LINGI2142-Project report

François Genon
Computer science student (0643-16-00)
UCLouvain
francois.genon@student.uclouvain.be

David Bodenghien
Computer science student (2222-16-00)
UCLouvain
david.bodenghien@student.uclouvain.be

Julie Caudroit
Computer science student (2163-20-00)
UCLouvain
julie.caudroit@student.uclouvain.be

***Abstract*—This document was made for the course LINGI2142 where we were asked to implement a transit network inspired by OVH and add to it stubs and other transit networks.**

I. Introduction

To get information about OVH network, we consulted OVH's weathermap and we selected parts we found interesting. We focus mainly on North America as it will be explained in point II. This does not mean that our network is a copy of OVH, only the topology was inspired and we added new services such as anycast and BGP communities on top of it.

II. Topology of the network

The topology is inspired by the OVH network and principally the North America network, as you can see we just take the biggest data-center in the world and make an interconnected network. You can see a diagram of the topology on Fig. 1 with the legend below.

- Route Reflector: blue circle
- BGP communities: green circle
- AS: cloud
- transit routers (Cogent, Telia, Level3): purple rectangle
- stub routers (Amazon, Charter): orange rectangle
- OVH routers (`nwk_1` & `nwk_5`): green rectangle
- OVH routers (`chi_1` & `chi_5`): yellow rectangle
- OVH routers (`ash_1` & `ash_5`): green rectangle
- OVH routers (`bhs_g1` & `bhs_g2`): light blue rectangle
- Europe router (`europe`): black rectangle
- Asia router (`asia`): blue rectangle
- eBGP session: red link
- physical OVH link (with cost) and Cogent & Telia routers: black link
- physical link for router Level3 : purple link
- physical link for router Amazon : green link
- physical link for router Charter : orange link

A. Our network (OVH)

Our network is inspired by the OVH network, but we were more interested in the North American OVH network because there were 8 data centers in 4 regions: New-York (`nwk_1`, `nwk_5`), Chicago (`chi_1`, `chi_5`), Ashburn (`ash_1`, `ash_5`) and Beauharnois (`bhs_g1`, `bhs_g2`). But to make a more realistic network of the world, we added a router representing Europe (`europe`) and directly connected to the routers of New-York (`nwk_1`, `nwk_5`) and another router in Asia (`asia`) directly connected to the routers of Chicago (`chi_1`, `chi_5`). The routers in Asia and Europe being connected to each other. This network representing the world is used to create the transatlantic and transpacific links.

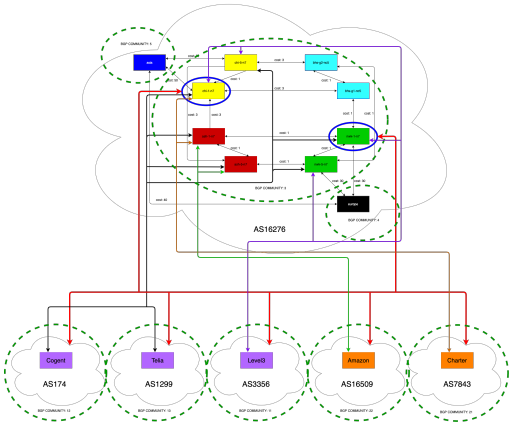

Fig. 1. Topology of our network

B. Peering Stub networks

For our stubs, we took Amazon and Charter. Our stub Amazon AS16509 represented in our network by the routers `as16509_r1` and `as16509_r2` connected to Ashburn (`ash_1`, `ash_5`). Charter in the real world is present in number in the Autonomous System among them the AS7843 (`as7843_r1`, `as7843_r2`), he is directly connected to the data centers of Chicago (`chi_1`) and Ashburn (`ash_1`).

C. Peering transit networks

We have 3 transit routers that are inspired by the actual transit of the OVH network namely Telia by the routers (`as1299_r1`, `as1299_r2`), Cogent by the routers (`as174_r1`, `as174_r2`) and Level3 by the routers (`as3356_r1`, `as3356_r2`). Telia is a Swedish telecommunications transit that is present in 45 Autonomous Systems around the world. We took Telia's AS1299 because it is connected to the data centers of New-York (`nwk_1`, `nwk_5`), Chicago (`chi_5`) and Ashburn (`ash_5`) present in the OVH network. Cogent is also a U.S. multinational telecommunications transit company present in more than 26 countries with more than 20 Autonomous Systems. We took Cogent's largest AS (AS174) because it is connected to the data centers

in Ashburn (ash_1, chi_5), Chicago (chi_1) and New-York (nwk_1, nwk_5). Our latest transit router is Level3, it's a US telecommunications company, it's one of the largest internet telecommunications operators. Level3 is present in 103 Autonomous Systems in the world including the AS3356 which we represented in our network because it is connected to the data centers of Chicago (chi_1, chi_5) and New-York (nwk_1, nwk_5).

III. IP CONFIGURATION

A. IP addressing of datacenters

We have chosen a type of IPv6 addressing according to the regions in the world where each datacenter is located.

- In America: 2010:1100:0000:0
- In Europe : 2010:1200:0000:0
- In Asia : 2010:1300:0000:0

So we're going to set the site prefix according to the region, which is logical.

Then, depending on the datacenter, we will change the subnetwork. Then, more precisely, at each link, we will change the interface of this same subnetwork.

For IPv4, the choice was made to represent addressing according to the city where the datacenter is located, or in some cases, for Europe and Asia, the region where a datacenter is located.

We therefore find :

- NWK = 160.72.241."
- BHS = "160.72.242."
- CHI = "160.72.243."
- ASH = "160.72.244."
- EUROPE = "160.72.245."
- ASIA = "160.72.246."

Then, depending on the router and its links, we will find a change in the address carried on the last bits of the IP address. Note that for the sake of clarity, the "1's" will be in .100, where the "5's" will be .110.

B. IP addressing of STUB and TRANSIT

Grouped as ASs, the addressing logic is identical for AS-specific routers determined by us. We then set fixed addresses with a logic according to the STUB and TRANSIT. A prefix is set, while the subnetwork may differ as its interface to arrive at a consistent address.

It can be noted, however, that the hosts linked to the datacenters will have addresses linked to the region in which the datacenter is located in IPv6 and linked to the city in IPv4. This is exclusive to the specific host, such as this one: `as16509_ash_1_amazon`. We then choose to take an addressing logic in relation to Ashburn (ASH), therefore the region of the USA, and in relation to the datacenter city.

Here is a concrete example:

- `ma1[as16509_ash_1_amazon].addParams (usa_ipv6 + "610::1/64", ash_ipv4 + "150/24")` : the code line represents the addressing of a STUB, here we use for a host the addressing related to the region in which it is located as well as the direct link to the city of the datacenter.
- `ma1[as16509_r1].addParams(as16509_ipv6 + "000::1/64", as16509_ipv4 + "101/24")` : We can see here that the addressing depends on the STUB and its router, it is specific to it and defined above according to the AS in which it is located:
  - `as16509_ipv4 = "162.61.1."`
  - `as16509_ipv6 = "2010:2100:0000:0"`

IV. OSPF CONFIGURATION

For our routers to be able to know the topology of the network, we decided to use OSPF. We opted to first implement the most decisive parameter of OSPF: the cost. See our OSPF cost at Fig. VII

A. OSPF metric

We wanted the cost to reflect reality of a real network so we added a cost that is proportional to the distance between routers. That's why intercontinental links have the biggest numbers, we try to rely on these links only when necessary. The fact that we most often rely on links of the same continent reduces our latency and thus detect failures as fast as possible. Our will to reduce latency in our network is a great way to introduce the next parameters.

B. OSPF intervals

Multiple parameters transmition parameters exist in OSPF, we will discuss the following:

- Transmission interval: the interval at which a router sends an HELLO packet to its neighbors
- Retransmission interval: the interval at which a router retransmits a HELLO packet to its neighbors if the sender did not receive an acknowledgment from its neighbors
- Dead interval: the rate at which we consider a link as dead

After some research, we decided to leave these values at the default one. By default, the intervals are fairly balanced, this is very important for our network because:

- if we increase either one of the parameters above or even all them, we may falsely assume that a link isn't dead and continue to send packets to a dead link that will later be retransmitted. This is very bad for our network because it would dramatically increase the latency in case of failure.
- if we reduce one or all of these value, we may falsely assume that a link is dead. This can happen when our dead interval and/or retransmission intervals are too low. When our transit network will be heavily demanded (for example on New Year's Eve), the transmission delay on our links may be increased and if these delays are above our dead interval treshold the network would consider the link dead and redirect traffic to another link. This will start a chain reaction because this new path will be heavily demanded and thus have a higher transmission interval,...

C. Other parameters

We configured our OSPF network to use a point to point type of connection and we left the default value for OSPF Priority. This means that we don't elect a DR (designated router) nor a BDR (backup designated router), our network is relatively small so we didn't had the need.

V. iBGP ORGANISATION

All our routers are connected using iBGP, even those from Asia and Europe. This helps us to have a great knowledge of accessible prefixes on our network.

Our network is not set up as a full mesh because of its size, we opted to use route reflectors.

A. Route reflectors

The route reflectors are `nwk_1` and `chi_1`. Having multiple reflectors allow us to have a backup in case of failure. The 2 RR's (route reflectors) were not chosen randomly, they are the routers with the most links and are in close proximity to Europe (from New-York) and Asia (from Chicago).

B. Decision process

In order to influence the iBGP decision process, many parameters can be set at our will:

- Local-Preference: The more it is big, the more will the route be preferred
- AS-Path length: AS prepending is possible
- MED: routes with a lower value are preferred but in certain cases the MED will not be taken in account.
- Router ID: this is a tricky one but it is technically possible.

On all the items above, I did not mention igp costs because they are set using OSPF and you can read more about it in section IV-A. Local preferences can be used to set primary links and backup links but we did not, in order the keep our network links balanced.

Being a transit network we want things to go smoothly. In our case, we did not implement AS prepending because the disadvantages outreached the positives sides. After studying our network we did not find the need of it and implementing it could cause security risks. For example, a "hostile" AS can decide to perform a local stripping of prepended ASes and then propagate the shortened AS path to its neighbours, the path appears perfectly plausible and the stripping of the prepending is invisible to the downstream AS. The result is potentially one of traffic re-direction.

For the MED value, we left the default setting for most of the link but we found one link to be particularly interesting. When setting up the network with default MED values, a loop sometimes occurred when Chicago's routers tried to connect to the CDN. AS 174 is connected both to Chicago and New-York, it will advertise the prefix he learned from New-York (where our CDN is based in North America) and Chicago's routers would go trough by AS 174. But AS 174 sometimes redirected the traffic back to Chicago and begins a packet loop. This is very bad for performance and will overwhelm routers from our network (Chicago) but also routers from peering networks ! To fix this problem, we set an higher med on the link between Chicago (MED=100) and AS 174 and a lower med on the link between New-York and AS 174 (MED=50). After this step, we did not find any loop in our network, nor when connecting to peering networks.

One other use case for local-preference and MED values are to set a certain link as primary and another one as a backup. We did not use these values to implement this into our network (we could have). As you can see in our topology, ASes and our CDN usually have multiple links with our network, this is a great thing for redundancy. But if you look closer, you will notice that our clients, CDN and peers don't have these multiple links going out to the same place. Our clients usually have outgoing links to different datacenters, that is why we decided to prioritize load balancing instead of primary-backup links. Since the links are usually have different destinations, it is better to use all of them to have an higher throughput available for our clients and because we are a transit network, we don't want to have delay in our network during peak hours because all the traffic is going through one link, when we can have two.
Check our diagrams for more details [VII]

VI. ADDITIONAL SERVICES

A. Anycast

Our transit network supports Anycast. To demonstrate this, we added a CDN to our network. Our CDN servers have instances on 3 different continents : Asia, Europe and of course North America. It supports iPv4 (10.0.3.2) and iPv6 (2001:3c::2).

Our CDN has been connected to our network using OSPF but we are also advertising the IP prefix of our CDN trough BGP so our peers and clients can know its existence and be able reach it (you can see some traceroute.

In North America, our CDN is directly connected to our New-York datacenters (`nwk_1`). We chose New-York because in case of failure traffic can still go trough Europe or Asia depending from somebody's location, because it's a route reflector, which means all routers on the network will already have `nwk_1`'s ip address in their routing table.

We also noticed a very interesting and useful behaviour happening when we calculate the route of every router on our network trying to connect to our CDN. In all our north American routers, Chicago's routers don't take a direct route to New-York, instead they go trough AS174 and then join back by New-York (see Fig. 7 for the traceroute). This behaviour diminishes the traffic from Chicago to New-York on certain part of our CDN, which is a non negligible benefit. Since as174 is our shared peer, it doesn't cost us more than directly going from Chicago to New-York.

When Chicago's router want to reach the CDN in New York, the packets will go trough AS174 without looping. Going trough AS174 is a normal behaviour here. `chi_1` and `chi_5` have two different paths to reach `nwk_1`. Let's use `chi_1` for our example:

- One path with a cost of 4: `chi_1` → `ash_1` → `nwk_1`
- One path trough our peer `as_174` with a cost of 1. If we consider that the cost to go trough Cogent only costs 1. Cogent will then forward the packet back to our network.

This may seem weird to go trough another AS but because of the nature that Cogent was set up as one of our peer. If this was a *client-provider* relationship, it would have been more costly to go trough Cogent and our router would have preferred to go trough the first path mentioned (`chi_1` → `ash_1` → `nwk_1`).
Check our diagram for more details at Fig. [6]

B. BGP communities

For better communication between routers, we have added BGP communities. So we have 8 communities, one for each Autonomous System connected to our network, namely our 3 transit Telia, Cogent, Level3 and our 2 Stub, Amazon and Charter as well as one community for each continent in our network (Europe, Asia and America). These BGP community attributes will be used to export and import with different filters, so upstream service provider routers can use these communities to apply specific routing strategies. In our case, here is how our communities are configured:

- AS3356 (Level3) Community 11
- AS174 (Cogent) Community 12
- AS1299 (Telia) Community 13
- AS7843 (Charter) Community 21
- AS16509 (Amazon) Community 22
- AS16276 (OVH North America) Community 3
- Europe Router Community 4
- Asia Router Community 5

These BGP communities helps us categories and localize prefixes that are communicated to us using iBGP and eBGP.

First, let's talk about categorical communities. If we receive a prefix with a community between

- 11-20, this means that the prefix was advertised by a transit.
- 21-30, this means that the prefix was advertised by a stub.

This can be used for different things. Our routers will be able to treat prefixes from stubs and transits differently.

- we could use a router to only store a part of prefixes of one peer and another router will store the rest. It will allows to store less prefixes on one router, by partitioning them on the different routers from one data-center.
- if we had a certain contract with a specific stub or transit, we can advertise different med to each of them and also set custom local preference if certain clients wants to create backup links.

The second type is geographical communities. They help us and customers of our transit network to customize the behaviour in prefixes advertising. By knowing where prefixes are from, we can detect failures or anormal behaviour on our network but also restrict prefixes to certain geographical area.

All these communities were implemented but we did not have time to perform a nice practical example for them.

C. BGP security

One of the major risks in BGP is BGP diversion. This allows a third party user to disclose BGP networks to third parties without the consent of the rightful owner. This is known as using the right filters to prevent malicious third parties from diverting traffic for criminal purposes. The first barriers to prevent this are :

- firewalls
- IDS/IPS systems

But this is often not enough for people specifically targeting the structure you are managing. In such cases, other methods have to be used. A secure routing solution is then used, so that the network cannot be exposed. This system is applied to BGP routers bordering the DFZ network (free zone by default). Routing will only be allowed via BGP sessions with clients, transit providers and colleagues. In addition, routing from their own and their customers' networks will be flagged with a combination of prefix, prefix length, originating AS and AS-pad.

- AS: An Autonomous System, or autonomous system, is a set of IP computer networks integrated with the Internet and whose internal routing policy is consistent. An AS is generally under the control of a single entity or organisation, typically an Internet Service Provider.
- PAD: Packet Assembler and Disassemble: Software for packet assembly (and disassembly on arrival) for transmission over a packet-switched network).

This level of security aims to reduce the number of potentially invalid messages that a network receives by rejecting invalid route messages and ensuring that only the owner can communicate the correct prefixes. By taking appropriate measures on edge BGP routers, it is possible to prevent the AS network from reaching an invalid prefix. This then makes it impossible to establish two-way communication and greatly complicates the propagation of threats.
We understood these different concepts on a theoretical standpoint but in a practice we didn't know how to implements these concepts.

VII. Conclusion

Our network is already well developed but it still misses some core functionalities such as blackholing using BGP communities and additional security.

References

https://blog.apnic.net/2019/10/25/as-prepending-in-bgp/
https://datacenter-magazine.fr/securite-des-reseaux-comprendre-le-detournement-bgp-pour-proteger-les-systemes-dentreprise/

APPENDIX
OSPF DIAGRAMS

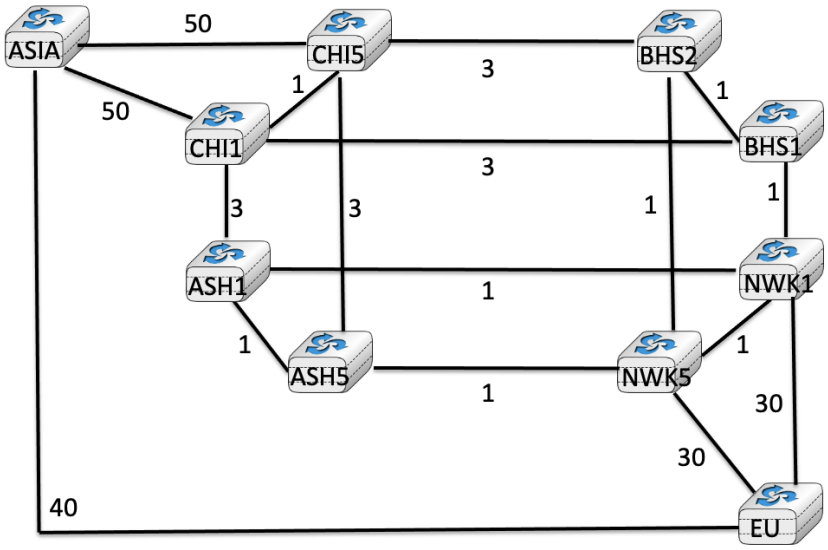

Fig. 2. OVH OSPF Model

BGP DIAGRAMS

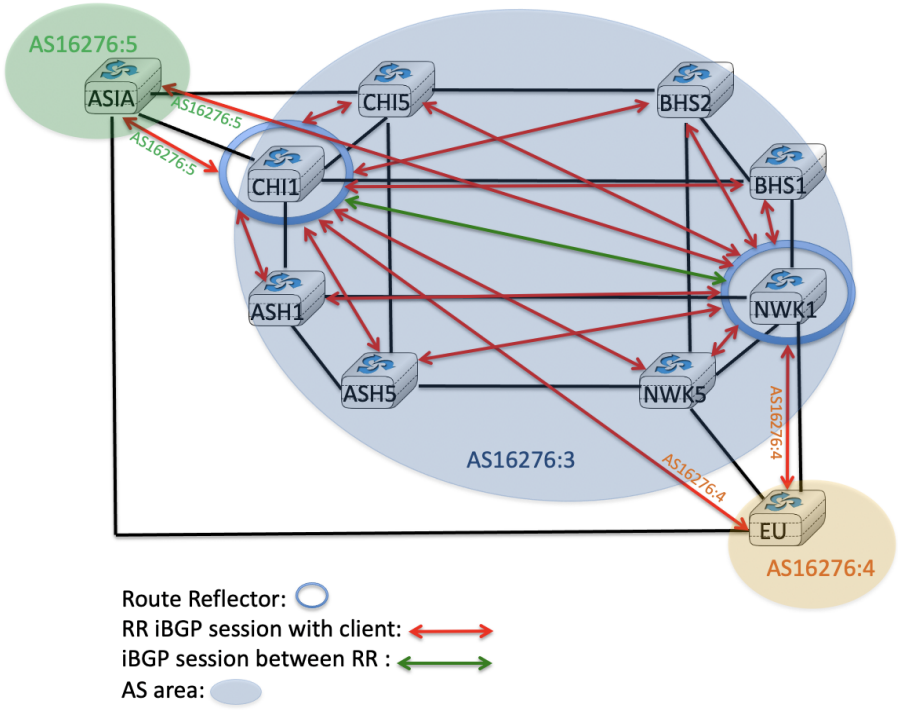

Fig. 3. OVH BGP Model

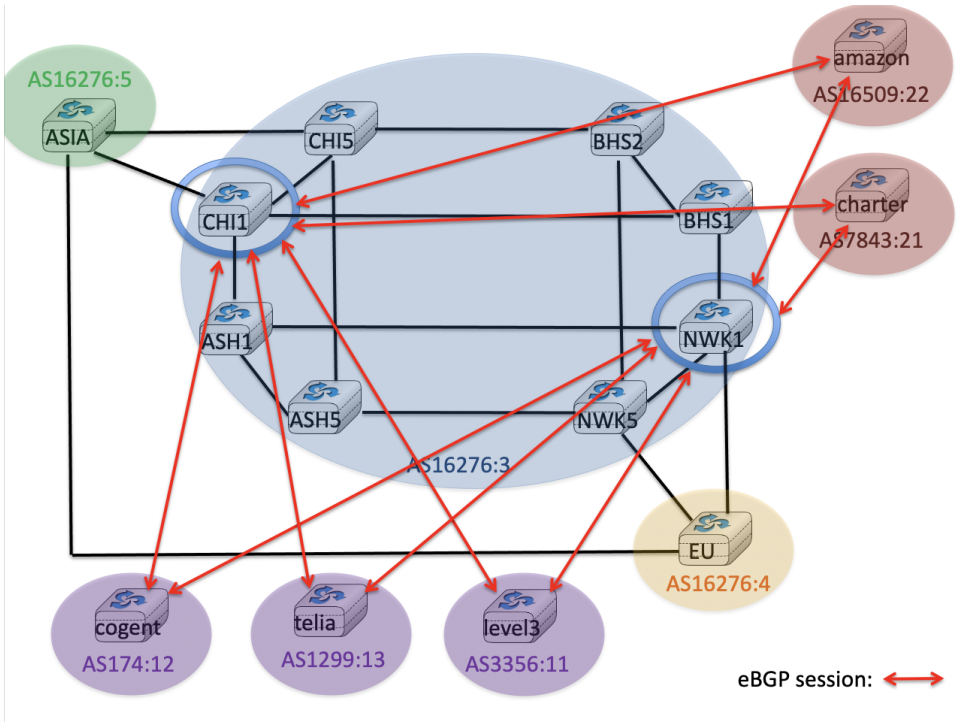

Fig. 4. OVH eBGP Model without Anycast

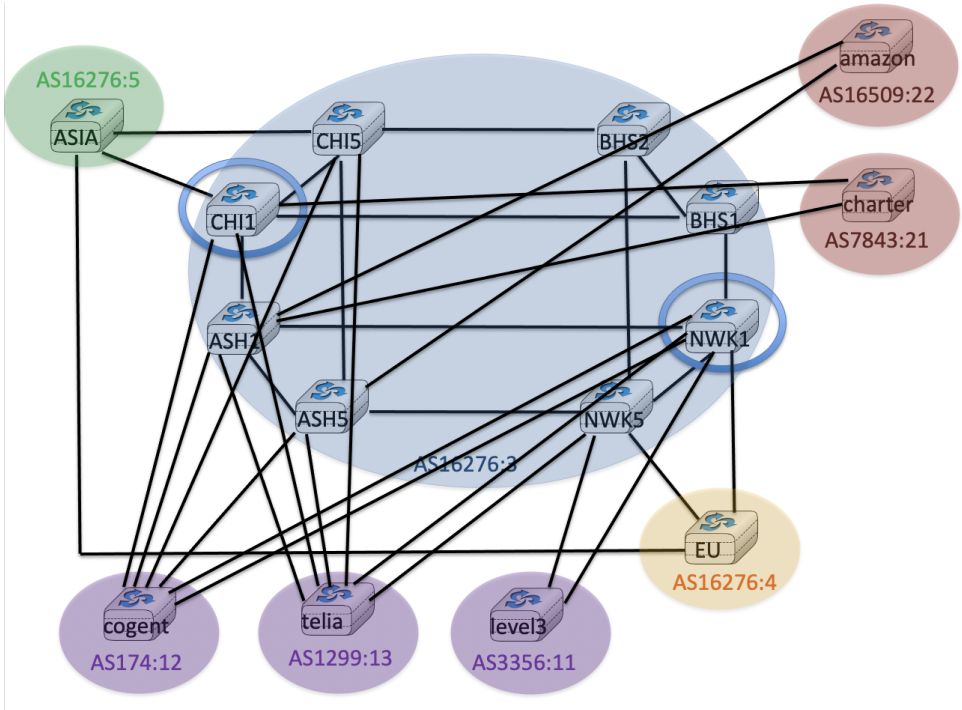

Fig. 5. Physical links between our network and all ASes, without Anycast

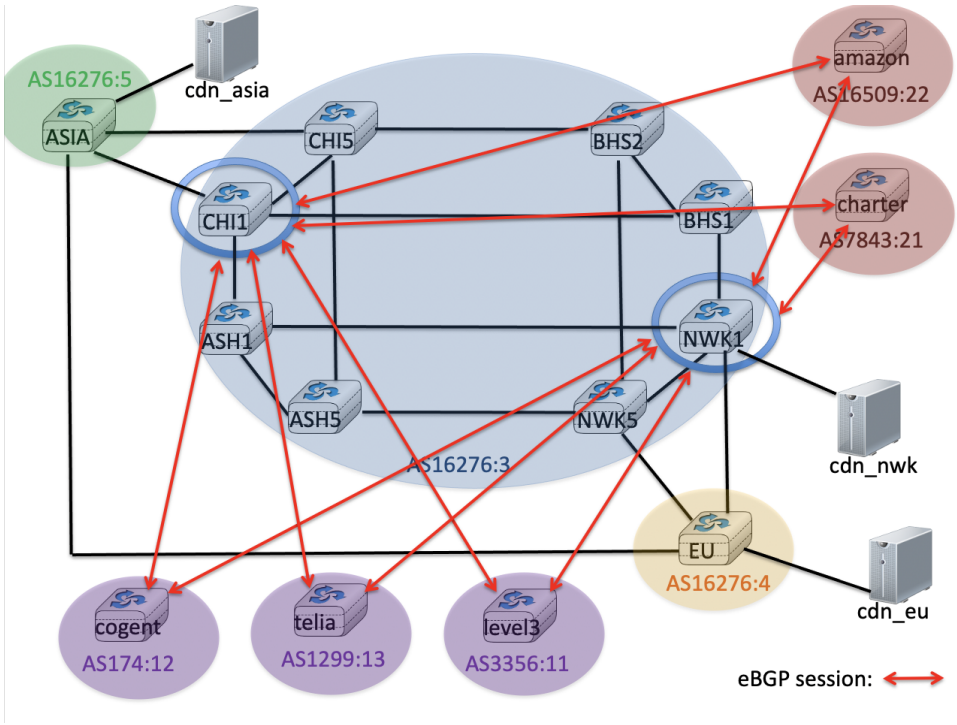

Fig. 6. OVH eBGP Model with Anycast

```
mininet> chi_5 traceroute cdn_host1
traceroute to 10.0.3.2 (10.0.3.2), 30 hops max, 60 byte packets
 1  as174_r1 (192.168.34.1)  0.027 ms  0.004 ms  0.004 ms
 2  nwk_5 (192.168.32.1)  0.012 ms  0.005 ms  0.005 ms
 3  nwk_1 (192.168.87.2)  0.017 ms  0.006 ms  0.006 ms
```

Fig. 7. command: chi_5 traceroute cdn_host1