Moanin

Kévin "Chewie" Sztern

Dec 10, 2020

CONTENTS:

CHAPTER

ONE

ABOUT THIS PROJECT

1.1 *Moanin'*

Welcome to the first NET1 project!

Like all my projects, the name comes from a song I like (or a song I'm listening to while writing the subject), and continuing the tradition of jazz songs for networking projects, this one is named Moanin'.

Moanin' is a jazz album by Art Blakey and the Jazz Messengers, and also the name of the first track of said album. Initially, the album was just titled "Art Blakey and the Jazz Messengers", but due to the popularity of the first track, the whole album came to be known as "Moanin'". A textbook example of the hard bop genre, this is typically the album I recommend to someone looking for a bluesy intro to the structure of jazz (after Kind of Blue, of course). Nevertheless, I hope moaning won't be what you do while you work on this project!

1.2 What you have to do

The main goal of this project is to install and configure GNS3, and learn how to configure the networking of a Linux box.

Sounds complicated? Don't worry! This subject will (hopefully) guide you through it!

CHAPTER

TWO

SETTING UP GNS3

2.1 The installation

First things first, you need to install GNS3!

There are two main ways of installing GNS3: with a helper VM that will run the appliances or without (only on Linux).

The appliances you will manipulate run on Docker, and configuring it can be tricker than expected. As such, I heavily recommend that you make use of the GNS3 VM no matter what, since it comes with everything preconfigured to run such appliances. I won't help you with your install if you want to configure Docker yourself!

GNS3 can be downloaded on the official site: https://gns3.com. The site also contains installation instructions if you're on Linux depending on your distribution. Don't forget to also download the VM appliance with the hypervisor of your choice (if in doubt, the VirtualBox one works fine)!

2.2 Installing the appliances

For the entirety of this course, we will only make use of three appliances : a end host, a switch, and a router. Let's see how to add them!

2.2.1 The host

The host is a custom appliance based on a docker image made by yours truly, and as such you need to create it manually. You can either click on the "new template" button on the bottom left and select "manually create a new template", or directly go to the settings of GNS3. There, you will find a tab for Docker based templates, and a button to add one.

When asked for a docker image, use the following:

```
chewiebeardy/gns3-host:2023
```

The ":2023" part is a tag to enforce a specific version, since I sometimes update the images with breaking changes from year to year.

I suggest you name it "Host", with only one adapter (it's an end device, after all), and leave all other configurations by default.

The appliance should appear in the "End Devices" section on the drawer on the left, try dragging it to your project and starting it!

2.2.2 The router

Adding the router is very similar to the host, except the image is the following:

```
chewiebeardy/gns3-router:2023
```

Name it "Router" (what a surprise), and make it have 4 adapters, it should be enough for our purposes (real life routers have dozens of ports, of course).

Afterwards, there are some quality of life configurations I recommend by asking to edit the template:

- Make it belong to the "Routers" category instead of the default "End Devices" one
- Change its symbol to visually distinguish it. The classic pack has an icon named "router" that does the trick.

2.2.3 The switch

Contrary to the first two appliances, we will not use a custom switch, but one already available!

Go ahead and click on "New template" on the bottom left, but this time choose "Install an appliance from the GNS3 server". On the next page, you will find an appliance named "Open vSwitch".

OpenvSwitch is an extremely popular virtual switch for Linux, and you will find it in many open source networking projects!

2.3 Exporting your project

In order to submit your project, you need to export it in the portable GNS3 format, which is basically a zipfile containing everything.

To do that, you can find the "Export portable project" in the "File" menu from the toolbar.

When asked if you would like to include base images or snapshots, choose "no" for both, the questions are not relevant for docker appliances.

If you're using version 2.2 or above, you can choose the compression method used. I advise you pick the default "Zip compression (deflate)".

Name the file *<login>-moanin.gns3project*, where *<login>* is your login. This is the file you will submit through Moodle. Be careful: it's "moanin", not "moaning"! Respect Art Blakey dammit.

Before submitting it, I strongly advise you to try importing it first and testing that everything works correctly, you can never be too sure! I'm not responsible for errors on your parts and won't help you if you forgot to test your project before submitting it.

CHAPTER

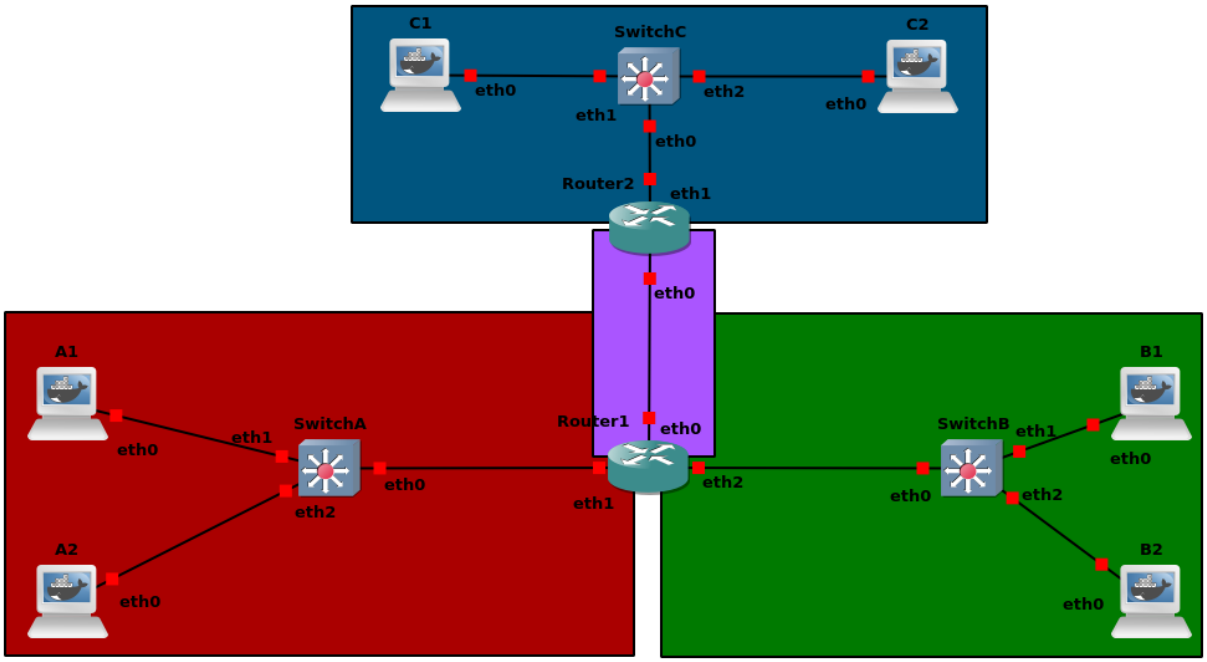
THREE

ON WITH THE PROJECT!

3.1 Creating the topology

Every project starts on the drawing board, let's create the machines and plug them before configuring them!

The topology I ask of you is this:



While subsequent subjects will be provided with a skeleton project containing the nodes, I want you to manually draw a topology at least once!

For added clarity, I added fancy colorful rectangles to represent each network, but you don't have to do it on your end. However, but **must** use the same names as me, and plug the cables on the same ports! I won't be able to grade you otherwise.

To make things clearer, you can activate the "show interface labels" option in the "View" menu from the toolbar.

Your goal is simple: every machine should be able to talk to every other machine, no matter how far! To do that though, you'll need to learn how to configure the network on the machines…

3.2 Configuring a machine

Linux being Linux, there are many ways of configuring the network depending on the distribution you're using, and even now we're no closer to an unified solution (I hope https://netplan.io will catch on beyond Ubuntu and win though).

To be more precise, configuring the network "on the fly" on a shell is more or less standardized, and everyone uses iproute2[1] for that. But persisting configuration files… that's a whole 'nother game. Welcome to configuration hell.

As such, don't understand the following section as "the way to configure networking on Linux" but one of the ways, one that historically is used by debian-based distributions and that GNS3 uses to configure the appliance before launching them (as in, technically the appliances don't configure themselves but GNS3 injects a tool to configure it on startup, but this is a minor technical detail you can ignore).

Here, the configuration will be done through the "/etc/network/interfaces" file present on each machine. GNS3 should have injected an empty one with commented out directives to make it easier.

For more information on this configuration format, have a look at the manpage[2]:

Note that the tool injected by GNS3 is a very simplified version that doesn't handle all the advanced features, so stick with the basics!

The four basic fields of interest are the following:

- "address": to specify the address, duh.
- "netmask": the network mask, in quad notation. This is **not** optional, remember that it doesn't make sense to configure an IP without it!
- "gateway": the default gateway, as in the route matching "0.0.0.0/0". Not all machines will have a default gateway, depending on your topology. Gateway is an old term synonymous with "router", they can be used interchangeably. Of course, there can only be at most one default gateway across all interfaces.
- "up": this takes an arbitrary command that will be executed once the interface is up, very useful to add an extra route for example, wink wink nudge nudge.

The "up echo nameserver" directive can be safely deleted, this is for DNS and we're not playing with DNS for now.

There are a couple of GNS3-specific quirks you should be aware of:

- Because this file isn't read by the appliance but by GNS3 itself, running "ifup" like the docs say won't work. I suggest you just restart the appliance to apply the new configuration. (If you must know, GNS3 injects a busybox in */tmp/gns3/bin* and runs this ifup, but it's a very barebones one and will fail for complicated reasons if you launch it manually. Just restart the appliance)
- For the same reason, if an error is present in the */etc/network/interfaces* file, the error will appear *before* the whole boot log text in the terminal. See the attached image for an example.

Note that switches shouldn't be configured this way: they are layer 2 devices, they don't care about IP addresses! No configuration of the switches is necessary for this project.

[1] https://baturin.org/docs/iproute2/
[2] https://manpages.debian.org/jessie/ifupdown/interfaces.5.en.html

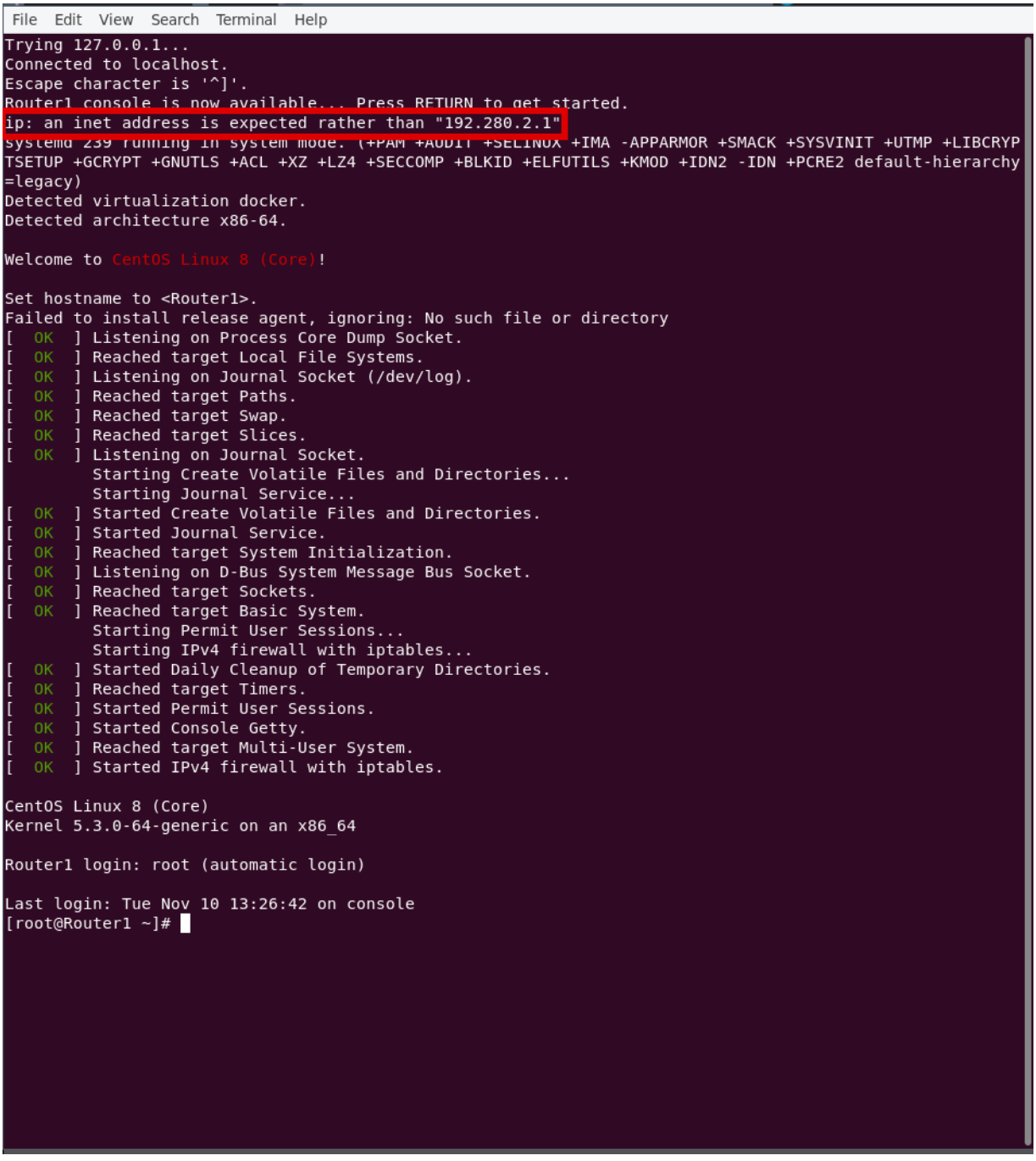


Fig. 1: errors with the configuration file appear up there.

3.3 Three little networks

Let's start small. Before talking about interconnecting, each network should work on its own and between its members! No default routes or static routes for now, just direct routes.

Each network has a prefix unique to it, and all machines inside this network have an address that begins with this prefix (and know about the prefix length, of course! Otherwise the machines cannot determine if they're talking locally or need to reach a distant network).

You will find in the following tables the addresses of each device on those simple networks. Fun fact, the three prefixes I chose are special use prefixes that are reserved for example and documentation, known in the RFC 5737 as TEST-NET-1, TEST-NET-2 and TEST-NET-3).

Name	IP Addresses
A1	• 192.0.2.2/24 (eth0)
A2	• 192.0.2.3/24 (eth0)
B1	• 198.51.100.2/24 (eth0)
B2	• 198.51.100.3/24 (eth0)
C1	• 203.0.113.2/24 (eth0)
C2	• 203.0.113.3/24 (eth0)
Router1	• 192.0.2.1/24 (eth1) • 198.51.100.1/24 (eth2)
Router2	• 203.0.113.1/24 (eth1)

Now login to a machine and type "ip address" (or "ip a" for short), you should see the address added to the correct interface!

More interestingly, try "ip route" (or "ip r" for short), and look at the output. What do you see?

For each address that you add, a route with "scope link" is automatically added. This is known as a "direct route", "local route", or "connected route", and its meaning is simple to decipher: if you want to talk to a neighbor (i.e. someone sharing the same prefix as you), you can send it on this interface!

Try pinging other machines in your network (including the router). At this stage, inside those three networks every machine should be able to communicate with their local neighbors.

Remote communication is still not configured though. Try pinging a distant machine. What error message do you get?

3.4 Working remotely in these unprecedented times

It's time to augment the *routing table* of our machines, and give them more than just direct routes!

Each time a machine tries to send a packet, it inspects its routing table and tries to match the entries (made up of network prefixes) with the destination IP, selecting the most specific match (i.e. the one with the longest prefix). If no entry is found, you get the wonderful "Network is unreachable" you should have seen earlier.

Non-direct routes (those not marked with "scope link") always specify a router to which the packet should be sent, so that the router repropagates this packet on the next network. A distant communication is just a succession of local communications! Of course, the indicated router must be reachable from a direct route, otherwise you get a chicken and egg problem.

Let's start with the end devices as they are trivial: since they only have one network interface, no matter the destination, the packet can only be sent through that interface.

As such, the only route we need to add is a default route, a catch-all entry with the lowest priority possible. The formal entry for the default is "0.0.0.0/0" (think about the longest-prefix match algorithm to see why this entry works as such), but everyone calls it the "default route" and most programs will show it as such.

Time for practice: add a default route to all the end devices. Depending on the device, which IP should you choose as the default gateway?

Now, can we talk to distant networks?

Funnily enough, this is enough for A and B to communicate! Since "Router1" is directly connected to both networks, it knows, through its direct routes, how to repropagate the packets sent from each of them, as long as the destination is one of them.

But reaching C might take a bit more work...

3.5 The legend of the fourth network

An easy trap to fall into would be to think that the topology has only three networks, but there are four! Remember that all machines connected to a switch are part of the same network, but routers act as boundaries between networks. As such, the link between the two routers is a network on its own!

This may sound a bit strange to you, but it's a network like all others, it's just that is has only two persons on it. This is what we call a "point-to-point network".

Just like the other networks, the two machines on it have an IP address for this network, and those two addresses share the same prefix.

Still, there is one useful trick you should be aware of: given that usually two addresses are reserved on a network (the "all zeroes" network address and the "all ones" broadcast address), the minimum prefix length should be a */30*, to allow four addresses in total. This is a 50% waste however, even more so given that broadcasting on a point-to-point link doesn't do much more than contacting the other peer!

For this reason, RFC 3021[3] allows one exception to this "always make room for two more" rule: for a point-to-point link, you can use a */31* instead of a */30*, and forget about network and broadcast addresses.

In this subject, the addressing must be as following:

[3] https://tools.ietf.org/html/rfc3021

Name	IP Addresses
Router1	• 42.42.42.42/31 (eth0)
Router2	• 42.42.42.43/31 (eth0)

Is this enough for A and B to reach C?

Not quite! "Router1" is directly connected to this point-to-point link and knows about it, but has absolutely no idea that a "203.0.113/24" network exists, since it's not directly attached to it! The same can be said of A and B for Router2, of course.

We need to explicitly inform the routers about the distant networks and tell them through which interface they can be reached. How do we do that? Through routes, of course!

3.6 Country route, take me home...

Let's start with "Router2". This router is connected to a stub network, that is to say a cul-de-sac. In such cases, the routing configuration should be fairly obvious: as long as it's not for the local network "203.0.113.0/24", the only way possible for the packet is through the other interface, eth0. This screams "default route"! Go ahead and add it, I trust you to find what IP to put as default gateway ;)

In some sense, we can say that Router2 is "lower" in the hierarchy than Router1, and defers all unknown routing matters to it. You can interpret a default route as saying "I don't know how to handle this one, take care of it!" and trusting the next router in the chain to know more (or defer again, and again...).

Router1 is an interesting case, because there are two valid ways to configure the routes, depending on the interpretation:

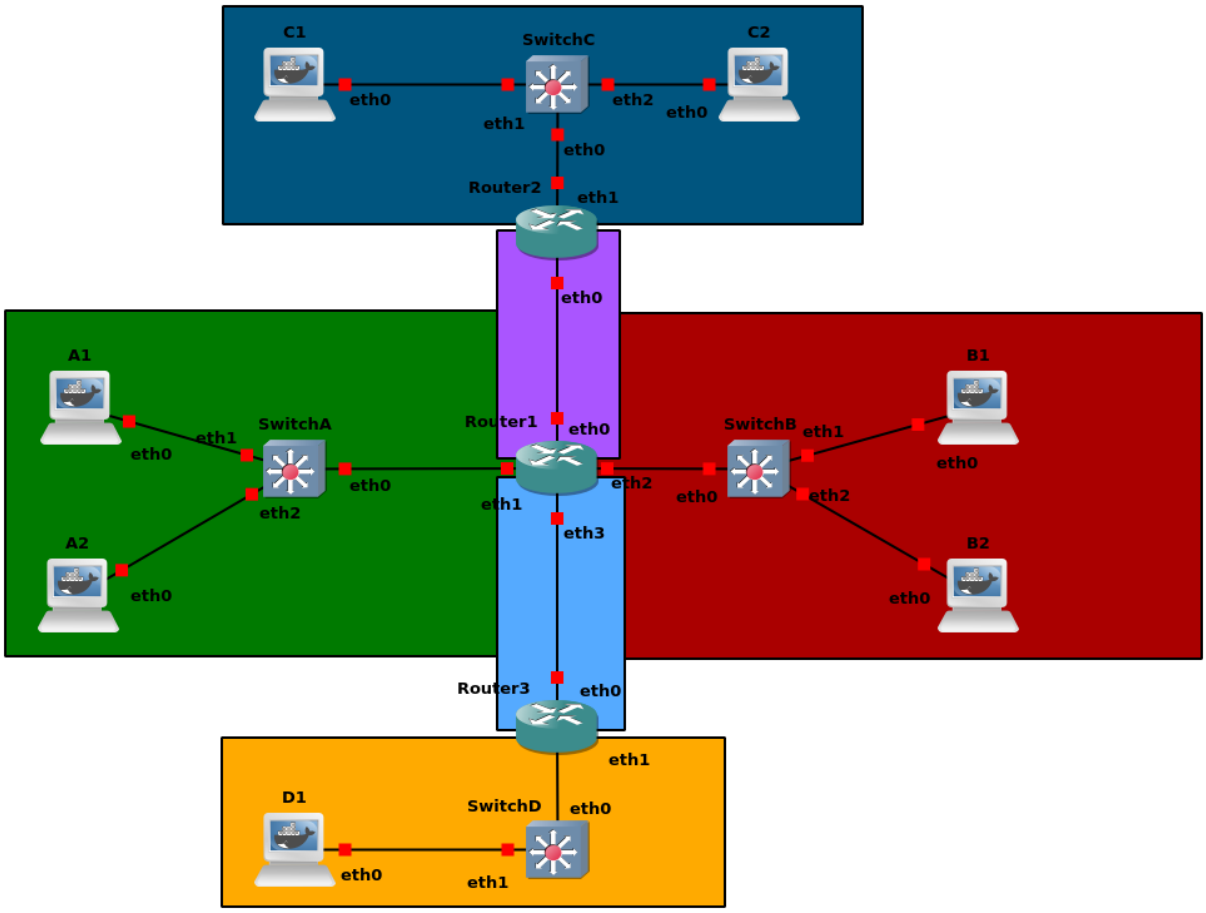
In this universe, there are only four networks in the world, and since Router1 already has route to three of them, the default route is strictly equivalent to a route exactly matching the fourth.

You could finalize the configuration and reach global connectivity by adding a default route via Router2, but let's think of the implication in terms of design. Does this configuration scale?

There is only one way to make this kind of problems visible, and that requires me to play a dirty, dirty trick on you, please forgive me...

3.7 This is getting out of hand, now there are 6 of them!

Surprise, the topology changed! A new network has been added, and the topology now looks like this:



Before you ask: yes, you **have** to include these new elements in your submission.

Don't be scared, it's only three new machines and it shouldn't take you more than 5 minutes to configure them.

Here is the addressing requested:

Name	IP Addresses
D1	• 128.66.42.2/24 (eth0)
Router3	• 42.42.42.51/31 (eth0) • 128.66.42.1/24 (eth1)
Router1	• 42.42.42.50/31 (eth3)

The routes for D1 and Router3 should be pretty obvious, their situation is the same as the C network.

Now, I didn't do that to make you angry, but because something very interesting has happened: having only a default route on Router1 doesn't work anymore!

What should be the default route for Router1 then?

Can't think of one? That's normal: there's not really a "upper router" just like Router1 is to Router2 and Router3. Router1 is at the top of the chain. No asking to the powers above!

Sure, you could just add an explicit case for D, and still rely on the default route to reach C, but what makes C so special as to deserve to be the default? It's a lower network just like the others!

In these situations, we say that Router1 lives in a "default-free zone", or has a "default-free routing table". That means exactly what you think it means: such routers must know **all** the routes explicitly, without relying on a default route! Many real-life routers are placed in such a situation and must hold the entirety of the Internet global routing table. As of writing this subject, this table holds approximately 850,000 prefixes[4]!

What we need to do, and the last stage of this project, is to remove the "gateway" field, and instead add *static routes* to Router1 To inform it of the C and D networks.

Static routes are just like the default route in that they specify a router that will transmit the message, but they match on a specific prefix. The term "static" here comes from the fact that those routes are "hardcoded" in the configuration and not discovered dynamically by some fancy protocol that is out of scope for now ;)

Unfortunately, the */etc/network/interfaces* file lacks a specific way to represent those routes, so we're stuck with adding them manually through the "up" field and using iproute2 command inside. Refer to the documentation linked in the footnotes and you should be golden!

Of course, such "up" commands should be put on the appropriate interface: if you add a route through eth0, don't do it through the "up" of eth1!

[4] See https://bgp.potaroo.net/ for other fun metrics

CHAPTER

FOUR

CONCLUSION

With this, you should have reached global connectivity, and every machine can ping every other machine no matter the network. Give yourself a pat on the back, you now know more about networking than 90% of the engineers in the industry!

I hope you found this project interesting and entertaining, and more importantly that you learned a lot through it. I did my best to make it as much of a "guided tutorial" as possible to ease the initial pain. The next projects will be less guided, but you should have no trouble handling them by then!