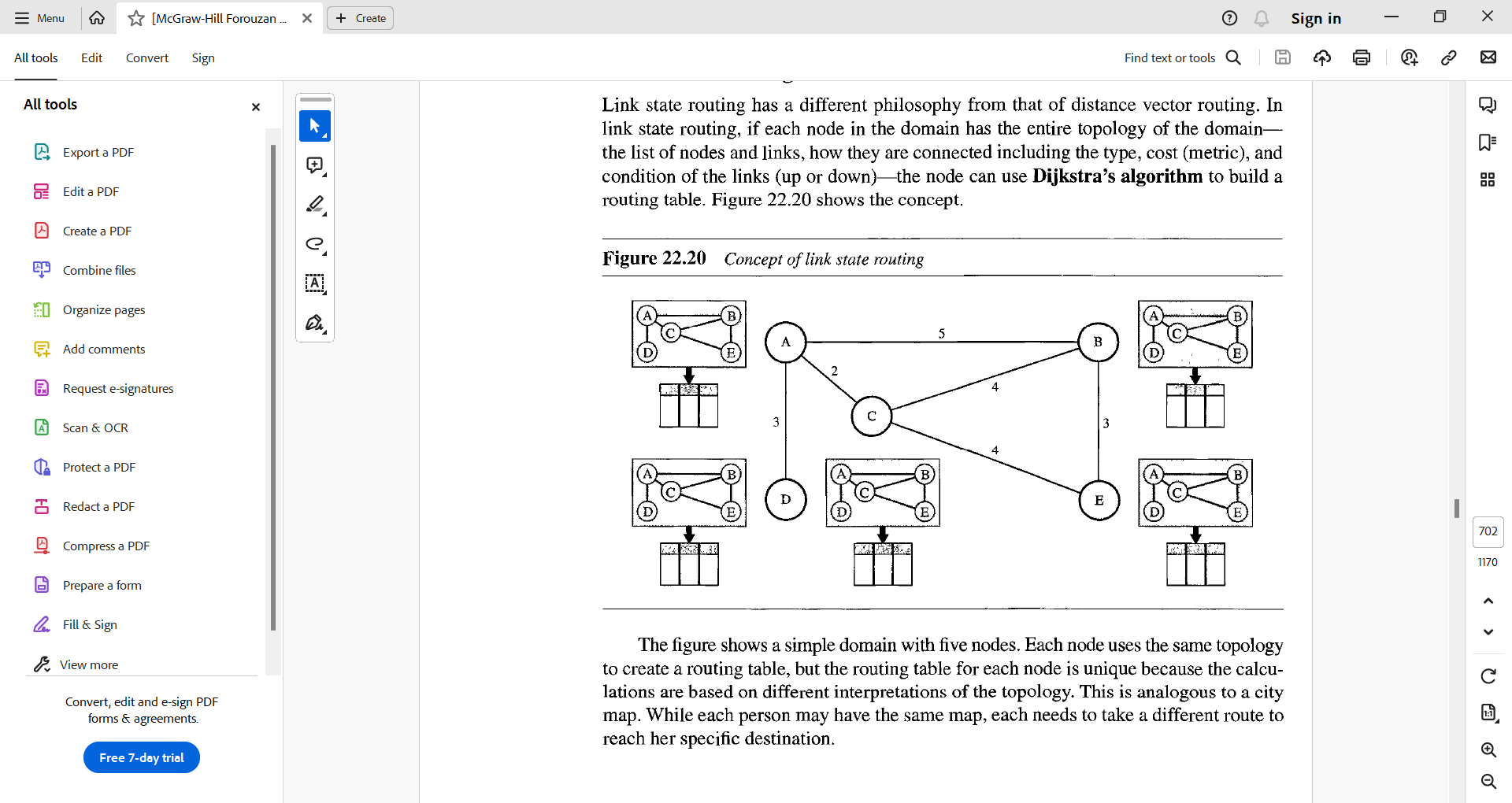


Figure 3.4.2.1 Concept of link state routing

* The figure shows a simple domain with five nodes.
* Each node uses the same topology to create a routing table, but the routing table for each node is unique because the calculations are based on different interpretations of the topology. This is analogous to a city map. While each person may have the same map, each needs to take a different route to reach her specific destination.
* The topology must be dynamic, representing the latest state of each node and each link.
* If there are changes in any point in the network (a link is down, for example), the topology must be updated for each node.
* How can a common topology be dynamic and stored in each node? No node can know the topology at the beginning or after a change somewhere in the network.
* Link state routing is based on the assumption that, although the global knowledge about the topology is not clear, each node has partial knowledge: it knows the state (type, condition, and cost) of its links. In other words, the whole topology can be compiled from the partial knowledge of each node.
* Figure 3.4.2.2 shows the same domain as in Figure 3.4.2.1, indicating the part of the knowledge belonging to each node.

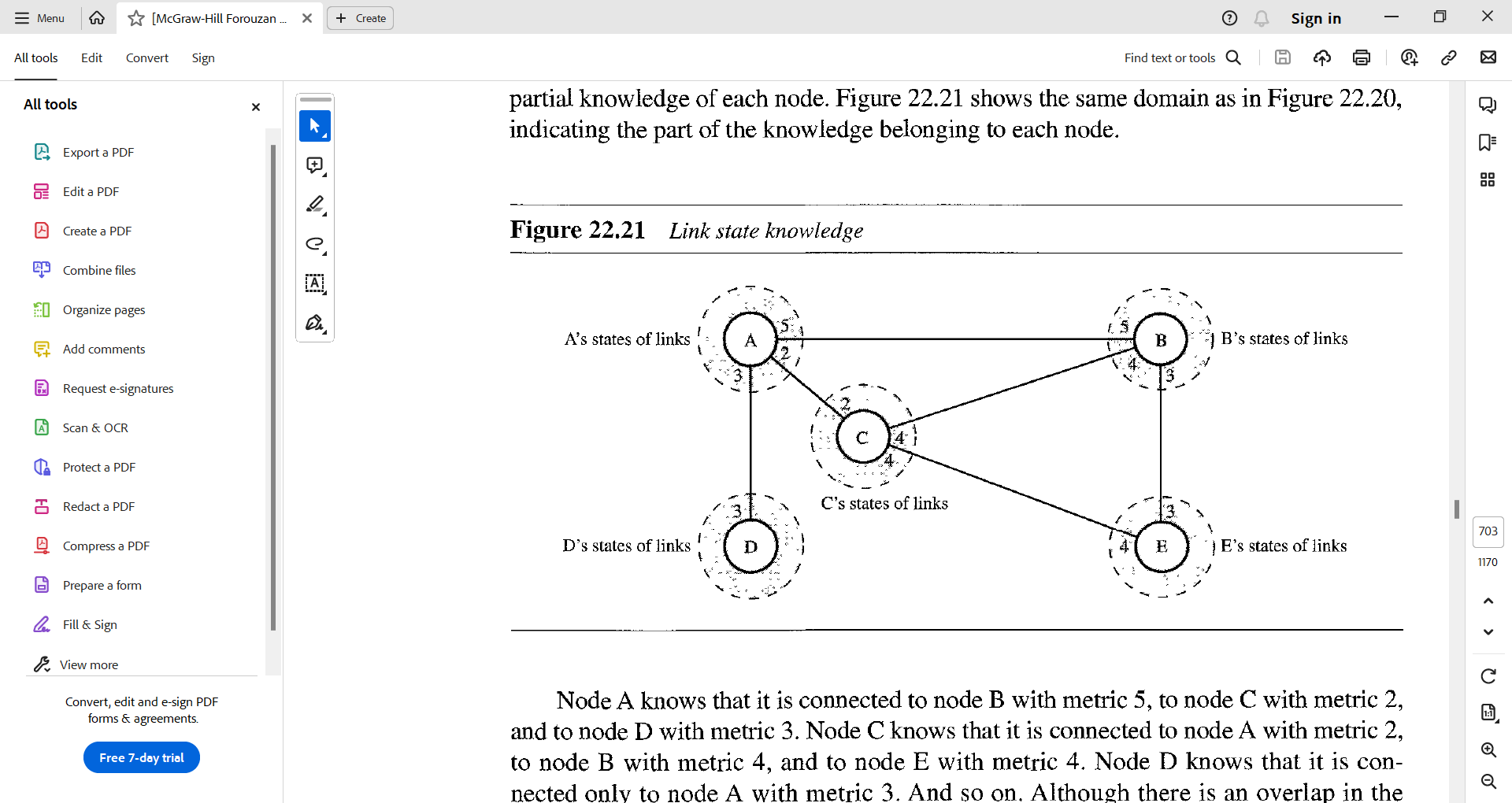


Figure 3.4.2.2 Link state knowledge

* Node A knows that it is connected to node B with metric 5, to node C with metric 2, and to node D with metric 3. Node C knows that it is connected to node A with metric 2, to node B with metric 4, and to node E with metric 4. Node D knows that it is connected only to node A with metric 3. And so on. Although there is an overlap in the knowledge, the overlap guarantees the creation of a common topology-a picture of the whole domain for each node.

***Building Routing Tables***

* In link state routing, four sets of actions are required to ensure that each node has the routing table showing the least-cost node to every other node.

1. Creation of the states of the links by each node, called the link state packet (LSP).
2. Dissemination of LSPs to every other router, called flooding, in an efficient and reliable way.
3. Formation of a shortest path tree for each node.
4. Calculation of a routing table based on the shortest path tree.

***Creation of Link State Packet (LSP)***

* A link state packet can carry a large amount of information. For the moment, however, we assume that it carries a minimum amount of data:
  + the node identity,
  + the list of links,
  + a sequence number,
  + and age.
* The first two, node identity and the list of links, are needed to make the topology.
* The third, sequence number, facilitates flooding and distinguishes new LSPs from old ones.
* The fourth, age, prevents old LSPs from remaining in the domain for a long time.
* LSPs are generated on two occasions:

1. When there is a change in the topology of the domain. Triggering of LSP dissemination is the main way of quickly informing any node in the domain to update its topology.
2. On a periodic basis. The period in this case is much longer compared to distance vector routing. As a matter of fact, there is no actual need for this type of LSP dissemination. It is done to ensure that old information is removed from the domain. The timer set for periodic dissemination is normally in the range of 60 min or 2 h based on the implementation. A longer period ensures that flooding does not create too much traffic on the network.

***Flooding of LSPs***

* After a node has prepared an LSP, it must be disseminated to all other nodes, not only to its neighbors. The process is called flooding and based on the following:

1. The creating node sends a copy of the LSP out of each interface.
2. A node that receives an LSP compares it with the copy it may already have. If the newly arrived LSP is older than the one it has (found by checking the sequence number), it discards the LSP. If it is newer, the node does the following:
   1. It discards the old LSP and keeps the new one.
   2. It sends a copy of it out of each interface except the one from which the packet arrived. This guarantees that flooding stops somewhere in the domain (where a node has only one interface).

Formation of Shortest Path Tree: Dijkstra Algorithm

* After receiving all LSPs, each node will have a copy of the whole topology. However, the topology is not sufficient to find the shortest path to every other node; a shortest path tree is needed.
* A tree is a graph of nodes and links; one node is called the root. All other nodes can be reached from the root through only one single route.
* A shortest path tree is a tree in which the path between the root and every other node is the shortest. What we need for each node is a shortest path tree with that node as the root.
* The Dijkstra algorithm creates a shortest path tree from a graph. The algorithm divides the nodes into two sets: tentative and permanent. It finds the neighbors of a current node, makes them tentative, examines them, and if they pass the criteria, makes them permanent.
* We can informally define the algorithm by using the flowchart in Figure 3.4.2.3.

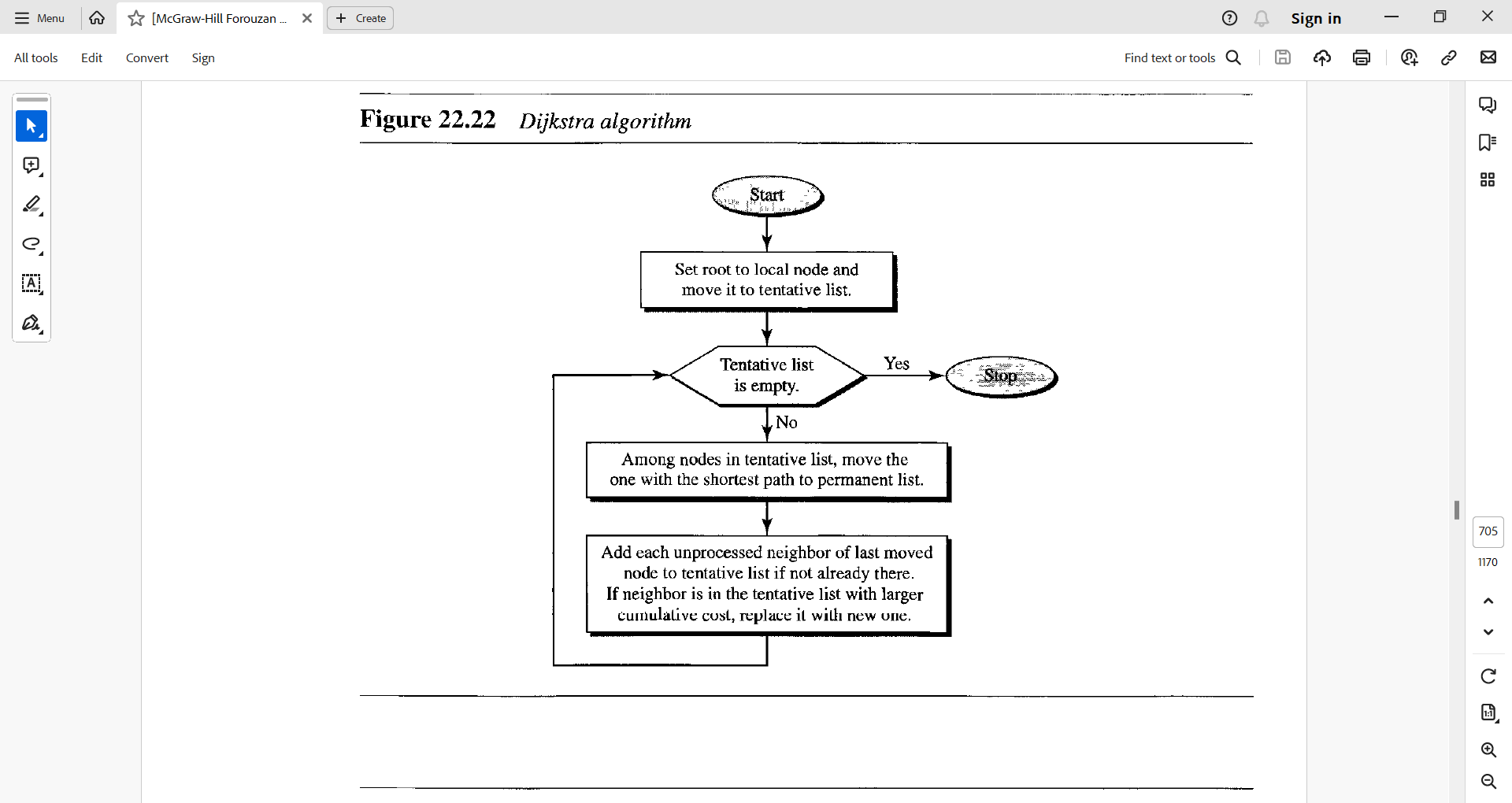


Figure 3.4.2.3 Dijkstra Algorithm

* Let us apply the algorithm to node A of our sample graph in Figure 3.4.2.4. To find the shortest path in each step, we need the cumulative cost from the root to each node, which is shown next to the node.

A screenshot of a computer

Description automatically generated

Figure 3.4.2.4 Example of formation of shortest path tree

* The following shows the steps. At the end of each step, we show the permanent (filled circles) and the tentative (open circles) nodes and lists with the cumulative costs.

1. We make node A the root of the tree and move it to the tentative list. Our two lists are

Permanent list: empty

Tentative list: A(O)

1. Node A has the shortest cumulative cost from all nodes in the tentative list. We move A to the permanent list and add all neighbors of A to the tentative list. Our new lists are

Permanent list: A(O)

Tentative list: B(5), C(2), D(3)

1. Node C has the shortest cumulative cost from all nodes in the tentative list. We move C to the permanent list. Node C has three neighbors, but node A is already processed, which makes the unprocessed neighbors just B and E. However, B is already in the tentative list with a cumulative cost of 5. Node A could also reach node B through C with a cumulative cost of 6 Since 5 is less than 6, we keep node B with a cumulative cost of 5 in the tentative list and do not replace it. Our new lists are

Permanent list· A(Q), C(2)

Tentative list· B(5), D(3), E(6)

1. Node D has the shortest cumulative cost of all the nodes in the tentative list. We move D to the permanent list Node D has no unprocessed neighbor to be added to the tentative list. Our new lists are

Permanent list· A(Q), C(2), D(3)

Tentative list: B(5), E(6)

1. Node B has the shortest cumulative cost of all the nodes in the tentative list. We move B to the permanent list. We need to add all unprocessed neighbors of B to the tentative list (this is just node E). However, E(6) is already in the list with a smaller cumulative cost. The cumulative cost to node E, as the neighbor of B, is 8. We keep node E(6) in the tentative list. Our new lists are

Permanent list: A(O), B(S), C(2), D(3)

Tentative list: E(6)

1. Node E has the shortest cumulative cost from all nodes in the tentative list. We move E to the permanent list. Node E has no neighbor. Now the tentative list is empty. We stop; our shortest path tree is ready. The final lists are

Permanent list: A(O), B(S), C(2), D(3), E(6)

Tentative list: empty

Calculation of Routing Table from Shortest Path Tree

* Each node uses the shortest path tree protocol to construct its routing table.
* The routing table shows the cost of reaching each node from the root.
* Table 3.4.2.1 shows the routing table for node A.

A screenshot of a computer

Description automatically generated

Table 3.4.2.1 Routing Table for Node A

* Compare Table 3.4.2.1 with the one in Figure 3.4.1. Both distance vector routing and link state routing end up with the same routing table for node A.

3.4.3 Path Vector Routing

* Distance vector and link state routing are both intradomain routing protocols.
* They can be used inside an autonomous system, but not between autonomous systems.
* These two protocols are not suitable for interdomain routing mostly because of scalability. Both of these routing protocols become intractable when the domain of operation becomes large.
* Distance vector routing is subject to instability if there are more than a few hops in the domain of operation.
* Link state routing needs a huge amount of resources to calculate routing tables. It also creates heavy traffic because of flooding.
* There is a need for a third routing protocol which we call path vector routing.
* Path vector routing proved to be useful for interdomain routing.
* The principle of path vector routing is similar to that of distance vector routing. In path vector routing, we assume that there is one node (there can be more, but one is enough for our conceptual discussion) in each autonomous system that acts on behalf of the entire autonomous system. Let us call it the speaker node.
* The speaker node in an AS creates a routing table and advertises it to speaker nodes in the neighboring ASs.
* The idea is the same as for distance vector routing except that only speaker nodes in each AS can communicate with each other. However, what is advertised is different. A speaker node advertises the path, not the metric of the nodes, in its autonomous system or other autonomous systems.

3.5 Unicast Routing Protocols

* In the previous section, we discussed unicast routing algorithms; in this section, we discuss unicast routing protocols used in the Internet. Although three protocols we discuss here are based on the corresponding algorithms we discussed before, a protocol is more than an algorithm.
* A protocol needs to define its domain of operation, the messages exchanged, communication between routers, and interaction with protocols in other domains.
* After an introduction, we discuss three common protocols used in the Internet:
  + Routing Information Protocol (RIP), based on the distance-vector algorithm,
  + Open Shortest Path First (OSPF), based on the link-state algorithm,
  + and Border Gateway Protocol (BGP), based on the path-vector algorithm.

3.5.1 Internet Structure

* Before discussing unicast routing protocols, we need to understand the structure of today’s Internet.
* The Internet has changed from a tree-like structure, with a single backbone, to a multi-backbone structure run by different private corporations today.
* Although it is difficult to give a general view of the Internet today, we can say that the Internet has a structure similar to what is shown in Figure 3.5.1.1

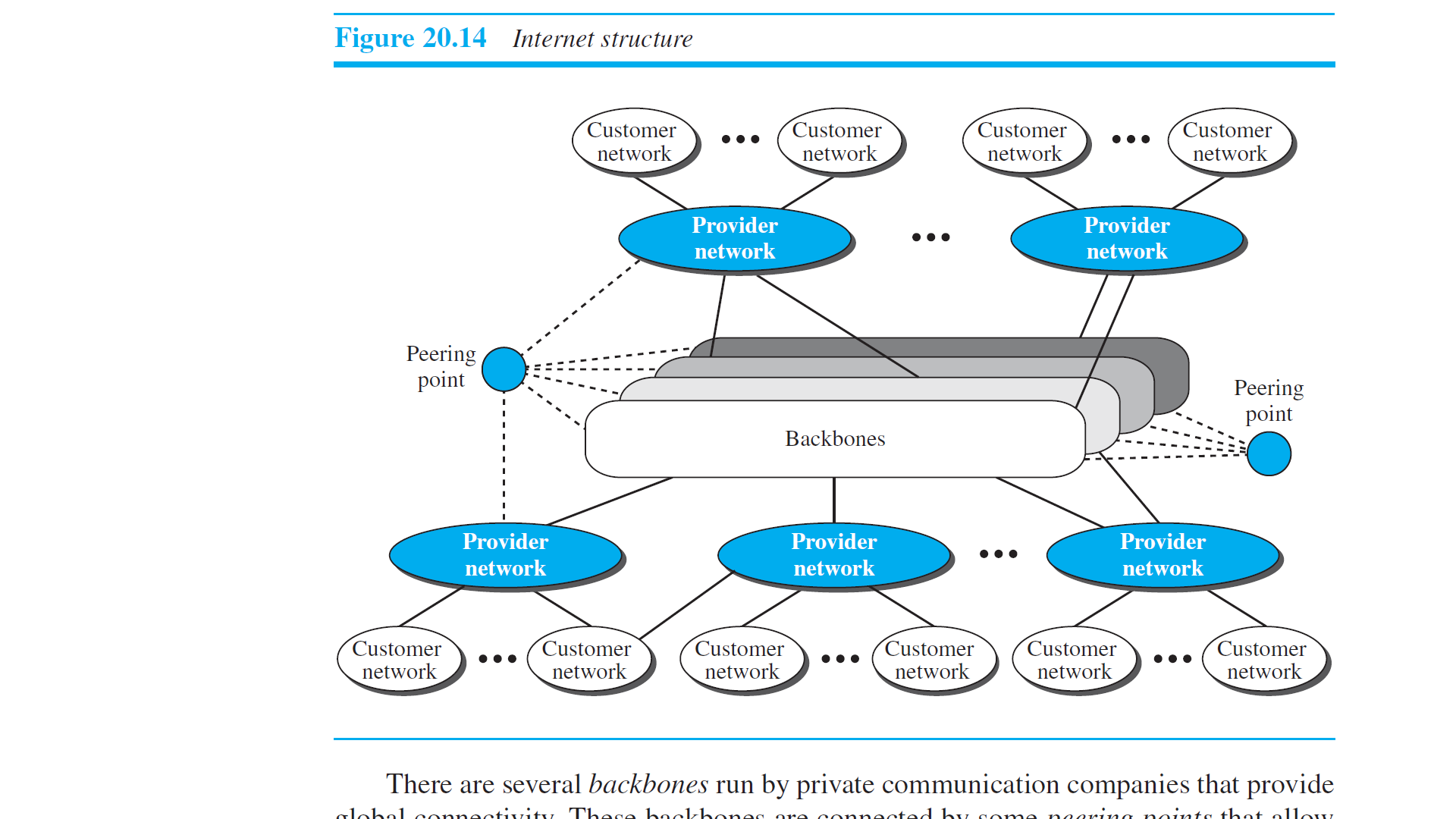


Figure 3.5.1.1 Internet Structure

* There are several backbones run by private communication companies that provide global connectivity.
* These backbones are connected by some peering points that allow connectivity between backbones.
* At a lower level, there are some provider networks that use the backbones for global connectivity but provide services to Internet customers.
* Finally, there are some customer networks that use the services provided by the provider networks.
* Any of these three entities (backbone, provider network, or customer network) can be called an Internet Service Provider or ISP. They provide services, but at different levels.

***Hierarchical Routing***

* The Internet today is made of a huge number of networks and routers that connect them.
* It is obvious that routing in the Internet cannot be done using a single protocol for two reasons:
  + a scalability problem
    - Scalability problem means that the size of the forwarding tables becomes huge, searching for a destination in a forwarding table becomes time-consuming, and updating creates a huge amount of traffic.
  + An administrative issue.
    - The administrative issue is related to the Internet structure described in Figure 3.5.1.1. As the figure shows, each ISP is run by an administrative authority.
    - The administrator needs to have control in its system. The organization must be able to use as many subnets and routers as it needs, may desire that the routers be from a particular manufacturer, may wish to run a specific routing algorithm to meet the needs of the organization, and may want to impose some policy on the traffic passing through its ISP.
* Hierarchical routing means considering each ISP as an autonomous system (AS).
* Each AS can run a routing protocol that meets its needs, but the global Internet runs a global protocol to glue all ASs together.
* The routing protocol run in each AS is referred to as intra-AS routing protocol, intradomain routing protocol, or interior gateway protocol (IGP); the global routing protocol is referred to as inter-AS routing protocol, interdomain routing protocol, or exterior gateway protocol (EGP).
* We can have several intradomain routing protocols, and each AS is free to choose one, but it should be clear that we should have only one interdomain protocol that handles routing between these entities.
* Presently, the two common intradomain routing protocols are RIP and OSPF; the only interdomain routing protocol is BGP. The situation may change when we move to IPv6.

**Autonomous Systems**

* As we said before, each ISP is an autonomous system when it comes to managing networks and routers under its control.
* Although we may have small, medium-size, and large ASs, each AS is given an autonomous number (ASN) by the ICANN.
* Each ASN is a 16-bit unsigned integer that uniquely defines an AS.
* The autonomous systems, however, are not categorized according to their size; they are categorized according to the way they are connected to other ASs.
* We have stub ASs, multihomed ASs, and transient ASs.
* The type, as we see will later, affects the operation of the interdomain routing protocol in relation to that AS.
  + Stub AS.
    - A stub AS has only one connection to another AS.
    - The data traffic can be either initiated or terminated in a stub AS; the data cannot pass through it. A good example of a stub AS is the customer network, which is either the source or the sink of data.
  + Multihomed AS.
    - A multihomed AS can have more than one connection to other ASs, but it does not allow data traffic to pass through it.
    - A good example of such an AS is some of the customer ASs that may use the services of more than one provider network, but their policy does not allow data to be passed through them.
  + Transient AS.
    - A transient AS is connected to more than one other AS and also allows the traffic to pass through.
    - The provider networks and the backbone are good examples of transient ASs.

**3.5.2 Routing Information Protocol**

* The Routing Information Protocol (RIP) is one of the most widely used intradomain routing protocols based on the distance-vector routing algorithm we described earlier.
* RIP was started as part of the Xerox Network System (XNS), but it was the Berkeley Software Distribution (BSD) version of UNIX that helped make the use of RIP widespread.

***Hop Count***

* A router in this protocol basically implements the distance-vector routing algorithm.
* First, since a router in an AS needs to know how to forward a packet to different networks (subnets) in an AS, RIP routers advertise the cost of reaching different networks instead of reaching other nodes in a theoretical graph. In other words, the cost is defined between a router and the network in which the destination host is located.
* Second, to make the implementation of the cost simpler (independent from performance factors of the routers and links, such as delay, bandwidth, and so on), the cost is defined as the number of hops, which means the number of networks (subnets) a packet needs to travel through from the source router to the final destination host. Note that the network in which the source host is connected is not counted in this calculation because the source host does not use a forwarding table; the packet is delivered to the default router. Figure 3.5.2.1 shows the concept of hop count advertised by three routers from a source host to a destination host.
* In RIP, the maximum cost of a path can be 15, which means 16 is considered as infinity (no connection). For this reason, RIP can be used only in autonomous systems in which the diameter of the AS is not more than 15 hops.

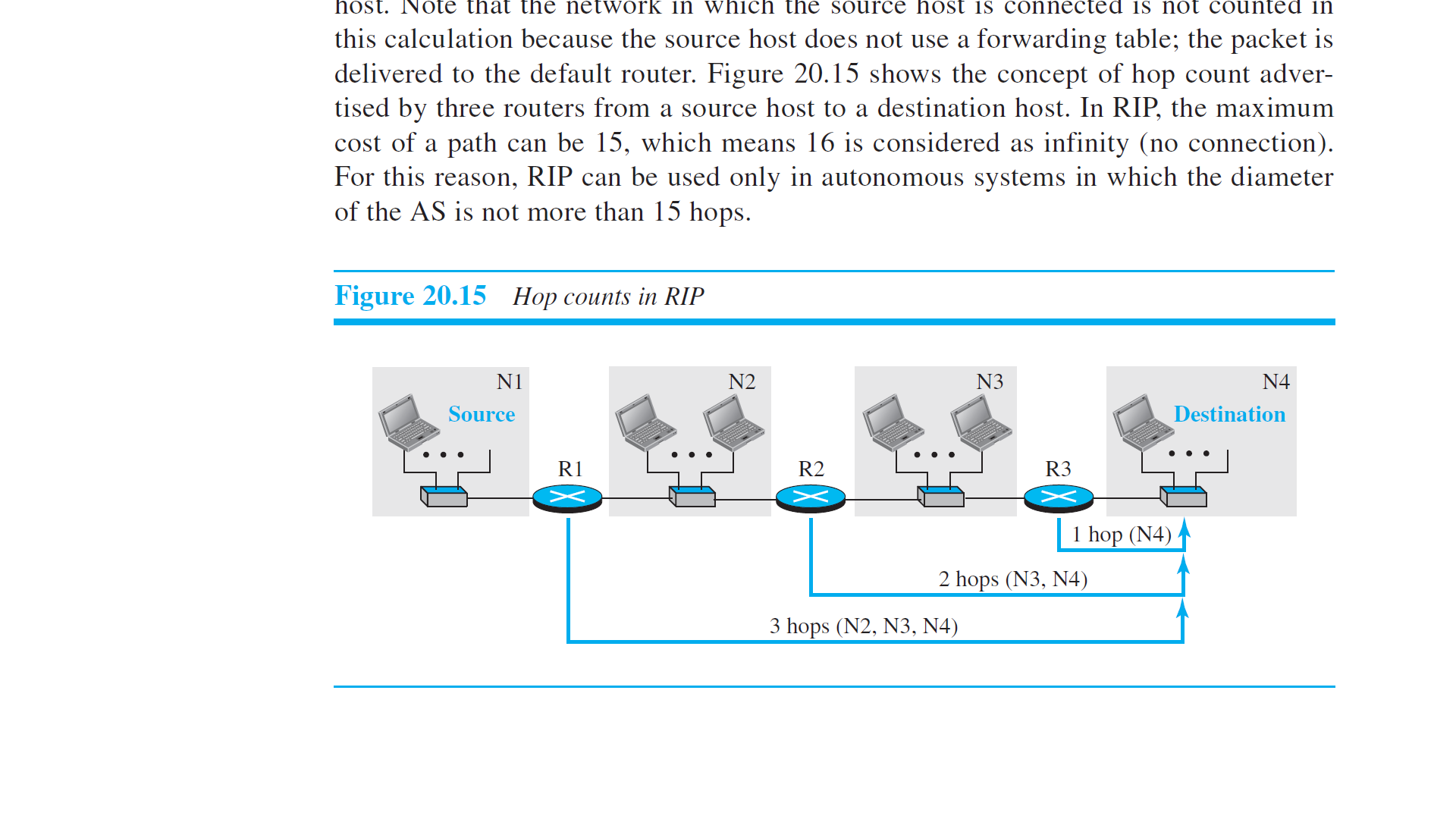


Figure 3.5.2.1 Hop counts in RIP

***Forwarding Tables***

* Although the distance-vector algorithm we discussed in the previous section is concerned with exchanging distance vectors between neighboring nodes, the routers in an autonomous system need to keep forwarding tables to forward packets to their destination networks.
* A forwarding table in RIP is a three-column table in which the first column is the address of the destination network, the second column is the address of the next router to which the packet should be forwarded, and the third column is the cost (the number of hops) to reach the destination network.
* Figure 3.5.2.2 shows the three forwarding tables for the routers in Figure 3.5.2.1. Note that the first and the third columns together convey the same information as does a distance vector, but the cost shows the number of hops to the destination networks.

A screenshot of a computer

Description automatically generated

Figure 3.5.2.2 Forwarding Tables

* Although a forwarding table in RIP defines only the next router in the second column, it gives the information about the whole least-cost tree based on the second property of these trees, discussed in the previous section.
* For example, R1 defines that the next router for the path to N4 is R2; R2 defines that the next router to N4 is R3; R3 defines that there is no next router for this path. The tree is then R1 → R2 → R3 → N4.
* A question often asked about the forwarding table is what the use of the third column is. The third column is not needed for forwarding the packet, but it is needed for updating the forwarding table when there is a change in the route, as we will see shortly.

***RIP Implementation***

* RIP is implemented as a process that uses the service of UDP on the well-known port number 520.
* In BSD, RIP is a daemon process (a process running in the background), named routed (abbreviation for route daemon and pronounced route-dee). This means that, although RIP is a routing protocol to help IP route its datagrams through the AS, the RIP messages are encapsulated inside UDP user datagrams, which in turn are encapsulated inside IP datagrams. In other words, RIP runs at the application layer, but creates forwarding tables for IP at the network later.
* RIP has gone through two versions: RIP-1 and RIP-2. The second version is backward compatible with the first section; it allows the use of more information in the RIP messages that were set to 0 in the first version. We discuss only RIP-2 in this section.

***RIP Messages***

* Two RIP processes, a client and a server, like any other processes, need to exchange messages.
* RIP-2 defines the format of the message, as shown in Figure 3.5.2.3.
* Part of the message, which we call entry, can be repeated as needed in a message.
* Each entry carries the information related to one line in the forwarding table of the router that sends the message.

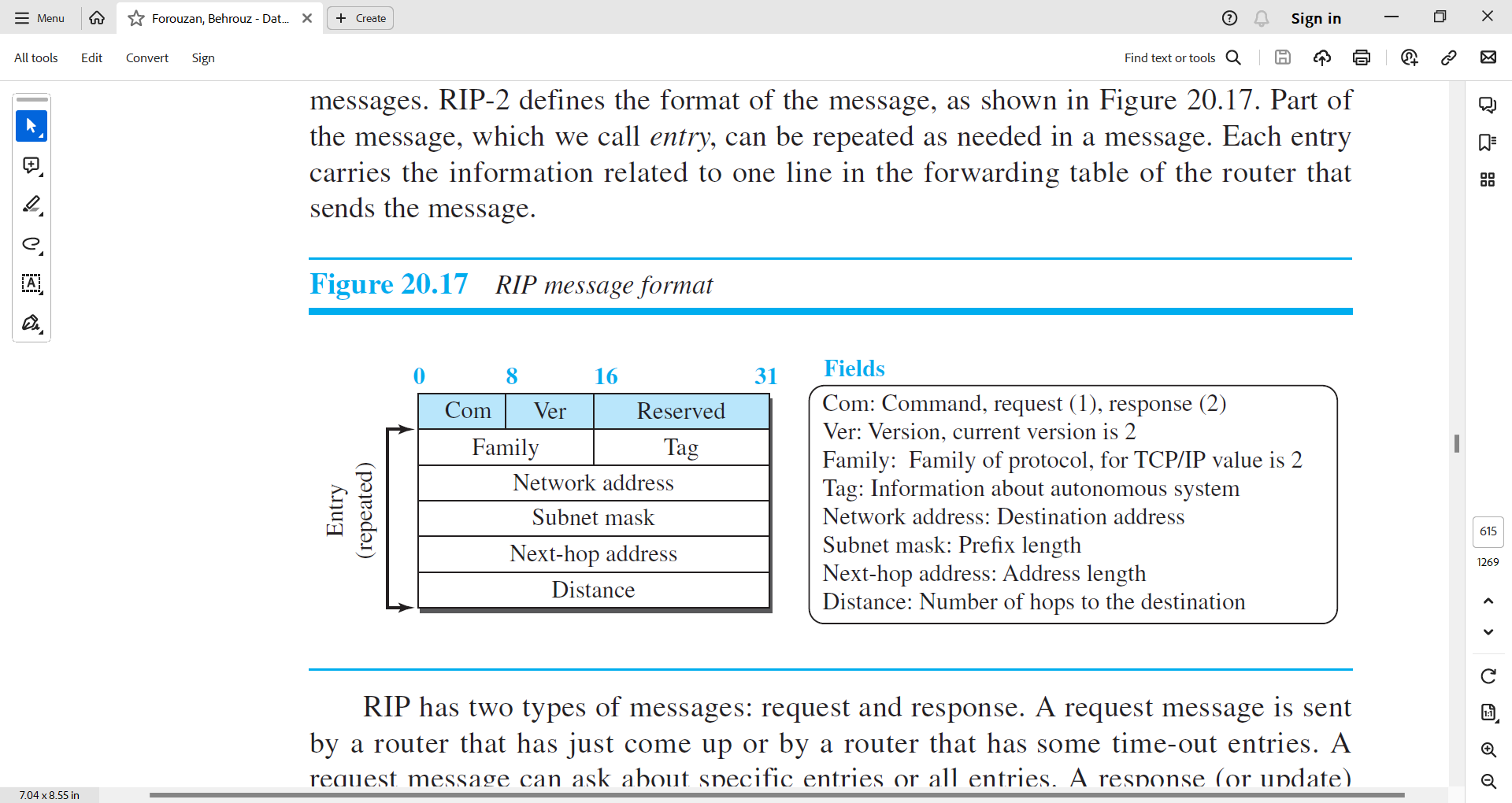


Figure 3.5.2.3 RIP Message Format

* RIP has two types of messages: request and response.
* A request message is sent by a router that has just come up or by a router that has some time-out entries. A request message can ask about specific entries or all entries.
* A response (or update) message can be either solicited or unsolicited. A solicited response message is sent only in answer to a request message. It contains information about the destination specified in the corresponding request message. An unsolicited response message, on the other hand, is sent periodically, every 30 seconds or when there is a change in the forwarding table.

***RIP Algorithm***

* RIP implements the same algorithm as the distance-vector routing algorithm we discussed in the previous section.
* However, some changes need to be made to the algorithm to enable a router to update its forwarding table:
  + Instead of sending only distance vectors, a router needs to send the whole contents of its forwarding table in a response message.
  + The receiver adds one hop to each cost and changes the next router field to the address of the sending router.
  + We call each route in the modified forwarding table **the received route** and each route in the old forwarding table **the old route**. The received router selects the old routes as the new ones except in the following three cases:
    - If the received route does not exist in the old forwarding table, it should be added to the route.
    - If the cost of the received route is lower than the cost of the old one, the received route should be selected as the new one.
    - If the cost of the received route is higher than the cost of the old one, but the value of the next router is the same in both routes, the received route should be selected as the new one. This is the case where the route was actually advertised by the same router in the past, but now the situation has been changed. *For example, suppose a neighbor has previously advertised a route to a destination with cost 3, but now there is no path between this neighbor and that destination. The neighbor advertises this destination with cost value infinity (16 in RIP). The receiving router must not ignore this value even though its old route has a lower cost to the same destination.*
  + The new forwarding table needs to be sorted according to the destination route (mostly using the longest prefix first).

***Timers in RIP***

* RIP uses three timers to support its operation.
  + **The periodic timer** 
    - It controls the advertising of regular update messages.
    - Each router has one periodic timer that is randomly set to a number between 25 and 35 seconds (to prevent all routers sending their messages at the same time and creating excess traffic).
    - The timer counts down; when zero is reached, the update message is sent, and the timer is randomly set once again.
  + **The expiration timer**
    - It governs the validity of a route.
    - When a router receives update information for a route, the expiration timer is set to 180 seconds for that particular route.
    - Every time a new update for the route is received, the timer is reset. If there is a problem on an internet and no update is received within the allotted 180 seconds, the route is considered expired and the hop count of the route is set to 16, which means the destination is unreachable.
    - Every route has its own expiration timer.
  + **The garbage collection timer**
    - It is used to purge a route from the forwarding table.
    - When the information about a route becomes invalid, the router does not immediately purge that route from its table. Instead, it continues to advertise the route with a metric value of 16.
    - At the same time, a garbage collection timer is set to 120 seconds for that route. When the count reaches zero, the route is purged from the table.
    - This timer allows neighbors to become aware of the invalidity of a route prior to purging.

***Performance***

* Update Messages.
  + The update messages in RIP have a very simple format and are sent only to neighbors; they are local.
  + They do not normally create traffic because the routers try to avoid sending them at the same time.
* Convergence of Forwarding Tables.
  + RIP uses the distance-vector algorithm, which can converge slowly if the domain is large, but, since RIP allows only 15 hops in a domain (16 is considered as infinity), there is normally no problem in convergence.
  + The only problems that may slow down convergence are count-to-infinity and loops created in the domain; use of poison-reverse and split-horizon strategies added to the RIP extension may alleviate the situation.
* Robustness.
  + Distance-vector routing is based on the concept that each router sends what it knows about the whole domain to its neighbors. This means that the calculation of the forwarding table depends on information received from immediate neighbors, which in turn receive their information from their own neighbors.
  + If there is a failure or corruption in one router, the problem will be propagated to all routers and the forwarding in each router will be affected.

**Example**

* Figure 3.5.2.4 shows a more realistic example of the operation of RIP in an autonomous system.
  + First, the figure shows all forwarding tables after all routers have been booted.
  + Then we show changes in some tables when some update messages have been exchanged.
  + Finally, we show the stabilized forwarding tables when there is no more change.

A screenshot of a computer program

Description automatically generated



Figure 3.5.2.4 Example of autonomous system using RIP.

3.5.3 Open Shortest Path First

* Open Shortest Path First (OSPF) is also an intradomain routing protocol like RIP, but it is based on the link-state routing protocol we described earlier in the chapter.
* OSPF is an open protocol, which means that the specification is a public document.( <https://www.rfc-editor.org/rfc/rfc2328.html>)

Metric

* In OSPF, like RIP, the cost of reaching a destination from the host is calculated from the source router to the destination network. However, each link (network) can be assigned a weight based on the throughput, round-trip time, reliability, and so on. An administration can also decide to use the hop count as the cost.
* An interesting point about the cost in OSPF is that different service types (TOSs) can have different weights as the cost.
* Figure 3.5.3.1 shows the idea of the cost from a router to the destination host network. We can compare the figure with Figure 3.5.2.1 for the RIP.

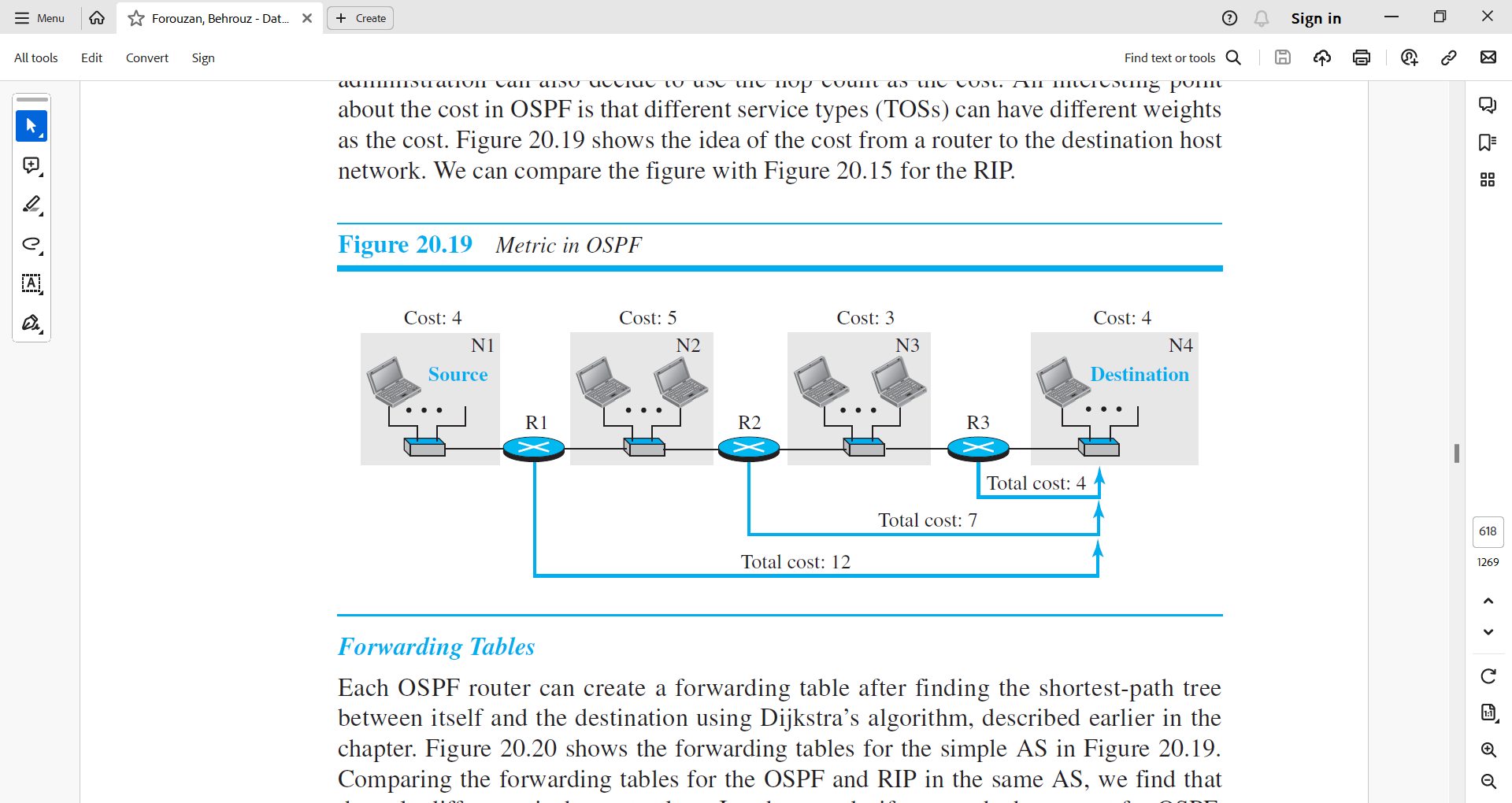


Figure 3.5.3.1 Metric in OSPF

Forwarding Tables

* Each OSPF router can create a forwarding table after finding the shortest-path tree between itself and the destination using Dijkstra’s algorithm, described earlier in the chapter.
* Figure 3.5.3.2 shows the forwarding tables for the simple AS in Figure 3.5.3.1.
* Comparing the forwarding tables for the OSPF and RIP in the same AS, we find that the only difference is the cost values. In other words, if we use the hop count for OSPF, the tables will be exactly the same.
* The reason for this consistency is that both protocols use the shortest-path trees to define the best route from a source to a destination.

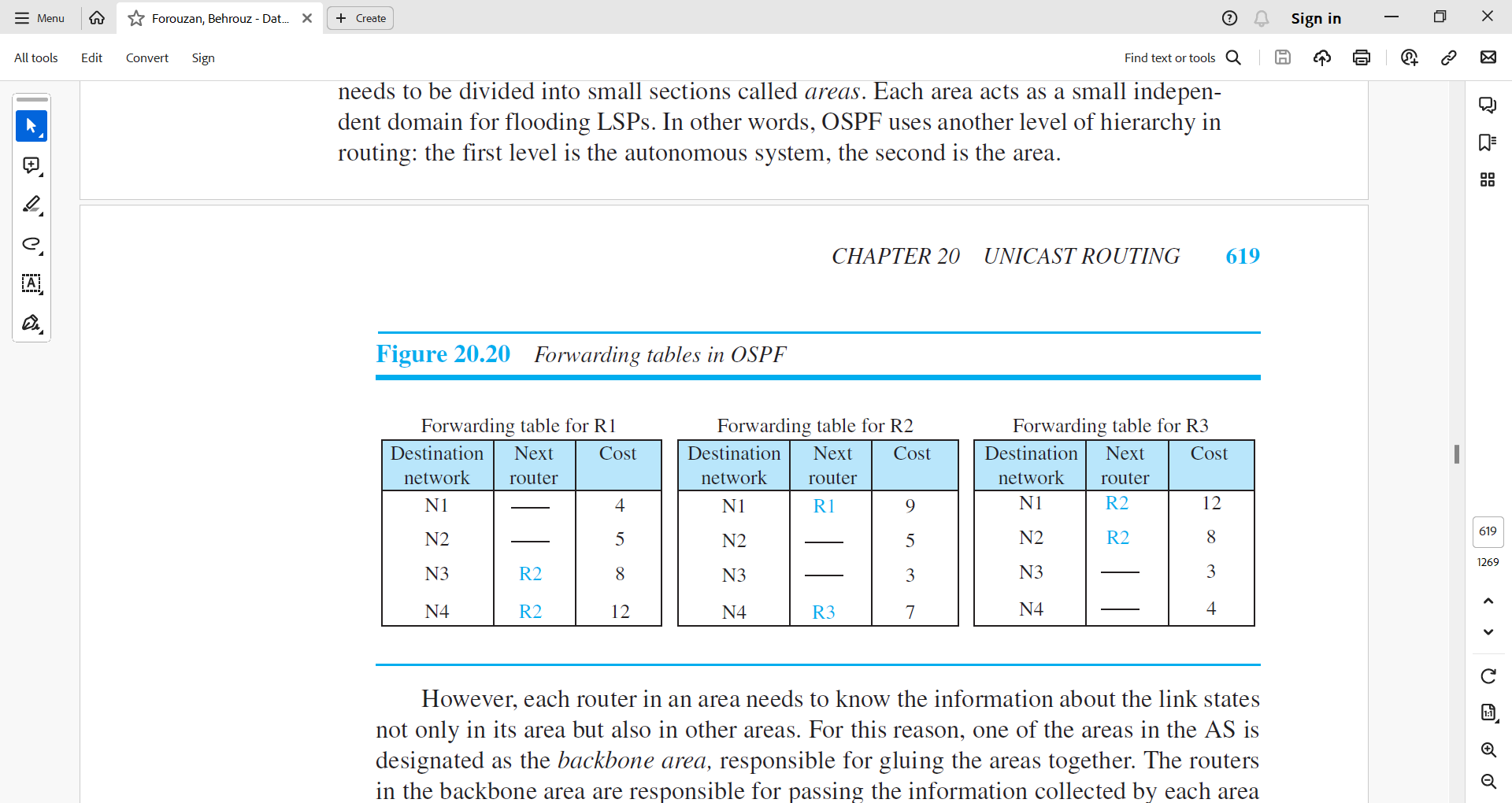


Figure 3.5.3.2 Forwarding tables in OSPF

Areas

* Compared with RIP, which is normally used in small ASs, OSPF was designed to be able to handle routing in a small or large autonomous system.
* However, the formation of shortest-path trees in OSPF requires that all routers flood the whole AS with their LSPs to create the global LSDB. Although this may not create a problem in a small AS, it may have created a huge volume of traffic in a large AS.
* To prevent this, the AS needs to be divided into small sections called areas. Each area acts as a small independent domain for flooding LSPs. In other words, OSPF uses another level of hierarchy in routing: the first level is the autonomous system, the second is the area.
* However, each router in an area needs to know the information about the link states not only in its area but also in other areas.
* For this reason, one of the areas in the AS is designated as the backbone area, responsible for gluing the areas together. The routers in the backbone area are responsible for passing the information collected by each area to all other areas.
* In this way, a router in an area can receive all LSPs generated in other areas. For the purpose of communication, each area has an area identification. The area identification of the backbone is zero.
* Figure 3.5.3.3 shows an autonomous system and its areas.

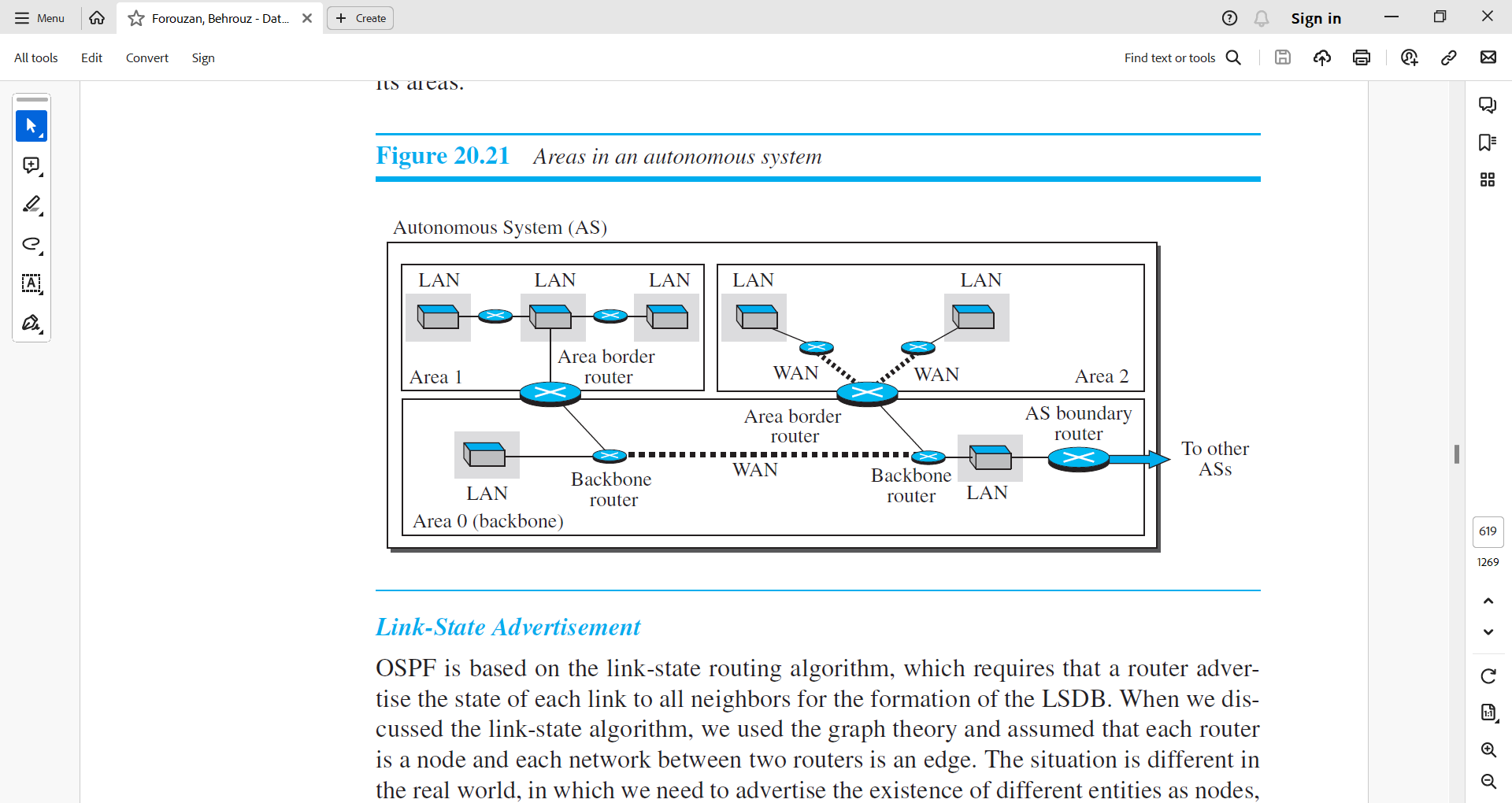


Figure 3.5.3.3 Areas in an autonomous system

Link-state advertisements

* OSPF is based on the link-state routing algorithm, which requires that a router advertise the state of each link to all neighbors for the formation of the LSDB.
* When we discussed the link-state algorithm, we used the graph theory and assumed that each router is a node and each network between two routers is an edge.
* The situation is different in the real world, in which we need to advertise the existence of different entities as nodes, the different types of links that connect each node to its neighbors, and the different types of cost associated with each link. This means we need different types of advertisements, each capable of advertising different situations.
* We can have five types of link-state advertisements:
  + router link,
  + network link,
  + summary link to network,
  + summary link to AS border router,
  + and external link.
* Figure 3.5.3.4 shows these five advertisements and their uses.

A screenshot of a computer

Description automatically generated

Figure 3.5.3.4 Five different LSPs

* Router link.
  + A router link advertises the existence of a router as a node.
  + In addition to giving the address of the announcing router, this type of advertisement can define one or more types of links that connect the advertising router to other entities.
  + A transient link announces a link to a transient network, a network that is connected to the rest of the networks by one or more routers. This type of advertisement should define the address of the transient network and the cost of the link.
  + A stub link advertises a link to a stub network, a network that is not a through network. Again, the advertisement should define the address of the network and the cost.
  + A point-to-point link should define the address of the router at the end of the point-to-point line and the cost to get there.
* Network link.
  + A network link advertises the network as a node.
  + However, since a network cannot do announcements itself (it is a passive entity), one of the routers is assigned as the designated router and does the advertising.
  + In addition to the address of the designated router, this type of LSP announces the IP address of all routers (including the designated router as a router and not as speaker of the network), but no cost is advertised because each router announces the cost to the network when it sends a router link advertisement.
* Summary link to network.
  + This is done by an area border router; it advertises the summary of links collected by the backbone to an area or the summary of links collected by the area to the backbone. As we discussed earlier, this type of information exchange is needed to glue the areas together.
* Summary link to AS.
  + This is done by an AS router that advertises the summary links from other ASs to the backbone area of the current AS, information which later can be disseminated to the areas so that they will know about the networks in other ASs. The need for this type of information exchange is better understood when we discuss inter-AS routing (BGP).
* External link.
  + This is also done by an AS router to announce the existence of a single network outside the AS to the backbone area to be disseminated into the areas.

OSPF Implementation

* OSPF is implemented as a program in the network layer, using the service of the IP for propagation.
* An IP datagram that carries a message from OSPF sets the value of the protocol field to 89. This means that, although OSPF is a routing protocol to help IP to route its datagrams inside an AS, the OSPF messages are encapsulated inside datagrams.
* OSPF has gone through two versions: version 1 and version 2.
* Most implementations use version 2.

OSPF Messages

* OSPF is a very complex protocol; it uses five different types of messages.
* In Figure 3.5.3.5 , we first show the format of the OSPF common header (which is used in all messages) and the link-state general header (which is used in some messages).
* We then give the outlines of five message types used in OSPF.
  + The hello message (type 1) is used by a router to introduce itself to the neighbors and announce all neighbors that it already knows.
  + The database description message (type 2) is normally sent in response to the hello message to allow a newly joined router to acquire the full LSDB.
  + The link state request message (type 3) is sent by a router that needs information about a specific LS.
  + The link-state update message (type 4) is the main OSPF message used for building the LSDB. This message, in fact, has five different versions (router link, network link, summary link to network, summary link to AS border router, and external link), as we discussed before.
  + The link-state acknowledgment message (type 5) is used to create reliability in OSPF; each router that receives a link-state update message needs to acknowledge it.

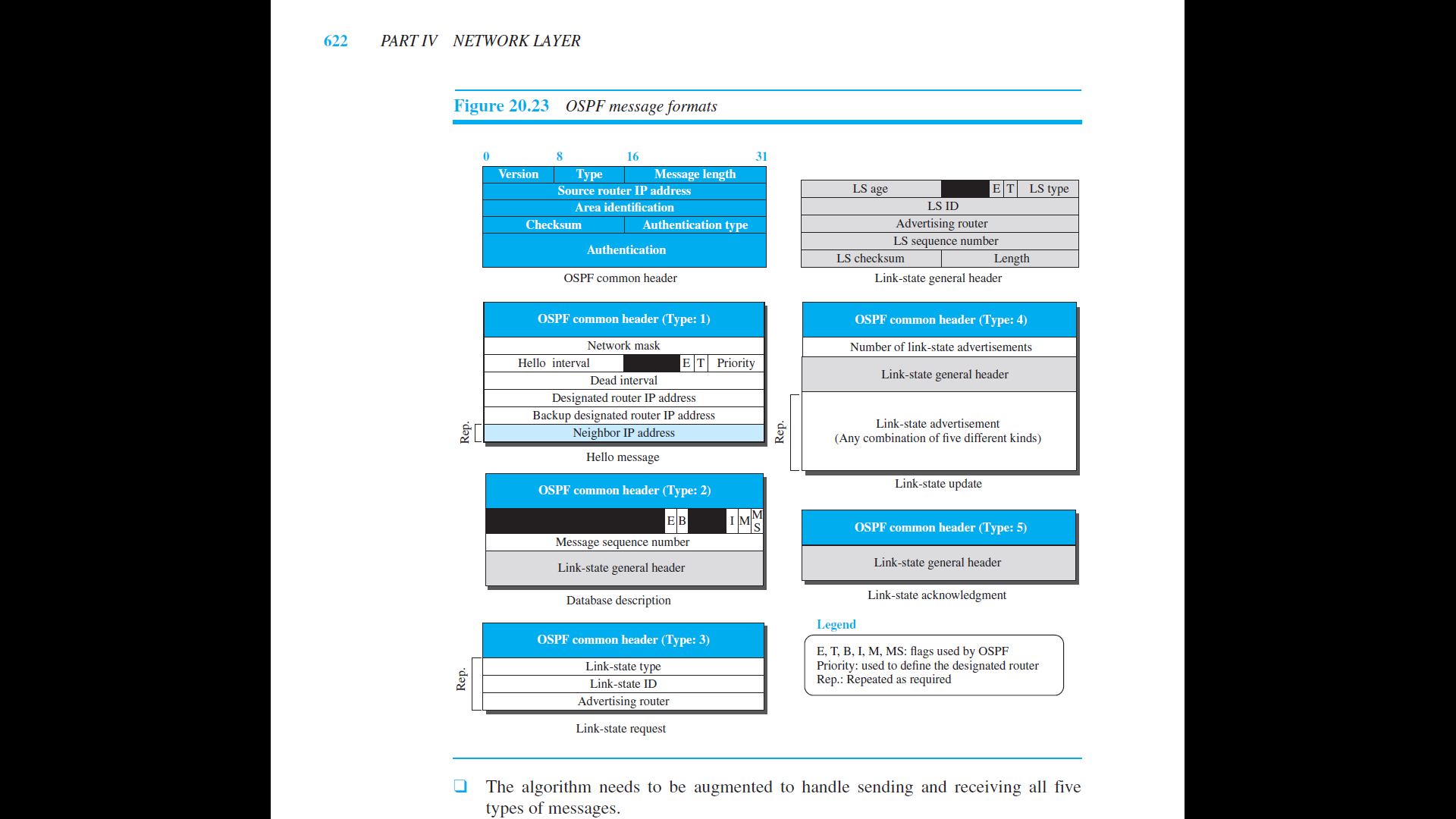


Figure 3.5.3.5 OSPF Message Formats

Authentication

* OSPF common header has the provision of Authentication in its common header.
* This prevents a malicious entity from sending OSPF messages to a router and causing the router to become part of the routing system to which it actually does not belong.

OSPF Algorithm

* OSPF implements the link-state routing algorithm we discussed in the previous section. However, some changes and augmentations need to be added to the algorithm:
  + After each router has created the shortest-path tree, the algorithm needs to use it to create the corresponding routing algorithm.
  + The algorithm needs to be augmented to handle sending and receiving all five types of messages.

Performance

* Update Messages.
  + The link-state messages in OSPF have a somewhat complex format.
  + They also are flooded to the whole area. If the area is large, these messages may create heavy traffic and use a lot of bandwidth.
* Convergence of Forwarding Tables.
  + When the flooding of LSPs is completed, each router can create its own shortest-path tree and forwarding table; convergence is fairly quick. However, each router needs to run Dijkstra’s algorithm, which may take some time.
* Robustness.
  + The OSPF protocol is more robust than RIP because, after receiving the completed LSDB, each router is independent and does not depend on other routers in the area.
  + Corruption or failure in one router does not affect other routers as seriously as in RIP.

3.5.4 Border Gateway Protocol Version 4 (BGP4)

* The Border Gateway Protocol version 4 (BGP4) is the only interdomain routing protocol used in the Internet today.
* BGP4 is based on the path-vector algorithm we described before, but it is tailored to provide information about the reachability of networks in the Internet.

Introduction

* BGP, and in particular BGP4, is a complex protocol. In this section, we introduce the basics of BGP and its relationship with intradomain routing protocols (RIP or OSPF).
* Figure 3.5.4.1 shows an example of an internet with four autonomous systems. AS2, AS3, and AS4 are stub autonomous systems; AS1 is a transient one. In our example, data exchange between AS2, AS3, and AS4 should pass through AS1.
* Each autonomous system in this figure uses one of the two common intradomain protocols, RIP or OSPF.

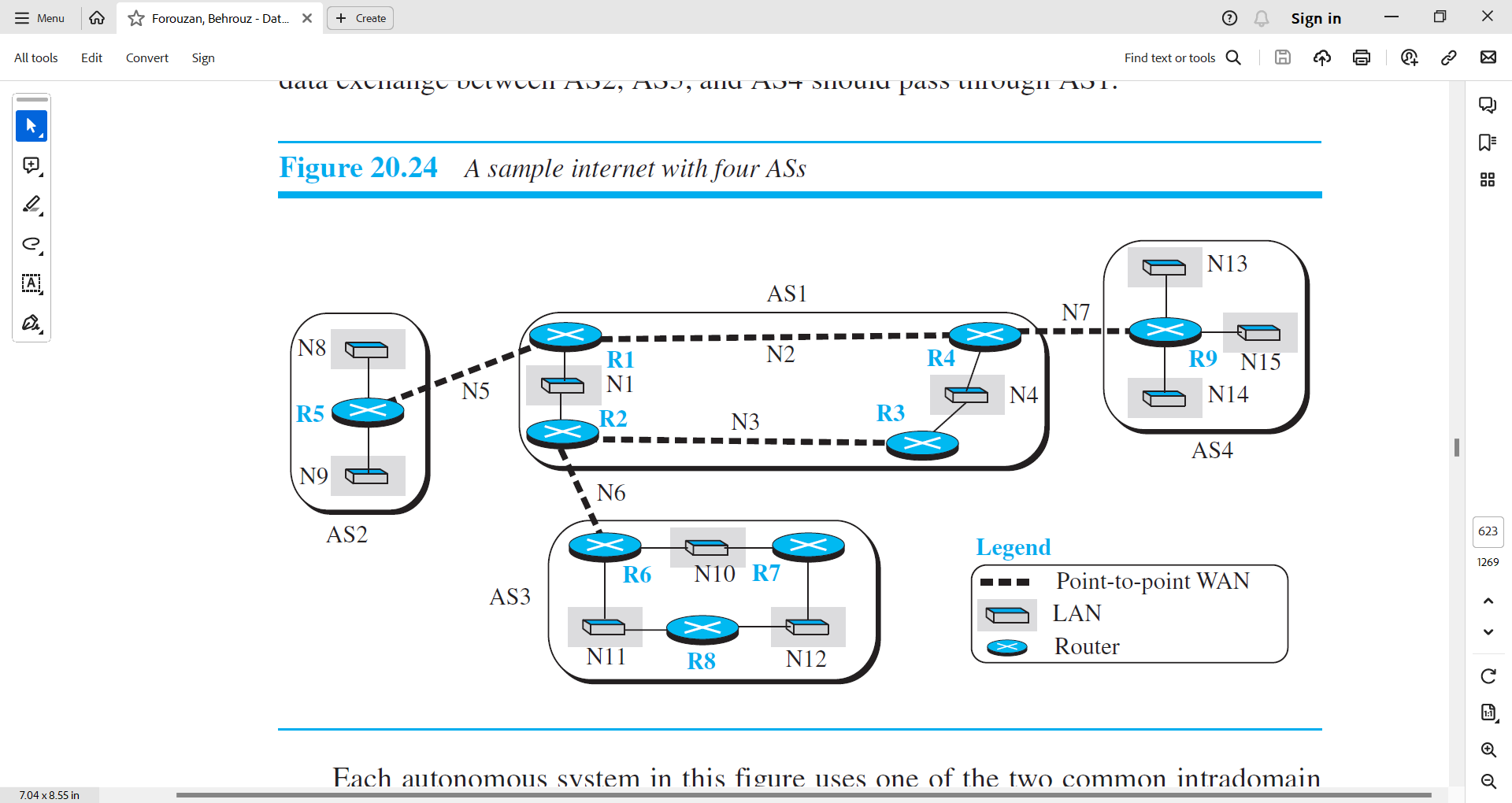


Figure 3.5.4.1 A sample internet with four ASs

* Each router in each AS knows how to reach a network that is in its own AS, but it does not know how to reach a network in another AS.
* To enable each router to route a packet to any network in the internet,
  + we first install a variation of BGP4, called external BGP (eBGP), on each border router (the one at the edge of each AS which is connected to a router at another AS).
  + We then install the second variation of BGP, called internal BGP (iBGP), on all routers.
* This means that the border routers will be running three routing protocols (intradomain, eBGP, and iBGP), but other routers are running two protocols (intradomain and iBGP).
* We discuss the effect of each BGP variation separately.

Operation of External BGP (eBGP)

* We can say that BGP is a kind of point-to-point protocol. When the software is installed on two routers, they try to create a TCP connection using the well-known port 179. In other words, a pair of client and server processes continuously communicate with each other to exchange messages. The two routers that run the BGP processes are called BGP peers or BGP speakers.
* We discuss different types of messages exchanged between two peers, but for the moment we are interested in only the update messages (discussed later) that announce reachability of networks in each AS.
* The eBGP variation of BGP allows two physically connected border routers in two different ASs to form pairs of eBGP speakers and exchange messages.
* The routers that are eligible in our example in Figure 20.24 (3.5.4.2) form three pairs:
  + R1-R5,
  + R2-R6,
  + and R4-R9.
* The connection between these pairs is established over three physical WANs (N5, N6, and N7).
* However, there is a need for a logical TCP connection to be created over the physical connection to make the exchange of information possible. Each logical connection in BGP parlance is referred to as a session. This means that we need three sessions in our example, as shown in Figure 20.25.

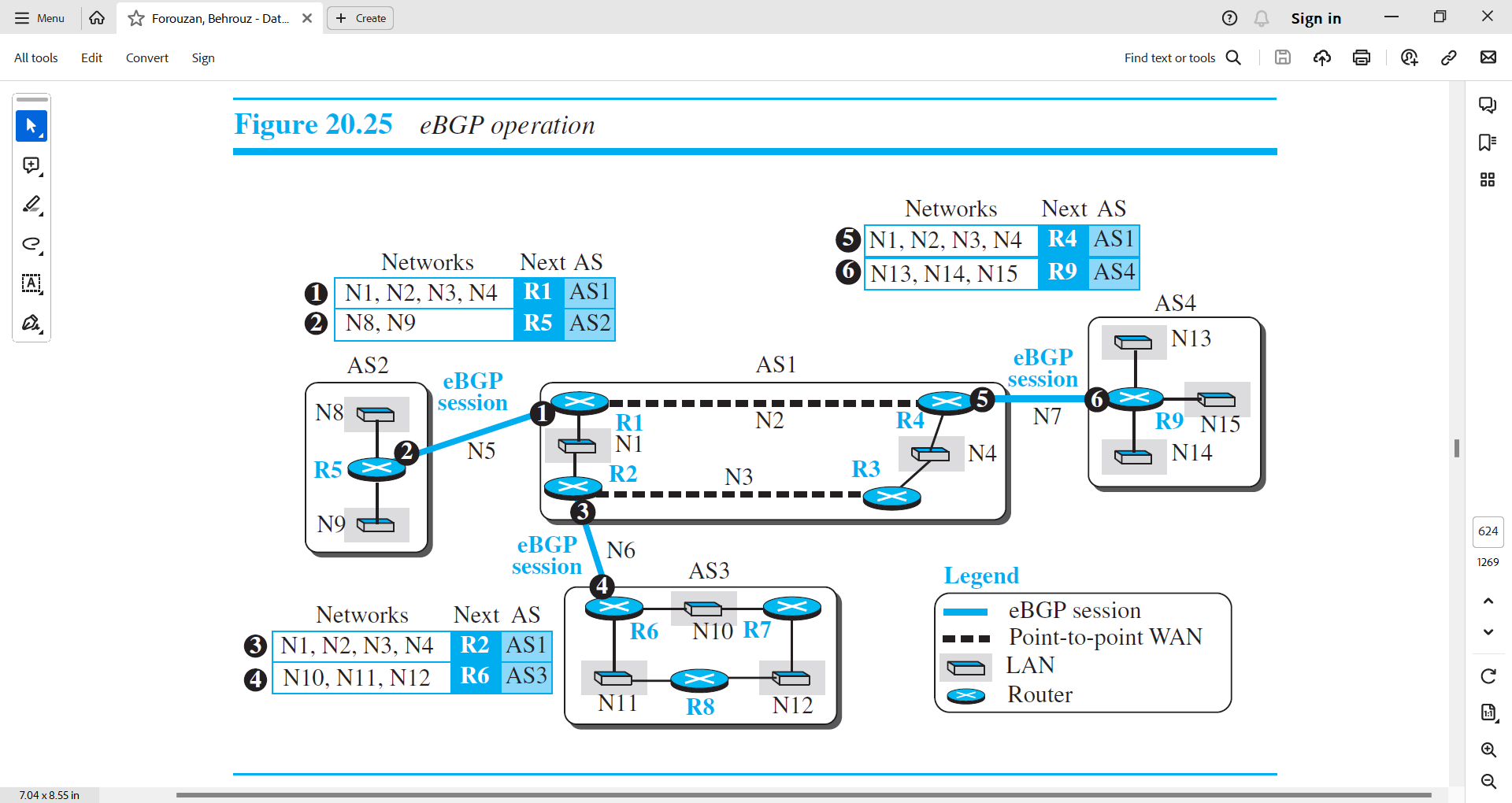


Figure 3.5.4.2 eBGP operation

* The figure also shows the simplified update messages sent by routers involved in the eBGP sessions.
* The circled number defines the sending router in each case.
  + For example,
    - Message number 1 is sent by router R1 and tells router R5 that N1, N2, N3, and N4 can be reached through router R1 (R1 gets this information from the corresponding intradomain forwarding table).Router R5 can now add these pieces of information at the end of its forwarding table. When R5 receives any packet destined for these four networks, it can use its forwarding table and find that the next router is R1.
* The reader may have noticed that the messages exchanged during three eBGP sessions help some routers to know, how to route packets to some networks in the internet, but the reachability information is not complete. There are two problems that need to be addressed:
  + 1. Some border routers do not know how to route a packet destined for nonneighbor ASs. For example, R5 does not know how to route packets destined for networks in AS3 and AS4. Routers R6 and R9 are in the same situation as R5: R6 does not know about networks in AS2 and AS4; R9 does not know about networks in AS2 and AS3.
  + 2. None of the nonborder routers know how to route a packet destined for any networks in other ASs. To address the above two problems, we need to allow all pairs of routers (border or nonborder) to run the second variation of the BGP protocol, iBGP.

Operation of Internal BGP(iBGP)

* The iBGP protocol is similar to the eBGP protocol in that it uses the service of TCP on the well-known port 179, but it creates a session between any possible pair of routers inside an autonomous system.
* However, some points should be made clear. First, if an AS has only one router, there cannot be an iBGP session. For example, we cannot create an iBGP session inside AS2 or AS4 in our internet. Second, if there are n routers in an autonomous system, there should be [n × (n − 1) / 2] iBGP sessions in that autonomous system (a fully connected mesh) to prevent loops in the system.
* In other words, each router needs to advertise its own reachability to the peer in the session instead of flooding what it receives from another peer in another session.
* Figure 3.5.4.3 shows the combination of eBGP and iBGP sessions in our internet.

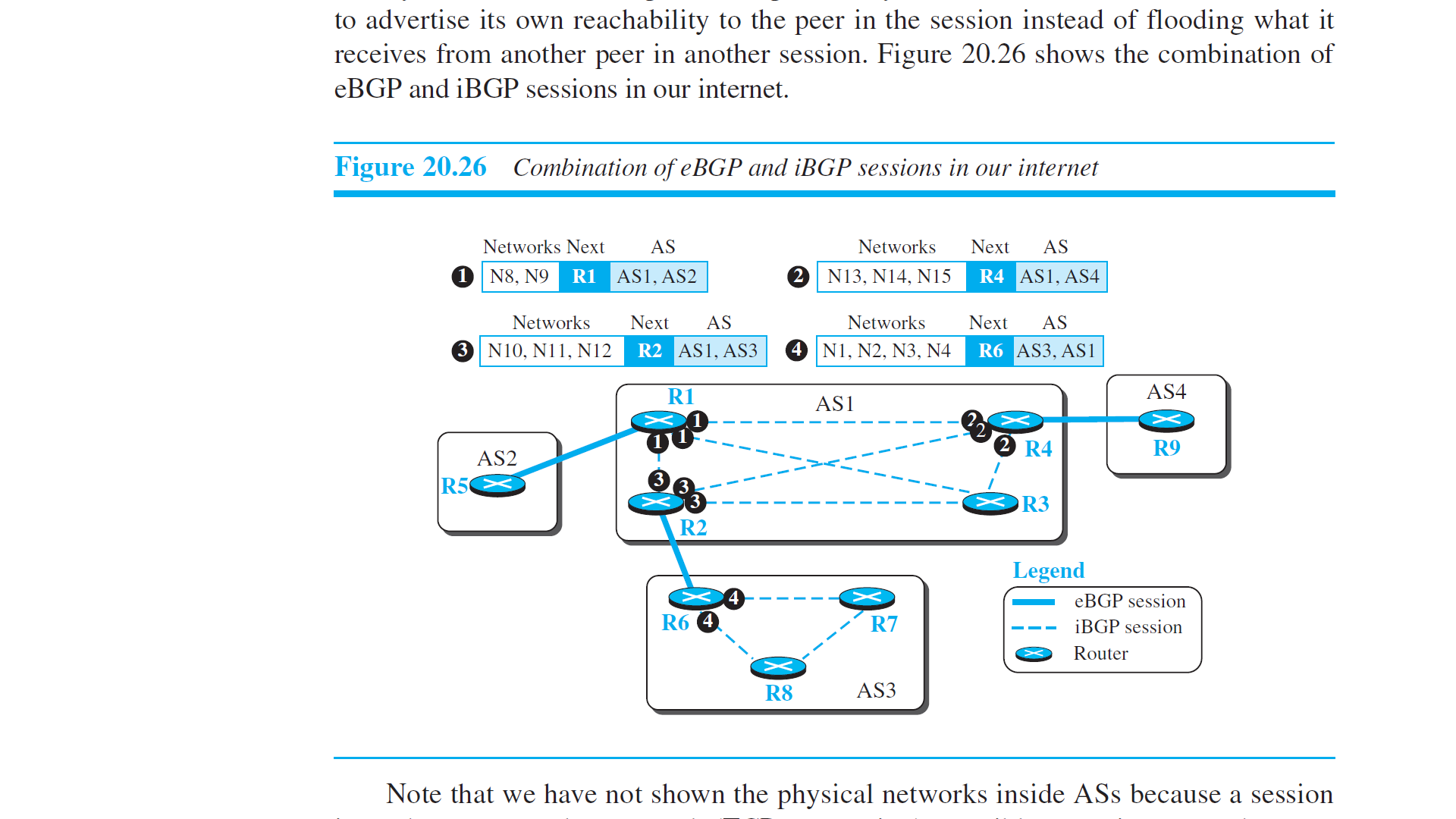


Figure 3.5.4.3 Combinations of eBGP and iBGP sessions in our internet

* Note that we have not shown the physical networks inside ASs because a session is made on an overlay network (TCP connection), possibly spanning more than one physical network as determined by the route dictated by intradomain routing protocol.
* Also note that in this stage only four messages are exchanged.
* The first message (numbered 1) is sent by R1 announcing that networks N8 and N9 are reachable through the path AS1-AS2, but the next router is R1. This message is sent, through separate sessions, to R2, R3, and R4.
* Routers R2, R4, and R6 do the same thing but send different messages to different destinations.
* The interesting point is that, at this stage, R3, R7, and R8 create sessions with their peers, but they actually have no message to send. The updating process does not stop here.
* For example, after R1 receives the update message from R2, it combines the reachability information about AS3 with the reachability information it already knows about AS1 and sends a new update message to R5. Now R5 knows how to reach networks in AS1 and AS3. The process continues when R1 receives the update message from R4.
* The point is that we need to make certain that at a point in time there are no changes in the previous updates and that all information is propagated through all ASs.At this time, each router combines the information received from eBGP and iBGP and creates what we may call a path table after applying the criteria for finding the best path, including routing policies that we discuss later.
* To demonstrate, we show the path tables in Figure 3.5.4.4 for the routers in Figure 3.5.4.1.

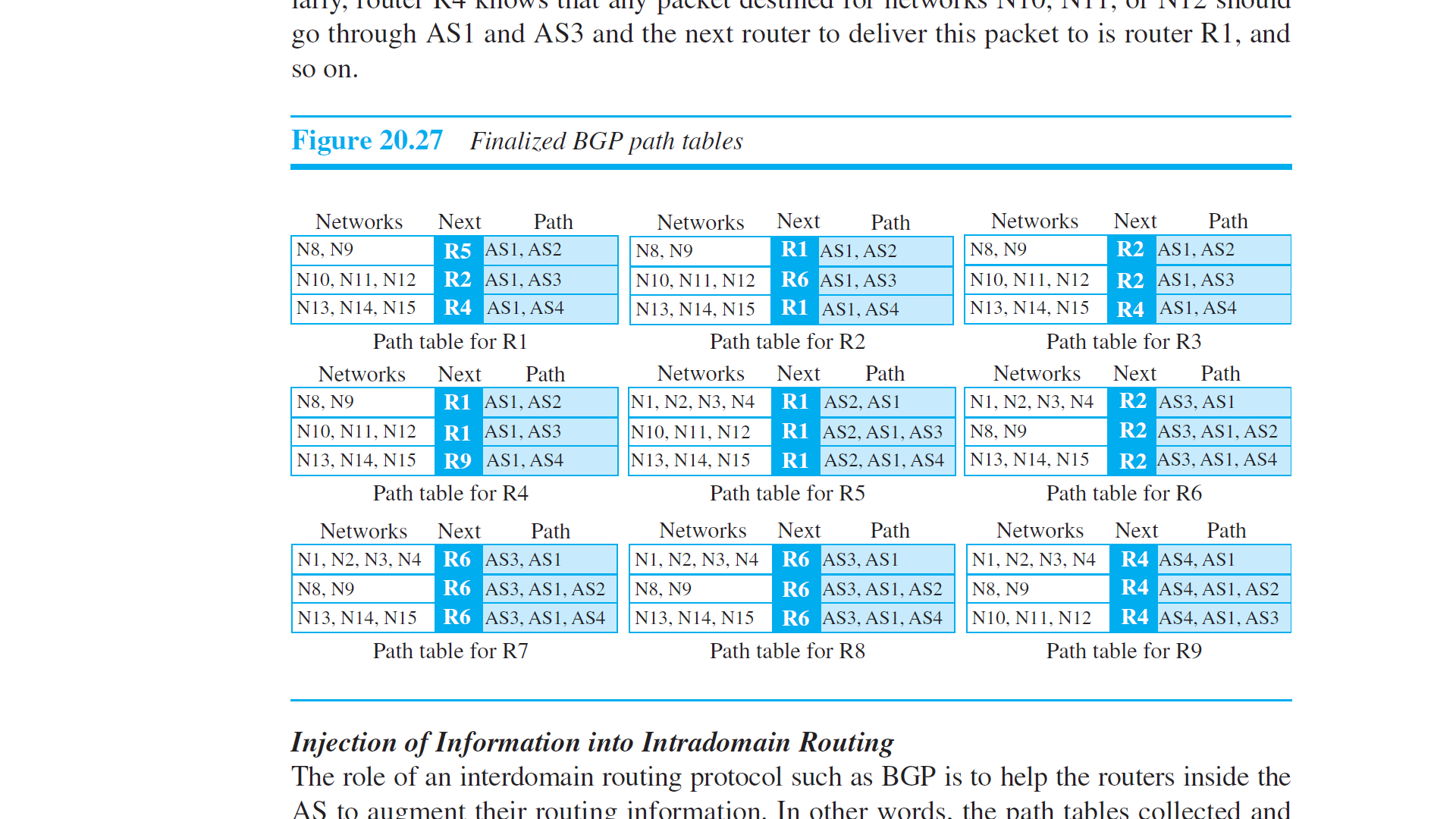


Figure 3.5.4.4 Finalized BGP Path tables

* For example, router R1 now knows that any packet destined for networks N8 or N9 should go through AS1 and AS2 and the next router to deliver the packet to is router R5. Similarly, router R4 knows that any packet destined for networks N10, N11, or N12 should go through AS1 and AS3 and the next router to deliver this packet to is router R1, and so on.

Injection of Information into Intradomain Routing

* The role of an interdomain routing protocol such as BGP is to help the routers inside the AS to augment their routing information. In other words, the path tables collected and organized by BPG are not used, per se, for routing packets; they are injected into intradomain forwarding tables (RIP or OSPF) for routing packets. This can be done in several ways depending on the type of AS.
* In the case of a stub AS, the only area border router adds a default entry at the end of its forwarding table and defines the next router to be the speaker router at the end of the eBGP connection. In Figure 3.5.4.1, R5 in AS2 defines R1 as the default router for all networks other than N8 and N9. The situation is the same for router R9 in AS4 with the default router to be R4. In AS3, R6 set its default router to be R2, but R7 and R8 set their default router to be R6. These settings are in accordance with the path tables we describe in Figure 3.5.4.4 for these routers. In other words, the path tables are injected into intradomain forwarding tables by adding only one default entry.
* In the case of a transient AS, the situation is more complicated. R1 in AS1 needs to inject the whole contents of the path table for R1 in Figure 3.5.4.4 into its intradomain forwarding table. The situation is the same for R2, R3, and R4. One issue to be resolved is the cost value. We know that RIP and OSPF use different metrics. One solution, which is very common, is to set the cost to the foreign networks at the same cost value as to reach the first AS in the path. For example, the cost for R5 to reach all networks in other ASs is the cost to reach N5. The cost for R1 to reach networks N10 to N12 is the cost to reach N6, and so on. The cost is taken from the intradomain forwarding tables (RIP or OSPF).
* Figure 3.5.4.5 shows the interdomain forwarding tables. For simplicity, we assume that all ASs are using RIP as the intradomain routing protocol. The shaded areas are the augmentation injected by the BGP protocol; the default destinations are indicated as zero.

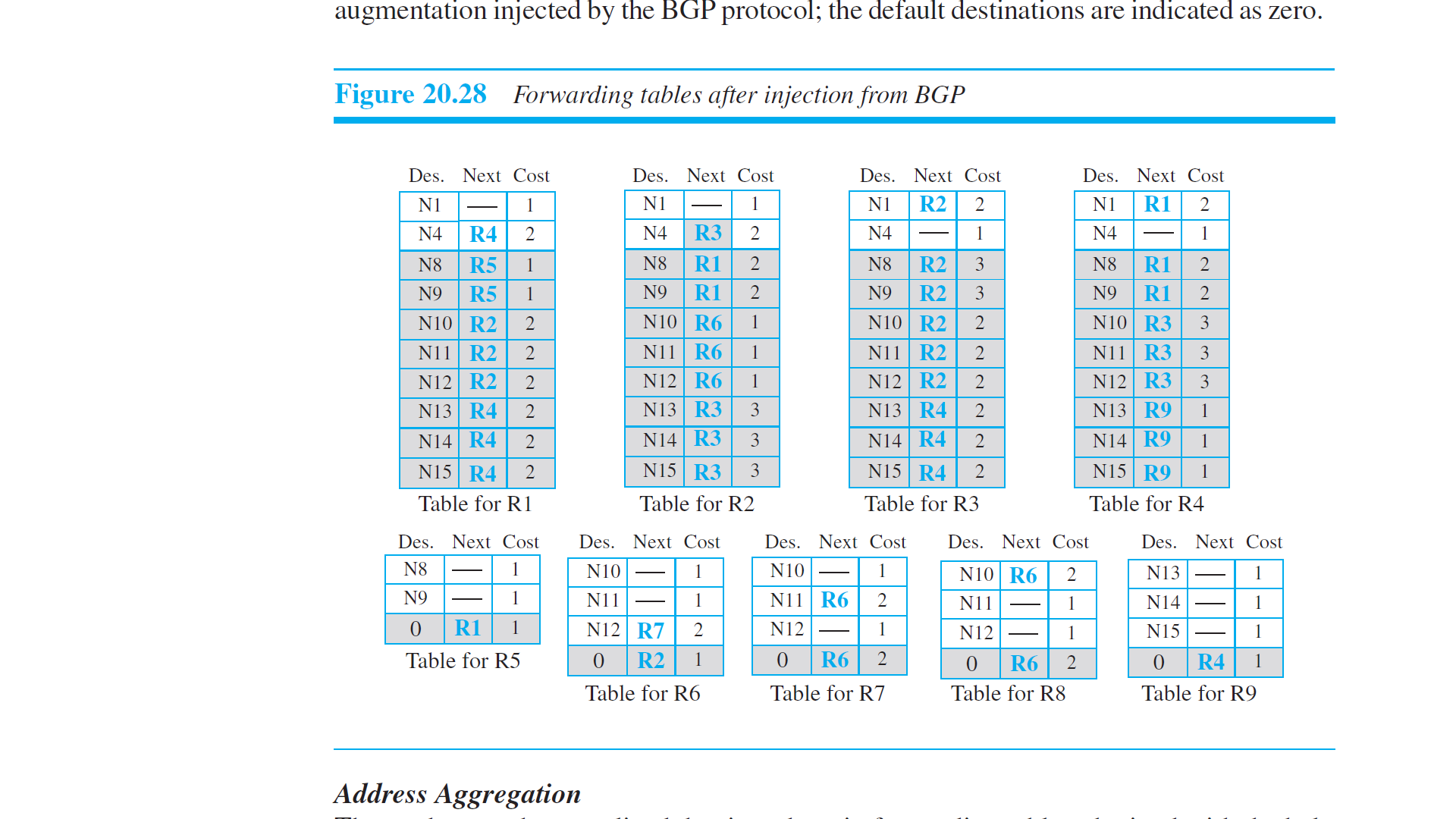


Figure 3.5.4.5 Forwarding tables after injection from BGP

Address Aggregation

* The reader may have realized that intradomain forwarding tables obtained with the help of the BGP4 protocols may become huge in the case of the global Internet because many destination networks may be included in a forwarding table.
* Fortunately, BGP4 uses the prefixes as destination identifiers and allows the aggregation of these prefixes, as we discussed in the Unit-1. For example, prefixes 14.18.20.0/26, 14.18.20.64/26, 14.18.20.128/26, and 14.18.20.192/26, can be combined into 14.18.20.0/24 if all four subnets can be reached through one path. Even if one or two of the aggregated prefixes need a separate path, the longest prefix principle we discussed earlier allows us to do so.

Path Attributes

* In both intradomain routing protocols (RIP or OSPF), a destination is normally associated with two pieces of information: next hop and cost. The first one shows the address of the next router to deliver the packet; the second defines the cost to the final destination.
* Interdomain routing is more involved and naturally needs more information about how to reach the final destination. In BGP these pieces are called path attributes.
* BGP allows a destination to be associated with up to seven path attributes. Path attributes are divided into two broad categories: well-known and optional. A well-known attribute must be recognized by all routers; an optional attribute need not be.
  + A well-known attribute can be mandatory, which means that it must be present in any BGP update message, or discretionary, which means it does not have to be.
  + An optional attribute can be either transitive, which means it can pass to the next AS, or intransitive, which means it cannot.
* All attributes are inserted after the corresponding destination prefix in an update message (discussed later).
* The format for an attribute is shown in Figure 3.5.4.6.

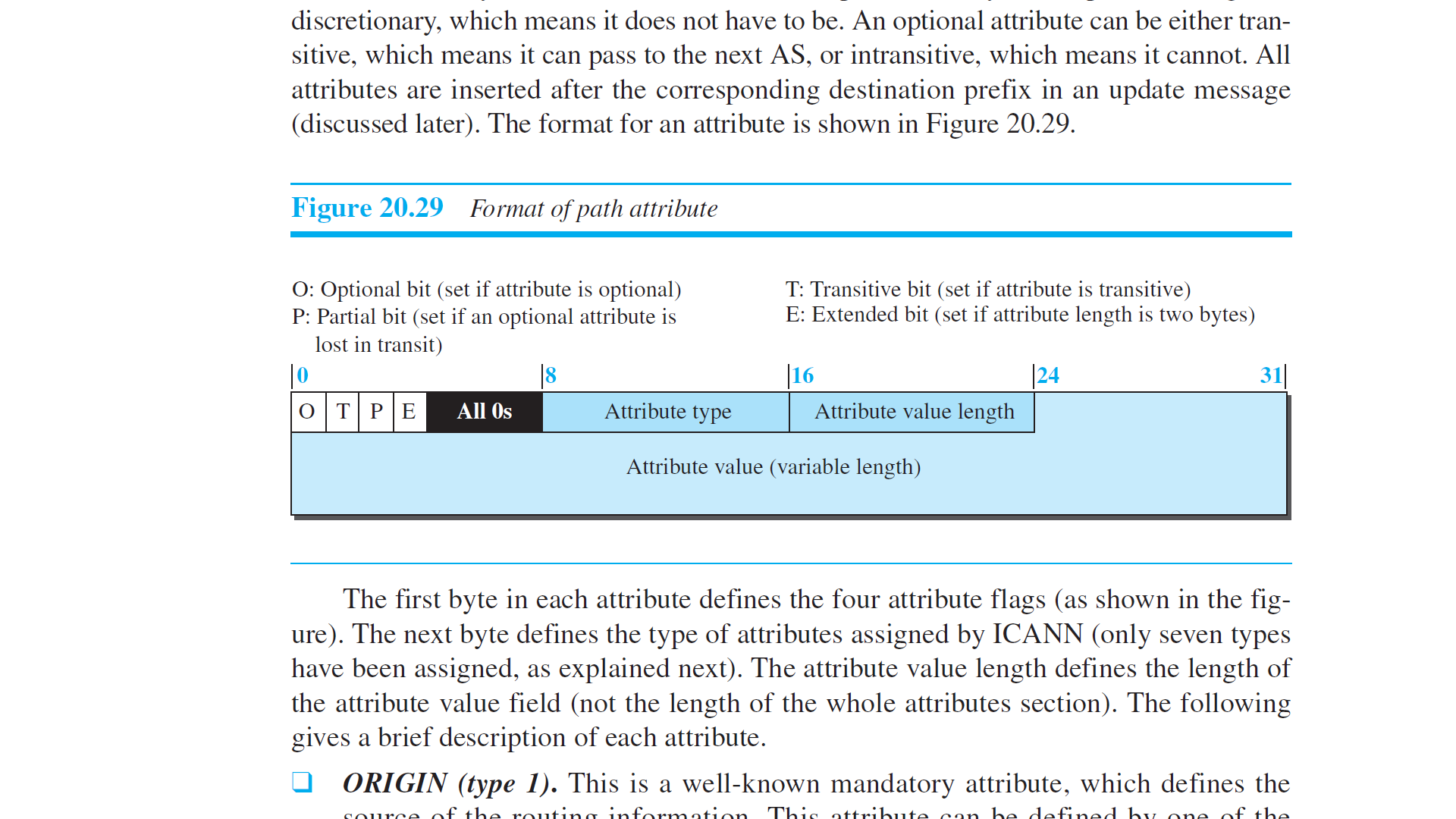


Figure 3.5.4.6 Format of Path Attribute

* The first byte in each attribute defines the four attribute flags (as shown in the figure). The next byte defines the type of attributes assigned by ICANN (only seven types have been assigned, as explained next). The attribute value length defines the length of the attribute value field (not the length of the whole attributes section). The following gives a brief description of each attribute.
* ORIGIN (type 1).
  + This is a well-known mandatory attribute, which defines the source of the routing information.
  + This attribute can be defined by one of the three values: 1, 2, and 3.
    - Value 1 means that the information about the path has been taken from an intradomain protocol (RIP or OSPF).
    - Value 2 means that the information comes from BGP.
    - Value 3 means that it comes from an unknown source.
* AS-PATH (type 2).
  + This is a well-known mandatory attribute, which defines the list of autonomous systems through which the destination can be reached. We have used this attribute in our examples.
  + The AS-PATH attribute, as we discussed in path-vector routing in the last section, helps prevent a loop. Whenever an update message arrives at a router that lists the current AS as the path, the router drops that path.
  + The AS-PATH can also be used in route selection.
* NEXT-HOP (type 3).
  + This is a well-known mandatory attribute, which defines the next router to which the data packet should be forwarded.
  + We have also used this attribute in our examples. As we have seen, this attribute helps to inject path information collected through the operations of eBGP and iBGP into the intradomain routing protocols such as RIP or OSPF.
* MULT-EXIT-DISC (type 4).
  + The multiple-exit discriminator is an optional intransitive attribute, which discriminates among multiple exit paths to a destination.
  + The value of this attribute is normally defined by the metric in the corresponding intradomain protocol (an attribute value of 4-byte unsigned integer).
  + For example, if a router has multiple paths to the destination with different values related to these attributes, the one with the lowest value is selected.
  + Note that this attribute is intransitive, which means that it is not propagated from one AS to another.
* LOCAL-PREF (type 5).
  + The local preference attribute is a well-known discretionary attribute. It is normally set by the administrator, based on the organization policy.
  + The routes the administrator prefers are given a higher local preference value (an attribute value of 4-byte unsigned integer).
  + For example, in an internet with five ASs,
    - the administrator of AS1 can set the local preference value of 400 to the path AS1 → AS2 → AS5, the value of 300 to AS1 → AS3 → AS5, and the value of 50 to AS1 → AS4 → AS5. This means that the administrator prefers the first path to the second one and prefers the second one to the third one. This may be a case where AS2 is the most secured and AS4 is the least secured AS for the administration of AS1. The last route should be selected if the other two are not available.
* ATOMIC-AGGREGATE (type 6).
  + This is a well-known discretionary attribute, which defines the destination prefix as not aggregate; it only defines a single destination network.
  + This attribute has no value field, which means the value of the length field is zero.
* AGGREGATOR (type 7).
  + This is an optional transitive attribute, which emphasizes that the destination prefix is an aggregate.
  + The attribute value gives the number of the last AS that did the aggregation followed by the IP address of the router that did so.

Route Selection

* So far in this section, we have been silent about how a route is selected by a BGP router mostly because our simple example has one route to a destination.
* In the case where multiple routes are received to a destination, BGP needs to select one among them.The route selection process in BGP is not as easy as the ones in the intradomain routing protocol that is based on the shortest-path tree.
* A route in BGP has some attributes attached to it and it may come from an eBGP session or an iBGP session.
* Figure 3.5.4.7 shows the flow diagram as used by common implementations.
* The router extracts the routes which meet the criteria in each step. If only one route is extracted, it is selected and the process stops; otherwise, the process continues with the next step.
* Note that the first choice is related to the LOCAL-PREF attribute, which reflects the policy imposed by the administration on the route.

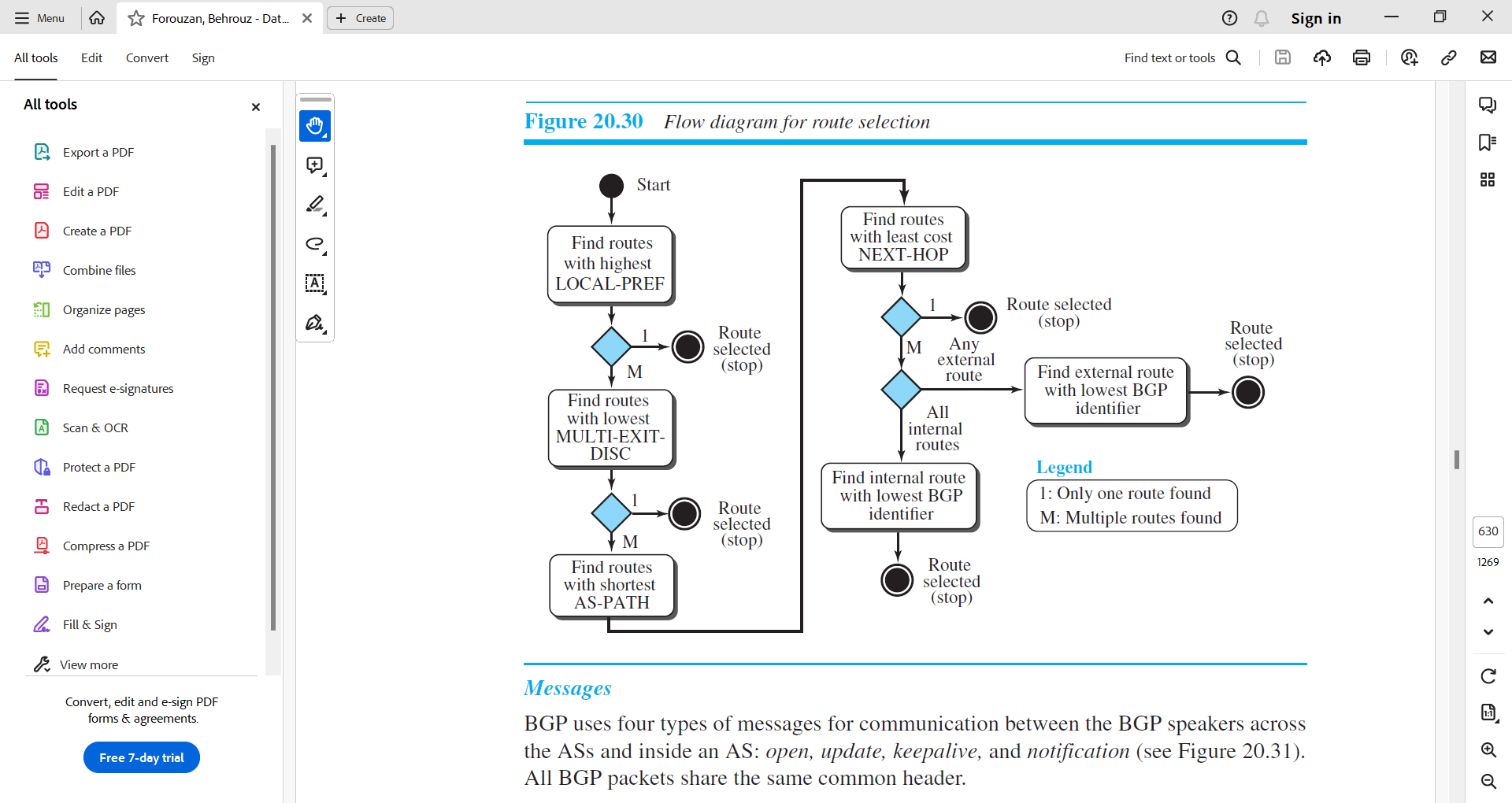


Figure 3.5.4.7 Flow diagram for route selection

BGP Messages

* BGP uses four types of messages for communication between the BGP speakers across the ASs and inside an AS: open, update, keepalive, and notification (see Figure 3.5.4.8).

A screenshot of a computer

Description automatically generated

Figure 3.5.4.8 BGP Messages

* All BGP packets share the same common header.
* Open Message.
  + To create a neighborhood relationship, a router running BGP opens a TCP connection with a neighbor and sends an open message.
* Update Message.
  + The update message is the heart of the BGP protocol.
  + It is used by a router to withdraw destinations that have been advertised previously, to announce a route to a new destination, or both.
  + Note that BGP can withdraw several destinations that were advertised before, but it can only advertise one new destination (or multiple destinations with the same path attributes) in a single update message.
* Keepalive Message.
  + The BGP peers that are running exchange keepalive messages regularly (before their hold time expires) to tell each other that they are alive.
* Notification.
  + A notification message is sent by a router whenever an error condition is detected or a router wants to close the session.

Performance

* BGP performance can be compared with RIP.
* BGP speakers exchange a lot of messages to create forwarding tables, but BGP is free from loops and count-to-infinity.
* The same weakness we mention for RIP about propagation of failure and corruption also exists in BGP.