Universidade de Aveiro

departamento de electrónica, telecomunicações e informática

Arquitetura de Redes Avançadas

ISP Network Design

Project 2017/2018

8240 - mestrado integrado em engenharia de computadores e telemática

Diogo Ferreira
NMec: 76425 | P1G12

Pedro Martins
NMec: 76551 | P1G12

2017-2018

Table of Contents

Chapter

Introduction

In the context of the course of Arquitetura de Redes Avançadas, we were proposed to design and to configure an Internet Service Provider (ISP) level network. The network specifications and engeneering choices will be explained in the following chapters.

Chapter

Network Design

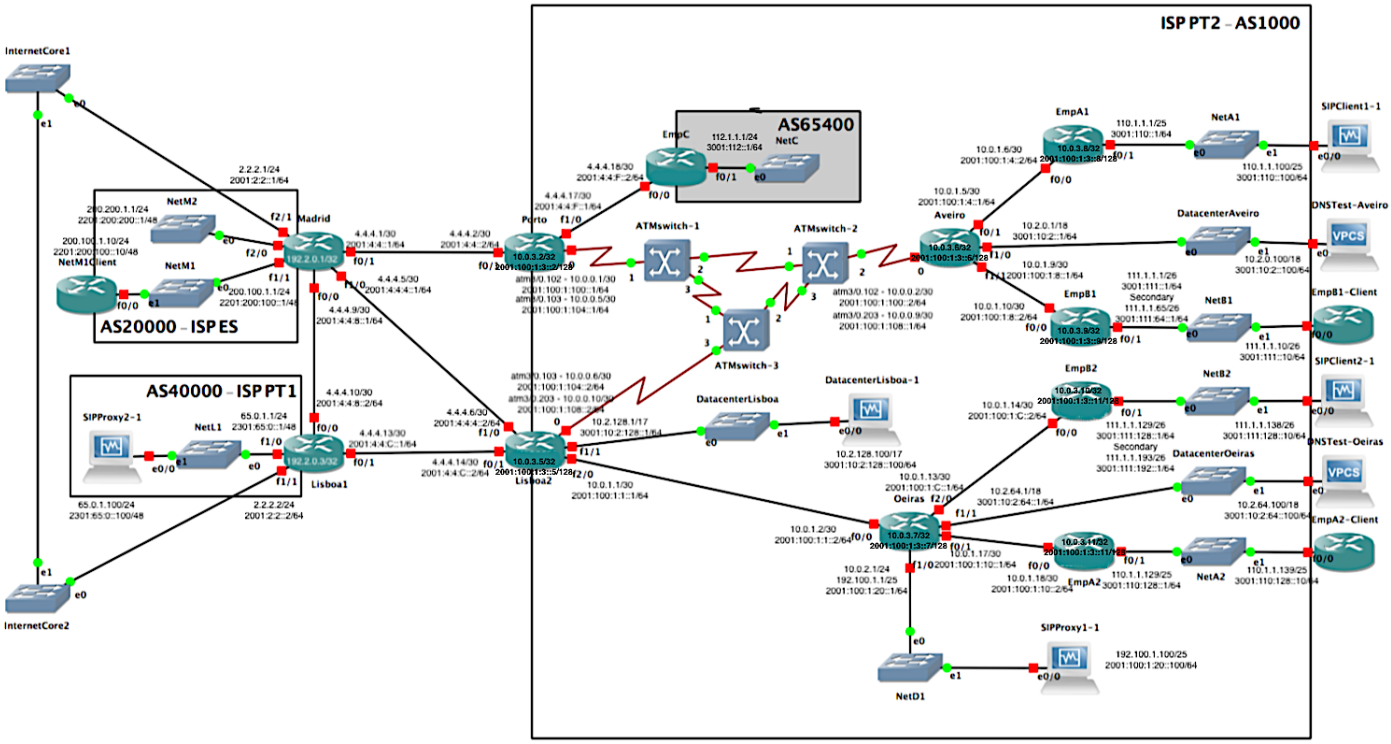

Figure .1: Network design.

A bigger version of the network design figure can be found at .1.

1 Scenario Description

From the provided Internet Protocol (IP) addresses list, the following tables were defined.

Table .1: IP addresses - part 1.

Network	**IPv4**	**IPv6**
Core	10.0.0.0/24	2001:100:1::/64
Porto - Aveiro	10.0.0.0/30	2001:100:1:100::/64
Porto - Lisboa2	10.0.0.4/30	2001:100:1:104::/64
Aveiro - Lisboa2	10.0.0.8/30	2001:100:1:108::/64
Lisboa2 - Oeiras	10.0.1.0/30	2001:100:1:1::/64
Aveiro - EmpA1	10.0.1.4/30	2001:100:1:4::/64
Aveiro - EmpB1	10.0.1.8/30	2001:100:1:8::/64
Oeiras - EmpB2	10.0.1.12/30	2001:100:1:C::/64
Oeiras - EmpA2	10.0.1.16/30	2001:100:1:10::/64
Loopback	10.0.3.0/24	2001:100:1:3::/64
Datacenter Lisboa	10.2.128.0/17	3001:10:2:128::/64
Datacenter Oeiras	10.2.64.0/18	3001:10:2:64::/64
Datacenter Aveiro	10.2.0.0/18	3001:10:2::/64
NetA1	110.1.1.0/25	3001:110::/64
NetA2	110.1.1.128/25	3001:110:128::/64
NetB1	111.1.1.0/26 111.1.1.64/26	3001:111::/64 3001:111:64::/64
NetB2	111.1.1.128/26 111.1.1.192/26	3001:111:128::/64 3001:111:192::/64
NetC	112.1.1.0/24	3001:112::/48
NetD1	10.0.2.0/24 192.100.1.0/25	2001:100:1:20::/64

Table .2: IP addresses - part 2.

Network	**IPv4**	**IPv6**
Madrid - Porto	4.4.4.0/30	2001:4:4::/64
Madrid - Lisboa2	4.4.4.4/30	2001:4:4:4::/64
Madrid - Lisboa1	4.4.4.8/30	2001:4:4:8::/64
Lisboa1-Lisboa2	4.4.4.12/30	2001:4:4:C::/64
Porto - EmpC	4.4.4.16/30	2001:4:4:F::/64

2 Basic Mechanisms and BGP

To have connectivity among terminals, routing mechanisms among networks need to be established.

In order to route traffic between Autonomous System (AS)s, we resorted to Border Gateway Protocol (BGP). Between ISP-PT2 border routers, EmpC, Madrid and Lisboa1 routers, routes were exchanged using External Border Gateway Protocol (eBGP) and the neighbor relations were established between the IPs of the point-to-point links. Thus, only directly connected peers established eBGP neighborhood between them.

Since Madrid and Lisboa1 routers provide access to the internet, to emulate this constraint, they were configured to announce a default route to AS1000 routers via BGP.

On the other hand, Internal Border Gateway Protocol (iBGP) neighbors inside ISP-PT2 communicate over loopback interfaces that were configured along all routers on its domain, even though only Porto, Lisboa2, Aveiro and Oeiras routers form a mesh as BGP neighbors.

Inside AS1000, an Open Shortest Path First (OSPF) process for intra-AS routing is also running (process 1 for Core routing, process 10 for EmpA and process 11 for EmpB routing between Costumer Edge (CE) and Provider Edge (PE) routers). To make the ISP-PT2 core and external networks aware of costumers networks, Aveiro and Oeiras routers must redistribute the internal OSPF processes via BGP. However, default routes must not be redistributed, so the following configurations were applied:

```
! Filter default routes
ip prefix-list pOut-priv-default seq 16 permit 0.0.0.0/0

ipv6 prefix-list pOut-default seq 16 permit ::/0

route-map routes-ospf deny 10
match ip address prefix-list pOut-priv-default
route-map routes-ospf permit 20

route-map routes-ospf6 deny 10
match ipv6 address prefix-list pOut-default
route-map routes-ospf6 permit 20

! BGP
router bgp 1000
address-family ipv4 unicast
redistribute static ip
redistribute ospf 11 route-map routes-ospf
address-family ipv6 unicast
redistribute ospf 10 route-map routes-ospf6
```

```
redistribute ospf 11 route-map routes-ospf6
```

Listing .1: Policies applied to redistribute CE OSPF processes into BGP

Nevertheless, Porto and Lisboa2 routers also announce via OSPF a default route, which will be chosen if the eBGP default routes are not available (e.g., for CE networks).

So that eBGP peers can "learn" the networks inside the AS (mainly those which are not directly connected to iBGP routers), OSPF routes are redistributed to BGP in Porto and Lisboa2. However, this means point-to-point private networks and OSPF default routes would also be redistributed to other ASs and conflicts could arise (e.g., loopback IPs would be either distributed via BGP and OSPF and would create loops on routing). To avoid that, Porto e Lisboa2 routers were configured to prevent BGP from redistributing those networks.

Another important aspect is that, in Porto and Lisboa2 BGP neighbor relations with external peers, a `route-map` was applied so that only internal networks to the AS can be announced to external neighbors, preventing AS1000 from becoming a transit-AS. However, the referred `route-map` also allows route updates with AS_PATHstarting by 65400, since the latter is a private AS; the neighbor relations with Madrid and Lisboa1 in Porto and Lisboa2 routers were also configured to remove private ASs from AS_PATH attribute of their BGP updates.

With that in mind, the following `prefix-lists` and `route-maps` were used:

```
! Filter private networks and default routes
ip prefix-list pOut-priv-default seq 10 permit 10.0.0.0/8 le 32
ip prefix-list pOut-priv-default seq 12 permit 172.16.0.0/12 le 32
ip prefix-list pOut-priv-default seq 14 permit 192.168.0.0/16 le 32
ip prefix-list pOut-priv-default seq 16 permit 0.0.0.0/0

ipv6 prefix-list pOut-default seq 16 permit ::/0

! Route-maps to filter routes redistributed to private ASs
route-map routes-ospf deny 10
match ip address prefix-list pOut-priv-default
route-map routes-ospf permit 20

route-map routes-ospf6 deny 12
match ip address prefix-list pOut-default
route-map routes-ospf6 permit 20

! Filter routes (only local routes)
ip as-path access-list 1 permit ^$
ip as-path access-list 1 permit 65400$

! Route-maps to filter routes redistributed to external ASs
```

```
route-map routes-out deny 10
match ip address prefix-list pOut-priv-default
route-map routes-out permit 12
match as-path 1

route-map routes-out6 deny 10
match ipv6 address prefix-list pOut-default
route-map routes-out6 permit 12
match as-path 1

! BGP
router bgp 1000
address-family ipv4 unicast
! Redistribution
redistribute ospf 1 route-map routes-ospf
...
! EmpC
neighbor 4.4.4.18 remote-as 65400
neighbor 4.4.4.18 next-hop-self
neighbor 4.4.4.18 route-map routes-ospf out
! Madrid
neighbor 4.4.4.1 remote-as 20000
neighbor 4.4.4.1 route-map routes-out out
neighbor 4.4.4.1 prefix-list netL1-in in
neighbor 4.4.4.1 remove-private-as

address-family ipv6 unicast
! Redistribution
redistribute ospf 1 route-map routes-ospf6
...
! EmpC
neighbor 2001:4:4:F::2 remote-as 65400
neighbor 2001:4:4:F::2 next-hop-self
neighbor 2001:4:4:F::2 routes-ospf6 out
! Madrid
neighbor 2001:4:4::1 remote-as 20000
neighbor 2001:4:4::1 route-map routes-out out
neighbor 2001:4:4::1 remove-private-as
neighbor 2001:4:4::1 activate
neighbor 2001:4:4::1 prefix-list netL1-in-6 in
```

Listing .2: Policies applied to avoid redistribution of private networks and default routes to BGP

Porto and Lisboa2 routers are also in charge of summarizing internal networks to their external neighbors. Only networks that are announced to the outside should be summarized.

Nevertheless there were other BGP constraints that needed to be addressed:

- **IP traffic towards Internet should be preferably routed via ISP PT1**: to follow this constraint, the first step was to increase Lisboa2

iBGP local preference inside AS1000, thus, all traffic destined to external networks is routed via Lisboa2. The second step was to increase Lisboa1 route announcements to Lisboa2 local preference (200). Since Lisboa2 was not considering routes announced by Madrid as having the default local preference (100), another `route-map` was created to set it to the default value, so that Lisboa1 (ISP-PT1) could be preferred.

- **IP traffic towards all AS20000 networks should be preferably routed via Porto from Aveiro, and via Lisboa from Oeiras**: since Lisboa2 local preference was already higher than default, all traffic towards AS20000 coming from Oeiras was already being routed through there. However, so that Aveiro could choose Porto as its output router to AS20000, a `route-map` was created in Lisboa2 (only applied to Aveiro neighborhood) to lower the local preference of all routes coming from AS20000 (it was set to 50). Regarding Porto, since eBGP has an administrative distance lower than iBGP, even though Lisboa2 announces those routes with an increased local preference, the chosen path will be routing traffic directly to Madrid.

- **IP traffic for remote SIP proxy 2 (to network netL1) cannot be routed via Porto using the direct peering link to ISP ES**: an `ip-prefix-list` denying all the prefixes of netL1 was used to filter incoming route advertisements from Madrid. This way, only Lisboa2 router will announce netL1 to the inside of AS1000.

3 MPLS

3.1 Client B Dedicated Channels

Since client B requested two bi-directional channels between its two branches, Multi-Protocol Label Switching Traffic-Engeneering (MPLS-TE) on the routers along the path was activated, i.e., Aveiro, Lisboa2 and Oeiras routers. Resource Reservation Protocol Traffic-Engeneering (RSVP-TE) will establish two dynamic bidirectional Label Switching Path (LSP) tunnels, being Aveiro and Oeiras the endpoints. We opted for using dynamic tunnels since there's only one path between Aveiro and Oeiras; if there was more than one (e.g., if the core had multiple paths from Lisboa2 to Aveiro, in this example, the sijmplification of the core implied that there were only two paths and they would be sharing link bandwidth in all but one link), static tunnels could be established in order to prevent them from using the same path and from consuming bandwidth one the same links.

On the other hand, since only traffic coming from and to client B branches needs to be routed along, there was no need to use the `autoroute` option. In order to route it correctly, two `access-lists` were defined, along with a `route-map` (one for Internet Protocol version 4 (IPv4) and another for Internet Protocol version 6 (IPv6)) to route the traffic to the corresponding tunnels. Since two tunnels were established, we subdivided each of client B branch networks in two subnetworks, each one running in each tunnel, i.e., only traffic between corporate B subnet A (111.1.1.0/27 and 111.1.1.128/27; 3001:111::/64 and 3001:111:64::/64) uses Tunnel 0 and only traffic between corporate B subnet B (111.1.1.64/27 and 111.1.1.192/27; 3001:111:128::/64 and 3001:192:64::/64) uses Tunnel 1.

Using policy based routing in the interfaces connected to corporate B CE routers in Aveiro and Oeiras, the following `route-maps` were applied (example for Aveiro):

```
access-list 110 permit ip 111.1.1.0 0.0.0.63 111.1.1.128 0.0.0.63
access-list 120 permit ip 111.1.1.64 0.0.0.63 111.1.1.192 0.0.0.63

route-map mpls-empB permit 10
match ip address 110
set interface tunnel 0
route-map mpls-empB permit 12
match ip address 120
set interface tunnel 1

ipv6 access-list mplstunnel10
permit ipv6 3001:111::/64 3001:111:128::/64
ipv6 access-list mplstunnel20
permit ipv6 3001:111:64::/64 3001:111:192::/64
```

```
route-map mpls-empB6 permit 10
match ipv6 address mplstunnel10
set interface tunnel 0
route-map mpls-empB6 permit 12
match ipv6 address mplstunnel20
set interface tunnel 1
route-map mpls-empB6 permit 20
```

Listing .3: Policies applied in Aveiro to route traffic to MPLS-TE tunnel.

3.2 Client A VPN

Client A requested for a Virtual Private Network (VPN) interconnecting its two branches (IPv4 only). The first step to achieve it was to create a Virtual Routing and Forwarding (VRF) in both Aveiro and Oeiras routers (importing and exporting Route Distinguisher (RD) 1000:10) and associate it with the interfaces connected to company A CE routers. Afterwards, `vpnv4` address family was configured at both routers (both defining the other as neighbor), so that, through Multi-Protocol Border Gateway Protocol (MP-BGP), routes between client A branches could get exchanged. It was also necessary to activate Label Distribution Protocol (LDP) along all routers on the path (Aveiro, Lisboa2 and Oeiras), so that packets could be routed via the Multi-Protocol Label Switching (MPLS) backbone.

However, as it was mentioned before, another OSPF process (10) at both routers was defined, running on the interfaces connected to company A CE routers. This process will also be redistributed via MP-BGP (configured on `ipv4 vrf VPN-1` address family).

After these steps, connectivity between branches will be established, and a MPLS Virtual Private Network (MPLS-VPN) will be created (packets between branches will be routed via the MPLS backbone). However, in order to make client A branches have connectivity with the exterior, two static routes were established (one for announcing a default route to the inside of the VPN pointing to the closest AS edge router - Aveiro to Porto and Oeiras to Lisboa - and another to anounce to public IP addresses to the networks exterior to the VPN):

```
! VPN static routes
ip route vrf VPN-1 0.0.0.0 0.0.0.0 10.0.3.5 global
ip route 110.1.1.128 255.255.255.128 FastEthernet0/1
```

Listing .4: Static routes to establish connectivy to the exterior configured on Aveiro

Those static routes were also redistributed via BGP and OSPF, so that other locations could be aware of company A networks.

Since it was only implemented a IPv4 VPN, we decided to redistribute the Open Shortest Path First version 3 (OSPFv3) process relativate to corporate A branches via BGP in both Aveiro e Oeiras, in order to have connectivity to the exterior over IPv6.

4 VoIP

A Voice over IP (VoIP) - Session Initiation Protocol (SIP) service for all ISP-PT2 corporate clients was also deployed. The service provides VoIP connectivity (through ISP proxy 1) between internal clients and forwards all other calls (including Public Switched Telephone Network (PSTN) numbers) to ISP-PT1 SIP proxy.

Using Asterisk as SIP server/proxy at SIP-Proxy1, some user accounts were created and configured SIP-Proxy2 as a peer with the remote host IP:

```
[PedroMartins]
type=friend
host=dynamic
secret=labcom
context=phones
allow=all

[DiogoFerreira]
type=friend
host=dynamic
secret=labcom
context=phones
allow=all

[PauloSalvador]
type=friend
host=dynamic
secret=labcom
context=phones
allow=all

[Proxy2]
type=peer
host=65.0.1.100
defaultuser=Proxy1
secret=labcom
context=phones
```

Listing .5: Snippet of SIP-Proxy1 `sip.conf` file configuration.

On SIP-Proxy2, since it's a proxy on a remote ISP and not "managed by us", SIP-Proxy1 was only configured as a peer (no user accounts were needed):

```
[Proxy1]
type=peer
host=192.100.1.100
secret=labcom
context=phones
```

Listing .6: Snippet of SIP-Proxy2 `sip.conf` file configuration.

For both proxies, a context was defined in `extensions.conf` file. At SIP-Proxy1, we opted to associate numbers of different corporates to different clients and place them in their respective networks (e.g. *PedroMartins* at EmpA1 and *PauloSalvador* at EmpB2).

Since there's no specific place to redirect all the other numbers of the respective companies, we chose to emulate that situation and redirect them all to one user (in a real life scenario, each company could be provided with a private proxy and all the calls with the corresponding company extensions would be redirected to it). For all the other calls, they are redirected to SIP-Proxy2.

```
[phones]
exten => 234100000,1,Dial(SIP/PedroMartins,10)
exten => 219100000,1,Dial(SIP/DiogoFerreira,10)
exten => 234110002,1,Dial(SIP/PauloSalvador,10)
exten => 23410XXXX,1,Dial(SIP/PedroMartins,10)
exten => 21910XXXX,1,Dial(SIP/PedroMartins,10)
exten => 23411XXXX,1,Dial(SIP/PauloSalvador,10)
exten => 21911XXXX,1,Dial(SIP/PauloSalvador,10)
exten => _X.,1,Dial(SIP/${EXTEN}@Proxy2,10)
```

Listing .7: Snippet of SIP-Proxy1 `extensions.conf` file configuration.

On the other hand, the context used in SIP-Proxy2 is really simple; it just accepts every call it receives (with at least one dialed digit) and plays back the dialed numbers (simplification of redirecting the call to the PSTN).

```
[phones]
exten => _X.,1,Answer(500)
exten => _X.,2,Playback(vm-received)
exten => _X.,3,SayDigits(${EXTEN:3})
exten => _X.,n,Playback(vm-goodbye)
exten => _X.,n,Hangup()
```

Listing .8: Snippet of SIP-Proxy2 `extensions.conf` file configuration.

5 CDN

5.1 Routing Service - Conditional DNS

In order to provide a Content Distribution Protocol (CDN) routing service for corporate clients, it was proposed to implement a conditional DNS service, able to redirect clients to the closest datacenter according to their location, i.e., terminals in Aveiro to Aveiro Datacenter, terminals in Oeiras to Oeiras Datacenter, and all the others to Lisboa Datacenter.

Since DNS, by itself, can't make a decision based on a geo-location, a database needed to be built where a set of network IPs (both IPv4 and IPv6) was mapped in a location (label). This database is described below:

```
acl "Lisboa"{
    10.2.128.0/17;
    3001:10:2:128::/64;
};

acl "Porto" {
    112.1.1.0/24;
    3001:112::/64;
};

acl "Aveiro" {
    10.2.0.0/18;
    110.1.1.0/25;
    111.1.1.0/25;
    3001:10:2::/64;
    3001:110::/64;
    3001:111::/64;
    3001:111:64::/64;
};

acl "Oeiras" {
    10.2.64.0/18;
    110.1.1.128/25;
    111.1.1.128/25;
    10.0.1.0/24;
    192.100.1.0/25;
    3001:10:2:64::/64;
    3001:110:128::/64;
    3001:111:128::/64;
    3001:111:192::/64;
    2001:100:1:20::/64;
};
```

Listing .9: DNS Access Control List

Now that all IPs are mapped in a geo-location, the remaining part is to choose where to route the requests, taking only into account the location label and the logic presented at the begining of this section, by defining one

view to each zone on the name configuration file:

- **Aveiro**:

```
view "aveiro" {
    match-clients { Aveiro; };
    recursion no;
    zone "aracdn.pt" {
            type master;
    file "/etc/bind/aracdn.pt-aveiro-db";
    };
};
```

- **Oeiras**:

```
view "oeiras" {
    match-clients { Oeiras; };
    recursion no;
    zone "aracdn.pt" {
            type master;
    file "/etc/bind/aracdn.pt-oeiras-db";
    };
};
```

- **Lisboa/Others**:

```
view "other" {
    match-clients { any; };
    recursion no;
    zone "aracdn.pt" {
            type master;
    file "/etc/bind/aracdn.pt-other-db";
    };
};
```

Each zone has its own zone file, which points to the respective datacenter IP:

- **Aveiro**:

```
            IN      NS      ns1.aracdn.pt.
            IN      A       10.2.0.100
            IN      AAAA    3001:10:2::100
    ns1     IN      A       10.2.128.100
```

- **Oeiras**:

```
            IN      NS      ns1.aracdn.pt.
            IN      A       10.2.64.100
            IN      AAAA    3001:10:2:64::100
    ns1     IN      A       10.2.128.100
```

- **Lisboa/Others**:

```
                IN      NS      ns1.aracdn.pt.
                IN      A       10.2.128.100
                IN      AAAA    3001:10:2:128::100
ns1             IN      A       10.2.128.100
```

5.2 Load balancing

In order to improve the CDN routing service, it was proposed to include a link/server load condition in the conditional DNS decision process. This was accomplished by creating a script that could manage the DNS filezones according to the load measured on one instant with Simple Network Management Protocol (SNMP).

The script was built in Python3, using `pysnmp` [1] and `pysnmp-mib` [2], and will be running in background, where it will analyze the load of each link/router every two minutes and perform the appropriate actions. This script is divided in four main stages:

- **Link measure**: Since SNMP Management Information Base (MIB)s don't provide a direct way to measure a link load, we need to compute it by obtaining in-bytes, out-bytes and the interface speed in two different instants, so that we can compute diferences in time (delta values) and get the usage percentage. The input and output usage can be calculated this way

$$\text{Input usage} = \frac{\delta ifInOctets * 8}{\delta seconds * ifSpeed}$$

$$\text{Output usage} = \frac{\delta ifOutOctets * 8}{\delta seconds * ifSpeed}$$

  where *ifInOctets*, *ifOutOctets* and *ifSpeed* are specific MIB objects and delta values are the difference between the values measured in the two distinct instants. Finally, the link usage percentage can be computed the following way:

$$\text{Link usage} = (\text{Input usage} + \text{Output usage}) * 100$$

  This process is executed several times (10 by default), so that short time peaks on the link can be ignored. All computed usage percentages are saved in order to post-process them and derive some conclusions (as we shall see later). We also analyze if the interface of a given router directly connected to a datacenter is operational. If not, that datacenter is automatically excluded. The following code shows how to get the values needed to compute the percentage, as mentioned above.

It's noteworthy the fact that this code only gets the **individual** object values; the delta values are not shown to avoid extending this report, but can be found in the source code.

```
for ip in links:
    for (errorIndication,
         errorStatus,
         errorIndex,
         varBinds) in nextCmd(SnmpEngine(),
            CommunityData('public'),
            UdpTransportTarget((ip, 161)),
            ContextData(),
            # get interface name
            ObjectType(ObjectIdentity('IF-MIB', 'ifDescr')),
            # get interface ip
            ObjectType(ObjectIdentity('1.3.6.1.2.1.4.20.1.2')),
            # get interface input load (bytes p/sec)
            ObjectType(ObjectIdentity('IF-MIB', 'ifInOctets')),
            # get interface output load (bytes p/sec)
            ObjectType(ObjectIdentity('IF-MIB', 'ifOutOctets'))
                ,
            # get interface speed (bits p/sec)
            ObjectType(ObjectIdentity('IF-MIB', 'ifSpeed')),
            lexicographicMode=False):

        (...)

        # find interface
        interface_ip_addr = varBinds[1][0].prettyPrint().split(
            '20.1.2.')
        if(len(interface_ip_addr) == 2 and interface_ip_addr[1]
             == ip):
            # get interface intput and output load
            in_load = int(varBinds[2][1].prettyPrint()) * 8
            out_load = int(varBinds[3][1].prettyPrint()) * 8
            # get interface speed
            speed = int(varBinds[4][1].prettyPrint())
```

The interface status is also analyzed to be able to simulate an overloaded link.

- **Post-processing**: After obtaining the usage percentage values, we needed to get the less stressed datacenter, in order to point the requests of the overloaded zones there. As we saved the several usage percentages measured, we needed to extract from them a single value that could approach the real usage percentage value; this is done by finding the median value over all values. After discovering the datacenter(s) with the lowest load, we had to find which datacenters are overloaded, by analysing the saved usage percentages and checking if they exceed the imposed limit (defined as being 85%. A link is consid-

ered overloaded only if we could find several usage percentages over the limit (5 by default), taking into account all the percentages measured in the first step.

- **Filezone changes**: When we find that a link is really overloaded (taking into account the definition of overloading mentioned above), it's time to react, by changing the filezone to the less loaded datacenter that was found in the post-processing stage. As one file of each zone is kept (Aveiro, Oeiras and others), this means we onlt need to point the filezone to the less loaded zone, by issuing a symlink to that file.

- **Replace original filezone**: If a link was flagged as overloaded (and consequently its filezone was changed), but now has a normal usage percentage, it will be possible to rollback the changes and replace the original filezone (by doing a symlink to the original zone file), and all the requests will be routed to original datacenters.

6 ATM Core

We were also proposed to use an Assynchronous Transmission Protocol (ATM) core instead of a traditional Ethernet core (in real scenarios, an ATM core is often favoured over an Ethernet one).

To emulate the core of the network, three ATM switches in a tringle configuration were used. Three point-to-point connections were defined (in comparison to multipoint connections, the latter ones consume much more bandwidth since those need to broadcast all the traffic to all its connected Permanent Virtual Circuit (PVC)s); considering a real world application, ATM is commonly used with point-to-point connections, just like circuit-switched networks:

- **pvc 102/0** - PVC to interconnect Porto and Aveiro;
- **pvc 103/0** - PVC to interconnect Porto and Lisboa2;
- **pvc 203/0** - PVC to interconnect Aveiro and Lisboa2.

The configuration is pretty straightforward, each ATM interface will have two subinterfaces, one for each PVC it is connected to (this led to the subdivision of the core network into point-to-point networks). Each ATM switch is also configured to only redirect one PVC from one input interface to the corresponding output. An example of ATM switch 1 configuration:

Table .3: ATM switch 1 configuration.

In (Port:Path:Channel)	**Out (Port:Path:Channel)**
1:102:0	2:102:0
1:103:0	3:103:0

The configured ATM core is represented in .2.

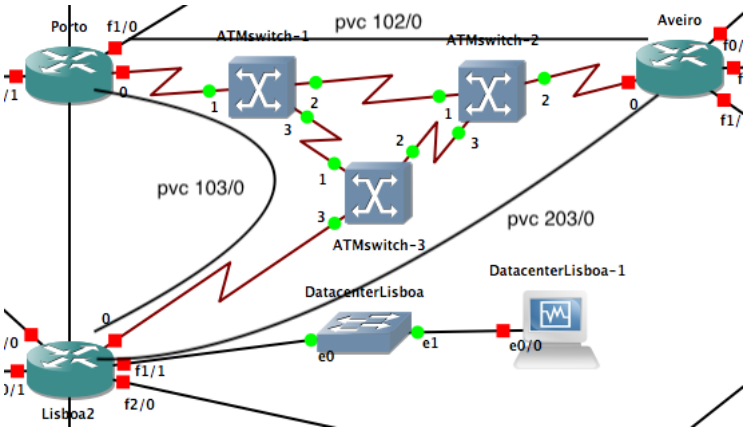

Figure .2: ATM Core with the defined PVC.

Chapter

Conclusions

ISP level networks are really complex and must be resilient to failures (annual uptime over 99.999%). Their routing mechanisms must be implemented in a way they can resist to multiple type of failures and all kinds of imposed constraints. These networks can also provide multiple services to their clients, such as VPNs, DNS and interconnection of VoIP content to the PSTN.

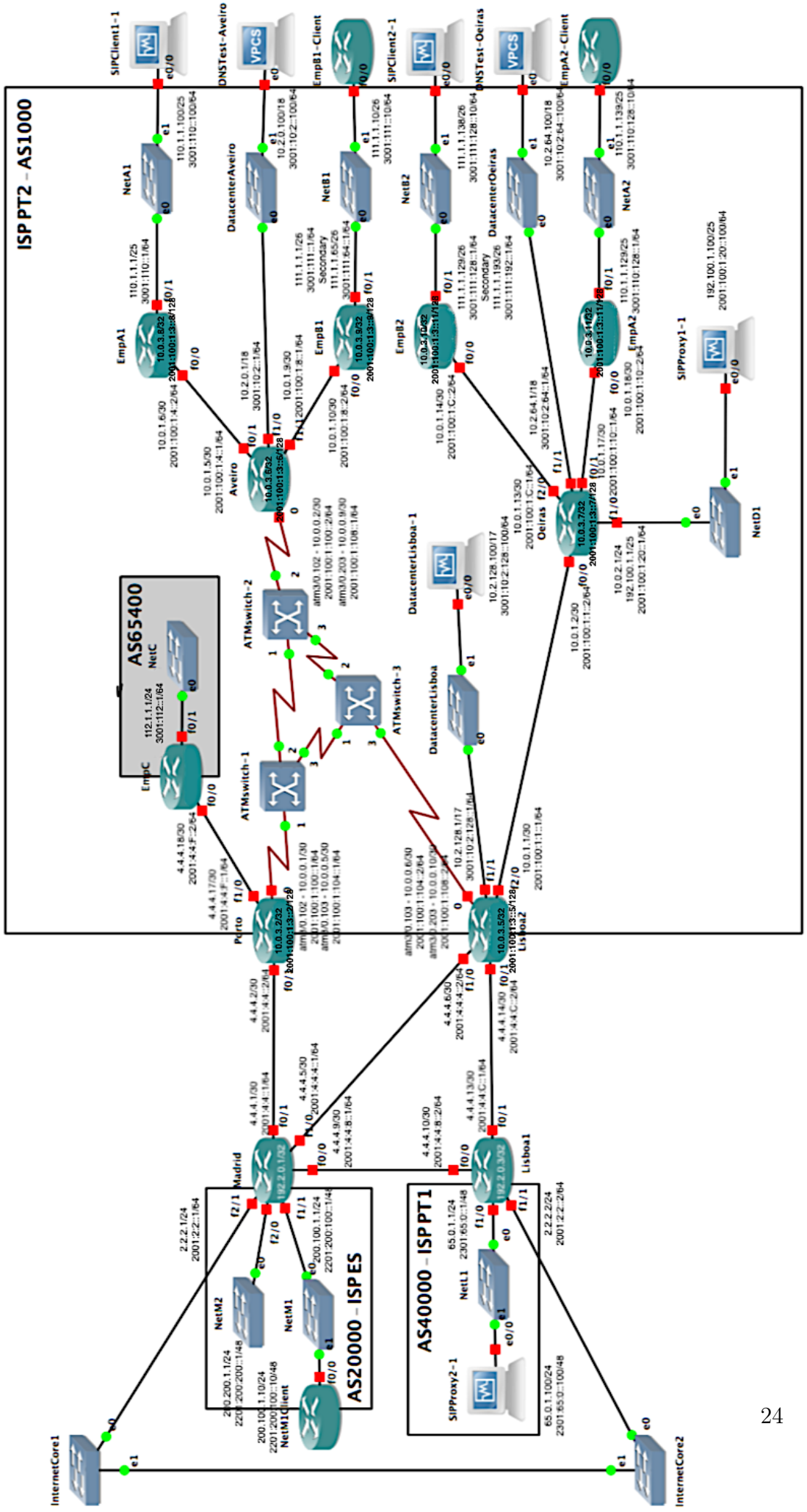

Figure .1: Network design.

Acrónimos

ISP Internet Service Provider

IP Internet Protocol

IPv4 Internet Protocol version 4

IPv6 Internet Protocol version 6

AS Autonomous System

BGP Border Gateway Protocol

MP-BGP Multi-Protocol Border Gateway Protocol

eBGP External Border Gateway Protocol

iBGP Internal Border Gateway Protocol

OSPF Open Shortest Path First

OSPFv3 Open Shortest Path First version 3

MPLS Multi-Protocol Label Switching

LDP Label Distribution Protocol

LSP Label Switching Path

MPLS-TE Multi-Protocol Label Switching Traffic-Engeneering

RSVP-TE Resource Reservation Protocol Traffic-Engeneering

MPLS-VPN MPLS Virtual Private Network

VPN Virtual Private Network

RD Route Distinguisher

CE Costumer Edge

PE Provider Edge

VRF Virtual Routing and Forwarding

CDN Content Distribution Protocol

DNS Domain Name System

SNMP Simple Network Management Protocol

VoIP Voice over IP

SIP Session Initiation Protocol

PSTN Public Switched Telephone Network

ATM Assynchronous Transmission Protocol

PVC Permanent Virtual Circuit

MIB Management Information Base

DNS Domain Name System

Bibliography

[1] Individual Contributors. Pysnmp. [Online; accessed in December 2017].

[2] Individual Contributors. Pysnmp-mibs. [Online; accessed in December 2017].