Group 2
William André
Bastien Claeys

Alexandre Jadin
Thomas Marissal
Mathieu Xhonneux

LINGI2142 — Computer networks: configuration and management

Project report

Introduction

This report describes how we implemented the different parts of our campus network, and the motivations behind our implementation choices. To know how to run and test our network, please refer to the `README` file provided with it.

1 Topology, addressing and multihoming

This section shall describe how we decided to organize our network, what addressing plan we chose, where the different services are hosted, the multi-homing feature, ...

1.1 Addressing plan

In our network, subnets prefixes are constructed following this pattern (each cell represents 4 bits, hence a hexadecimal character) :

f	d	0	0		0	{2,3}	0	0		0	0	0	2		U	L	R	R	::/64

U represents the type of machines connected to the subnet. Only the following values are valid, each of them corresponding to a specific type of network user :

- 0 : Routers and Services (DHCP, DNS, Web, etc)
- 1 : Staff
- 2 : Students
- 3 : Guests
- 4 : Phones (VoIP)

Except for the use 0, all others uses need a specific VLAN for the subnet to work. Specific firewalls and QoS policies are set for each type of users, but these are defined in the relevant sections of this report.

L represents a location in the network. A location could contain subnets in the same building but belonging to different routers, or a set of geographically close routers, ... If lots of routers are added, and thus subnets, in order to keep the routing tables small, it could be possible with this plan to advertise the networks in other locations through with IGP /56 prefixes, instead of advertising all the /64 subnets.

RR represents the ID of the router, and is location-dependent. Thus, each router can only have one subnet per user type (a router having two prefixes for students is not possible).

There are mainly three things to discuss concerning this plan :

- We didn't reserve bits at the beginning of the prefixes for differentiating multiple campus sites. It is a deliberate choice, thus if a second site should be added, it couldn't share the same /48 prefixes.
- We decided to put **U** before **L**. This means that our routing tables will be longer since each router will advertise up to 5 /64 subnets (instead of 1 /52 per location if **L** were before **U**), but the advantage is that our firewall policies will contain much lesser rules. Since generally, the firewall is slower than the packet routing mechanisms, we decided it was more important to reduce the number of firewalling rules.
- Both the 200 and 300 prefixes follow the same structure, but we shall see shortly after that they're not used in the same way.

1.2 Network topology

The whole network topology is described in figure 1.

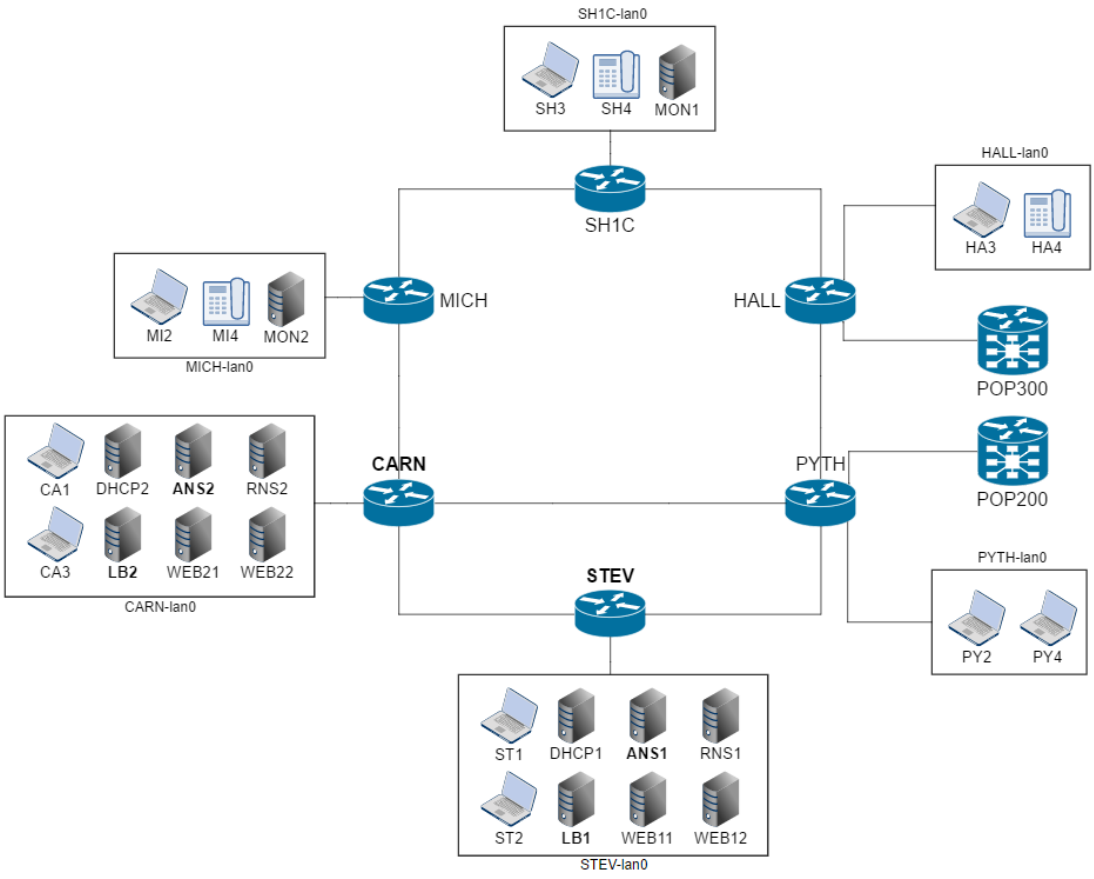

Figure 1: Network topology

This topography is fully described in the *routing/lans.conf* configuration file, also available in listing 8.1 (in the annexes). Most of our scripts use this file to derive subnets, VLANs, IP addresses, Bird configurations, ...
Two aspects of the addressing plan are not described in this figure : locations and uses.

- *SH1C* and *HALL* belong to location 0, *PYTH* and *STEV* to location 1, and finally *CARN* and *MICH* to location 2.
- Per router, all 5 subnets (each for one usage) are not always activated. Only the default use 0 is mandatory, but the VLANs (1, 2, 3 and 4) must be set in the *routing/lans.conf* configuration file.

All laptops and phones correspond to end-host users. For instance, CA1 belongs to a staffer and CA3 to a guest. MI2 belongs to a student, MI4 is a VoIP phone, etc. The number always corresponds to the type of user. These machines receive an IP address through DHCP.

All others machines (except the routers) are servers hosting services, and are described in the following sections. It is important to mention that by default, as indicated in *routing/lans.conf*, all machines only receive subnets from the fd00:200:2::/48 prefix. Only the machines with their name in bold own subnets from fd00:300:2::/48. The reason why is explained in the following subsection.

1.3 Leveraging the multi-homing possibility

Since our network is connected to two different autonomous systems, each of them delegating a different /48 prefix, we had to find a way to leverage this opportunity.

We arbitrary decided to treat the AS 300 link as a failover link for the web services. By default, all traffic (of end-hosts and all services, except for the DNS service, which we shall discuss in its respective section) goes through the 200 link. When this link goes down, a fail-over mechanism is activated. This ensures that our web services are still available from the outside, such that the web traffic is routed through the 300 link.

This type of configuration would make sense if the peering with AS 300 is really expensive, thus only allowing the web services to be rescued, and not the end-host users (who are not critical). This is why only the ANS1, ANS2 (authoritative DNS servers) and LB1, LB2 (HTTP load-balancing servers) have IP addresses in the fd00:300:2::/48 prefix.

This multi-homing configuration uses some features which will be described in the routing, DNS and monitoring parts of this report.

2 Routing (Mathieu Xhonneux)

Routing is done using the Bird routing daemon. Static IP addresses, LANs and routes are automatically set-up by the *routing/setup_lan.py* script, which itself uses the *routing/lans.conf* configuration file to derive all (V)LANs, IP addresses and default routes. This script is called by each machine, either host or router, when it's starting.

In a similar fashion, all Bird configurations files are also generated dynamically using the *routing/bird6.conf.tpl* template. Before the networks starts, the script *routing/gen_birdconfs.py* is executed to generate these files. This script also uses the *routing/lans.conf* configuration file to generate proper on-the-fly Bird configurations.

2.1 Internal routing

The internal routing is managed using OSPF, which is a Link-state routing protocol. RIP could also have been a good choice for our setup, since the network is quite small (only 6 routers), but we chose OSPF because it is theoretically faster to respond to link failures.

All of our 6 routers use OSPF. The OSPF configuration itself is quite straightforward. Only the /64 prefixes of each router's LANs are advertised, and the default route to Internet provided by one of the two BGP-connected routers. Proper import and export filters have been set. All

OSPF communications are point-to-point, no authentication has been used, and proper timing values has been used. This assures that if one internal link fails, all on-line OSPF routers will quickly adapt to the failure and update their routing tables accordingly.

2.2 External routing and multihoming management

The external routing is managed using BGP, but its configuration is a bit particular because we wanted to leverage the multihoming functionality, as explained previously. Both HALL and PYTH maintain an active BGP connection to their respective peer and advertise the fd00:{200,300}:2::/48 prefixes over BGP. We define two states in which the network can be :

Default state : This state is running when PYTH is online and has a valid BGP peering with the AS 200. This state has the following properties :

- Only PYTH advertises the ::/0 prefix internally to the other routers, so all outgoing Internet traffic is routed to him
- PYTH has static source routing rules defined to redirect all traffic coming from fd00:300:2::/48 towards the Internet to HALL.
- HALL has static source routing rules defined to redirect all traffic coming from fd00:300:2::/48 towards the Internet to its pop300 interface.
- HALL can be online or offline. If it is offline or without a valid BGP peering, only the fd00:300:2::/48 traffic will be impacted, but in this state, only DNS requests can be carried through the AS 300 peering.

Failover state : This state is running when PYTH is offline or when it has no valid BGP peering with the AS 200. This state has the following properties :

- Only HALL advertises the ::/0 prefix internally to the other routers, so all outgoing Internet traffic is routed to him. Yet only outgoing traffic coming from fd00:300:2::/48 will be routed correctly.
- PYTH doesn't advertise the ::/0 prefix.

Both PYTH and HALL have, for each state, their own Bird configuration files. The monitoring servers MON1 and MON2 are responsible to determine in which state the network is, and switch the Bird daemons to the proper configuration files if needed.

The source routing policies are available in the *project_ cfg/HALL_ start* and *project_ cfg/PYTH _ start* files.

3 End-users management

The end-users management is divided in two main parts. The first part is the address assignments and the second one the domain name resolution.

3.1 Dynamic address assignment (Thomas Marissal)

For dynamic address assignement, several possibilites were available : SLAAC only, SLAAC + stateless DHCPv6, DHCPv6 only, SLAAC (gateway) + DHCPv6, ...

After some researches, we chose the DHCPv6 server + SLAAC for gateway advertising solution. We considered that a DHCPv6 server was needed, at least stateless, to indicate a TFTP server to the VoIP phones. Besides, having a DHCPv6 server offers much more control on our network (some addresses can be forbidden, others can be statically assigned to a specific peripheral, some

MAC addresses can be banned, ...) than the SLAAC protocol provides.

In order to simplify the configuration of our DHCPv6 servers, we also installed RA daemons on each router, indicating to each host the gateway. DHCPv6 provides thus only an IP address and the DNS servers, whereas SLAAC provides the gateway. A full DHCPv6 solution was of course also possible.

To make the dynamic assignment failure resilient, we have installed two DHCPv6 servers, DHCP1 and DHCP2. Each DHCP request is transferred to both servers, and if possible, both servers will respond. The client has then to choose which IP address he will use. To avoid conflicts, each DHCP server has per subnet its own pool of IP addresses (DHCP2 has all addresses ending in the $[1, 10]$ range, and DHCP1 has $[11, 20]$, these pools can easily be extended). This is not the most elegant solution, but it works as flawlessly as a master/slave setup and is perfectly robust if any of the two servers fails.

As demonstration, 12 DHCP hosts are configured, through the 4 users types and with 2 per router. Each of them receives correctly the gateway and a valid IP address.

3.1.1 VLANs and functioning

Each end-user has its own type (staff, student, guest or phone), and must be allocated an IP address in the corresponding subnet. In order to being able to differentiate the different types of clients connected to a router, each router sets up VLANs according to the `routing/lans.conf` file. For instance, *CARN* sets two VLANs, 1 and 3. CA1 is connected to VLAN 1 and CA3 to the VLAN 3.

The drawback of this configuration is that the DHCP servers are only accessible in the LAN it belongs to, thus not the VLANs. To make both servers accessible from all routers and from all VLANs, we had to setup relays on all the outers. We used `dhcrelay`. Each relay listen to the configured VLANs on the router and then dispatches the requests to both DHCP servers.

3.1.2 SLAAC

To indicate the gateway to our hosts, we use the SLAAC protocol with the `radvd` daemon on each router. Its configuration is minimal : it only broadcasts itself as gateway, and sets minimal and maximal time spans between two unsolicited advertisements.

3.2 DNS servers (Mathieu Xhonneux)

4 DNS servers are configured in our network. All use Bind9's *named*. ANS1 and ANS2 are authoritative name servers, whereas RNS1 and RNS2 are two recursive name servers.

3.2.1 Authoritative Name Servers

Both servers are authoritative on the `group2.ingi.` and `0.2.0.0.0.0.0.2.0.0.0.d.f.ip6.arpa.` zones. AAAA records are set on the first zone, and PTR records on the latter. Each router or server on the network has its own AAAA and PTR records (`lb1.group2.ingi`, `dhcp2.group2.ingi`, `stev.group2.ingi`, ...). All zones are described in the `db.group2.ingi` and `db.200.ingi` files in ANS1 configuration folder, TTL's values are either 1 one week or 1 day (thus these records are not supposed to change often).

By default, the `www.group2.ingi` domain is served by two AAAA records, one pointing to fd00:200:2:103::4, the other to to fd00:200:2:204::4, which respectively correspond to LB1 and LB2. This allows a DNS round-robin setup where the load of web requests should be divided equally to LB1 and LB2. It is important to note that these IP addresses belong to the fd00:200:2::/48 prefix, thus all web requests are routed through the POP200 link.

Moreover, ANS1 and ANS2 are configured in a master/slave fashion, where ANS1 is the master, thus making them failure resilient. ANS1 notifies ANS2 of any changes in the records, which happens if the state of the network changes.

3.2.2 Recursive Name Servers

RNS1 and RNS2 are two recursive name servers. Their configuration is simple, they forward all incoming DNS requests to fd00::d, ANS1 and ANS2, and cache them. Besides, an ACL has been set in order to accept only queries from within our internal network (thus coming from fd00:200:2::/48 or fd00:300:2::/48). Both servers are totally independent.

All machines of our network (including the ones configured by DHCP) use these two name servers.

3.2.3 Multi-homing management

As described in the routing section, the state of the network can change from *default* to *failover*, if PYTH or the AS 200 links fail. If this happens, all web traffic must be routed through the POP300 link.

As stated previously, by default the `www.group2.ingi` record points to the load-balancers, using the fd00:200:2::/48 prefix. MON1 and MON2 are allowed by ANS1 to send Dynamic DNS updates. Once a monitoring server detects that the AS 200 link failed, he will withdraw the two 200 AAAA records to `www.group2.ingi` from ANS1 and replace them with two AAAA records pointing to the same IP addresses, but using the fd00:300:2::/48 prefix (and vice-versa if the link goes back up). ANS1 will then notify ANS2 of the change.

Since the TTL on `www.group2.ingi` is 1 minute, this ensures that all web clients will quickly be redirected through the AS 300 link.

4 Services (Alexandre Jadin)

In this project, we configured two types of services in our network : SSH and web servers using load-balancing. This section shall briefly describe how we configured them.

4.1 SSH

SSH service is provided by `openssh` and is configured on all routers, servers and hosts of our network. All SSH servers share a common configuration, which itself is quite classic, yet we'd like to stress some specific points.

Firstly, we created a bash script, `ssh/initialize.sh` whose goal is to generate for each SSH server its master keys and configuration file. It then creates, again for each server, a pair of RSA keys, whose public key is authorized by the corresponding server. These client keys can then be used to log in to any machine in the network, and are also copied to MON1 and MON2 for monitoring purposes.

We also made some modifications to the default `sshd_config` file :

- `ListenAddress ::`, the server only listens to IPv6 addresses
- The paths to the `HostKey` and `AuthorizedKeysFile` have been changed in order to benefit from the NetNS /etc/ functionality, so each server can use its own master keys and authorized keys
- `PermitRootLogin no`, disallowing SSH connections as root to improve security
- `PasswordAuthentication no`, for the same reason

Details about how to connect with SSH are explained in the `README` file.

4.2 HTTP service

We chose `nginx` as web server. It's a solid choice, since it is robust, has high-performance and offers all the functionalities we need. Yet, the Debian package doesn't offer IPv6, SSL and load-balancing, so we have to compile nginx ourselves with these extensions at the VM's creation.

The setup is quite straightforward. As detailed on image 1, we have two main datacenters, and each datacenter contains one load-balancing server (LB1, LB2) and two web servers (WEB11, WEB12, WEB21, WEB22).

The WEB** servers are configured to display only one static page (`<h1>This page has been served by web11.group2.ingi !</h1>`), and the load-balancers are configured to dispatch the HTTP requests they receive to the static web servers in the same datacenter, with the same priority (or weight). It's important to note that we have thus two level of redundancy : one at the DNS level, and one at the HTTP load-balancing level.

Concerning multi-homing, both LB1 and LB2 have an IP address in the fd00:300:2::/48 range. This makes them reachable from both AS 200 and 300, according to the current DNS configuration.

5 Security (Bastien Claeys)

The security of our network is handled by `ip6tables`, a module of `Netfilter` which implements firewall within Linux kernel. This part will talk about the rules we deployed in the network to prevent specific kind of attacks.

Preamble `ip6tables` works with three default *chains* : `INPUT`, `OUTPUT` and `FORWARD`, corresponding to the packets incoming, outgoing and passing by the host. They will be mentioned later and they allow to easily apply the right behavior for the right packet. Concerning the use itself of `ip6tables`, one must know that rules follow precedence, each packet will be tested through a set of rules in a specific order until it matches one or reach the default policy.

5.1 Main rules

Each router owns his own file in the folder `firewall`. Each file contains the rules deployed at the start of the network. Before going into the details, please note that in a general way, we have the following rules :

White-listing policy It seems more careful to allow only some packets rather than to forbid others with the risk to forget to filter some illegal packets. We thus use a white-listing policy, that is, each router has a DROP default policy for each chain.

Existing connections Is allowed the traffic noted as `ESTABLISHED` and `RELATED` from anywhere to anywhere as well as the local traffic. This stateful firewall involves that existing connections will be allowed.

Multihoming Every rule has been set taking into account the use of multihoming.

ICMPv6 Ping packets are allowed inside the network for everyone, but from the outside only the border routers can be "pinged".

5.2 Border routers and traffic from the outside

Our first concern is to protect the network from external attacks. Rules have been set up to allow traffic from outside the network only in some cases. To achieve this, the border routers, `HALL` and `PYTH`, look upon every packet coming from `pop300` or `pop200` respectively. Here are the packets we allow from these interfaces :

Spoofed IP addresses Since these packets come from the outside, they are not supposed to have a source address containing the prefix of our network (`fd00:200:2::/48` and `fd00:300:2::/48`). We thus drop all of them because it can only mean that their IP addresses have been spoofed by an attacker.

BGP, OSPF and DHCP packets In order to allow connection to the Internet, we must allow the BGP packets addressed to our border routers. Concerning the OSPF and DHCP packets used to configure the network, there is no need to allow them to leave it, or to allow others to enter it. They are both however allowed to be forwarded anywhere else in the network.

Outside traffic Concerning the services requested from the outside, we allow SSH connections, SMTP, DNS, HTTP/HTTPS.

5.3 Users policy

As mentioned previously, we have five types of user in our network : routers/services, staff, students, guests and phones. Here are the policies deployed for each one of them.

5.3.1 Routers/services policy

Limited access to routers Unless the user types `0 (services)` and `1 (staff)`, nobody should have access to the routers. This is why we prevent any guest, student or phone to access to it. Only ICMPv6 packets can be received from anyone.

Multicast address for DHCP Packets going to `ff02::1:2` are allowed. DHCP makes use of it to communicate with neighboring (i.e., on-link) relay agents and servers `[RFC 3315]`.

Traceroute The packets passing by ports used by `traceroute` are accepted by the firewall.

Restricted ports on hosts Each host has only its used ports open. By default we disabled any incoming traffic going to ports that should not be used.

5.3.2 Staff policy

Full access This kind of user is for administrators and thus has full access.

Monitoring servers Only the staff can access the monitoring servers MON1 and MON2.

5.3.3 Students policy

Services allowed We allow the students to use SSH, DNS, HTTP/HTTPS, SMTP and VoIP.

Not a host We do not want students to be able to connect to other users. Thus each router forbids students to connect to `guests`, other `students`, or `phones`.

5.3.4 Guests policy

Strongly restricted We constrained the access to the network for the `guest` users. They are only allowed to access to four hosts : the load balancers, `LB1`, `LB2` and the DNS servers `RNS1` and `RNS2`. They are not allowed to access to the internet, only our own website. Their packet will not be forwarded if they don't concern these hosts.

5.3.5 Phones policy

VoIP Only the DNS and VoIP ports are activated for the phones.

5.4 Monitoring packets

The logs of ip6tables have been activated through `ulogd`. The logs of all packets are gathered at `/var/log/ulog/syslogemu.log` and are easily generated each time a packet matches with a line containing `-j NFLOG -nflog-prefix "<text>"` in an iptables rule. In our network, we print a log each time a packet is dropped with the default policy.

5.5 Security check

Some tests are provided in `firewall/run_tests.sh`. They aim to verify the policies deployed for different kind of users. We did not replicate the possibles attacks but a bunch of tools are available to test the configuration of the network. You can easily see that `ping6` should be allowed anywhere and towards any router (`ping6 carn.group2.ingi`), host (`ping6 ca2.group2.ingi`) or website (`ping6 google.be`), but not from the outside, to prevent the discovery of our network topology. On the other hand, you can check that the ports are open through `nmap`, and get a web page with `wget` or also try to establish a connection to a determined port with `netcat`. For instance, a connection can be established by running `nc6 -l -p <portNumber> -v` on one instance and `nc6 -6 <address> <portNumber> -vvv` on another.

6 Quality of Service (William André)

We implemented QoS policies in order to prioritize some types of traffic. This section shall describe what types of traffic we want to prioritize, how we splitted them into classes and the final three we came up with.

6.1 Classes

6.1.1 Critical traffic

The most critical packets are VoIP packets because it is really difficult for humans to have a conversation with delays and information loss. This traffic should have a high priority and doesn't need any fairness since they should quit the queue immediately. If they aren't sent immediately, they could as well be dropped. VoIP connections are not heavy so we can easily give them the highest priority.

An other kind of prioritised traffic is the one for security cameras. The QoS tree built supports it, but since there aren't any cameras on the network, we suppose that the <USE> bits of the IPv6 addresses reserved for these cameras equal 5.

The last general priority given is from staff over other users. The staff should be able to use the network even in case of heavy load.

6.1.2 Destination

Since we have a link with 5 Gbps for "commercial" purpose and 10 Gbps for "research" purpose, the queuing policies should be adapted. Therefore, there we divide the network queuing in 3 parts : internal, toward research networks (other universities,...) and other destinations (commercial). Internal network gets the highest priority, with up to 50% of the mawimal bandwidth. The commercial traffic gets 33% of the traffic not used by internal traffic, therefore that leaves 66% for the research traffic, which is the same ratio as the one from the uplinks.

If the traffic is internal for a ssh connexion, it also gets a higher priority since it can be used for maintenance.

The filter used for the research traffic is a dummy one because we don't know which IP are given to this kind of traffic.

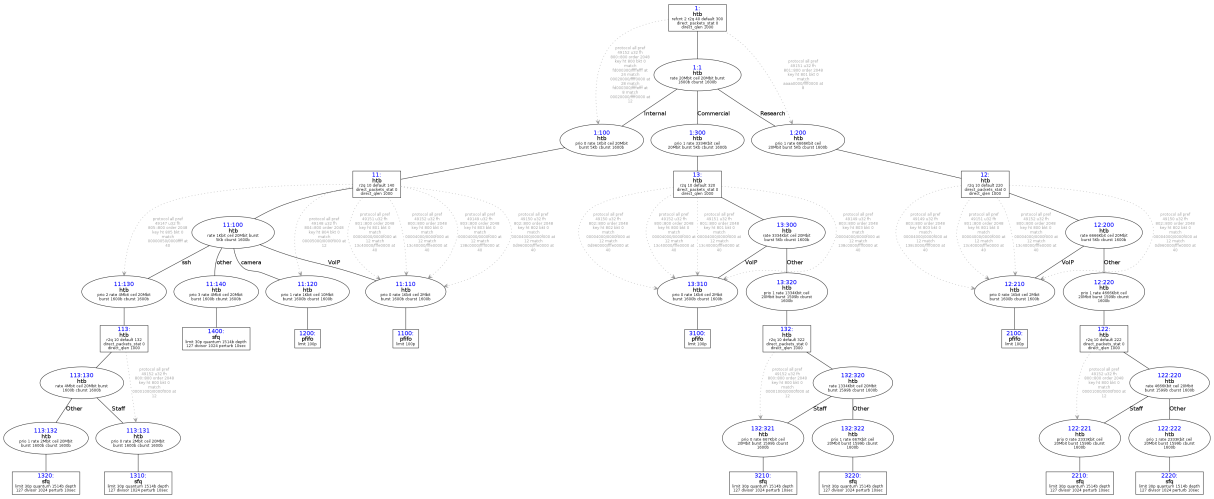

Figure 2: Entire queuing tree

6.2 Queuing disciplines

The general discipline used is *Hierarchy Token Bucket*. On the leaves of the tree, there are two kind of disciplines : *pfifo* and *Stochastic Fairness Queuing*.

Pfifo is used only for VoIP with a limited queuing size since the packets are useless if they are sent with too much delay.

SFQ is used for everything else, helping having more fairness by preventing any single flow from drowning out the rest.

6.3 Starvation and free bandwidth

Since we use SFQ and every class is guaranteed to get at least a certain amount of bandwidth, no starvation should occur. The only place where a limit on bandwidth is set if for VoIP (because it gets the highest priority), but the limit should be above the maximum utilisation of the service, which should be a lot less than the maximum capacity of the link. Of course, a burst is allowed to let packets like HTTP request and responses flow.

6.4 Tree

You can see the whole tree used for the QoS on Figures 2, 4 and 3. The subtree dedicated to network toward research nodes is very similar to the one toward commercial nodes and the zoom is therefore not given. The only difference is the minimum rate allowed to the root. The rate of the children is in the same proportion as for the commercial subtree.
All the rates displayed on the figures are automatically computed given a link of maximum 20 Mbps.

6.5 Testing

The tests consisted on using `iperf` or `nc` coming from different IPs and ports and `watch -dc tc -p -s -d qdisc show dev $INTERFACE` on the router in order to see if the traffic was going in the right flow and if the limits were respected. The fact that the visualization of the tree is correctly generated from a script (see tcviz[1]) also indicates that the tree is well built.

[1] `https://github.com/ze-phyr-us/tcviz`

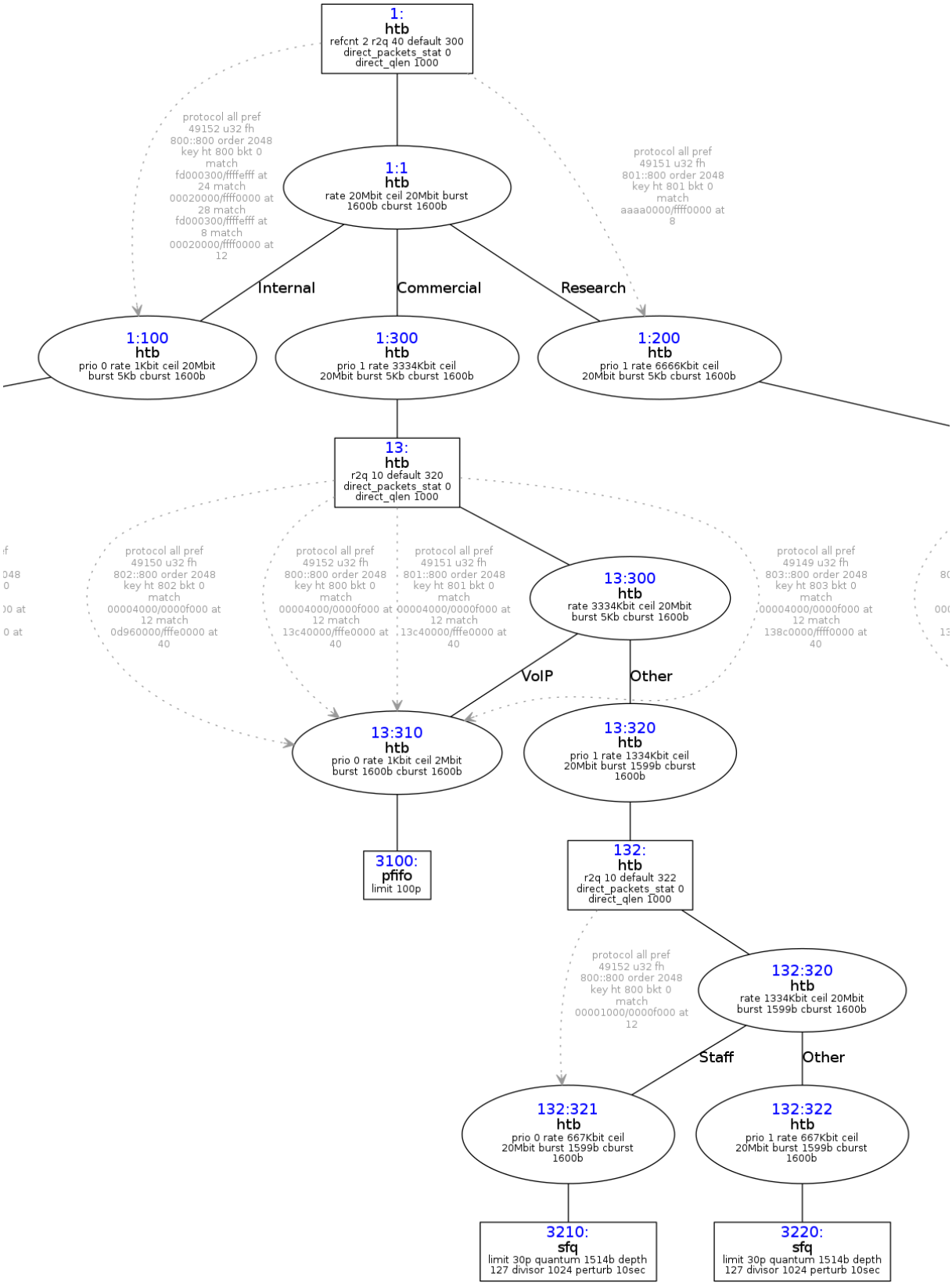

Figure 3: Commercial network part of the tree

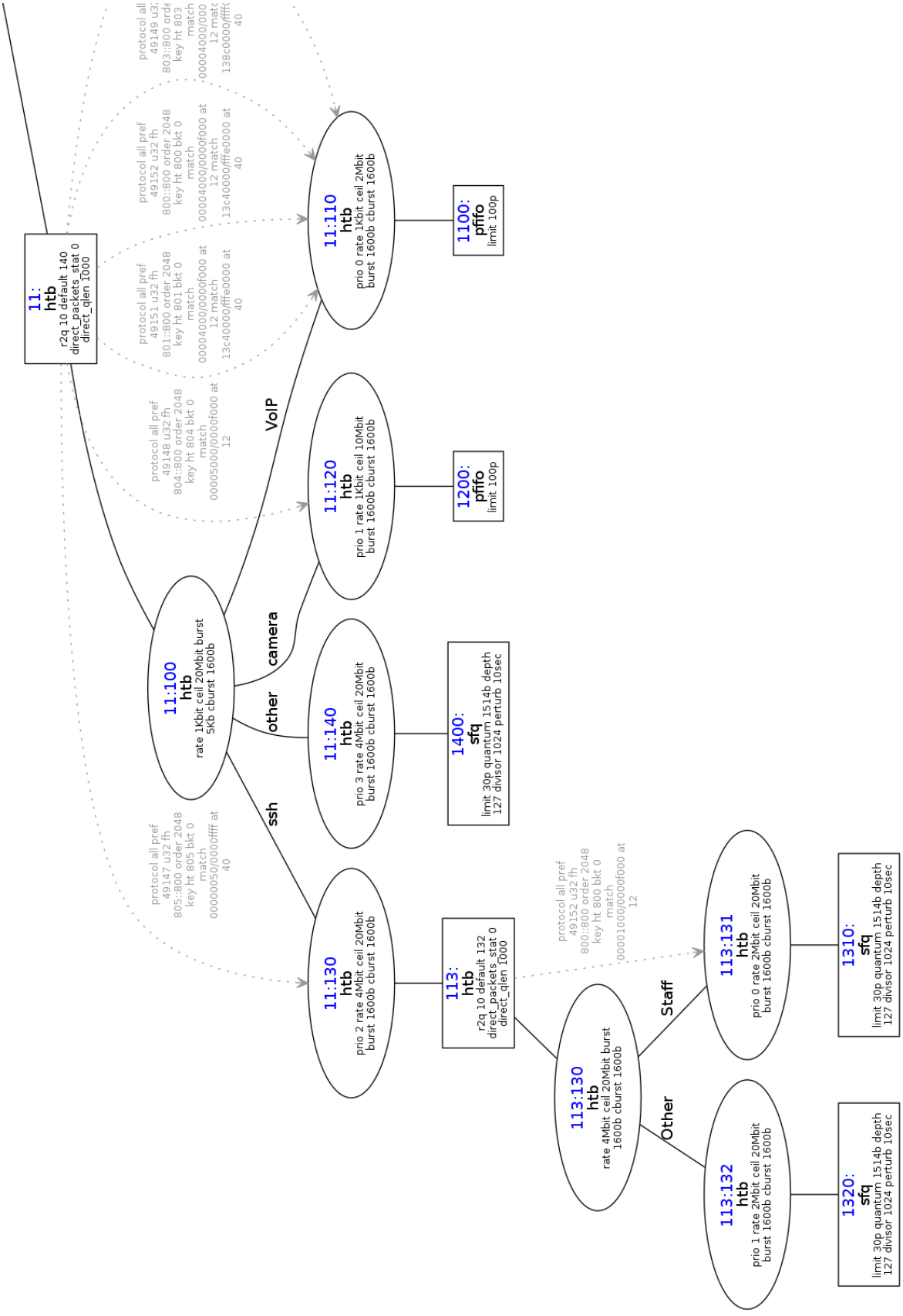

Figure 4: Internal network part of the tree

7 Monitoring (Mathieu Xhonneux)

We have set up two monitoring servers, MON1 and MON2, outside the two main datacenters. They are mainly responsible for assuring our multihoming failover feature.

Both server run the script `monitoring/failover.sh`, which maintains the network's state and checks, each 10 seconds, if PYTH has a BGP peering with AS 200. If the script notices that the peering has just been lost or recovered, it changes the network to the correct state, by updating HALL's and PYTH's Bird daemons configurations, and the 2 AAAA www records of ANS1 (and thus ANS2 since ANS1 is its master).

Furthermore, a web server is running on `mon1.group2.ingi` and `mon2.group2.ingi`. Both serve the same monitoring status report, which includes the following information :

- The current network's state
- The results of the command `dig @ns1.group2.ingi www.group2.ingi AAAA`
- For each router, the results of the two following `birdc` commands : `show protocols all` and `show ospf neighbors`.

For security purposes, this page can only be reached from a staff user on the network (thus CA1 or ST1). This web page is updated through the `monitoring/web.sh` script and an example of this page is provided next to this report.

The monitoring servers communicate to the routers through SSH, and to ANS1 through the Dynamic DNS protocol.

8 Bibliography

Routing

[1] http://bird.network.cz/?get_doc&f=bird.html

[2] https://gitlab.labs.nic.cz/labs/bird/wikis/BGP_example_2

[3] http://www.tldp.org/HOWTO/Adv-Routing-HOWTO/lartc.rpdb.simple.html

DNS

[1] https://www.garron.me/en/go2linux/how-setup-dns-server-master-slave-bind.html

[2] https://www.sixxs.net/faq/dns/?faq=reverse

Services

[1] http://nginx.org/en/docs/beginners_guide.html

[2] https://doc.ubuntu-fr.org/ssh

Security

[1] https://tools.ietf.org/html/rfc3315

[2] https://www.ietf.org/rfc/rfc4890

[3] https://home.regit.org/2014/02/logging-connection-tracking-event-with-ulogd/

[4] http://connect.ed-diamond.com/GNU-Linux-Magazine/GLMFHS-041/Ulogd2-journalisation-avancee-avec-Netfilter

[5] https://rlworkman.net/howtos/ulogd.html

[6] http://ipset.netfilter.org/iptables-extensions.man.html

[7] https://www.jumpingbean.co.za/blogs/mark/set-up-ipv6-lan-with-linux

[8] https://www.jethrocarr.com/2013/02/09/ip6tables-ipv6-icmp-vs-icmp/

[9] https://www.digitalocean.com/community/tutorials/how-to-set-up-a-firewall-using-iptables-

[10] https://www.g-loaded.eu/2006/11/06/netcat-a-couple-of-useful-examples/

Annexes

8.1 *routing/lans.conf* configuration file

```
{
  "SH1C" : {"type":"router", "location":0, "id":0, "lans":[0, 3, 4], "as":[200], "links":2},
  "HALL" : {
            "type":"router", "location":0, "id":1, "lans":[0, 3, 4], "as":[200], "links":2,
            "bgp":{"source_ip":"fd00:300::2", "neighbor_ip":"fd00:300::b", "neighbor_as":300, "use":"failover"}
            },
  "PYTH" : {
            "type":"router", "location":1, "id":2, "lans":[0, 2, 3], "as":[200], "links":3,
            "bgp":{"source_ip":"fd00:200::2", "neighbor_ip":"fd00:200::b", "neighbor_as":200, "use":"default"}
            },
  "STEV" : {"type":"router", "location":1, "id":3, "lans":[0, 1, 2], "as":[200, 300], "links":2},
  "CARN" : {"type":"router", "location":2, "id":4, "lans":[0, 1, 3], "as":[200, 300], "links":3},
  "MICH" : {"type":"router", "location":2, "id":5, "lans":[0, 1, 2, 3, 4], "as":[200], "links":2},

  "DHCP1": {"type":"host", "router":"STEV", "sub_id":1, "lan":0, "as":[200]},
  "ANS1" : {"type":"host", "router":"STEV", "sub_id":2, "lan":0, "as":[200, 300]},
  "RNS1" : {"type":"host", "router":"STEV", "sub_id":3, "lan":0, "as":[200]},
  "LB1"  : {"type":"host", "router":"STEV", "sub_id":4, "lan":0, "as":[200, 300]},
  "WEB11": {"type":"host", "router":"STEV", "sub_id":5, "lan":0, "as":[200]},
  "WEB12": {"type":"host", "router":"STEV", "sub_id":6, "lan":0, "as":[200]},

  "DHCP2": {"type":"host", "router":"CARN", "sub_id":1, "lan":0, "as":[200]},
  "ANS2" : {"type":"host", "router":"CARN", "sub_id":2, "lan":0, "as":[200, 300]},
```

```
  "RNS2" : {"type":"host", "router":"CARN", "sub_id":3, "lan":0, "as":[200]},
  "LB2"  : {"type":"host", "router":"CARN", "sub_id":4, "lan":0, "as":[200, 300]},
  "WEB21": {"type":"host", "router":"CARN", "sub_id":5, "lan":0, "as":[200]},
  "WEB22": {"type":"host", "router":"CARN", "sub_id":6, "lan":0, "as":[200]},

  "MON1" : {"type":"host", "router":"SH1C", "sub_id":1, "lan":0, "as":[200]},
  "MON2" : {"type":"host", "router":"MICH", "sub_id":1, "lan":0, "as":[200]},

  "ST1" : {"type":"host", "router":"STEV", "sub_id":"dhcp", "lan":1, "as":[200]},
  "ST2" : {"type":"host", "router":"STEV", "sub_id":"dhcp", "lan":2, "as":[200]},

  "CA1"  : {"type":"host", "router":"CARN", "sub_id":"dhcp", "lan":1, "as":[200]},
  "CA3"  : {"type":"host", "router":"CARN", "sub_id":"dhcp", "lan":3, "as":[200]},

  "HA3"  : {"type":"host", "router":"HALL", "sub_id":"dhcp", "lan":3, "as":[200]},
  "HA4"  : {"type":"host", "router":"HALL", "sub_id":"dhcp", "lan":4, "as":[200]},

  "MI2"  : {"type":"host", "router":"MICH", "sub_id":"dhcp", "lan":2, "as":[200]},
  "MI4"  : {"type":"host", "router":"MICH", "sub_id":"dhcp", "lan":4, "as":[200]},

  "SH3"  : {"type":"host", "router":"SH1C", "sub_id":"dhcp", "lan":3, "as":[200]},
  "SH4"  : {"type":"host", "router":"SH1C", "sub_id":"dhcp", "lan":4, "as":[200]},

  "PY2"  : {"type":"host", "router":"PYTH", "sub_id":"dhcp", "lan":2, "as":[200]},
  "PY3"  : {"type":"host", "router":"PYTH", "sub_id":"dhcp", "lan":3, "as":[200]}
}
```