Securing BGP Communities with Blockchain

Computer engineering final project

Roger Coll Aumatell

Director: Albert Cabellos

Information Technologies

20/01/2020

Table of contents

1. Context 6
    1.1 Information 6
    1.2 Internet 6
    1.3 Border Gateway Protocol 7
    1.4 BGP Communities 9
2. Justification 11
    2.1 Motivation 11
    2.2 Existing Solutions 11
3. Scope 13
    3.1 Problem 13
    3.2 Objectives 14
    3.3 Ultimate aim 14
    3.4 Possible obstacles 14
    3.5 Concerned parties 15
    3.6 FIB context 15
4. Organization 17
    4.1 Division of labor 17
    4.2 Methodology 17
5. Description of tasks 19
    5.1 Project calendar 19
    5.2 Resources 20
        5.2.1 Personal resources 20
        5.2.2 Equipment 21
6. Gantt 23
7. Risk management 25
    7.1 Obstacles 25
    7.2 Possible solutions 25
8. Economic management 28
    8.1 Human resources costs 28
    8.2 Material costs 29
        8.2.2 Indirect costs 30
    8.3 Contingency and unforeseen events 30

8.4 Management control and total costs 31
9. Sustainability 33
9.1 Economical dimension 33
9.2 Social dimension 33
9.3 Environmental dimension 33
9.4 Matrix of sustainability 34
10. Implementation 36
10.1 BGP Communities Parser 36
10.1.1 Previous Concepts 36
10.1.2 Technical concepts 39
10.2 Blockchain 43
10.2.1 Previous concepts 43
10.2.2 Technical concepts 44
10.3 Compiler 47
10.3.1 Previous Concepts 47
10.3.2 Technical concepts 47
11. Experiments 50
11.1 Localhost experiment 50
11.2 Real-world deployment 51
12. Results and benchmarks 54
13. Conclusions 59
13.2 Future Scope 60
14. Acknowledgment 62
15. Bibliography and references 64

Abstract

Internet is used in our daily life in any technological device we handle. Confidential data is also sent between countries in just few seconds through the Internet. But what if I say that Internet is not perfectly secure and that there are known and opened vulnerabilities?

This project aims to solve some of these vulnerabilities explained in a research paper [1] using a decentralized approach and, more specifically, generate multiple benchmarks in order to validate the solution performance.

The complete development of the system exceeds the scope for a single technical engineering project. The work has been divided in distinct sections and distributed between two engineers. As all the sections are complementary this document contains the details of all of them.

Hyperledger [2] has been used as the base decentralized system, nevertheless, many features have been modified to fulfil our project dependencies.

1. Context

1.1 Information

Information [3] can be defined as knowledge obtained from investigation, study or instruction. In our daily life we use information constantly, since the day we born we start getting information from our parents and surroundings to understand more what we are and what we see is.

Some years ago, information was transferred with oral tradition. Oral tradition is a form of human communication wherein knowledge, art, ideas, and cultural materials is received, preserved and transmitted orally from one generation to another. Fastly, the first book was created, it was a new way to preserve information without the need of a human brain. It was a huge step for the humanity as our nowadays knowledge comes from books.

In 1920s, television was born and in 1950 was the primary medium for influencing public opinion. With the television we were able to transmit oral information for any point in the world to another one at the same time.

Finally, the prizewinner comes in 1960s with Internet.

1.2 Internet

On October 29, 1969 at 10:30 PM, Internet history was made, as it was born [4] with the transfer of one simple message.

At 10:30 p.m., a student programmer at UCLA(University of California, Los Angeles) named Charley Kline sent the letter “l” and the letter “o” electronically more than 350 miles to Stanford Research Institute Computer in Menlo Park, California. The letters stood for “login”, and the effort led to a system crash immediately afterward. But a technological revolution had begun.

As many of the biggest world discovers came from military research. After the nuclear discover in World War 2 the world was frightened of a nuclear apocalypse, which caused that millions of dollars were destined to military research. The military needed a way to command and control their computers remotely in the case of an attack, and from this need Internet was born, in that times it was called ARPANET (Advanced Research Projects Agency Network).

What they didn’t know at that moment is that ARPANET, closed at 1989, will become the most common way to transmit and store any kind of information worldwide, nowadays called Internet.

The Internet is the global systems of interconnected computer networks that use the Internet protocol suite to link devices worldwide. This link allows humans to read, write and transfer information between two computers instantly.

A protocol [5] is a standard set of rules that allow electronic devices to communicate with each other. These rules include what type of data may be transmitted, what commands are used to send and receive data, and how data transfers are confirmed, among others. For the correct operation of Internet, it also uses multiple protocols.

There are many different protocols to be used in a communication, but Internet uses TCP(Transmission Control Protocol) or UDP(User Datagram Protocol) as the transport protocol and later on uses BGP(Border Gateway Protocol) in the upper layer, as the application layer.

1.3 Border Gateway Protocol

The Internet protocol suite [6] is the conceptual model and set of communication protocols used in the Internet. Provides end-to-end communication specifying how data(information) should be packetized, addressed, transmitted, routed and received. This functionality is organized into different layers, each of them with different scopes and protocols. The layers represent data transfer operations common to all types of data transfers among cooperating networks.

The International Standards Organization (ISO) described an ideal structure made of seven layers, each of which has one or more protocols associated with it.

However, as mentioned before, OSI model describes an idealized network communication protocol. Internet works thanks to TCP/IP protocols which does not correspond to OSI model directly, as it either combines several OSI layers into a single layer.

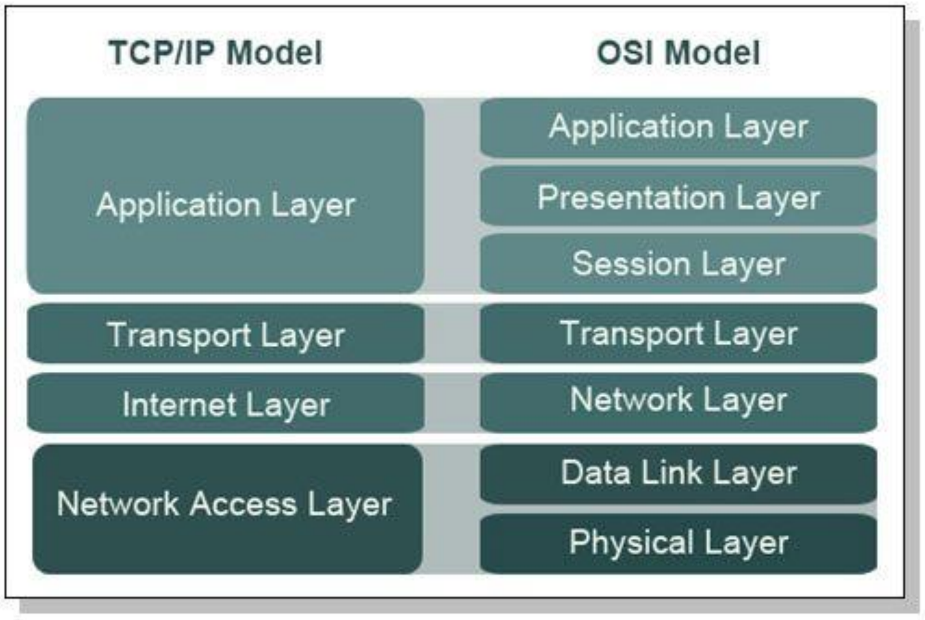


Figure 1 TCP/IP vs OSI model

Each layer can be implemented with different protocols depending of the final use of the application and the proposal of the layer, each protocol has different functionalities.

For example, the Transport layer is the responsible of managing the transfer of data and ensures that the data received and transmitted are identical. TCP and UDP are protocols examples that can be used in that layer. Another one is the Network layer, which manages data addressing and delivery between networks, the most well-known protocol of that layer is called IP (Internet Protocol).

For our research we will focus on the top layer of the TCP/IP model, the Application layer, which englobes the Application, Session, Presentation layers of the OSI model. This layer focuses on:

- Managing the connections and terminations between cooperating computers
- Ensures that the information is delivered to the receiving machine in a form that it can be processed
- Services with standard communication and applications that everyone can use

There are many protocols in this layer, some of them are: DHCP, DNS, FTP, HTTP, HTTPS, IMAP, LDAP, MGSCP, LDAP, NTP, POP, SMTP, SSH, RIP, TLS/SSL, Telnet, etc. Indeed, this final project will focus in a particular protocol of this layer, the BGP (Border Gateway Protocol) protocol.

Border Gateway Protocol is designed to exchange routing and reachability information among Autonomous Systems on the Internet. Routing is the process of moving packets (amount of information) across the network from one host to another, it is usually performed by dedicated devices called routers. Autonomous Systems (AS) are organizations that manages and supervises a network, or a collection of networks and they have their own Internet prefixes. Basically, BGP protocol is used by Autonomous Systems to transmit information between them, usually this information allows them to keep the Internet resources updated.

1.4 BGP Communities

During a BGP communication between two ASes, routing information is exchanged using BGP routers. Routing information can be explained as information related to one or more Internet resources. One Internet resource is a point where an end-user will retrieve information and can be identified with an address IP(Internet Protocol), furthermore, a prefix(aggregation of IP addresses).

A BGP community is a bit of "extra information" that Autonomous Systems can add to one or more prefixes during a BGP communication between BGP neighbors. BGP communities are an optional transitive [7] BGP attribute that traverse from one Autonomous System to another one. A BGP community is a 32-bit number that can be included with a route.

A BGP community by itself won't change any behavior on a BGP router, in fact, the community same value can operate very differently depending on the Autonomous System. Normally, the actions that must be performed when an Autonomous Systems receives a BGP community are discussed previously between it's BGP neighbors using a telephone, email or any other information device. Nevertheless, there are some well-known communities [8] which means the same for all the Autonomous systems:

- NO_EXPORT(0Xffffff01): tells a BGP router it should only propagate any prefixes this community is attached over Internal BGP, and not propagate it over External BGP (other ASes) to external autonomous systems.
- NO_ADVERTISE(0Xffffff02)
- NO_EXPORT_SUBCONFED(0Xffffff03)
- NOPEER(0Xffffff04)

The well-known communities are very few compared with the total number of communities that can be defined (32-bit field), thus, generates a lot of misconfigurations and leaks.

2. Justification

2.1 Motivation

The paper *BGP Communities: Even more Worms in the Routing Can (November 2, 2018)* [1] proves different vulnerabilities found in BGP Communities that can affect the hole functionality of Internet. The most well-known attack that can be deployed using BGP communities is hijacking a route, that corresponds to announce a prefix for which the Autonomous System is not responsible for. Nevertheless, many other attacks are recognized, from disabling an access point for some minutes till some hours for the whole world, to changing the path of a prefix for another one just for economic interests and, among others, making a man in the middle attack within a prefix.

It is very interesting, that in the mentioned document only specify some attacks that the researchers of the article found or thought, but as they say the attacks that can be released due to this BGP communities vulnerability are not closed and neither small.

The main motivation is that all the vulnerabilities mentioned before and, in the paper, and also the ones that are not yet found or made can be parched if BGP communities are treat in a secure way.

The main idea of the project is to find a valid solution for securing BGP communities. Another stimulus is using blockchain as a possible solution, due to its newness and the ideal scenario that seems to present for using this technology.

2.2 Existing Solutions

After some research we have found only two proposed solutions:

- RPKI: Resource Public Key Infrastructure [9] is a public key infrastructure framework designed to secure the Internet's routing infrastructure, specifically the Border Gateway Protocol. Basically, RPKI provides a way to connect an Internet prefix (such as IP addresses) to a trust anchor. The main problem of this solution is that is a centralized solution, and Internet is the most decentralized thing, there exist multiple entities that manages Internet information in each continent but there isn't a consensus between them, and nearly impossible to achieve it.
- BGPSec: Border Gateway Protocol Security [10] is a security extension of BGP. Provides to receivers of valid BGPsec UPDATE messages cryptographic verification of the routes they advertise. BGPsec replaces the BGP AS_PATH attribute with a new BGPsec_Path attribute. This new change entails to change the whole Autonomous Systems infrastructure, which is impossible.

As said before, there isn’t still a valid solution and our proposed solutions solves the impossibilities that prevents the previous projects to succeed.

3. Scope

3.1 Problem

As mentioned before, has been proved that BGP communities have vulnerabilities. Malicious people can take advantage of this vulnerabilities and exploit them to take profit or even for just for fun. The problem is that if an attack to BGP communities is successful several access points to the Internet can be unreachable for a certain amount of time. The paper mentioned before explains different kind of attacks which can be deployed in BGP communities and that they aren't solved yet.

Indeed, the main problem remains because BGP communities are not formally defined, this means that the meaning of the community value is agreed among groups of collaborating ASes. This issue, coupled with the lack of security mechanisms protecting BGP communities, means that communities are error-prone and a source of many misconfigurations in inter-domain routing.

BGP route leaks cause important service disruption on the Internet. As a recent example, in June 24, 2019, several websites started to have performance issues, including Discord, AWS services and OneTrust. According to [11], on that day Allegheny Technologies Inc. propagated prefixes received from one of its providers (DQE Communications) to another provider (AS701 – Verizon). As shown in figure 2, AS33154 sent to his customer AS396531 routes for several popular Internet destinations such Amazon, Facebook and Cloudflare. Due to a BGP misconfiguration, AS396531 leaked such routes to AS701. In a normal scenario, a customer should never send routes learned from a provider to another provider. As a result, AS701 accepted these routes and advertised them, leaking such routes further on a global scale. AS701 was not suitably equipped to deal with this drastic increase in traffic, causing disruption in service. Consequently, there was a major outage of such popular customer-facing services, this last for roughly 126 minutes which is a lot for those ultra-high availability services.

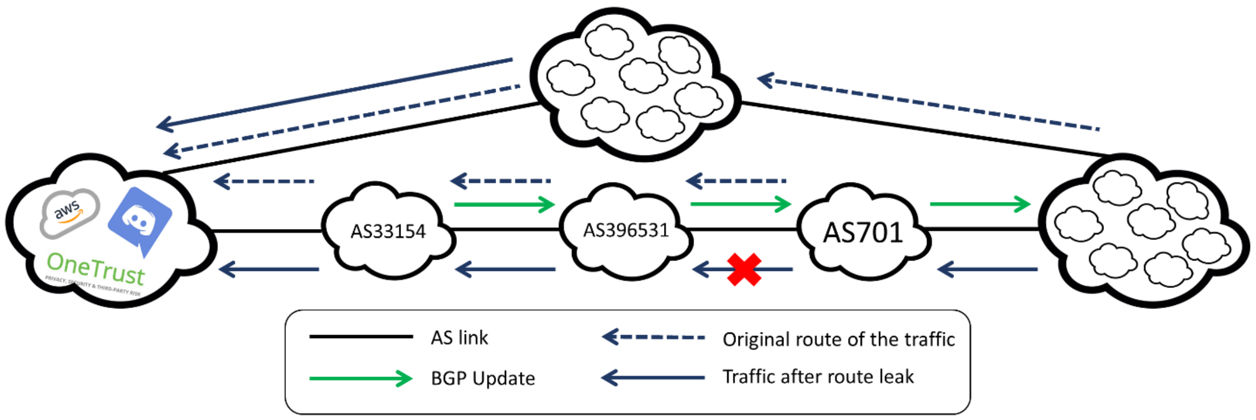


Figure 2: Important service disruption In June 24, 2019

3.2 Objectives

The main objective of this project is to prevent some attacks that can be performed BGP communities. These includes:

- Fully understanding of how BGP communities work and how are they used in the real world.
- Propose a formal, expressive and rich language for BGP communities.
- Use blockchain technology to build a distributed database of BGP communities that can be used between BGP neighbors.
- Collect a real dataset of communities and transform into computer readable files for subsequently use them in our blockchain.
- Provide benchmarks of the implemented solution.
- Write a research paper.

Our solution works as follows. First, we define a formal language used to describe BGP routing policies. The language is used by network operators (Autonomous Systems workers) to define their routing policy. This will prevent misconfigurations due to ambiguity or the ill-defined nature of BGP communities. Later, the policy is uploaded to the distributed ledger. The distributed ledger(blockchain) is shared among the participant ASes and it is used to securely communicate the BGP routing policy.

3.3 Ultimate aim

In the end of the project, is expected to have a prototype based on the objectives described previously. This prototype should be able to be tested with the collected dataset of BGP communities and export some benchmarks.

The final aim is to write a research paper with the extracted results with the measurements made and, finally, have feedback of the project of any real autonomous system.

3.4 Possible obstacles

The principal obstacle that can be thought is finding real BGP communities. When saying real, it is referring to BGP communities that Autonomous Systems use in their daily life. It can be difficult due to communities are used only internally in their network and BGP neighbors, there's no need to be public.

Another clear obstacle could be the difficulty to transform those communities to a computer readable format, because they are defined as human text. Also, the creation a formal language to exchange BGP routing policies.

For the blockchain part, it will be used Hyperledger technologies which is relatively recent and together with the creation of a compiler for the blockchain communities could also be a barrier for the project progress.

3.5 Concerned parties

The main companies that can be interested in the project are the ones in the networking sector, especially the ones which are Autonomous Systems. Moreover, everyone that uses Internet could be interested due to its objectives, making Internet more secure and preventing global break downs.

3.6 FIB context

The project involves multiple technologies, as networking, routers, blockchain, APIs and multiple programming languages (from high level ones like Go to very low level as the ones used for configuring routers). All those technologies are explained and some of them learnt in my FIB's specialty, information technologies, indeed, the projects fit in perfectly.

4. Organization

4.1 Division of labor

As it is a very wide project with clearly distinguish sectors, it will be divided in the following parts:

- Find real BGP communities
- Parser: Make a program which can transform human written BGP communities to configuration files which a computer can interpret.
- Blockchain: Using Hyperledger technologies design a chain code able to transfer communities and simulate entities as Autonomous Systems
- Compiler: Design a compiler which can transform blockchain communities to actual BGP routing policies
- Testing: Make different tests and benchmarks with the implemented solution
- Paper: Write a research paper

The test part is defined as an individual part, nevertheless, each of the previous parts will contain a unit testing to prevent huge rollbacks.

4.2 Methodology

In the project we can distinct four distinct phases:

- Documentation: This part might be the largest one due to the newness of this technologies and the need to fully understanding for choosing the best way and practices for the project implementation.
- Implementation: Blockchain and BGP communities implementation will be done while the concepts are understood.
- Testing: Test will also be done while implementations releases are done.
- External feedback: It the test are the projects succeed it will be great to get some external feedback, for example, of someone working on an Autonomous System.

We have chosen to use Agile methodology, instead of Cascade. The main reason is the newness of the technologies unlike Cascade, Agile allows to implement while also documenting. That is great because due to the final results uncertainty many doubts and rollbacks will appear.

Weekly meetings will be done with the project director, where we will explain the progress and problems or doubts that may arise during the implementation and testing.

5. Description of tasks

The hole project can be divided in multiple tasks:

- Begging of the project: It refers on the first month of the project and it involves the bureaucratic parts like documentation and pre-registration. It also includes some research of the fields that will be primary enrolled in the project.
- Project management: During this task it will be defined how the work will be distributed, the purpose of the project and some related fields that are related to the study, like finances and sustainability.
- Implementation: This part can be divided in multiple sections, depending on the things to implement, but each of them is involved in basically three actions; information collection, design and code implementation.
- Testing: To prove that a program works, tests must be done. Luckily, unit tests will be defined and executed so we can advance as we keep developing and we not bet everything on a final test.
- Benchmarks: For a research paper it's interesting to show some results and comparisons to previous implementations. Benchmarks will be done for either test or real input to reach this goal.
- Documentation: Once we collect real results and benchmarks, we will write a research paper to make it public for the world community and a memory will be written for this project.

5.1 Project calendar

The following table list all the project tasks with the estimation of the hours for its realization, it must be taken into account that the calendar is an estimation done before the realization and with technological developments the real time spent can vary a lot.

Task	Responsible	Hours
Begging of the project		**50**
1. Start up	Director	10
2. Investigation	Director and developer	40
Project management		**90**
1. Scope	Engineer and director	20
2. Planification	Engineer and director	20
3. Economy management	Engineer	10
4. Sustainability definition	Engineer	10
5. Final document and presentation	Engineer	30
Implementation		**200**

BGP Communities samples		10
Parser		50
1. Analysis of requirements	Engineer	10
2. Design	Developer	20
3. Testing	Developer	10
4. Parsing real communities	Developer	10
Blockchain		70
1. Analysis of requirements	Engineer	20
2. Design	Developer	30
3. Testing	Developer	10
4. Tests with parser outputs	Developer	10
Compiler		70
1. Analysis of requirements	Engineer	20
2. Design	Developer	40
3. Testing	Developer	10
Testing		**50**
1. Tests creation	Developer	40
2. Tests execution	Developer	10
Benchmarks		**50**
1. Definition	Engineer	20
2. Implementation	Developer	20
3. Execution	Developer	10
Documentation		**100**
1. Research paper	Engineer	25
2. Memory	Engineer	50
3. Feedback internal/external	Director	25
TOTAL(director)		**75**
TOTAL(development and engineer)		**365**

Table 1: Estimation of the hours per task

5.2 Resources

5.2.1 Personal resources

As mentioned before the project started in June-July so the final duration of the project is five month (excluding August). This was done because as it is a research project there's a lot of risk and many failures possibilities. Another thing to consider is that the student is currently working six hours per day and making two other subjects.

To fulfill the marked times, it is estimated a dedication of 10-20 hours per week. This hours can vary depending on the many variables, the load of work, feedback needs, external help, implementation changes, etc.

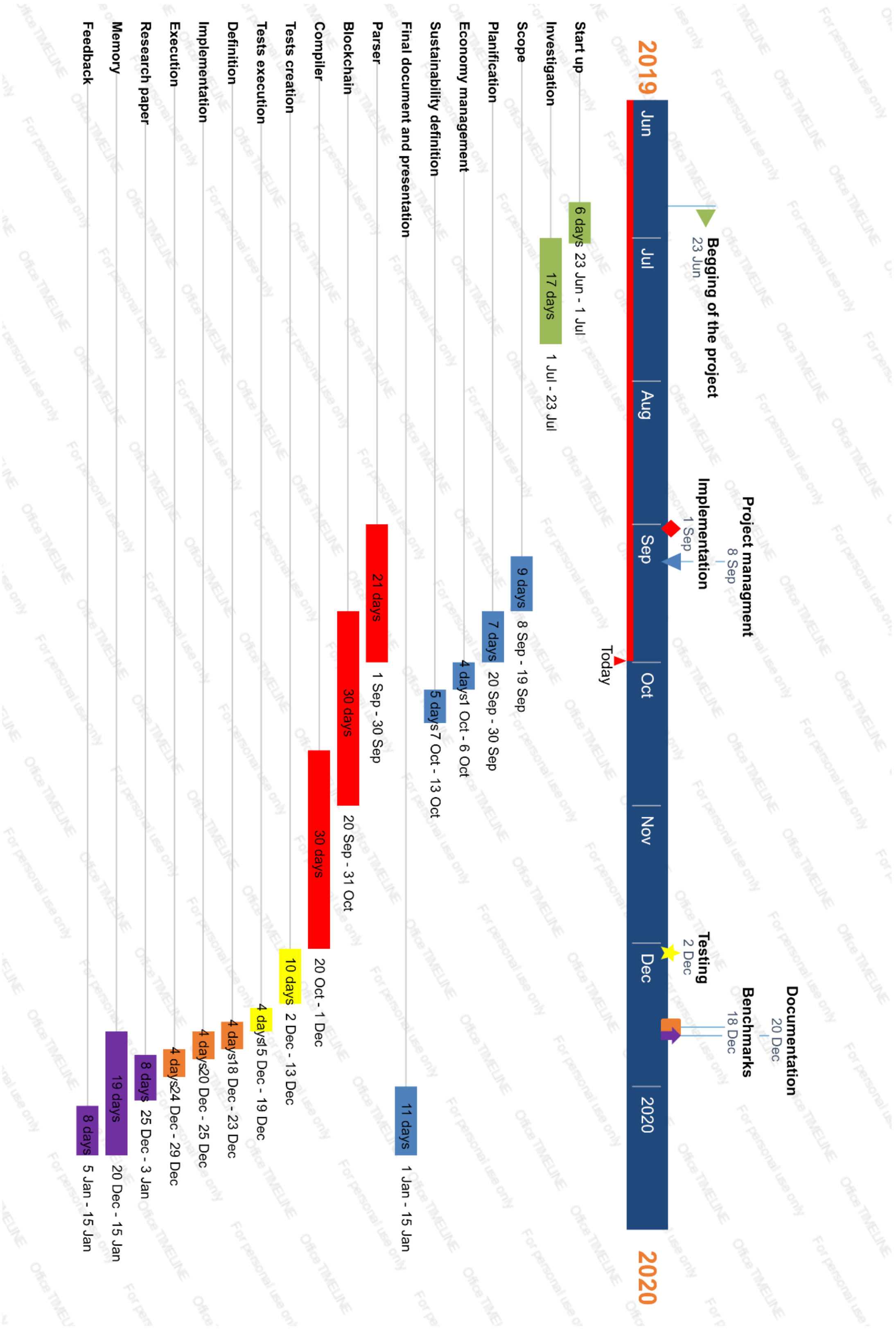
5.2.2 Equipment

In this part will be defined the required equipment to develop the project, also the tools that will be used for communication and documentation:

- **Desktop computer:** The different members only have available desktop computer for the development, due that there's no need for big computation, desktop computer will be enough.
- **Ubuntu:** Ubuntu operating system will be used because blockchain dependences. Hyperledger is a very new open source blockchain project and right now it's only available for Ubuntu environments.
- **Github:** Will be used for keeping updated the code and for director's revision at any time. Also, because its facility installs Github code in any machine.
- **Hyperledger:** Open source blockchain project made for IBM and Linux Foundation. Our chain code will be implemented over Hyperledger.
- **Antlr:** A Java plugin used for parsing programming languages. It can be useful during the parsing task.
- **Google Cloud Platform:** Cloud for the blockchain deployment, due to its facility to run multiple instances install. Moreover, Google Cloud Platform provides a natural language API that can be considered if Antlr fails.
- **OfficeTimeline.com:** Online tool to design Gantt diagrams.
- **Skype:** Communication tool that will be used weekly to keep track of the project between the members of it.

- **Word and PowerPoint:** Tools that are already installed on the members desktop computer for writing the documentation and designing the presentation.
- **Git:** Software tool for the versions control of the project.

6. Gantt


2019
2020
Jun
Jul
Aug
Sep
Oct
Nov
Dec
2020
Begging of the project
23 Jun
Implementation
1 Sep
Project managment
8 Sep
Today
Testing
2 Dec
Benchmarks
18 Dec
Documentation
20 Dec
Start up
6 days
23 Jun - 1 Jul
Investigation
17 days
1 Jul - 23 Jul
Scope
9 days
8 Sep - 19 Sep
Planification
7 days
20 Sep - 30 Sep
Economy management
4 days
1 Oct - 6 Oct
Sustainability definition
5 days
7 Oct - 13 Oct
Final document and presentation
11 days
1 Jan - 15 Jan
Parser
21 days
1 Sep - 30 Sep
Blockchain
30 days
20 Sep - 31 Oct
Compiler
30 days
20 Oct - 1 Dec
Tests creation
10 days
2 Dec - 13 Dec
Tests execution
4 days
15 Dec - 19 Dec
Definition
4 days
18 Dec - 23 Dec
Implementation
4 days
20 Dec - 25 Dec
Execution
4 days
24 Dec - 29 Dec
Research paper
8 days
25 Dec - 3 Jan
Memory
19 days
20 Dec - 15 Jan
Feedback
8 days
5 Jan - 15 Jan
Office TIMELINE
For personal use only

7. Risk management

After some research together with the director, it has been seen that there are two tasks with high risk of failure. The following paragraphs will expose the problems which can appear and multiple solutions that could work:

7.1 Obstacles

- Parser: The main problem is evident; the main function of the parser is transforming human sentences to computer readable files. It seems like with only the parser it's enough for a final degree project. In advantage, normally these sentences are in BGP community's context so only a few of human sentences might be treated. Nevertheless, there isn't any available project that could help us with this task and the incapacity of develop this tool can happen.

- Compiler: This problem has some similitudes with the previous one. In these cases, the output of the blockchain might be processed and transformed to BGP routers language. A first sight, it seems easier because it's from machine language to another machine language, but we have no experience with that kind of low-level languages as the BGP routers one and routing policies can be very complex.

7.2 Possible solutions

- Parser solution: First of all, we will try to parse the BGP communities using Antrl [12]. As Antrl is quite old and we are not used to it, a possible alternative is using Google Cloud Natural Language API [13]. Although the API is a good option, it only helps Antlr by providing the part of speech of each word. If any of the previous tasks is not successful, the last chance is to do the transformation manually. Thus, reading real BGP communities and transforming them to configuration files made of our proposed formal language, this will slow down the process considerably, and thus an increase on implementation time.
- Compiler solutions: As mentioned before, this case has minus risk than the previous one. If the main idea does not work, we can make a simulation of a BGP router and do the tests and benchmarks on the blockchain. The last solutions is very similar as the final solution of the previous obstacle, transforming to BGP programming language manually.

As it is an investigation project the timeline will be followed strictly, due to the purpose of publishing the research paper before Christmas. If a lot of time is spent in an obstacle we will move to the following defined solution.

8. Economic management

Many different types of technologies and methodologies are used in the project, so for a better understanding of the economic management the cost will be divided in the following sections:

- Human resources
- Hardware and software (material resources)
- Contingency and global costs

As the final product is a software solution, the hardware needed for its execution will not be counted as material costs, indeed, the client might have the already the corresponding hardware.

8.1 Human resources costs

To detail the human costs, we will use the table used in previous sections, indeed, we will use the table that describes the hours which will be spend in task of the project and who is responsible of it(Gantt table). As during the project, there are multiple roles involved and corresponds with different responsibilities different salaries will be also considered:

- Director: 22 €/h
- Engineer: 20 €/h
- Developer: 15 €/h

Those salaries are estimations based on writer work experience. In Barcelona, those values can vary a lot depending of the company type and its roots(nation).

Task	**Responsible**	**Hours**	**Costs (€)**
Begging of the project		**50**	**960**
1. Start up	Director	10	220
2. Investigation	Director and developer	40	740
Project management		**90**	**1840**
1. Scope	Engineer and director	20	370
2. Planification	Engineer and director	20	370
3. Economy management	Engineer	10	150
4. Sustainability definition	Engineer	10	150
5. Final document and presentation	Engineer	30	450
Implementation		**200**	**3300**
BGP Communities samples	Engineer	10	200
Parser		50	800
1. Analysis of requirements	Engineer	10	200
2. Design	Developer	20	300

3. Testing	Developer	10	150
4. Parsing real communities	Developer	10	150
Blockchain		70	1150
1. Analysis of requirements	Engineer	20	400
2. Design	Developer	30	450
3. Testing	Developer	10	150
4. Tests with parser outputs	Developer	10	150
Compiler		70	1150
1. Analysis of requirements	Engineer	20	400
2. Design	Developer	40	600
3. Testing	Developer	10	150
Testing		**50**	**750**
1. Tests creation	Developer	40	600
2. Tests execution	Developer	10	150
Benchmarks		**50**	**850**
1. Definition	Engineer	20	400
2. Implementation	Developer	20	300
3. Execution	Developer	10	140
Documentation		**100**	**2050**
1. Research paper	Engineer	25	500
2. Memory	Engineer	50	1000
3. Feedback internal/external	Director	25	550
TOTAL(director)		**75**	**1570**
TOTAL(development and engineer)		**365**	**8180**

Table 2: Human costs/task

8.2 Material costs

In previous sections we defined the material resources (hardware and software) that will be used during the project. The intention is to use as much open source software as we can, so many of the tools that will be used are free:

- Ubuntu, GitHub, Hyperledger, Antlr, OfficeTimeline.com (free edition), Skype(free videocalls service) and Git.

Some others, generally hardware, have a cost associated with them. For example, for the computer laptop dependency will be used the Lenovo ThinkPad T480(i5- 8250U, 8GB of RAM and 512 of SDD) due to director's preferences. Word and PowerPoint, that was chosen because a request of the engineer also have economical costs and the impression of all the documentation including the memory, too. The following table represents these costs:

Equipment	Units	Price (€)/Unit	Costs (€)
Computer laptop	3	1279	3837
Office 365 (Word and PowerPoint)	3	10	30
Impression	15	40	600
Total			4467

Table 3: Material costs

The price of the computer laptop and Office correspond to the market value found in Amazon.es in the date on October 7, 2019. The price of the impression it has been asked to the director, due to his experience in previous projects.

8.2.2 Indirect costs

Those costs correspond on extra needs for a correct development of the project. All members of the project live in Barcelona, but we all need a transport method to go to the office. An office will be rent for six months, the ideal should be five months but as the project start in the beginning of July, August must be paid too. The office is located very close to UPC Campus Nort because it's the nearest location for each member. The following table includes all costs related to office maintenance (water, electricity, Internet, management, insurance, etc):

	Units	Price (€)/Unit	Costs (€)
Office rent	6	400	2400
Transport	18(3x6months)	50	900
Total			3300

Table 4: Indirect costs

8.3 Contingency and unforeseen events

Many things can happen in an investigation project and due to the previous costs are estimation based on few working experience addition costs must be added. The riskiest part is the human resources costs. This is because is an investigation project and some implementation or investigation tasks can take much more time that is expected, and thus, increase the number of hours of the workers. The materials resources are more unluckily to suffer unforeseen events as it is part of our daily life, nevertheless, for example, the probability to broke down a laptop is always positive, so to prevent disagreements, additional costs will be added too.

Depending on the risk we apply a different additional percentage to the final cost of the resource:

- Human resources: 15% Total=> 9407 €
- Material resources: 10% Total => 8543,7 €

8.4 Management control and total costs

The components of the project have the aim to prevent the previous additional cost, due to its high and competitive salaries. To work on that, some steps will be mandatory during the project:

- Meeting: Every Tuesday there will be an appointment using Skype mandatory for all the project members. In those meetings each member will explain the improvements done during the week.
- Testing: To prevent going back to previous steps, unitary test will be done for each implementation.
- Security: Each member of the project is responsible of his laptop, this includes lost, theft, etc. Nevertheless, things that cannot be known as battery freeze are excluded from this condition.

We want that the total deviation of the project to be minimal. At the end of the project the deviation will calculated comparing the real cost with the expected (defined in the previous sections); real costs – expected costs. To have a better understanding on how investigation projects work we will also compare the our invested during the implementation, for example, for the blockchain implementation:

- Real hours spent with the analysis of requirements – expected hours defined in the Gantt table.
- Real hours spent in the design – expected hours

In the end of the project, thanks to these comparisons we will be able to know in which task part normally takes more time.

Afterall, the total expected cost is 17950,7€.

9. Sustainability

9.1 Economical dimension

One of the biggest aims of blockchain technology is to prevent control of entities over other people. Normally those entities have a commission on each transaction that is done in the application, blockchain disables this central entity and thus, no extra charges are made. Using blockchain have real economical social benefits.

To see real economic effects the project must be tested in a real environment. It is obvious that this is a big issue, due to that is not an application that one can test at home. To test the application a mandatory hardware resource is a BGP router, but not only this, to use this kind of router one needs to be part of a real Autonomous System and have access to the Internet network. All in all, the economy impact is very hard to estimate and the only possibilities to calculate it; is to know someone working in an AS or the research paper to have enough impact to make big companies' interest.

It might be pinpoint that the solution that purposes this project does not involve the substitution of the current hardware that is used by the Autonomous Systems, indeed, the solutions could work with the actual BGP routers.

For the tests of this project we will use abstract BGP routers.

9.2 Social dimension

This aspect does not affect directly to social life, nevertheless they data security and high availability Internet does. This project refers to very internal aspects of the Internet which only few specialized personalities or big companies know, so it's does not behave on a big change of people's daily life.

However, this project implements some technologies that are indirectly beneficious for social aspects. One of them is the use of a decentralized system, decentralized systems prevents the need to have a central entity which controls all data that flows on that software. During this project the use of decentralized systems will be explained and promoted for other projects/opportunities.

If the project is successful, real attacks to Internet that had happened could be secured, furthermore, would prevent global Internet break downs and thus the final user will have a higher availability of its Internet resources.

9.3 Environmental dimension

There's a big point in this project that was not thought for environment purposes, but it really helps to decrease its enemy. The main software technology that will be used is blockchain, till now, nearly all the blockchain projects, for example, Bitcoin, uses a consensus algorithm know as PoW(Proof of Stake) [14]. Basically, this algorithm is based on the computational force that the members of the blockchain have. The more computational power one has, more powerful is in the blockchain network, more capabilities and decision capacity have over the other. This algorithm makes big investors to buy massive amounts of hardware and to use a lot of electricity. In the following graph we can see some country comparisons:

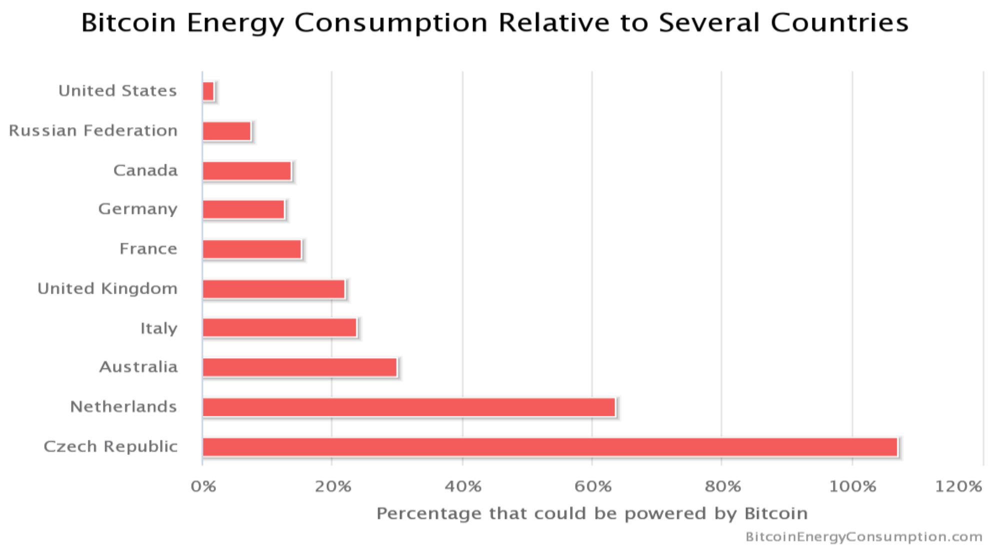


Figure 3: Bitcoin energy consumption comparison [15]

Indirectly, this has a huge impact in our environment. Our project, instead of using PoW algorithm will use PoS (Proof of Stake) algorithm. This algorithm is not based on computation power, instead uses a stake and a random parameter of the blockchain members. That behaves in not a race for hardware resources and thus, much less electricity is spent.

9.4 Matrix of sustainability

	PPP	**Exploitation**	**Risks**
Environmental	8	7	-8
Economic	3	3	0
Social	7	5	-5
Sustainability: 20			

Table 5: Matrix of sustainability

10. Implementation

The whole implementation was done by a team of two members. Miquel Ferriol, which is studying and a researcher of the UPC and Roger Coll which is finishing Computer Engineering in UPC and the writer of this document. The implementation was divided in multiple sections, as seen in previous sections, thus, we hand out those sections between us. Although this division, we work side by side in order to solve multiple problems faster.

Feature	Responsible/s
BGP Communities Parser	Roger Coll
Blockchain	Miquel Ferriol and Roger Coll
Compiler	Miquel

Table 6: Final distribution of the different implementation features

10.1 BGP Communities Parser

10.1.1 Previous Concepts

We started from the need of having input data to test in the final experiment. There was the possibility to generate random data to test but for a research project it wins much more reputation if data that has been used in the real worlds is used to generate the results and benchmarks.

The BGP protocol is not a well-known protocol as Internet Protocol (IP) could be, consequently it was hard to find real data exchanged with the protocol. After some discussions and research, we decided to use the dataset provided by OneStep [16], which is a consulting firm related to BGP Conferences. This dataset provides real BGP Communities that have been used in the wild between multiple Autonomous Systems, in total, 94 Autonomous Systems shared their data with OneStep.

All the BGP Communities shared in OneStep are made of a string value and a description. This description explains what actions must be performed when an Autonomous Systems receives this BGP Community value. Only the string value is sent through the BGP Protocol and the description is shared with the corresponding Autonomous Systems via telephone or any other out-of-band mechanism. Our goal was to transform those descriptions to a defined formal language and share the Community value together with the corresponding actions. The secret remains with the implemented compiler which, is able to understand those actions(formal language) and configure the BGP router automatically and autonomously.

Community String	Local Pref	Effect
174:10	10	Set customer route local preference to 10 (below everything-least preferred)
174:70	70	Set customer route local preference to 70 (below peers)
174:120	120	Set customer route local preference to 120 (below customer default)

Table 7: Example of Local Preference BGP Communities found in OneStep [17]

Community String	Effect
174:3000	Do not announce.
174:3001	Prepend 174 1 time.
174:3002	Prepend 174 2 times.
174:3003	Prepend 174 3 times.

Table 8: Example of Prepend BGP Communities found in OneStep [17]

The next step was to read all the BGP communities (Community value and description) in the dataset and transform them into instructions/actions that our posterior blockchain and compiler could understand. The OneStep dataset has more than 600 BGP Communities, thus it would be a waste of time to transform them one by one and by hand. In order to solve it we decided to make a tool which was able to transform all of them autonomously. This kind of tools are called parsers. The easy part was the community value, as we can see in the Table 7 and 8, the Community String has always the same format; Autonomous System value plus a double dots sign(:) and the BGP community value. The problem remains with the Effect/Description which has been written by a human and from one Autonomous System can differ a lot, even if the actions to be performed to a BGP router for that BGP community are the same.

The tool needed to understand human sentences and transform them into defined actions. In order to be easier, we saw that most of the BGP Communities in the data set can be divided in different known types, this facilitates a lot the transformation. In the following sections we will talk about our defined language instead of defined actions, the difference between the human language sentences and our defined language can be seen in Table 9 and Table 10.

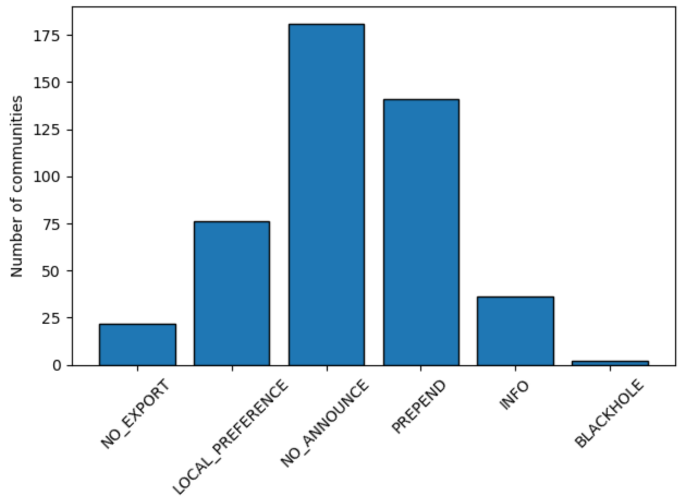


Figure 4: Number of BGP Communities by type found in OneStep

Human language
286:70 ➔ Set Local Preference to 91
286:14 ➔ Prepend 4 times to european peers
286:49 ➔Do not announce to US peers

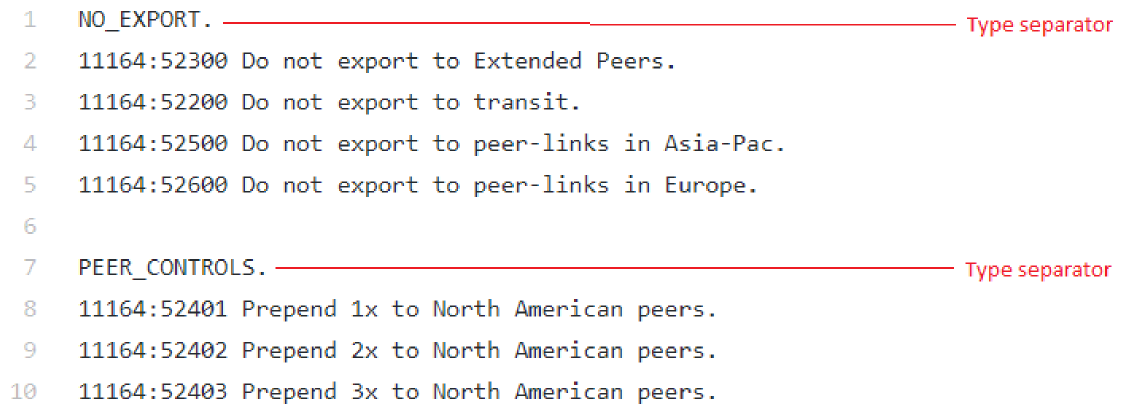
Table 9: Example of human-readable community definition

ASN	**CN**	**Rule**	**Value**	**To**
286	70	LOCAL_PREFERENCE	91	
286	14	PREPEND	4	EZ
286	49	NO_ANNOUNCE	-	US

Table 10: Example of proposed community definition

10.1.2 Technical concepts

The parser had a problem, if it didn't know any previous information of which type of BGP Community was going to read, it would have to know all human English words(all the descriptions are in English) and all the sentences behavior in order to transform them into instructions. For now, this is an impossible task, there isn't any computer in the world which is able to understand all human phrases. To solve this, we had to do a manual part. First, copy all the BGP Communities of one Autonomous System into a text file, read all of them and add a separator between the different known types shown in Figure 4. The structure of a file can be seen in the Figure 5.



```
NO_EXPORT.
11164:52300 Do not export to Extended Peers.
11164:52200 Do not export to transit.
11164:52500 Do not export to peer-links in Asia-Pac.
11164:52600 Do not export to peer-links in Europe.

PEER_CONTROLS.
11164:52401 Prepend 1x to North American peers.
11164:52402 Prepend 2x to North American peers.
11164:52403 Prepend 3x to North American peers.
```

Figure 5: Portion of the BGP Communities file for the Autonomous System 11164

Once all the files were ready, they were passed throw the parser tool which firstly only knew what type of BGP Communities was reading in each moment thanks to the type separator. The problem was that each description was written differently, and the structure of the sentence was different too; some had de indirect complement at the beginning, others in the end, some uses always the same verbs, etc. We needed some help in understanding the structure of each sentence, moreover, the parsed needed to know the meaning of each word. After some research on the Internet we discovered the Google Cloud Natural Language platform [13].

Google Cloud Natural Language uses Google machine learning to revel the structure and meaning of unstructured text. It can extract information about people, places, nouns, verbs, and better understanding of the text. Indeed, it has a Natural Language API that lets the developers work with natural language understanding features including sentiment analysis, entity analysis, content classification, and syntax analysis.

Basically, the BGP Parser tool gets the BGP Community of each Autonomous System file, sends the Communities description to the Natural Language API, which sends back to the BGP Parser a syntax analysis of each description, and finally with the information provided by the API the

BGP Parser generates the configuration file that the compiler can understand to create BGP router instructions.

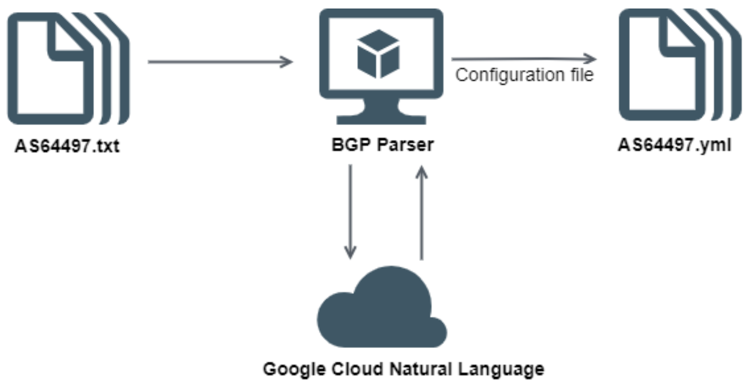
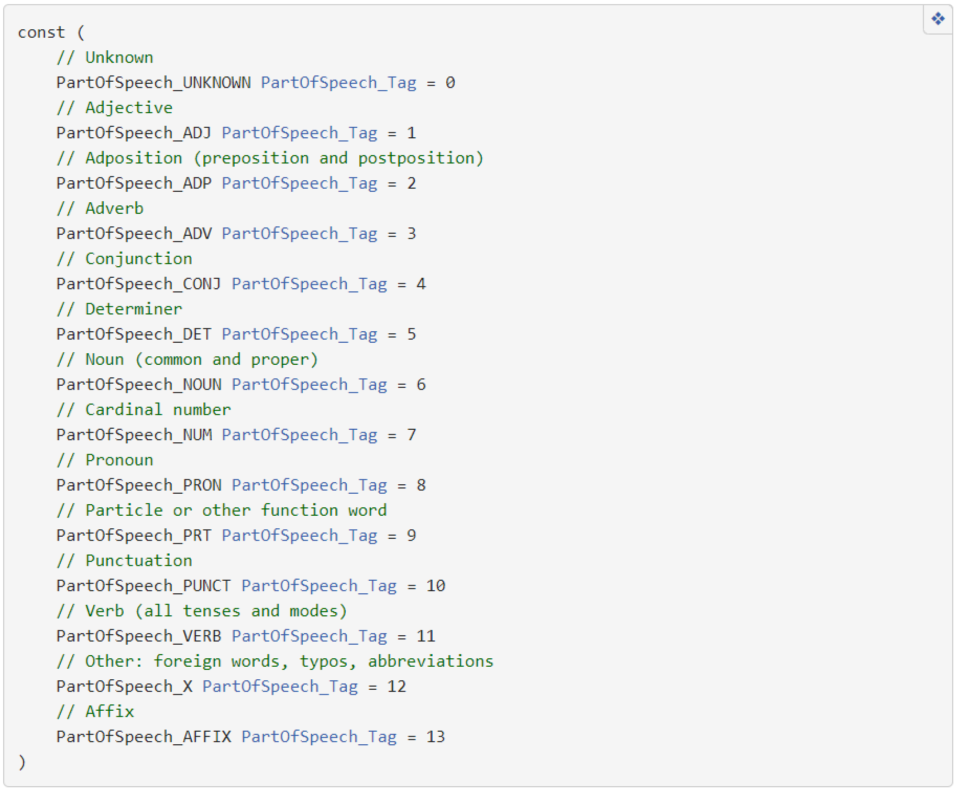


Figure 6: Parser workflow

For example, the BGP Community description “Prepend 174 2 times.” contains the verb prepend, the number “174” which refers to the Autonomous System 174, the number “2” that represents the number of times that the route must be prepended, and finally the noun times. All the prepend BGP Communities for the Autonomous System 174 follow that sentence format, but for example, the prepend Communities for the Autonomous System 2914 have this format; “Prepend 2x to all EU peers.”. Thanks to the separator the BGP Parser knows that each description of every type of BGP Community have a similar structure/syntax, as seen in the previous examples, both of them uses the same verb, a number that indicates the times that the route must be prepended and a destination. The separator was crucial, even with only the Google Cloud Natural Language API would have been impossible to treat all the possibilities for each possible description.

The BGP Parser separate each BGP Community as independently text and made a call to the Natural Language API for that text. The API run the machine learning for that text and for every word in the text sent back to the Parser a Json struct, which had the following format:



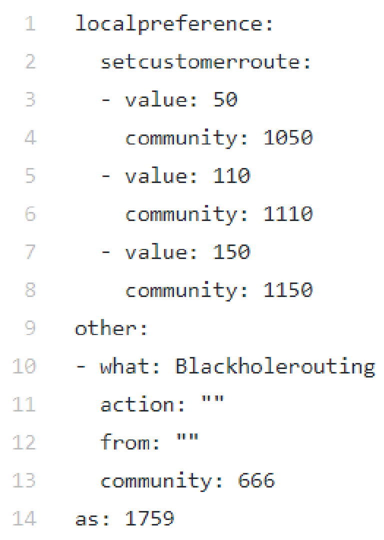
```
const (
    // Unknown
    PartOfSpeech_UNKNOWN PartOfSpeech_Tag = 0
    // Adjective
    PartOfSpeech_ADJ PartOfSpeech_Tag = 1
    // Adposition (preposition and postposition)
    PartOfSpeech_ADP PartOfSpeech_Tag = 2
    // Adverb
    PartOfSpeech_ADV PartOfSpeech_Tag = 3
    // Conjunction
    PartOfSpeech_CONJ PartOfSpeech_Tag = 4
    // Determiner
    PartOfSpeech_DET PartOfSpeech_Tag = 5
    // Noun (common and proper)
    PartOfSpeech_NOUN PartOfSpeech_Tag = 6
    // Cardinal number
    PartOfSpeech_NUM PartOfSpeech_Tag = 7
    // Pronoun
    PartOfSpeech_PRON PartOfSpeech_Tag = 8
    // Particle or other function word
    PartOfSpeech_PRT PartOfSpeech_Tag = 9
    // Punctuation
    PartOfSpeech_PUNCT PartOfSpeech_Tag = 10
    // Verb (all tenses and modes)
    PartOfSpeech_VERB PartOfSpeech_Tag = 11
    // Other: foreign words, typos, abbreviations
    PartOfSpeech_X PartOfSpeech_Tag = 12
    // Affix
    PartOfSpeech_AFFIX PartOfSpeech_Tag = 13
)
```

Figure 7: Natural Language API result used by the BGP Parser

As shown in the table, after sending the BGP Community description each word description had a *PartOfSpeech_Tag,* consequently the BGP Parser knew the grammatical type of each word of the description. Finally, knowing the type of the BGP Community and the information provided by the Natural Language API the BGP Parser was able to generate the corresponding configuration files.

The BGP Parser was made with Golang programming language due to its ease of integration with the Google Cloud Natural Language API. The code and the generated configuration files are open-sourced and can be found in Github [18].

The Natural Language API service that provides Google is not free because it uses its resources to retrieve information, all in all, for all the communities of the Autonomous Systems in the dataset it cost around 2.50$.



```yaml
localpreference:
  setcustomerroute:
  - value: 50
    community: 1050
  - value: 110
    community: 1110
  - value: 150
    community: 1150
other:
- what: Blackholerouting
  action: ""
  from: ""
  community: 666
as: 1759
```

Figure 8: Configuration file with the proposed language generated for the Autonomous System 1759

10.2 Blockchain

10.2.1 Previous concepts

All done in the previous section was because the need to test real data, but the solution to the problem of BGP communities itself remains with the blockchain implementation.

Blockchain is a decentralized system, it does not store any of its information in a central location. Instead, the blockchain is copied and spread across all the network. Every time a new block (information) is added, every peer on the network updates its own copy of the blockchain to reflect the change. By spreading the information across the network, rather than storing it on a central database, blockchain becomes harder to tamper with. It is also immutable, after a transaction (in our case a BGP Community) is validated by the network (Autonomous Systems), it is added into the blockchain in the form of a block. Each block references the previous one via a hash. When the information contained in a block is modified in any way its hash changes and makes the blockchain inconsistent and invalid, thus, ease to find what and who is trying to modify it.

Blockchain is the ideal system to solve the specified problems in previous sections that generate BGP communities. In addition, most of the modern blockchain implementation provides privacy, that means that apart from the first block, the desired blocks can be only understood by some parts of the chain.

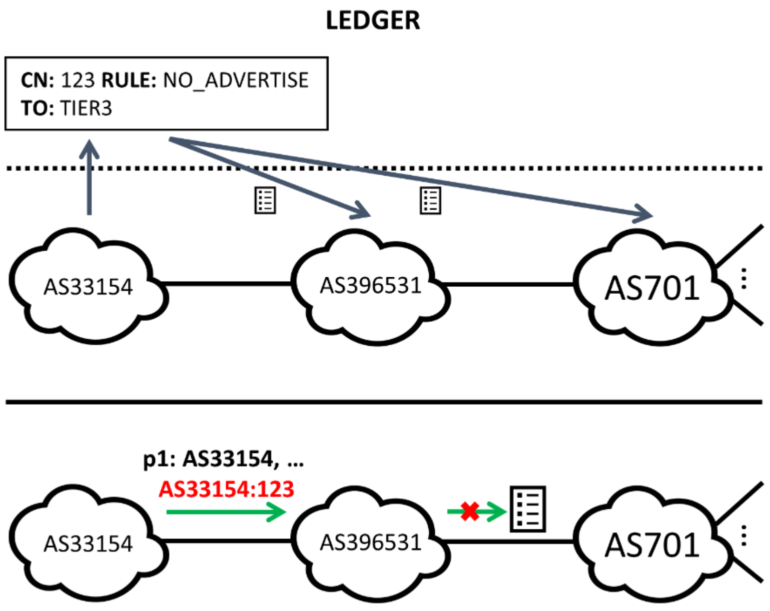


Figure 9: Diagram of the blockchain operation among three different Autonomous Systems

10.2.2 Technical concepts

In order to take advantage of all the features that a blockchain provides we decided to use a developed blockchain platform, the Hyperledger project. Hyperledger [2] is a project of open source blockchains, started in December 2015 by the Linux Foundation, and has received contributions from IBM and many other companies, to support the collaborative development of blockchain-based distributed ledgers.

Hyperledger provides many different frameworks depending on your needs, for example, if you are a bank entity, if Ethereum [19] code chain is needed, if mobile integration is a must, etc. There isn't a relationship between them and the most well-known are: Hyperledger Fabric, Hyperledger Iroha, Hyperledger Sawtooth and Hyperledger Besu.

For this project we decided to use Hyperledger Fabric, a part of that is the one with more support, the main reasons are:

- Permissioned blockchain
- Modules architecture
- Delineation of roles between nodes
- Configurable consensus and membership
- Implementation can be done with Node.js, Java or Golang

Although the assets, the information inside each block, can be fully customized, the structure of the blockchain network can have different roles. A blockchain network is made of nodes, peer or orderer nodes(endorsers), peer nodes make submissions to the blockchain that can only be validated with a consortium algorithm between the orderer nodes. In our experiment, the main idea was to simulate a real BGP scenario. Each Autonomous Systems would have multiple peers able to submit their BGP Communities into the blockchain, and an orderer node that together with the orderers of other ASs and a consortium algorithm was able to validate any submitted BGP Community.

In addition, Hyperledger Fabric provides designed applications that can be customized and deployed as you want. We decided to use the Commercial paper [20] as the base application, and modify it depending on our needs.

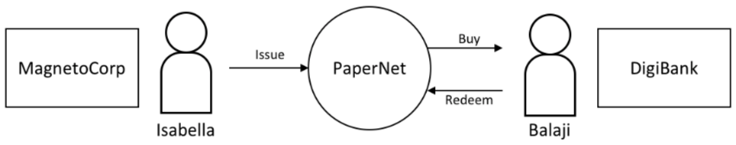


Figure 10: Basic workflow of the default Commercial paper application

We parted form the workflow shown in Figure 10. MagnetoCorp and DigiBank simulate our Autonomous System and the PaperNet represents a transaction, which in our case will be a BGP Community.

First of all, we had to create a new asset by replacing the PaperNet for a BGP Community asset. The resulting BGP Community asset was able to be generated from the configuration files done by the BGP Parser. The BGP Community asset that the Autonomous Systems would exchange have the following parameters:

- Autonomous System number
- BGP Community number
- BGP rule or BGP routing policy
- Destination
- Value (in case of Local Preference or Prepend Community)
- Datetime

Finally, we needed to create the nodes for the Autonomous Systems and the channels which they will use to exchange the BGP Community assets between them. All the nodes of the network use asymmetric public/private key to authenticate in the blockchain. To generate those nodes, Hyperledger uses a configuration file(yaml) for each blockchain node which can be fully customized, in order to set the node name (Autonomous System name), certificates or any possible parameter, that you want.

Hyperledger uses access control lists to manage access to resources by associating a policy, which specifies a rule that evaluates to true or false, given a set of identities. This property is quite interesting since it gives the Autonomous Systems (blockchain peers) the possibility of giving read-only access to some other Autonomous Systems, and not all of them. This feature is implemented with configuration files(yaml) too and are called channels. In the experiment section, we will talk about what channels we configured to test the blockchain.

All the services and nodes in the blockchain run in separate Linux containers, thus if one of them fails the blockchain is still consistent. Hyperledger uses Docker [21] as the container's management.

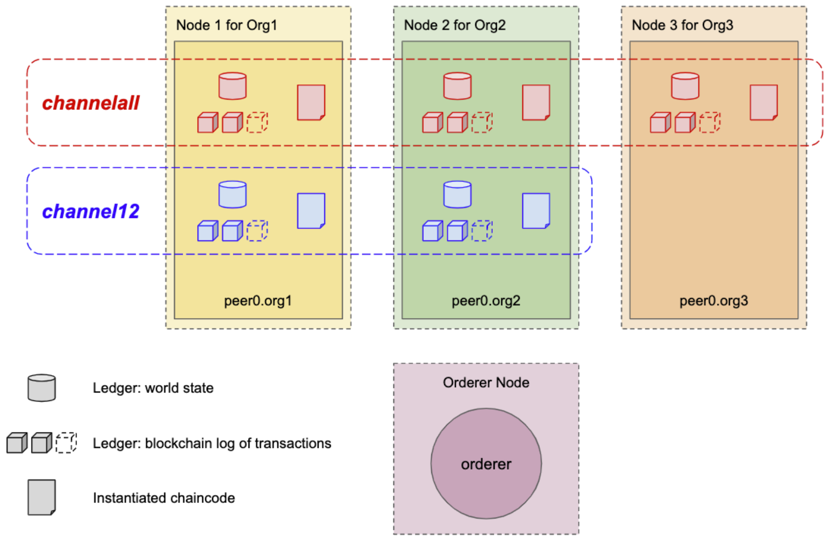


Figure 11: Diagram of a basic Hyperledger network made of 3 organization with one orderer node.

10.3 Compiler

10.3.1 Previous Concepts

All the aspects and features of the BGP Compiler were done by Miquel, consequently not many details can be mentioned in this document. Moreover, to maintain the thread of all the project we though that explaining the basic operation would be valued.

First of all, the blockchain stores in its blocks with BGP communities in the formal language that we proposed ourselves. Specifically, we transform the informal definitions (human sentences) of the OneStep dataset to our formal language using the BGP Parser, in order to be used in the blockchain. Lastly, we need to transform the output of the blockchain, that it is in our formal language, to be used in real BGP routers. The problem is that BGP routers only understand BGP commands, the solution was to transform our formal language into BGP commands. Therefore, we needed to build a compiler.

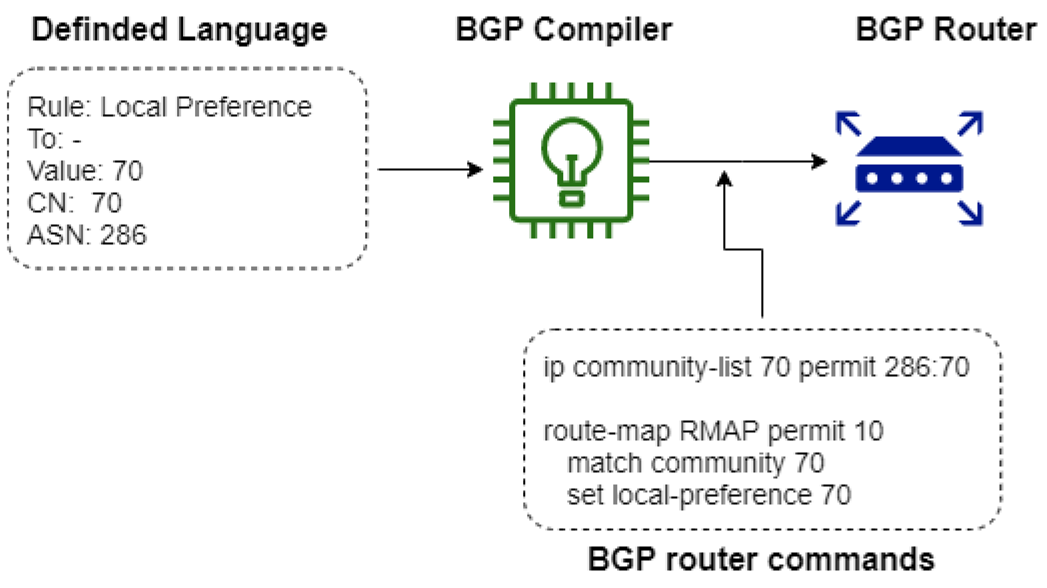
The developed compiler translates our language to ACL(Access Control Lists) BGP filters using Quagga [22]. ACL BGP filters are used to permit or deny traffic from specific IP addresses to specific destination IP address and port. It also allows you to specify different types of traffic such as ICMP, TCP, UDP, etc. Quagga is a routing software suite, providing implementations of OSPFv4, RIP and BGP, between others, for Unix platforms. The Quagga project aims at providing open source versions of routing protocols.

10.3.2 Technical concepts

The compiler was developed with JavaScript programming language, due to ease compatibility with the blockchain, as it is developed with JavaScript too.

This tool must run in every simulated BGP router in a manner that when a BGP community in the blockchain is verified for that router, the compiler automatically transforms the BGP Community of the blockchain to the real BGP command and executes it.

The compiler uses multiple maps to make the transformations. Basically, we know nearly all the possible BGP commands that can be executed and the relationship with our defined language. This relationship has been extrapolated with various maps programming structure. When a BGP community enters to the compiler as a string, the compiler divides the string in different parts; ASN(Autonomous system), CN(Community Number), Rule, Value and To(Destination). The rule part is linked to different maps which make the relationship to the real BGP commands. Finally, it maps the Rule part to the corresponding BGP command, and it appends all the other parts to get the final and applicable command.



```
ip community-list 70 permit 286:70

route-map RMAP permit 10
  match community 70
  set local-preference 70
```

Figure 12: BGP Compiler workflow

11. Experiments

11.1 Localhost experiment

Hyperledger is still a very recent project, as the first long-term-support of Hyperledger Fabric it was not announced till January 2019. There are also some remarkable prerequisites to install and run the development tools for Hyperledger as; Ubuntu 14.04/16.04 (both 64-bit), Docker Engine (Version 17.03 or higher), Docker-Compose (Version 1.8 or higher), among others.

Considering all the previous requirements and that there's isn't a big support behind we decided to make a first experiment in our laptops. Indeed, the first test was the basic Commercial paper application that Hyperledger fabric provides in order to check that everything run as expected and that we met all the requirements. Once the provided application was running, we decided to start changing the applications code by parts, starting for the asset that is exchanged in the blockchain. While performing every change we rerun the blockchain to check that the blockchain was still able to validate transactions in the correct way. Finally, once the blockchain was able to make transactions with the BGP Communities generated by the BGP Parser, we decided to start changing the nodes in order to simulate real Autonomous Systems. A real BGP Confederation contains multiple Autonomous Systems and different links between them. We decided that those links will be simulated in the blockchain with channels. In the application provided by Hyperledger there were only two nodes, in our final experiment we setup ten nodes simulating different Autonomous Systems and different channels of communication.

The problem of running the blockchain in localhost is that the times and latency between transactions are not realistic because they tend nearly to zero. Consequently, a real-word experiment was done.

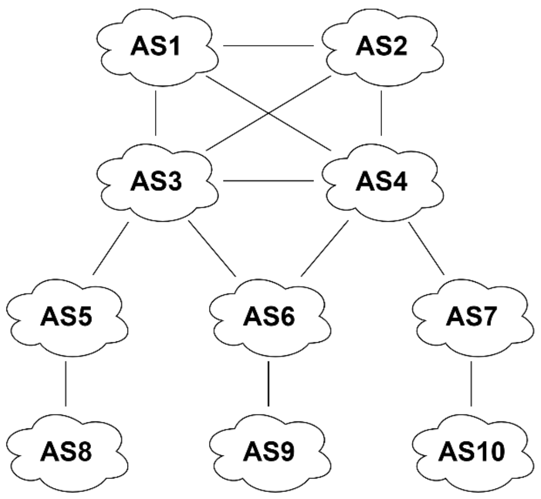


Figure 13: Topology used for the final experiment

11.2 Real-world deployment

The main objective was to test the developed blockchain in multiple nodes in different locations. To accomplish that objective, we needed multiple computers and a dedicated public IP address for each one. The best option was to use the Cloud. Using a Cloud provider give us the ability to fulfil all the requirements for that operation in the quickest and cheapest manner possible. There exist four main Cloud providers in the world: Google Cloud Platform, Amazon Web Services (AWS), Azure (Microsoft) and Alibaba Cloud. We decided to use Amazon Web Services because we were more comfortable working with it, due to previous experience with other projects.

The first step was to design the infrastructure. One of the services of AWS is CloudFormation designer [23], which its purpose is to design AWS architecture using a visual interface. It also generates automatic configuration files (JSON or YAML) in order to make the deployment autonomously.

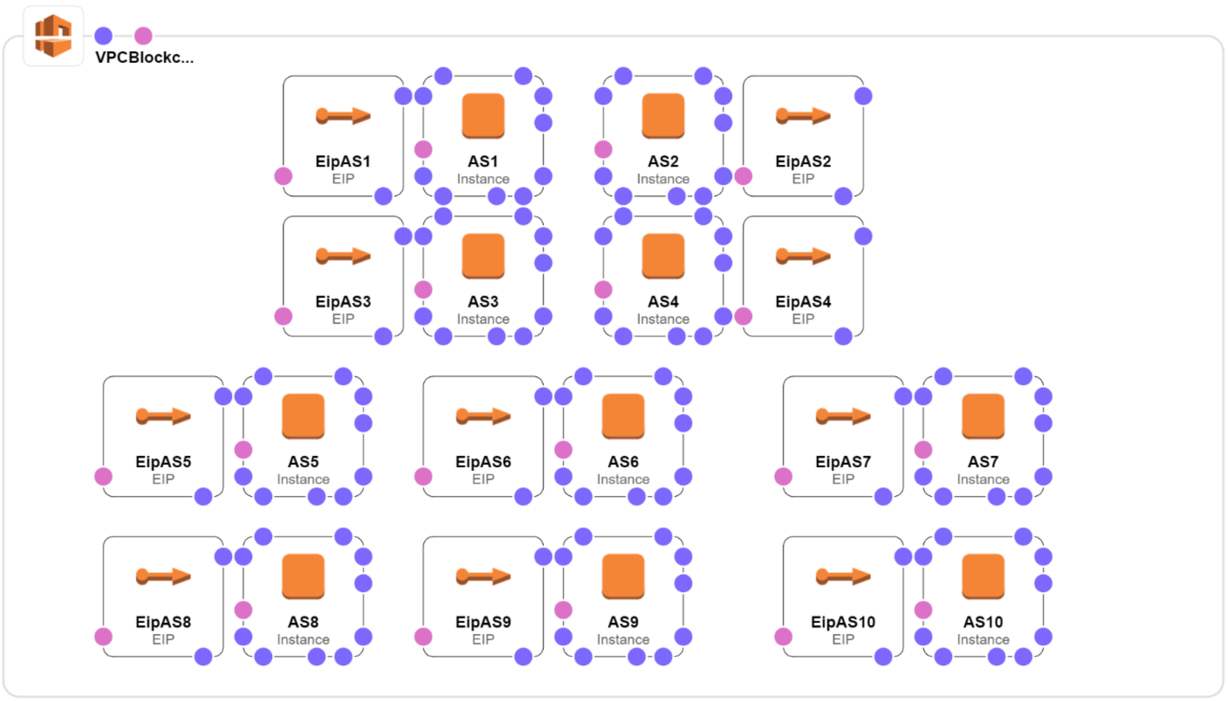


Figure 14: AWS CloudFormation design for the BGP experiment

As seen in the Figure 14, the final experiment in the cloud consist of ten orange squares linked with an orange arrow each of them. Each orange square is an Elastic Compute Cloud(EC2), in other words, a virtual machine in the Cloud that will simulate the node in the blockchain. Each virtual machine has 4GB of RAM, as it is a minimum requirement of Hyperledger. The orange arrows are Elastic IP addresses, each Elastic IP address is a static IPv4 address designed for dynamic cloud computing. That feature was very useful because in case we stopped our instances (during the night) we didn’t lose our public IP and consequently neither the communication between them.

Once the experiment design in AWS was clear we had to reserve all the Elastic IP addresses that we used and update the configuration files of the blockchain. In brief, we changed all the localhost mentions to the corresponding Elastic IP addresses in the blockchain configuration files. Secondly, we thought that installing all the requirements of Hyperledger in all ten virtual machines would be a very repetitive and slow task, because you can only choose the operating system, of software decisions, when launching a new EC2. The most effective solution for that problem was to create an Amazon Machine Image. First, we launched a sample EC2 instance with Ubuntu 16.04 as operating system, succeeding we installed all the requirement to run the blockchain in that instance, among others:

- Golang programming language
- Docker
- Node: 8.9
- Npm
- Git
- Python
- Hyperledger
- SSH key to manage the instance
- Blockchain application files (Gitlab repository)

When all those features were installed, AWS provides a tool to autogenerate Amazon Machine Images. Basically, it freezes the instance and makes a software and usable copy of it. Thanks to this tool we were able to set up all the EC2 instances ready to test the blockchain very quickly.

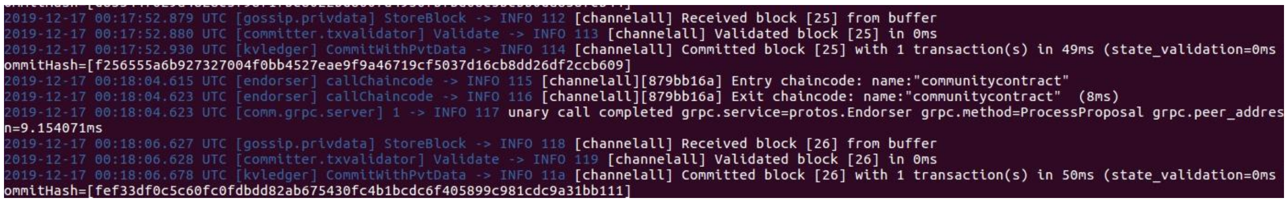
In addition, the communication between the instance had to be configured. For this, we take advantage on the Amazon Security Groups feature, which allow the user to create networking rules between EC2 instances. These rules are set using a very friendly interface and are very similar to ACL (Access Control List) rules.

At this point, all the communication between the instances, blockchain nodes, were correctly working and the blockchain itself was initialized too. All that remains is to extract the results, latency times, from the BGP Communities (generated with the BGP Parser) exchanged between the configured nodes. To build this experiment, for each instance found in the figure 34 a random Autonomous System from the dataset generated by the BGP Parser is selected. Then, the different transactions are built and committed to the blockchain and finally, the elapsed time is calculated for every node.

12. Results and benchmarks

In order to test more than one BGP Community at once and not manually, we generate a JavaScript script that sent to the blockchain multiple BGP Communities automatically. This script run in every blockchain node.

As mentioned in previous sections, all the blockchain, Hyperledger, components run with Docker containers and thus, makes it very easy to keep track and logs of them. By default, the logs of a container in docker can be seen by just running the following Linux command*: docker logs <container_name>*. In those logs many items are recorded, for example, the time to generate a block, the time to receive a block, time to validate a block, among many others.



```
2019-12-17 00:17:52.879 UTC [gossip.privdata] StoreBlock -> INFO 112 [channelall] Received block [25] from buffer
2019-12-17 00:17:52.880 UTC [committer.txvalidator] Validate -> INFO 113 [channelall] Validated block [25] in 0ms
2019-12-17 00:17:52.930 UTC [kvledger] CommitWithPvtData -> INFO 114 [channelall] Committed block [25] with 1 transaction(s) in 49ms (state_validation=0ms
ommitHash=[f256555a6b927327004f0bb4527eae9f9a46719cf5037d16cb8dd26df2ccb609]
2019-12-17 00:18:04.615 UTC [endorser] callChaincode -> INFO 115 [channelall][879bb16a] Entry chaincode: name:"communitycontract"
2019-12-17 00:18:04.623 UTC [endorser] callChaincode -> INFO 116 [channelall][879bb16a] Exit chaincode: name:"communitycontract"  (8ms)
2019-12-17 00:18:04.623 UTC [comm.grpc.server] 1 -> INFO 117 unary call completed grpc.service=protos.Endorser grpc.method=ProcessProposal grpc.peer_addres
n=9.154071ms
2019-12-17 00:18:06.627 UTC [gossip.privdata] StoreBlock -> INFO 118 [channelall] Received block [26] from buffer
2019-12-17 00:18:06.628 UTC [committer.txvalidator] Validate -> INFO 119 [channelall] Validated block [26] in 0ms
2019-12-17 00:18:06.678 UTC [kvledger] CommitWithPvtData -> INFO 11a [channelall] Committed block [26] with 1 transaction(s) in 50ms (state_validation=0ms
ommitHash=[fef33df0c5c60fc0fdbdd82ab675430fc4b1bcdc6f405899c981cdc9a31bb111]
```

Figure 15: Example of Docker logs for an Hyperledger container

Figure 15 shows logs of one of our nodes in the blockchain. The first blue part is a summary of that log; the date of the log, who has registered that log and a title of the log. The white part displays information about that log, in our cases, event in the blockchain. Those logs were very outright and therefore, our primary source to generate all the reports and results of the experiment.

In order to analyze the performance, feasibility, scalability and reliability of the experiment we analyze multiple parameters generated by the logs. First of all, we analyze the chain size with respect of the number of BGP Communities stored in the chain. It has something to do with the fact that perhaps the blockchain stores the information in a different way and occupies more space than though and to test that the assets are not superimposed once stored in the corresponding nodes.

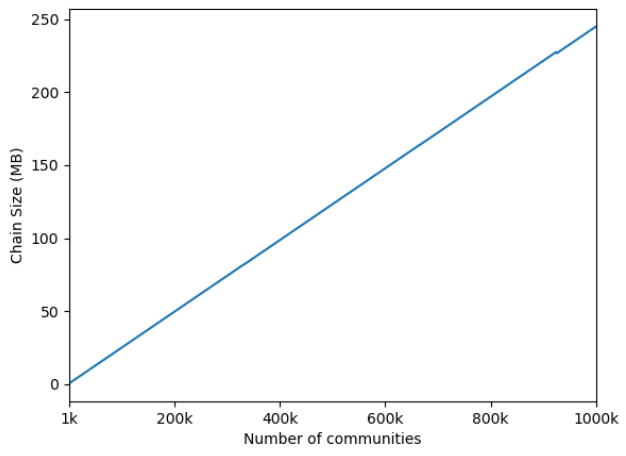


Figure 16: Total chain size by number of communities

As the figure shows the size grows linearly with the amount of communities, thus the size of the data in the blockchain will be always proportional to the total number of communities. This shows that it is feasible to store the routing policy of a large number of participating Autonomous Systems. It is also remarkable that we made that test for 400k communities which is considerably a big number of them, therefore the performance of the system seems reliable enough.

The second blockchain item that we wanted to validate was the latency between the nodes. Indeed, we wanted to analyze the write latency depending on the number of the blockchain nodes. The write latency encompasses the time it takes to write a new policy using the formal language into the distributed ledger. The reason was because a system like that the latency must remain linear due to its big number of participants in the wild. In order to make it more realistic we analyzed with endorsers which is a type of blockchain node that is responsible for the approval of a transaction when in proposed by the other nodes. The results in Figure 17 show that the latency is also linear with the amount of endorser.

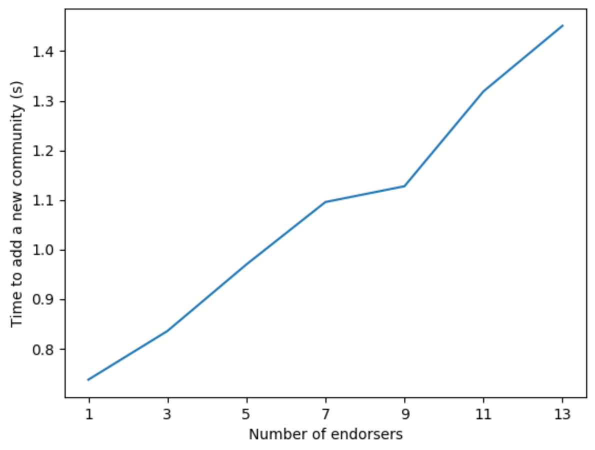


Figure 17: Write latency as a function of the number of endorser nodes

The last feature of the blockchain itself that we wanted to analyze was the time to compile a community, in other words, the time it takes to read the policy from the blockchain and to transform it into a BGP routing filter using the BGP Compiler. Figure 18 shows that the times is also linear with respect the number of communities.

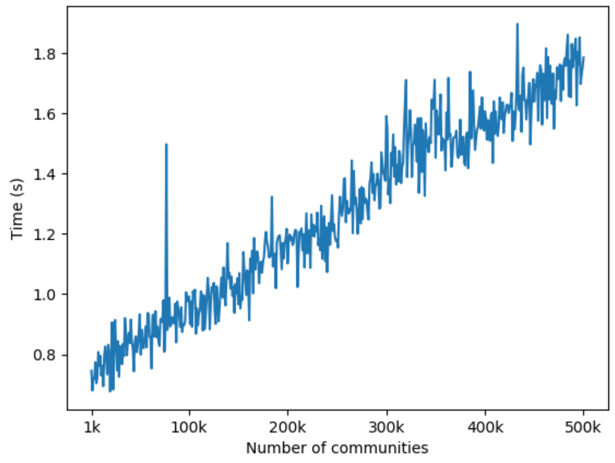


Figure 18: Compiling time by number of communities

In the begging those previous reports were not planned, as they are not related to security and performance of the solution. Unexpectedly, as the results were as desired, making those reports help us a lot and create a comfortable atmosphere to continue with the project.

The last reports that we wanted to perform were for analyzing the performance of our proposed architecture (10 Autonomous Systems) with the real-world data generated with the BGP Parser. In the previous measurements was used always the same input for comfort and quickness, the performance won't change by changing the input because the input is not scanned by the blockchain therefore it does not modify its functionalities. The objective of this last experiment was to experimentally measure the end-to-end latency of the proposed architecture. This process involves the writing, propagation, validation and storage of the policy in the blockchain. Each node of the last experiment represents one Autonomous System.

We calculated three different times for the last experiment. First of all, the contract execution time that tells us the time it takes a transaction to be executed. Secondly, the block commit time which is the time since a transaction is added to a block, till this block is send and committed to the chain.

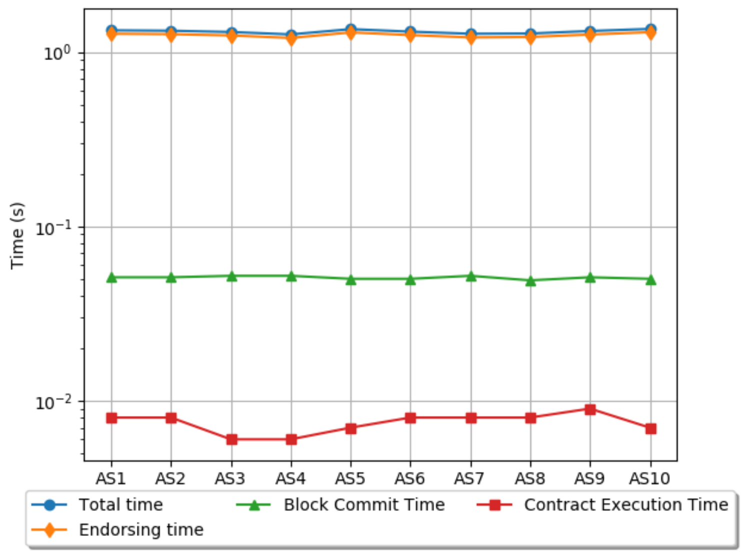


Figure 19: Elapsed time of the real-world scenario

Figure 19 shows the results for all three the calculated times for each of the Autonomous Systems, blockchain nodes. The contract execution time took values of 10-^{2} seconds magnitude; however the block commit time took values around one second. The block commit times takes a bit more since all the information on the chain needs to be updated.

And finally, the total time, that is the sum between the previous two times plus the communication time between two peers. It must be remarked that we remove the batch timeout. The batch timeout is an Hyperledger default time that is added in all the nodes. Basically, is the amount of time that a node waits after the first transaction, waiting for additional transactions to be added to the block before cutting it. Decreasing this value will improve latency but decreasing it too much may decrease throughput by not allowing the block to fill to its maximum capacity.

13. Conclusions

The principal objective of the project was to create a new system where BGP communities can be used safer than are used today. To achieve that goal, we develop a distributed solution, a blockchain, because with it all the BGP communities used in the system where confidential, available and nonrepudiation, among others. Using this approach some of the vulnerabilities mentioned in previous sections are solved, for example, the clearest one is BGP route leaks.

The blockchain shares the distributed ledger among the participants Autonomous Systems and it is used to securely communicate the routing policy and the BGP community. The distributed ledger makes the routing policy from the source Autonomous System available to the participants in a secure way, providing privacy and authentication. Encryption of the blockchain data can also be provided by any participant, in that way an Autonomous System can encrypt a policy in order than can only be decrypted by the desired participants.

We have also seen that machine learning is getting very powerful and able to make unthinkable things, for example, the sentiment analysis of any sentence or as check in this project, the ability to know all the syntaxis characteristics of an English word.

To sum up, the main objective was achieved. Using the implemented distributed solution, BGP communities where able to be distributed securely, preventing for example, the BGP route leak (explained in scope section) that happened in June 2019, that caused and important service disruption on the Internet. The error from that event was because an Autonomous System (AS396531) did not filter correctly a BGP community, and sent it to another Autonomous System (AS701) which was unable to fulfil that BGP community route requirements. With the proposed solution the previous Autonomous System (AS396531) would have checked in the distributed ledger that the BGP community was only for it, and the other AS701 would have never received that BGP community, preventing the global service disruption.

A research paper explaining the parts of this project has been written. The link to it cannot still be provided because it isn’t public yet. The paper has been presented to a networking conference, the IFIP Networking 2020 [24], and the deadline for submitting papers is January 21, 2020. Writing a paper takes more time than thought, all the project parts might be as synthesized as possible, and a lot of iterations for the correctness must be performed.

13.2 Future Scope

The implement solution is just an approach of a possible system that could be developed for the BGP protocol. The possibility of changing a protocol and all the devices that work with it is extremely low, nevertheless, our solution worked as expected and solves many actual problems with that protocol. This distributed ledger would not change the relationship between Autonomous Systems as we parted from the point that the blockchain participants are Autonomous Systems that have a business relationship and thus, have an incentive to participate.

For now, all that can be done is to wait till the networking conference days. During these days, perhaps some attendant is interested with the project and makes a better reputation about it.

14. Acknowledgment

Finally, I gratefully acknowledge the assistance of the project director, Albert Cabellos, for dedicating his time and effort resolving multiple problems that arose during the duration of the whole project and also for his involvement and commitment for the goodness of the final result.

As might be expected, give many thanks to Miquel Ferriol that is studying a Master degree in FIB, that has lead many parts of the project's implementation, for his dedication, compromise and backing.

To conclude, must be remarked the involvement of the Departament de Arquitectura of UPC which give us support in terms of theoretical questions and economical resources, as a paid Amazon Web Services account.

15. Bibliography and references

[1] Florian Streibelt, Franziska Lichtblau, Robert Beverly, Anja Feldmann, Cristel Pelsser, Georgios Smaragdakis and Randy Bush. "BGP Communities: Even more Worms in the Routing Can". November, 2018. https://people.mpi-inf.mpg.de/~fstreibelt/preprint/communities-imc2018.pdf

[2] Hyperledger Project, The Linux Foundation Projects. https://www.hyperledger.org/

[3] Dictionary, https://www.merriam-webster.com/dictionary/information

[4] A History Of the Internet And Its Invention, November 2019. https://allthatsinteresting.com/internet-history

[5] Protocol definition, https://techterms.com/definition/protocol

[6] Internet protocol suite, https://en.wikipedia.org/wiki/Internet_protocol_suite

[7] BGP communities are an optional transitive BGP attribute, http://www.ciscopress.com/articles/article.asp?p=2756480&seqNum=12

[8] Well-known BGP communities, https://www.noction.com/blog/understanding-bgp-communities

[9] Liu, Xiaowei & Yan, Zhiwei & Geng, Guanggang & Lee, Xiaodong & Tseng, S.s & Ku, C.-H. (2016). RPKI Deployment: Risks and Alternative Solutions. 387. 299-310. 10.1007/978-3-319-23204-1_30.

[10] li, qi & Liu, Jiajia & Hu, Yih-Chun & Xu, Mingwei & Wu, Jianping. (2018). BGP with BGPsec: Attacks and Countermeasures. IEEE Network. PP. 1-7. 10.1109/MNET.2018.1800171.

[11] BGP Leak Highlights the Fragility of the Internet with

Real Consequences, https://blog.catchpoint.com/2019/06/26/%20bgp-leak-internet-fragility/

[12] Antlr, parsing tool https://www.antlr.org/

[13] Google Cloud Natural Language API https://cloud.google.com/natural-language/

[14] Mirko Schmiedl, October 2, 2019. Research Report: Is Proof of Stake better than Proof of Work? https://www.stakingrewards.com/journal/research/Proof-of-Work-vs-Proof-of-Stake-Research-report/

[15] Bitcoin Energy Consumption Index, Digiconomist. https://digiconomist.net/BITCOIN-ENERGY-CONSUMPTION

[16] BGP communities' dataset, OneStep. https://onestep.net/communities/

[17] AS174 OneStep dataset. https://onestep.net/communities/as174/

[18] Parsing OneStep dataset tool, Roger Coll. https://github.com/rogercoll/BGPcommunities

[19] What is Ethereum? https://blockgeeks.com/guides/ethereum/

[20] Commercial paper, Hyperledger application. https://hyperledger-fabric.readthedocs.io/en/release-1.4/tutorial/commercial_paper.html

[21] What is Docker? https://opensource.com/resources/what-docker

[22] Quagga Routing Suite https://www.quagga.net/

[23] Why Use AWS CloudFormation Designer? https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/working-with-templates-cfn-designer-why.html

[24] IFIP Networking June 22-26, 2020, Paris, France. https://networking.ifip.org/2020/