

Practical 2 – Network Configuration and Testing

Coursework Weight: 40%

Contents

Introduction	1
Submission and Assessment.....	2
Marking Criteria.....	2
Introduction to IMUNES.....	3
Using traceroute and ping	4
Setting the link characteristics	5
Setting the router characteristics.....	5
Measuring Network Bandwidth.....	6
Route Distribution with OSPF	7
The OSPF Cost Attribute.....	9
Tuning the default OSPF cost	10
Router configuration syntax	11
Changing the bandwidth of an interface.....	12
Configuring OSPF routing areas	14
Self-assessment tasks	16
Construct and configure a network.....	16
Configure OSPF costs	17
Configure OSPF areas	17
Network debugging	17

Introduction

In this practical, you will be building, configuring and testing computer networks. This exercise is designed to give you a better understanding of how and why computer networks work the way that they do, the ways that they can be configured and built, and to give you an appreciation of what goes on behind the scenes when your software application uses this resource.

For the purposes of this practical, we will be using the Integrated Multiprotocol Network Emulator/Simulator (IMUNES). This software-based network emulation tool allows us to design, build and run networks without the need for physical hardware setups. We can also use IMUNES to recreate networks far larger than what would be possible in a lab-based environment. It has an easy to use graphical interface, which makes working with the tool

much more intuitive. Finally, because IMUNES runs unmodified protocol stacks, the traffic that traverses our virtual networks is close to that which we would observe in the real world and is particularly genuine in this respect.

Submission and Assessment

The format of this assessment is in an in-lab test, to take place in Week 20. You will be asked to implement specific emulation scenarios using IMUNES, and briefly document your solution to explain your rationale. Feel free to include screenshots and text in this. However, we want to be clear that you do not need to document *everything*, but instead provide a short explanation of what you did and why. At the end of the test lab session, you will submit both your topology file and documentation on Moodle. Note that the problems will be different among different lab sessions.

Marking Criteria

In each case, marks will be awarded for satisfactorily demonstrating that you have completed each task and that you are familiar with concepts contained within. You will provide evidence of such through the participation in the in-lab assessment and through documentation of the process.

Introduction to IMUNES

To get started with IMUNES, we will be using the same virtual machine image as used in Coursework 1. Open a terminal window and run IMUNES using:

```
sudo imunes
```

This should start the IMUNES emulator, which will load with a blank canvas, similar to this:

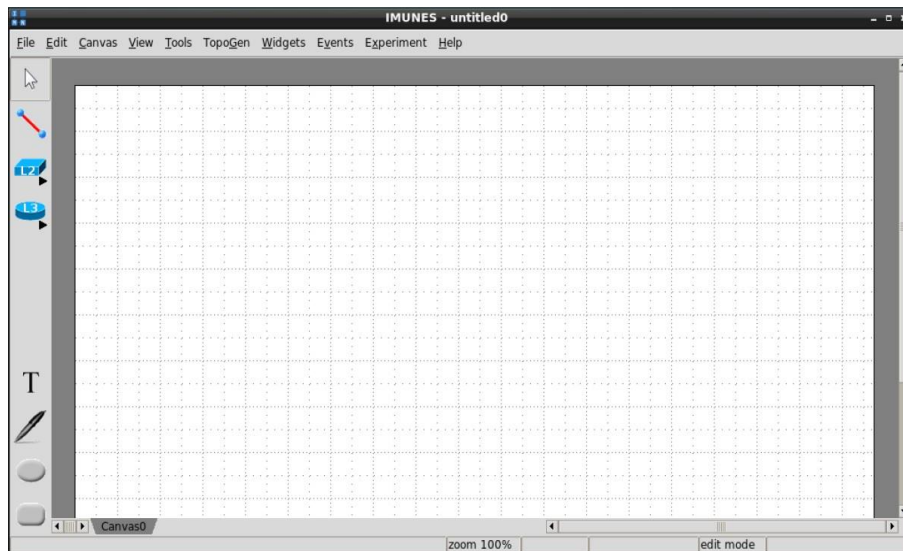


Figure 1: The IMUNES canvas

From here, we can load an IMUNES scenario file. To do this, click *File* → *Open* and navigate to the file you wish to load. To get you started, we have provided you with such a file on Moodle. Once you have loaded this, your canvas should look like this:

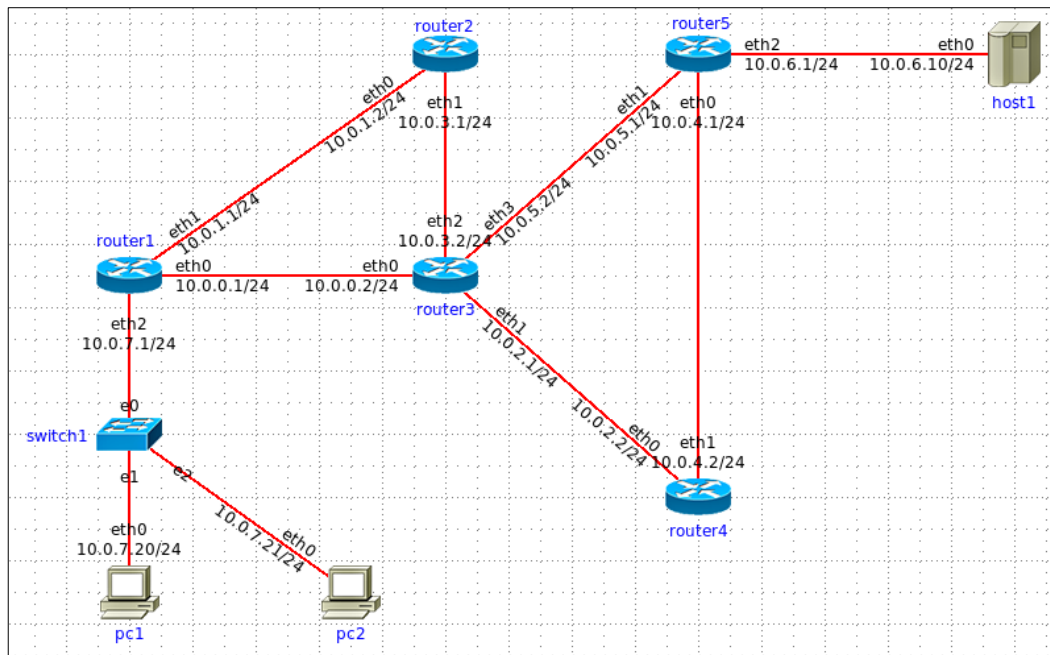


Figure 2: Network topology of the IMUNES scenario file

IMUNES supports a number of different node types, including switches and routers. These can be connected together using links to build a network. Finally, IMUNES also supports the creation of hosts, which represent end points in the network (much like the PC or laptop you are using now). Together, these nodes enable the emulation of a full computer network. The scenario file we have provided contains a wide range of these node types, including hosts (PCs), routers and switches.

IMUNES also enables the display of various types of information on the canvas. This can be configured using *View* → *Show* and selecting the items of information to be shown. This includes useful information such as IP addresses (which are automatically assigned by the emulator), node labels (used to identify each node) and interface names (useful when a device has multiple interfaces visible, such as the router). For the purposes of this practical, we will not be using IPv6 addresses, so these can be disabled here too.

Once you are happy with the setup and information displayed, you can run the emulation by selecting *Experiment* → *Execute* from the topmost menu.

This brings the emulation into execution mode, whereby the hosts, links and networking elements are active. At this point, if setup correctly, the network should be able to send packets, and communication between one or more hosts should be possible.

Using traceroute and ping

IMUNES allows us to create terminal windows, which give us access to each host. With the emulation in execution, double-click a host to open its terminal window, which should look something like this:



Figure 3: You can bring up the command-line terminal of a host or router by double-clicking its icon while the emulation is in execution. This allows us to run common UNIX utilities, such as `ping`, `traceroute`. Try running some traceroutes to experiment with this feature, for example between *pc1* and *host1*.

To end the emulation and stop the activity within the network, select *Experiment* → *Terminate* from the topmost menu.

Setting the link characteristics

The link characteristics can be changed by clicking on each link and then choosing the *Configuration* option, which will bring up the link configuration window (Figure 4). From here, the bandwidth, delay and loss (Bit Error Rate or BER) can be individually set.

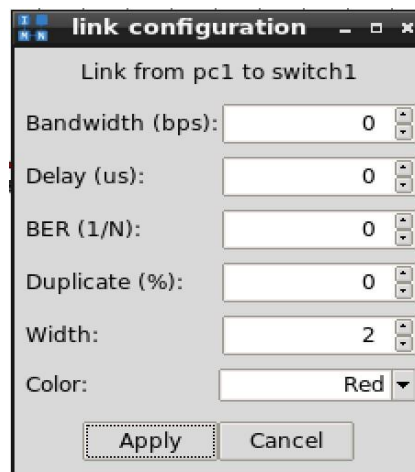


Figure 4: The link configuration window allows you to set the link properties

Setting the router characteristics

With the emulation stopped, double-click on the icon of a router to open its *router configuration* panel. From here, you can edit an array of router properties, such as its model, services, protocols and interfaces.

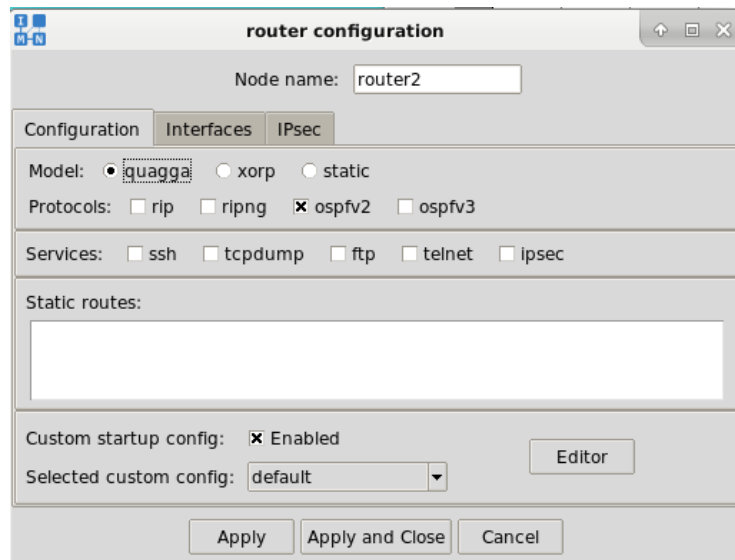


Figure 5: The router configuration panel

The *Interfaces* tab allows you to edit the properties of the router interfaces, such as their IP addresses and their MTU (Maximum Transmission Unit).

Two connected interfaces need to belong in the same subnet and to have the same MTU value in order to be able to exchange traffic directly.

Measuring Network Bandwidth

Iperf is a command-line tool that allows you to test the bandwidth between two network hosts. One host must act as a server that listens for incoming connections. The other host acts as a client that connects to the server and exchange data.

Open the terminal of *host1* and run the following command to start the iperf server:

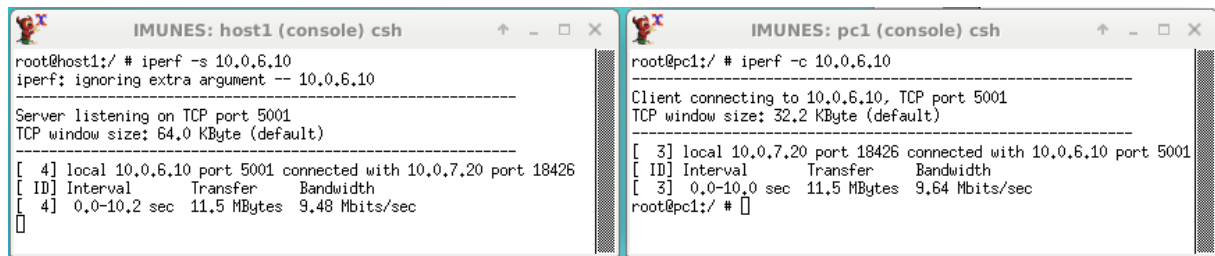
```
iperf -s 10.0.6.10
```

The `-s` flag means that you executed the server mode of iperf, while 10.0.6.10 is the IP address of host1.

Now open the terminal of *pc1* and run the following command to measure the bandwidth between *pc1* and *host1*:

```
iperf -c 10.0.6.10
```

The `-c` flag means that you executed the client mode of iperf, and you connect to the IP address of host1. After completing a test data exchange, iperf will print the measurement results as shown in Figure 6.



```

IMUNES: host1 (console) csh
root@host1:/ # iperf -s 10.0.6.10
iperf: ignoring extra argument -- 10.0.6.10
-----
Server listening on TCP port 5001
TCP window size: 64.0 KByte (default)
-----
[  4] local 10.0.6.10 port 5001 connected with 10.0.7.20 port 18426
[ ID] Interval      Transfer    Bandwidth
[  4] 0.0-10.2 sec  11.5 MBytes  9.48 Mbits/sec

IMUNES: pc1 (console) csh
root@pc1:/ # iperf -c 10.0.6.10
-----
Client connecting to 10.0.6.10, TCP port 5001
TCP window size: 32.2 KByte (default)
-----
[  3] local 10.0.7.20 port 18426 connected with 10.0.6.10 port 5001
[ ID] Interval      Transfer    Bandwidth
[  3] 0.0-10.0 sec  11.5 MBytes  9.64 Mbits/sec
root@pc1:/ #

```

Figure 6: Execution of iperf bandwidth measurement between host1 and pc1

[Here](#) you can find more information on how to use iperf.

Now you have a basic introduction to the operation of IMUNES. In this practical, you will be using this emulator extensively to conduct a number of experiments and tasks. These are detailed in the remainder of this document.

Route Distribution with OSPF

This task involves examining how routing works, and how it adapts to changes in the network.

A router makes forwarding decisions based upon information that it holds about how to reach certain destinations. This decision is determined by the IP address of an arriving packet. The information is held in a table, which contains a number of entries, each representing another segment of the network, and the accompanying port through which the packet should be sent to reach its destination. This information is spread between routers using standard protocols. In this practical, we will be focusing on OSPF.

Open Shortest Path First (OSPF) is a routing protocol used internally in large networks; that is, within a single Autonomous System (AS). As an implementation, it monitors the network for changes, including the availability of routers. After such a change, it recalculates the routing structure very quickly. OSPF is a link-state algorithm that computes the shortest path to a destination based upon Dijkstra's algorithm. As such, each link has an associated cost metric, which is a value based on the total capacity of each link along the path. Each router selects the optimal path to a destination IP prefix based on its total cost and advertises the selected path to its neighbouring routers.

For more information, please see Chapter 5.3 of the course text book.

To view the current routing table, select *Widgets → IPv4 Routing Table* from the main menu bar. With the emulator running, hovering over the router will produce a widget with the following information:

```
n2# netstat -4 -rn
```

Kernel IP routing table							
Destination	Gateway	Genmask	Flags	MSS	Window	irtt	Iface
10.0.0.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
10.0.1.0	0.0.0.0	255.255.255.0	U	0	0	0	eth1
10.0.2.0	0.0.0.0	255.255.255.0	U	0	0	0	eth2
10.0.3.0	10.0.2.1	255.255.255.0	UG	0	0	0	eth2
10.0.4.0	0.0.0.0	255.255.255.0	U	0	0	0	eth3
10.0.5.0	10.0.2.1	255.255.255.0	UG	0	0	0	eth2
10.0.6.0	10.0.4.1	255.255.255.0	UG	0	0	0	eth3
127.0.0.1	0.0.0.0	255.255.255.255	UH	0	0	0	lo

Note that this screenshot may not directly match what you see.

Within the emulator, it is also possible to get more information on these routes and manipulate them in different ways. To do this, we need to open a terminal for the router. Right-click on one of the router elements and select *Shell window* → *vtysh*:

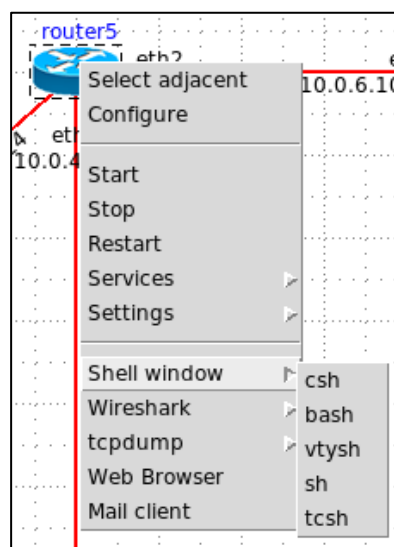


Figure 7: The vtysh shell window allows access to routing commands

This should open up an interface, subtly different to those opened previously on the host machines; this terminal interacts directly with the routing daemon running on the node.

To show a more detailed view of the routes, use:

```
show ip route
```

This should give you information such as the one shown in Figure 8.


```

Hello, this is Quagga (version 1.2.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

router5# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, P - PIM, A - Babel, N - NHRP,
       * - selected route, * - FIB route

O>* 10.0.0.0/24 [110/20] via 10.0.5.2, eth1, 00:07:45
O>* 10.0.1.0/24 [110/30] via 10.0.5.2, eth1, 00:07:40
O>* 10.0.2.0/24 [110/20] via 10.0.4.2, eth0, 00:07:45
                        via 10.0.5.2, eth1, 00:07:45
O>* 10.0.3.0/24 [110/20] via 10.0.5.2, eth1, 00:07:45
O  10.0.4.0/24 [110/10] is directly connected, eth0, 00:08:30
C>* 10.0.4.0/24 is directly connected, eth0
O  10.0.5.0/24 [110/10] is directly connected, eth1, 00:08:30
C>* 10.0.5.0/24 is directly connected, eth1
O  10.0.6.0/24 [110/10] is directly connected, eth2, 00:08:30
C>* 10.0.6.0/24 is directly connected, eth2
O>* 10.0.7.0/24 [110/30] via 10.0.5.2, eth1, 00:07:40
C>* 127.0.0.0/8 is directly connected, lo0
O>* 127.0.0.1/32 [110/0] is directly connected, lo0, 00:08:30
router5#

```

Figure 8: The routing table of router5 annotated with the basic attributes of a route

For each route the routing tables provides the following basic information:

- ❶ Indicates the destination IP prefix of the route
- ❷ The first number in the brackets is the *administrative distance* of the information source; the second number is the *metric (cost)* for the route. The administrative distance is by default 110 for all routes learned through OSPF, and is used to select the most preferable route only if multiple routing protocols are used. For routes with equal administrative distance, the route with the lowest cost is used.
- ❸ The IP address of the next router to the destination
- ❹ The interface through which the destination can be reached.
- ❺ The last time the route was updated in *hours:minutes:seconds*. As you can see not all paths are inserted in the routing table at the same time. Some paths take up to 50 seconds more. This happens because the routes need to be propagated from one router to the other, so the destinations that are further from the local router take longer time to be established.

For a more detailed explanation, you can consult the [Cisco documentation](#).

There are also number of additional commands that you may find useful:

```

show ip route – shows all routes
show ip ospf route – shows OSPF routes
show ip ospf interface – show info about the router's interfaces
show ip ospf neighbor – show info about the router's neighbours

```

The OSPF Cost Attribute

Every router along the path to a destination IP prefix calculates the OSPF cost as follows:

$$\frac{\text{Reference Bandwidth}}{\text{Interface bandwidth}}$$

The default Reference Bandwidth is 100 Mbps, while the default interface bandwidth for IMUNES is 10 Mbps. Therefore, the default cost of an OSPF route is 10.

After a router calculates the cost of route, it propagates the <IP prefix, cost> tuple to its neighbouring routers. When a router receives a route from a neighbour, it also calculates the cost based on the capacity of its own interface and adds it to the received cost. Therefore, at each router the cost to an IP prefix is the cumulative cost of all the routers along the path to that prefix.

For example, consider the routing table of *router5* (Figure 8). Why the cost to the IP prefix 10.0.7.0/24 is 30?

The cost of the prefix at *router1* is 10 because *router1* is the origin of the prefix; therefore, *router1* propagates the prefix 10.0.7.0/24 to *router3* with cost 10. *Router3* will receive the prefix 10.0.7.0/24 from *router1* and it will add another 10 to the cost of the route, therefore *router3* propagates the prefix 10.0.7.0/24 with cost 20. Finally, *router3* will add another 10 to the cost of the route for its own interface for a total cost of 30. Like that, OSPF can install the shortest path.

Now test what would be the cost of the same route if the link between *router3* and *router5* fails. With the emulation execution terminated, right click on the link and select *Delete*. Query the routing table of *router5* again and take note of the new cost¹.

Measure the latency between *router5* and *pc1* when the link *router3* – *router5* is deleted, and when the link is restored. How does the average RTT change?

Tuning the default OSPF cost

The OSPF cost calculation selects the most efficient path if all the links have the same characteristics. However, if different links have different performance characteristics, the OSPF selection may become suboptimal.

Change the configuration of the *router3* – *router5* link and set the delay to 100 us. Query the routing table of *router5* again. The routing table has not changed because OSPF considers only the interface properties when calculating path costs, not the link properties.

Re-run the ping measurements between *router5* and *pc1* to examine how latency has changed. Now the path to *pc1* through *router4* will have higher cost compared to the path through *router3*, but lower latency. Therefore, the default OSPF cost is suboptimal.

Fortunately, it is possible to override the default cost calculation and tune it to the network topology. To do that, you need to create a custom router configuration.

¹ You may need to wait some time before the 10.0.7.0 appears in the routing table of *router5* due to the propagation delay.

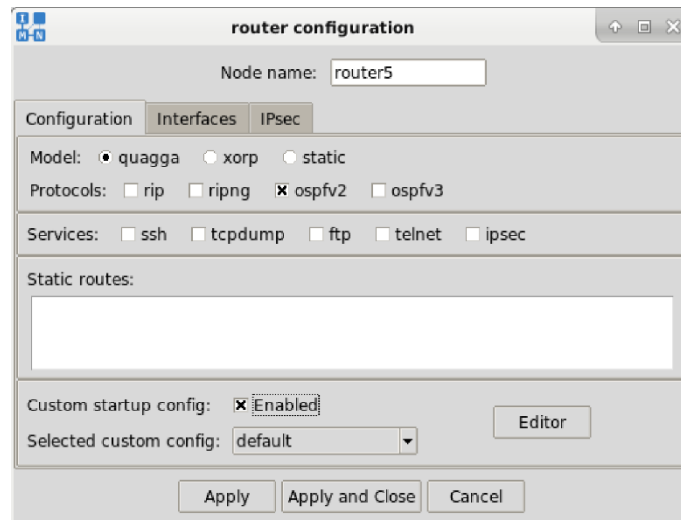


Figure 9: How to enable the custom router configuration

To create a custom routing configuration, with the execution terminated follow the steps below for *router5*:

1. Open the router's configuration window (*double-click on the router's icon or right-click → configuration*) as shown in Figure 9.
2. In the *Configuration* tab, check the *Enabled* checkbox next to the “Custom startup config” label.
3. Click the *Editor* button.
4. A new window with title “Custom configurations” should appear. Click *Create* in this new window to create a new custom configuration.
5. A new configuration editor should appear with a “default” Configuration ID. Click the *Fill Defaults* button.
6. The configuration textbox is filled with a default configuration, which you will extend to achieve the desired routing behaviour.

Router configuration syntax

The default configuration should be similar to the one below (the actual IP addresses may differ):

```
interface lo0
ip address 127.0.0.1/8
!
interface eth0
ip address 10.0.0.1/24
!
interface eth1
ip address 10.0.2.1/24
!
router ospf
 redistribute static
 redistribute connected
 redistribute rip
 network 0.0.0.0/0 area 0.0.0.0
!
```

The exclamation marks are just used to assist in readability by dividing the different sections of the documentation.

An **interface** <interface_name> line indicates that the next lines will configure the corresponding interface of the router.

The **router** <protocol> line indicates that the next lines will configure the corresponding routing protocol of the router. Each **redistribute** <protocol> line instructs OSPF to send the routes known to this router through the corresponding protocol to its neighbouring routers. Static routes are the routes that are “hardcoded” (statically configured) in the router’s routing table. Connected routes are those learned from directly connected routers. RIP routes are those learned from the RIP protocol.

The **network** <interface_ip> **area** <area_id> line determines the routing area in which a router interface belongs. The IP 0.0.0.0/0 covers the entire address space, which means that it covers all the interfaces of the router. By default, IMUNES places every router in the 0.0.0.0 area.

Changing the bandwidth of an interface

To override the default OSPF cost of an interface you can change its default bandwidth using the **bandwidth** <kbps> command.

For example, you can set the bandwidth of the **eth1** interface that connects router5 to router3 to 1000 Kpbs by appending the line “**bandwidth 1000**” at the end of the **interface eth1** configuration block, as shown in Figure 10:

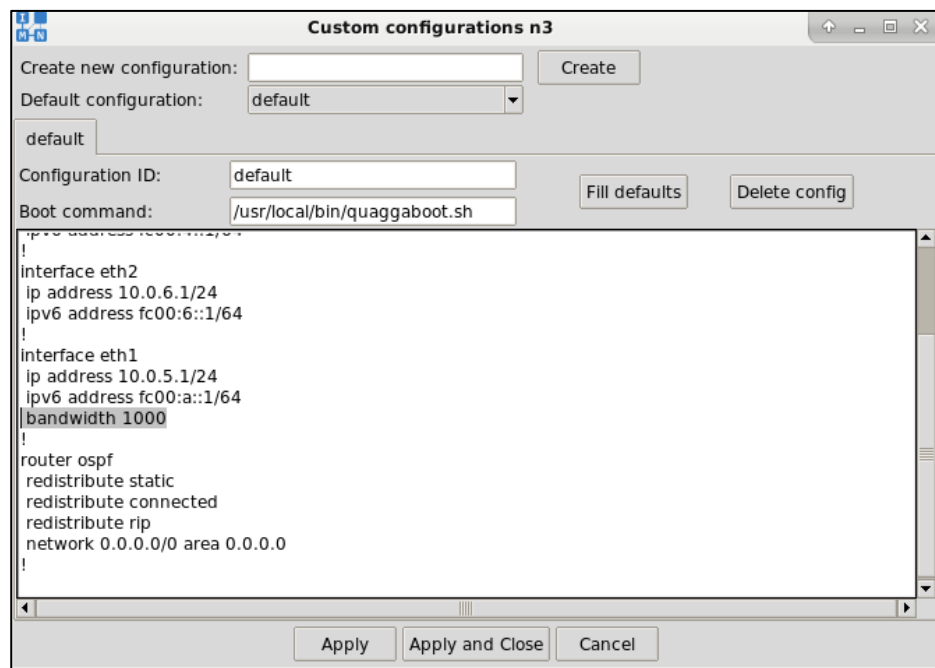


Figure 10: Setting the bandwidth of interface eth1 to 1000 kbps

Note that in your case the interface name may be different.

After clicking 'Apply and Close' to both configuration windows, re-run the emulation and query again the routing table of router5. Now the next hop to 10.0.7.0/24 has shifted from router 3 to router 4. If you stop router 4 while the emulation is still running, what is the new route cost to 10.0.7.0/24?

Note that the bandwidth command only sets an informational parameter to the routing protocol; you cannot adjust the actual bandwidth of an interface using this command.

Setting the cost of an interface directly

The bandwidth command does not allow you to the bandwidth of an interface to a value higher than its physical bandwidth. Therefore, the maximum bandwidth you can assign to an IMUNES interface is 10000 Kbps (10 Mbps). Any value higher than that will have no impact on the cost calculation.

Since it is not possible to set the bandwidth interface to a value higher than 10000 Kbps, it is not possible to lower the cost of an interface by setting its bandwidth. However, in certain scenarios, you may need to achieve finer-grained control of the interface cost than what the bandwidth command permits.

The `ip ospf cost <value>` allows you to set directly the cost of an interface to a value between 1 and 65535. For *router5*, set the cost of interface eth0 to 4 by adding the line "`ip ospf cost 4`" in the interface's configuration block, as shown in Figure 11:

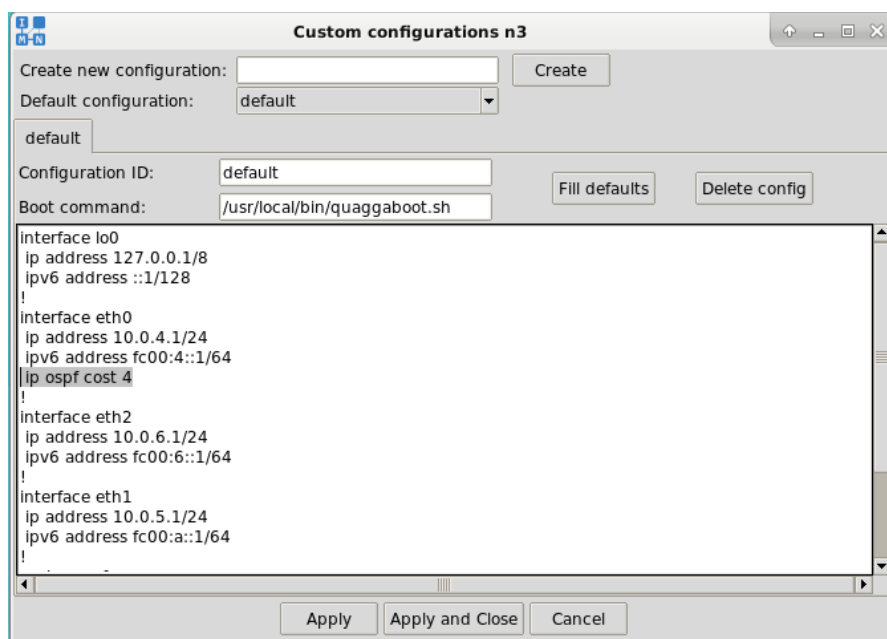


Figure 11: Setting directly the OSPF interface cost

Re-run the emulation and check router5's routing table. What are the new costs?

Configuring OSPF routing areas

OSPF uses areas to simplify administration and optimize traffic and resource utilization.

An area is simply a logical grouping of routers. All routers in the same area have the same topology table and do not know about routers in the other areas. The main benefits of using areas in an OSPF network are that the routing tables on the routers are reduced, and the OSPF algorithm requires less time to run.

When a network is divided in multiple areas, each area in an OSPF network must be connected to a backbone area or a backbone router that provides reachability among all others routing areas.

Figure 12 shows a way to partition our topology to two separate areas that are connected through *router3* which functions as a backbone router.

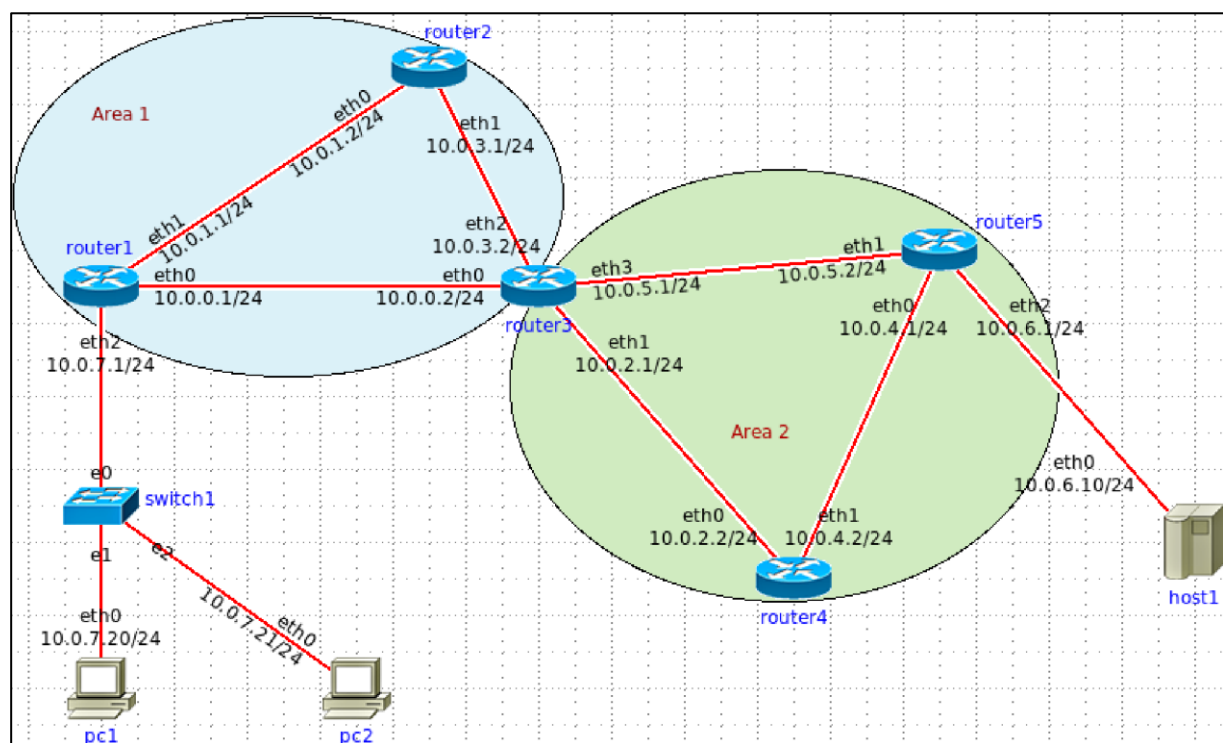


Figure 12: An OSPF topology with two areas and a backbone router

To achieve the partitioning of Figure 12 you can leave the routers in Area 2 in the default area (0.0.0.0), and create a custom startup configuration for the routers in Area 1 to place them in a different area.

Follow the steps listed in page 11 to create the default custom startup configurations for *router1* and *router2*. Then change the line “network 0.0.0.0/0 area 0.0.0.0” to “network 0.0.0.0/0 area 1.0.0.0”. Now these two routers will be in a logical routing domain with area ID 1.

If you run the emulation now you will notice that the routers in Area 1 cannot communicate with the routers in Area 2. This happens because you have not yet configured *router3* to operate as a backbone router. To do so, you need to assign the interfaces of *router3* connected to Area 1 routers with area ID 1, and the interfaces connected to Area 2 with area ID 0.0.0.0 (the default area ID).

```
interface eth2
ip address 10.0.3.2/24
ipv6 address fc00:3::2/64

interface eth3
ip address 10.0.5.1/24

router ospf
 redistribute static
 redistribute connected
 redistribute rip
 network 0.0.0.0/0 area 0.0.0.0
 network 10.0.0.2/24 area 1.0.0.0
 network 10.0.3.2/24 area 1.0.0.0
```

Figure 13: Dividing the interfaces of a router between two different areas

Edit the router configuration as shown in Figure 13 to add two extra commands with syntax:

network <interface_ip> area <area_id>

Now restart the emulation, and after waiting to converge check again the routing tables of the network routers. Make sure that the routers in Area 1 have paths to the routers in Area 2, and vice versa. You can also use ping and traceroute to check the connectivity.

Self-assessment tasks

The purpose of these tasks is only to help you test your understanding. They are not assessed and you do not need to submit a report.

Construct and configure a network

Construct the topology shown in Figure 14. You can add network devices, links, and annotation in your canvas using the menu bar on the left of the canvas (Figure 15).

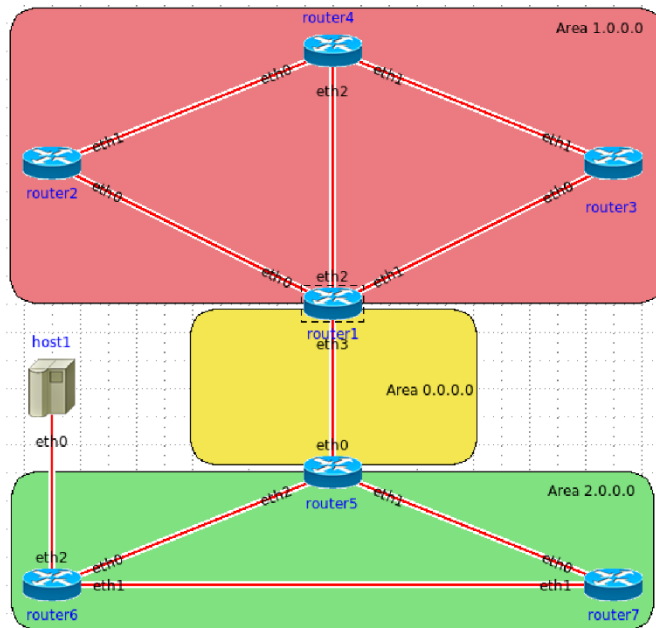


Figure 14: Re-construct the depicted topology in your IMUNES emulator

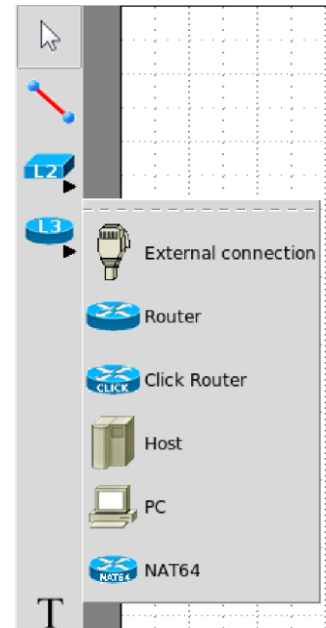


Figure 15: The menu of network devices that you can add in your emulation

Before adding routers to the topology, it is important to set the default routing protocol. To do this, select Tools → Routing Protocol Defaults from the main menu. Deselect both RIP options, and select the ospfv2 option, as shown in Figure 16. OSPFv3 is used for IPv6 addressing which we do not use in this task, so you don't have to select the ospfv3 option.

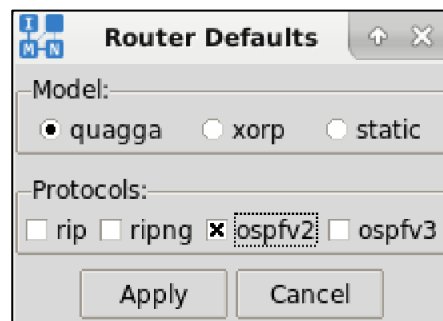


Figure 16: Setting the default router protocol to ospfv2

Configure OSPF costs

Configure the network so that:

1. *Router 4* reaches *host1* through *router 3*
 - a. Can you do the same without changing the configuration of *router 4*? Remember that OSPF costs are cumulative.
2. The bandwidth of *router 6* interface *eth0* should be lower than the bandwidth of interface *eth1*.
 - a. What is the impact of this change on the routing paths?

Configure OSPF areas

Configure the following areas in the topology:

- All the interfaces of *router 5*, *router 6* and *router 7* should be in area 2.0.0.0
- All the interfaces of *router 2*, *3*, *4* should be in area 1.0.0.0
- The interfaces *eth1* and *eth2* of *router 5* should be in area 2.0.0.0. The interface *eth0* should be in area 0.0.0.0
- The interfaces *eth0*, *eth1*, and *eth2* of *router 1* should be in area 1.0.0.0, while interface *eth3* should be in area 0.0.0.0

Check that the routers in area 1.0.0.0 can reach the routers and host in area 2.0.0.0

Network debugging

Open the configuration panel of *router 1* and change the IP interface of the *eth3* interface (the interface through which *router 1* connects to *router 3*) to 9.0.0.1.

Run a traceroute measurement between *router 4* and *host1*.

Does the traceroute path reach the destination? If not in, which hop does it fail?

Network Performance

Measure the bandwidth between *router 6* and *router 7* using *iperf*.

Now reduce the bandwidth of the *eth1* interface on *router 6* and measure the bandwidth again. Do you observe any substantial change in the two bandwidth measurements? Why?

Now reduce the MTU size of the connected *router 6* and *router 7* interfaces and measure the bandwidth again. Do you observe any change this time?

Inter-domain Routing with BGP

The Internet is a network of networks; it's broken up into hundreds of thousands of smaller networks known as **Autonomous Systems (AS)**. Each of these networks is essentially a large pool of routers run by a single organization. Autonomous Systems typically belong to ISPs or other large high-tech organizations, such as tech companies, universities, government agencies, and scientific institutions. Each autonomous system wishing to exchange routing information must have a registered **Autonomous System Number (ASN)**.

Typically, Internet traffic between two remote hosts traverses multiple ASes. To enable such end-to-end reachability ASes exchange routing information among each other using the **Border Gateway Protocol (BGP)**. BGP provides the allows network operators to implement complex routing policies and override shortest path routing so that they can tailor their paths to their business model and operational requirements².

In this lab you will experiment with BGP configurations in order to implement policy-based routing. You will learn how to:

- Set-up BGP routers and BGP sessions
- Announce IP prefixes through BGP
- Specify path preference when multiple paths are available to the same destination IP
- Selectively advertise prefixes to conform to AS business relationships

Now we do not want to use OSPF. To deactivate OSPF for every router by default, select *Tools* → *Routing protocol defaults* from the top menu bar, which opens the *Router Defaults* window. Make sure *quagga* is selected as the router model and uncheck all the *Protocols* checkboxes (Figure 17).

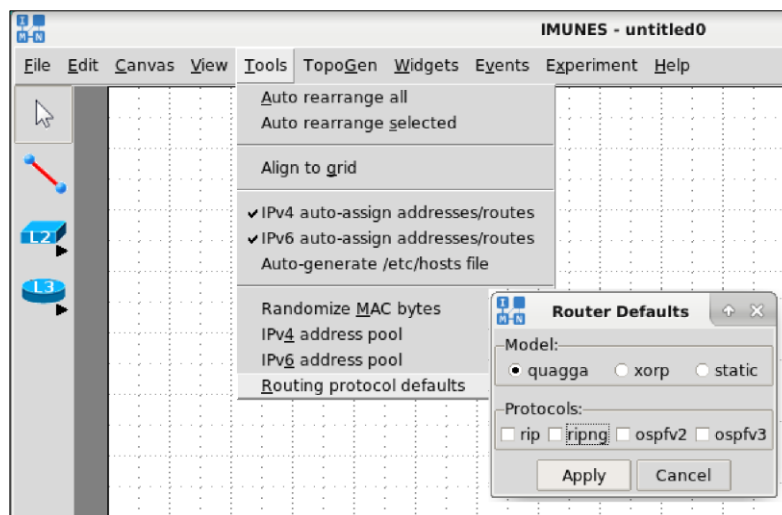


Figure 17: Deactivate all Protocols from the "Routing protocol defaults" menu

Now create the topology depicted in Figure 18. Note that the actual IP addresses may differ in your machine. In this topology we assume that each router belongs to a different AS. Since

² For a short visual introduction on BGP: <https://youtu.be/AIKXPqIqNZ4>

you deactivated all the routing protocols, the routers do not run any routing protocol by default.

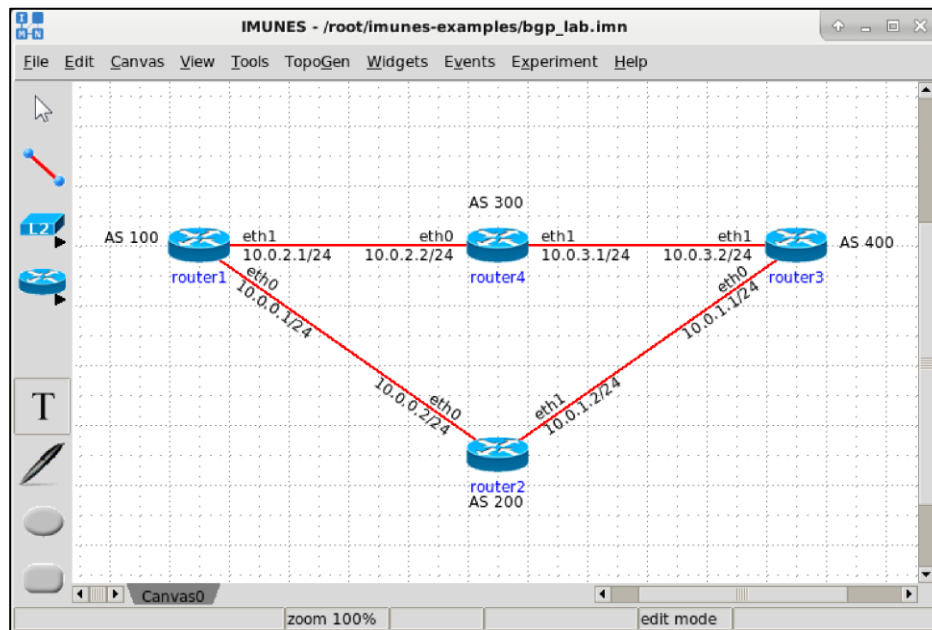


Figure 18: BGP toy topology

To enable BGP, in each router you need to enable and create a custom startup configuration like you did for OSPF. This time the default configuration will include not only interfaces. For instance, the defaults for router2 will resemble the one shown in Figure 19.

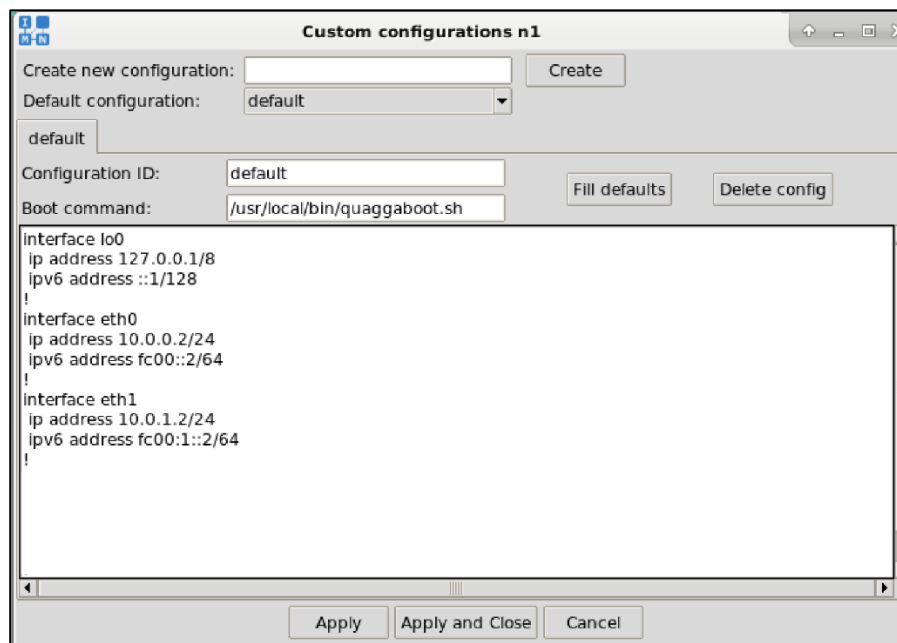


Figure 19: Defaults for custom startup configuration of router2 before enabling BGP

For router2, append the following lines at the end of the configuration to enable BGP in router2:

```

router bgp 200
 redistribute connected
 neighbor 10.0.0.1 remote-as 100
 neighbor 10.0.1.1 remote-as 400
 !

```

The **router bgp <as-number>** command enables BGP and assigns the AS number (ASN) to the local router. In this case, you assign the ASN 100 to router2.

The **redistribute connected** command instructs the router to advertise the IP address it learns through one connected router to another connected router.

The **neighbour <ip-address> remote-as <as-number>** creates configures the IP address and the ASN for a remote BGP router connected to the local router to establish a new BGP session.

Now try to configure accordingly the rest of the routers in order to enable BGP. Assign to each router the appropriate ASN and establish the BGP sessions depicted in *Figure 18*. You can find the correct configuration for each router in Appendix I, but before checking the solution first try to configure the routers yourself to test your understanding.

After you complete the setup for all the routers, execute the emulation (from the top menu choose *Experiment* → *Execute*). Then open the vtysh terminal to view the routing table (right-click on a router icon → *Shell window* → *vttysh*).

You can use the **show ip route** command to list the table entries, but for BGP it does not provide all the details. Instead, you can get a detailed BGP routing table (*Figure 20*) using the command:

show ip bgp

```

IMUNES: router4 (console) vtysh
Hello, this is Quagga (version 1.2.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

router7# show ip bgp
BGP table version is 0, local router ID is 10.0.3.1 ①
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

② Network      ③ Next Hop      Metric LocPrf Weight Path ④
* 10.0.0.0/24   10.0.3.2         1         0 400 200 ?
*>             10.0.2.1         1         0 100 ?
* 10.0.1.0/24   10.0.2.1         1         0 100 200 ?
*>             10.0.3.2         1         0 400 ?
* 10.0.2.0/24   10.0.2.1         1         0 100 ?
*>             0.0.0.0         1        32768 ?
* 10.0.3.0/24   10.0.3.2         1         0 400 ?
*>             0.0.0.0         1        32768 ?
* 10.0.4.0/24   10.0.2.1         1         0 100 200 ?
*>             10.0.3.2         1         0 400 200 ?

Displayed 5 out of 10 total prefixes
--More--(END)

```

Figure 20: The BGP routing table of router 4

❶ The **local router ID** field shows the ID of the router, which usually corresponds to its interface with the highest value.

❷ The **Network** column shows the destination IP prefix of each path.

❸ The **Next Hop** column shows the IP interfaces of the remote router that is the next hop along the path.

❹ The **Path** column shows the sequence of ASes along the path to the destination. Note the difference between OSPF and BGP on how routes are propagated among routers in a network. An OSPF router propagates only the link cost to a destination. In contrast, BGP propagates the full AS path. This is why OSPF is a *link state* protocol, while BGP is a *path vector* protocol.

BGP Path Selection Process

Note that for some IP prefixes the router has more than one paths available. For instance, prefix 10.0.1.0/24 is reachable both through the path AS100 – AS200, and through AS400. The greater-than symbol > points to the most preferable route.

When more than one paths are available to the same IP destination, BGP applies the following steps to rank paths. The process stops at the step for which there is no tie.

1. Prefer the path with the highest **Weight**: The default weight for learned routes is 0 and the default weight for a locally originated route is 32768.
2. Prefer the path with the highest **LocPrf** (Local Preference): If multiple paths to the same destination have the same Weight, BGP uses the LocPrf attribute to break ties. By default, LocPrf has a value of 0 or 100.
3. Prefer the shortest path in terms of AS hops.
4. Prefer paths learned through an intra-domain routing protocol (IGP) over paths received from a different AS (namely paths learned through external BGP session).
5. Prefer the path with the **lowest Metric**: If a router has multiple paths to the same destination with the same next-hop AS, and their attributes above are equal, BGP uses the Metric attribute to break ties.
6. The path that was learned first (oldest path) is preferred.
7. Prefer the route that comes from the BGP router with the lowest router ID. The router ID is the highest IP address on the router.

You can check details on a BGP route using the command:

```
show ip bgp <prefix>
```

Figure 21 shows the details for the BGP paths to 10.0.0.2 for *router4*. The *Last Update* field corresponds to the date and time the route was installed in the routing table. There are two paths to 10.0.0.2/24. Both have all their attributes equal including their path length. Therefore, the path that was learned first is selected as the best path, which is the path “400 200”.

```

Hello, this is Quagga (version 1.2.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

router4# show ip bgp 10.0.0.2
BGP routing table entry for 10.0.0.0/24
Paths: (2 available, best #2, table Default-IP-Routing-Table)
  Advertised to non peer-group peers:
    10.0.2.1
  100 200
    10.0.2.1 (metric 1) from 10.0.2.1 (10.0.2.1)
      Origin IGP, localpref 100, valid, external
      Last update: Wed Mar  4 11:59:06 2020
  400 200
    10.0.3.2 (metric 1) from 10.0.3.2 (10.0.1.1)
      Origin IGP, localpref 100, valid, external, best
      Last update: Wed Mar  4 11:59:03 2020
--More--(END)

```

Figure 21: The output of the "show ip bgp <prefix>" command with the Last Update date annotated

Note that the precision of the last update field is in seconds while the actual differences between update times can be milliseconds, which may cause sometimes the misleading impression that two paths are updated at the same time.

Now let's restart router3 and examine the routing table of router4 again. While the emulation is in execution mode, right click on router3 and click *Restart*. Re-run the following command in the vtysh terminal of router4:

```
show ip bgp 10.0.0.2
```

You should observe that the best path switched to "100 200", since now it is the path that is considered the oldest known path. A second restart of router3 will not affect the routes. This example illustrates why BGP prefers the oldest path: if a neighbouring router or link is prone to failures and interruptions it won't affect connectivity.