Introduction

The assignment requires building a small tool to monitor and analyse which peers contribute, and how much, to the download of a content form the IPFS network. The goals of the assignment are to download large files from the IPFS, to query the bitswap agent to get the CIDs of the partners in order to know how much they contribute. Optionally it was required to detect the location of these peers around the globe, this task has been implemented using an external geolocation API called `ipapi.co`.

The tool is written in Python and it exploits HTTP calls to interact with the API provided by the agent. It consists of 2 modules and a main. The modules implement utilities functions used for plotting and requesting services to the API.

The plots are made with the `pygal` module that is used to output them in files with `svg` (interactive plots, they can be opened within a browser) and `png` format. Additionally some tables created with `matplotlib` are used to show other information about the peers like the locations, agent versions, protocol versions and latencies. During the execution, the tool also logs in the `stdout` and in a file some other information related to the analysed files (like the size, average speed, duplicate data ecc.) and eventual errors.

The files that were used for the analyses are xkcd (**QmdmQXB2mzChmMeKY47C43LxUdg1NDJ5MWcKMKxDu7RgQm**), Old Internet Files (**QmbsZEvJE8EU51HCUHQg2aem9JNFmFHdva3tGVYutdCXHp**), Presidential Daily Briefs (**Qme6epvZDj3vzHcFKdF1nZhbixjw8Bn4imGcKnbUyBJL89**) and IETF RFC Archive (**QmNvTjdqEPjZVWCvRWsFJA1vK7TTw1g9JP6we1WBJTRADM**). All these files are available in the IPFS dataset.

Usage

To install the dependencies needed for the tool, just run (preferably in a virtual environment) the command:

```
pip install -r requirements.txt
```

It is necessary that the IPFS agent is installed and the repository is initialized (i.e. `ipfs init`). It is not required to have the daemon up and running as the tool will

manage to start and shutdown it during the analyses. The tool is fully tested in the Linux environment and, to have a better download experience, it is recommended to run the command (as `root`) `sysctl -w net.core.rmem_max=2500000` in order to increase the buffer size that in the default settings is quite small (as explained here https://github.com/lucas-clemente/quic-go/wiki/UDP-Receive-Buffer-Size).

The tool accepts some other options that can be useful to get more information:

- `-s` (`--swarmnodes`): option not required, it can be used to plot some information on a small part of the swarm (only 70 CIDs analysed).
- `-b` (`--bootnodes`): option not required, it can be used to plot some information on the bootstrap nodes that are used on startup.
- `-c` (`--cids`): option not required, it allows to specify a list of CIDs (space separated) that will be used in the analyses. If not specified, only the CID of Old Internet Files and xkcd are used.
- `-o` (`--output`): option required, path of where the download files will be stored.

Example of usage (in midterm directory):

```
python ipfs_analyzer.py -s -b -o /home/apuccia/downloads
```

Files analyzed

For each file analysed, three charts are generated:

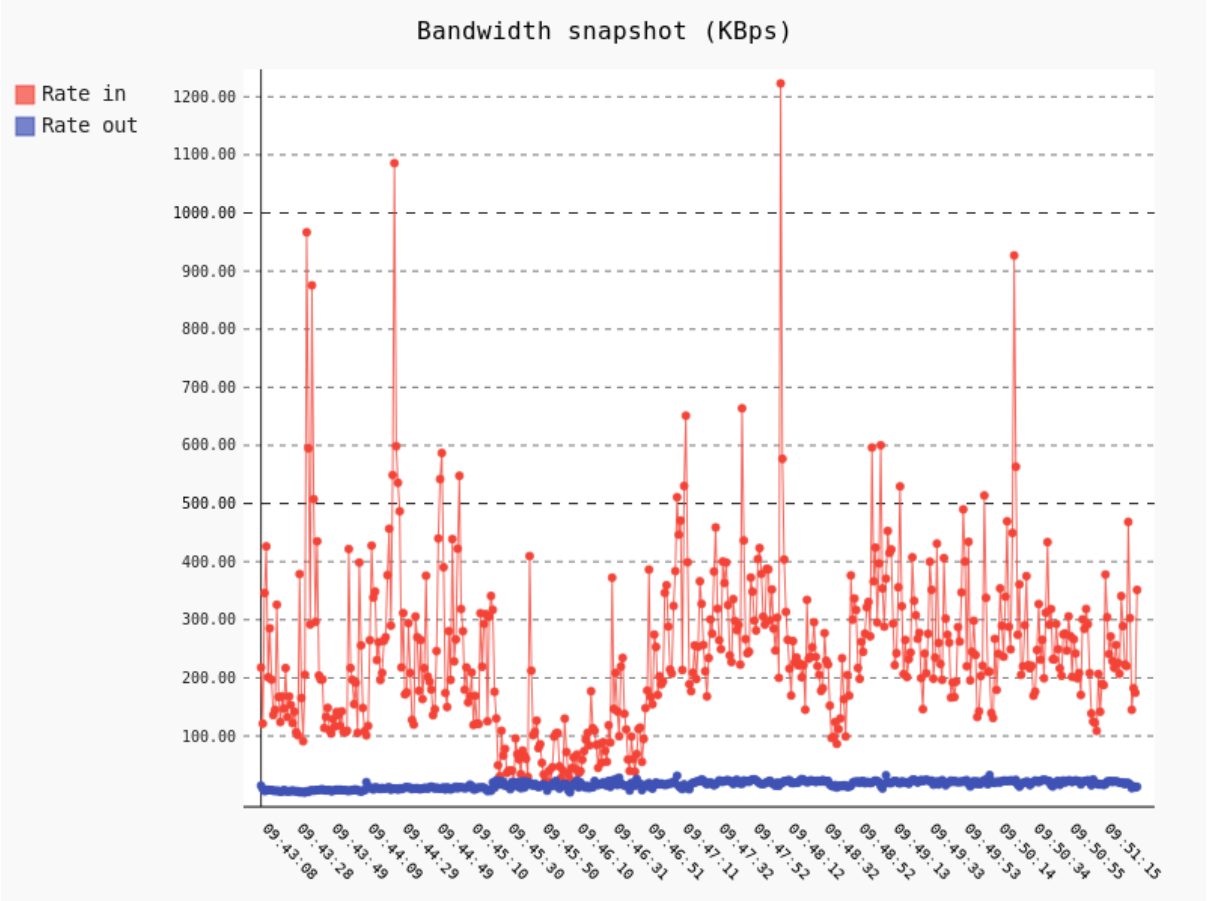
- A pie chart showing the KB that are received/sent from/to the peers.
- A line chart that captures for a timespan (500 seconds maximum) the bandwidth rate in input and output. Every second a query to retrieve the bandwidth is issued. This query specifies the protocol to monitor as `/ipfs/bitswap/1.2.0` (i.e. the last version of the protocol).
- A bar chart that shows the blocks that are exchanged with the peers.
- A “world chart” that shows for each country a colour intensity to indicate the locations of the peers that contribute.

In addition 2 tables are created:

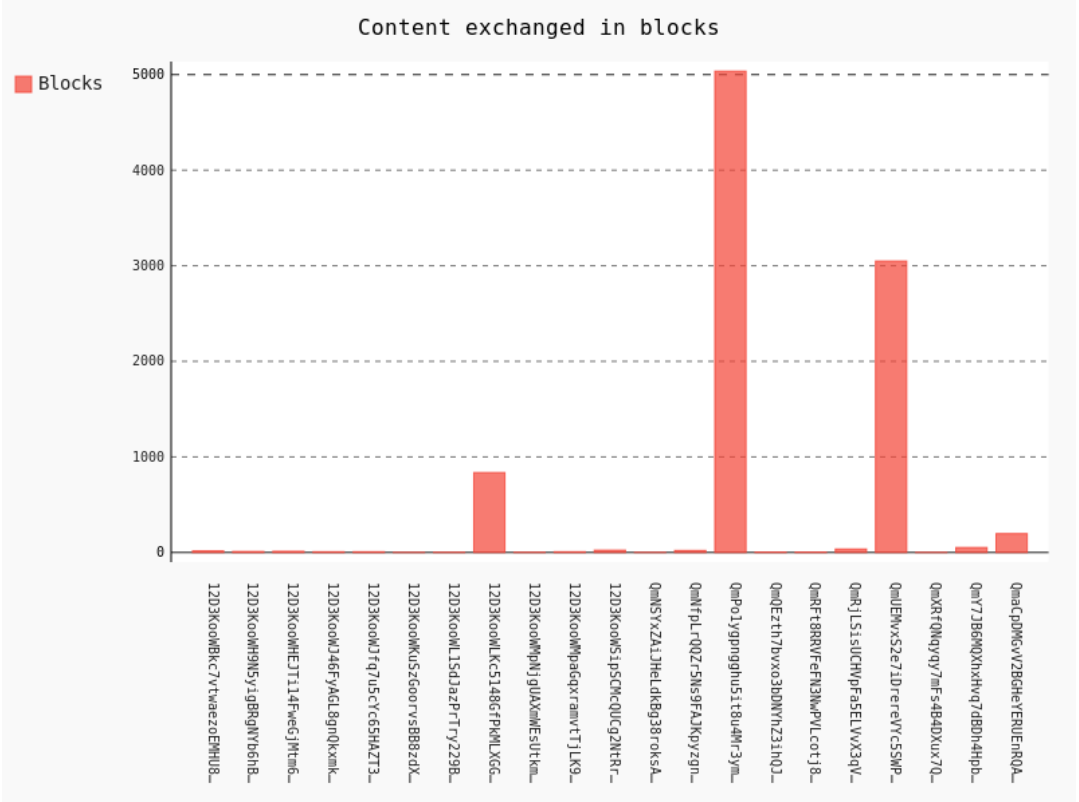
- A table that shows the same information of the pie and bar chart plus information about latency, agent and protocol version.
- A table that shows for each peer CID their IPs and geolocation information such as country, region and city.

xkcd

The xkcd comics have a total size of 106.9 MB and the download lasted about 14 minutes. As we can see from the chart, the download was highly irregular, with a period of really low input rate, overall the average download speed was 242.43 KBps.



Considering the exchanged blocks the first peer from which we exchanged the most, sent almost the double blocks compared to the second one and 5 times more than the third one.

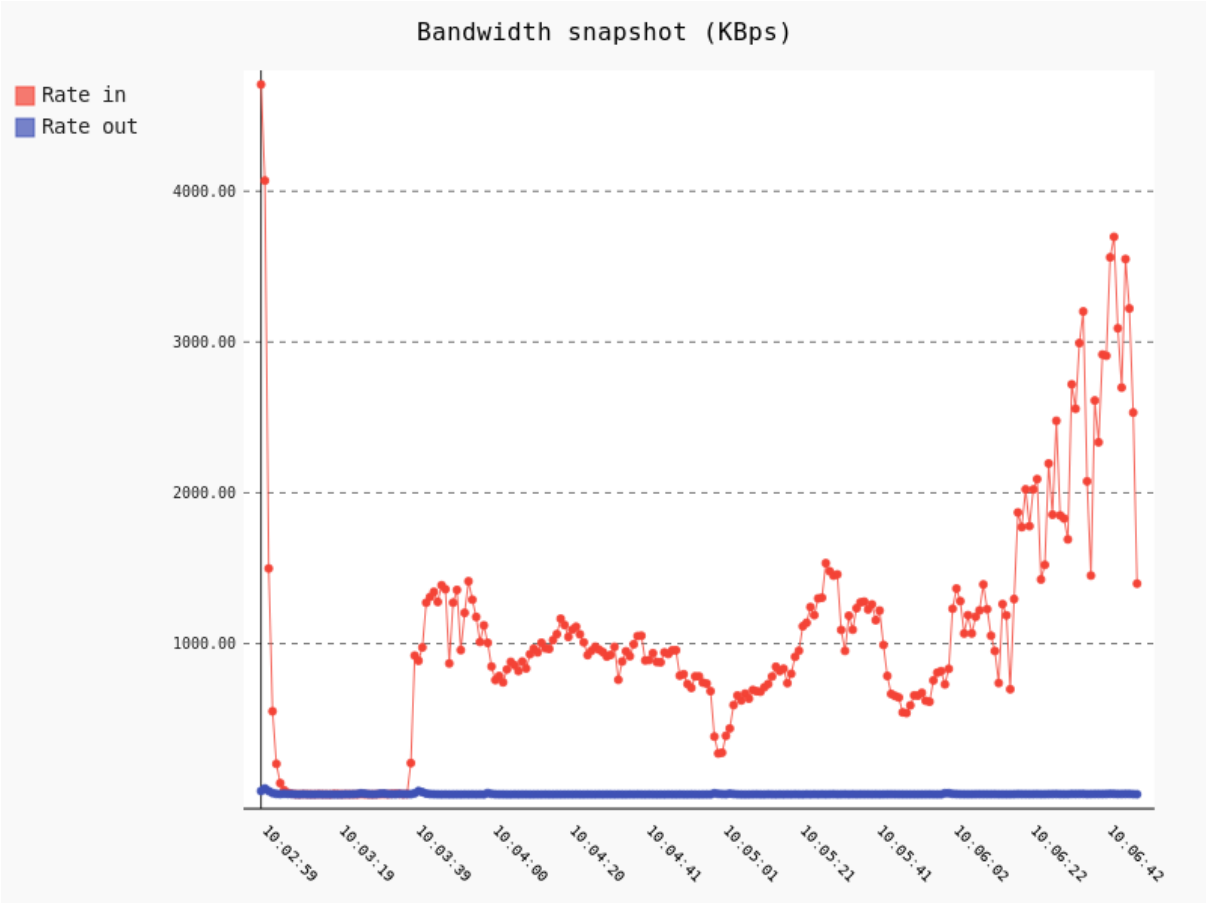


In this table it is possible to see that the peers that interacted with us were mainly from the United States and Germany, but the peers that exchanged the most blocks are from Germany.

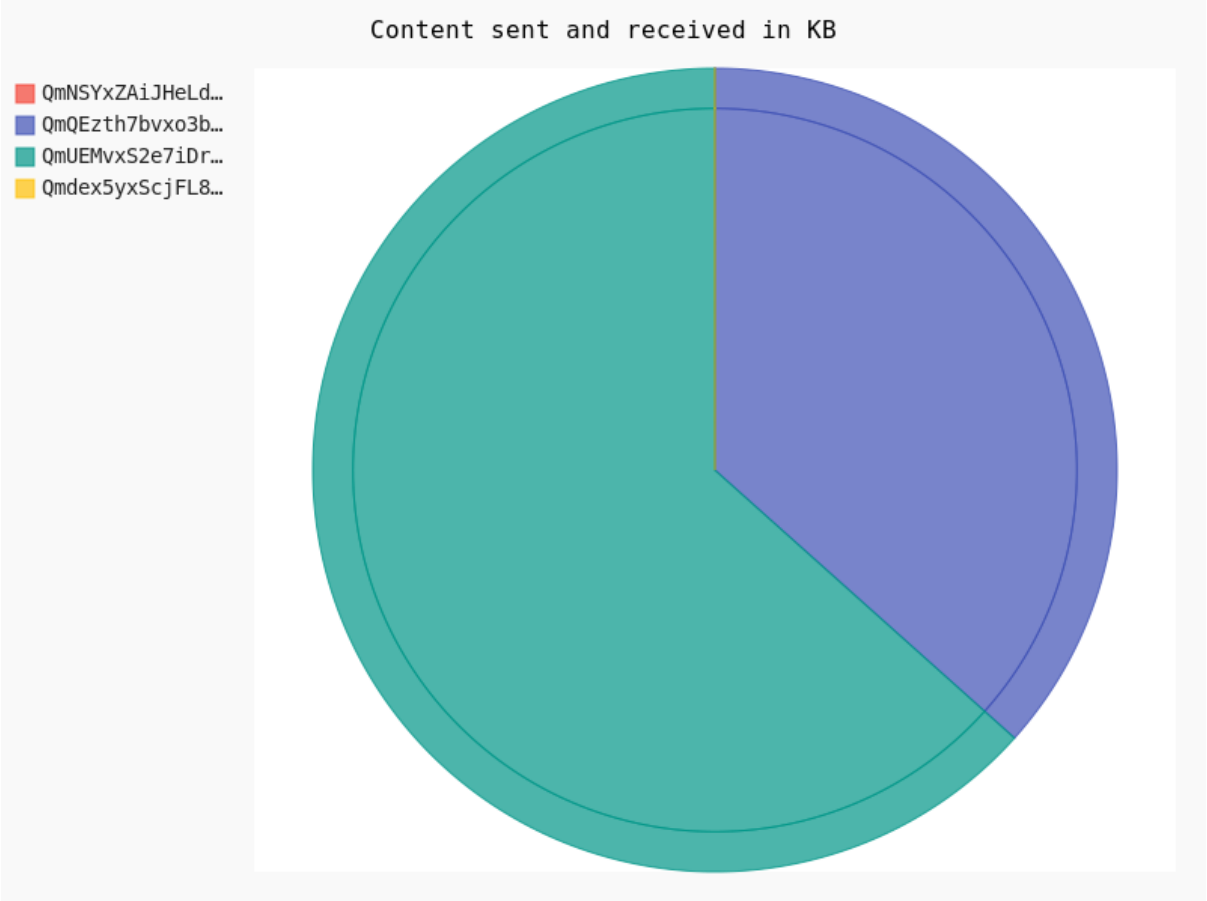
	IP	Country	Region	City
12D3KooWBkc7vtwaezoEMHU8MgMxJSFRQxJFYhWKpZV44keMkiMW	71.114.104.198	United States	Maryland	Ft. Washington
12D3KooWH9N5yigBRgNYb6hBokhgP3URs7a4QVR55sARQ2SqwULy	54.221.18.242	United States	Virginia	Ashburn
12D3KooWHEJTi14FweGjMtm6G7tTTJ7JUTTe3QQbwETTF4KaVq6K	96.77.146.145	United States	Texas	Missouri City
12D3KooWJ46FyAGL8gnQkxmkZH4UTibC8kaSRpHFRM9oEUwKMZLB	34.224.26.109	United States	Virginia	Ashburn
12D3KooWJfq7u5cYc65HAZT3mE3irDH5KYSUdFxXTdf2KTB7FQgf	136.144.57.15	United States	Virginia	Ashburn
12D3KooWKuSzGoorvsBB8zdXCYVMEfybYKa7sDgXT3AkofAzX3FV	47.144.0.144	United States	California	Camarillo
12D3KooWL1SdJazPrTry229BUjE39mBHT9fo9J9KTCKD1iFyyhfe	54.92.128.128	United States	Virginia	Ashburn
12D3KooWLKc5148GfPkMLXGGBCXYsrwrhPuFu6zNxQ8eRntPY9iF	207.180.196.26	Germany	Bavaria	Nuremberg
12D3KooWMpNjgUAXmWEsUtkmDTJLtRTksTad74hqXMsTMwMEHqxB	3.91.3.156	United States	Virginia	Ashburn
12D3KooWMpaGqxramvtTjLK9jLdr1MqWKkwEYbcmEzzeUfuHok4C	3.92.152.122	United States	Virginia	Ashburn
12D3KooWSipSCMcQUCg2NtRrs3fAm55WzwxHdGJmXTEKnX6NHXHS	8.9.30.45	United States	Illinois	Chicago
QmNSYxZAiJHeLdkBg38roksAR9So7Y5eojks1yjEcUtZ7i	138.201.67.220	Germany	Bavaria	Gunzenhausen
QmNfpLrQQZr5Ns9FAJKpyzgnDL2GgC6xBug1yUZozKFgu4	167.71.55.120	Germany	Hesse	Frankfurt am Main
QmPo1ygpngghu5it8u4Mr3ym6SEU2Wp2wA66Z91Y1S1g29	64.225.105.42	Germany	Hesse	Frankfurt am Main
QmQEzth7bvxo3bDNYhZ3ihQJ8tCpLvnFPjsWHFK2TCXugN	167.99.132.247	Germany	Hesse	Frankfurt am Main
QmRFt8RRVFeFN3NwPVLcotj8pffsgk73e9Y6GheWkmJTDZ	52.255.200.232	United States	Virginia	Washington
QmRjLSisUCHVpFa5ELVvX3qVPfdxajxWJEHs9kN3EcxAW6	64.227.27.182	United States	New Jersey	North Bergen
QmUEMvxS2e7iDrereVYc5SWPauXPyNwxcy9BXZrC1QTcHE	94.130.135.167	Germany	Saxony	Falkenstein
QmXRfQNqyqy7mFs4B4DXux7QhFBHRtbrRYSyYFAxP2VpaV	148.251.9.69	Germany	Saxony	Falkenstein
QmY7JB6MQXhxHvq7dBDh4HpbH29v4yE9JRadAVpndvzySN	147.75.87.119	Netherlands	North Holland	Amsterdam
QmaCpDMGvV2BGHeYERUEnRQAwe3N8SzbUtfsmvsqQLuvuJ	104.131.131.82	United States	California	San Francisco

Old Internet Files

The Old Internet Files have a total size of 207.67 MB and the download lasted about 4 minutes. Differently from the xkcd comics, the average speed in this period was about 1 MBps, this is reflected in the chart that appears more regular and with only one period in which the download completely stopped.



Four peers actively participated in the sharing but only two, both from Germany, sent the most content. The latencies computed with the ping messages to these German peers is low compared to the fourth peer.

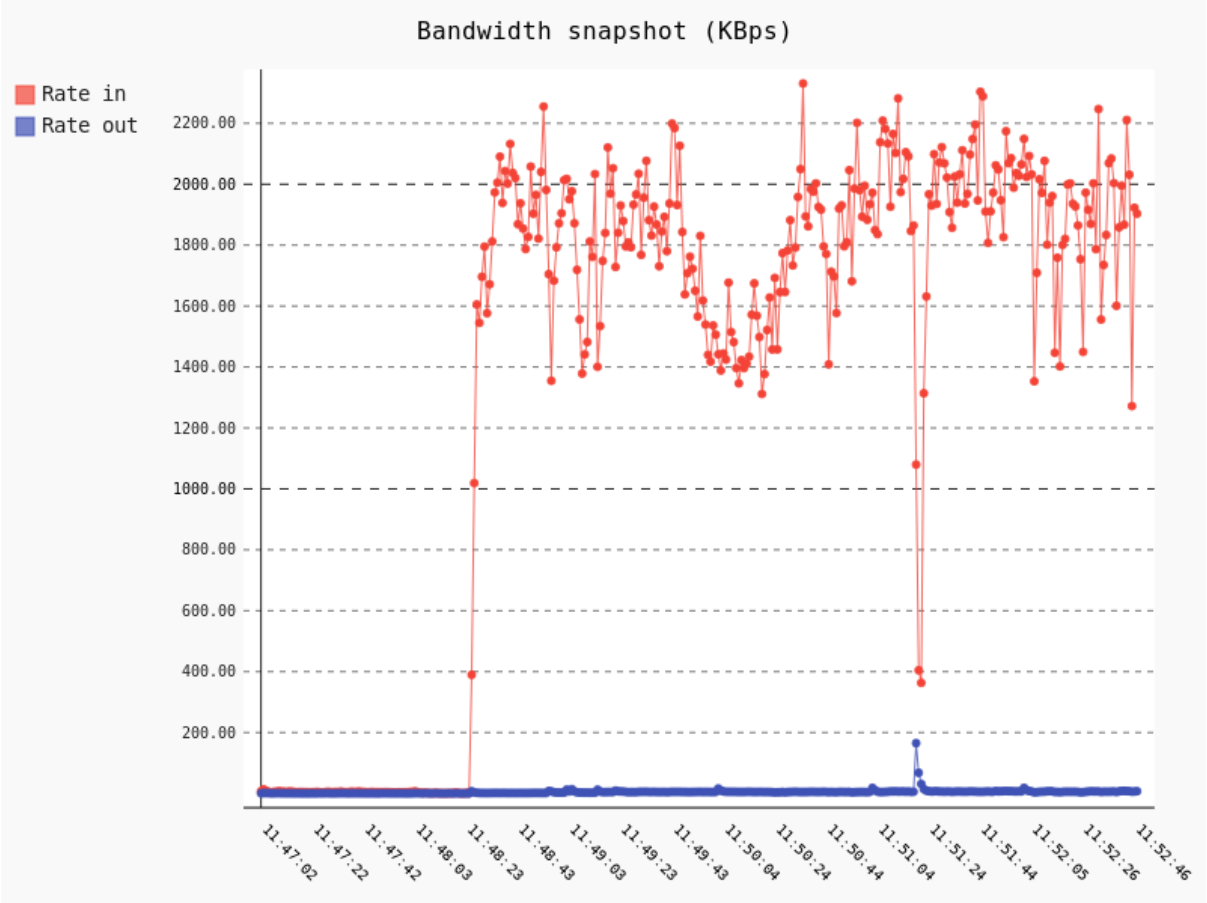


	Agent version	Protocol version	Latency	Received	Sent	Blocks exchanged
QmNSYxZAiJHeLdkBg38roksAR9So7Y5eojks1yjEcUtZ7i	go-ipfs/0.8.0/	ipfs/0.1.0	57.21ms	1.98 KB	0 B	14
QmQEzth7bvxo3bDNYhZ3ihQJ8tCpLvnFPjsWHFK2TCXugN	go-ipfs/0.8.0-rc2/	ipfs/0.1.0	48.98ms	76.05 MB	0 B	376
QmUEMvxS2e7iDrereVYc5SWPauXPyNwxcy9BXZrC1QTcHE	go-ipfs/0.8.0/	ipfs/0.1.0	53.41ms	131.62 MB	0 B	581
Qmdex5yxScjFL8QVaLXWCjYntvy1gfFXYZmAYBANZeasiz	go-ipfs/0.9.0-dev/fb4542b	ipfs/0.1.0	194.70ms	8.01 KB	0 B	2

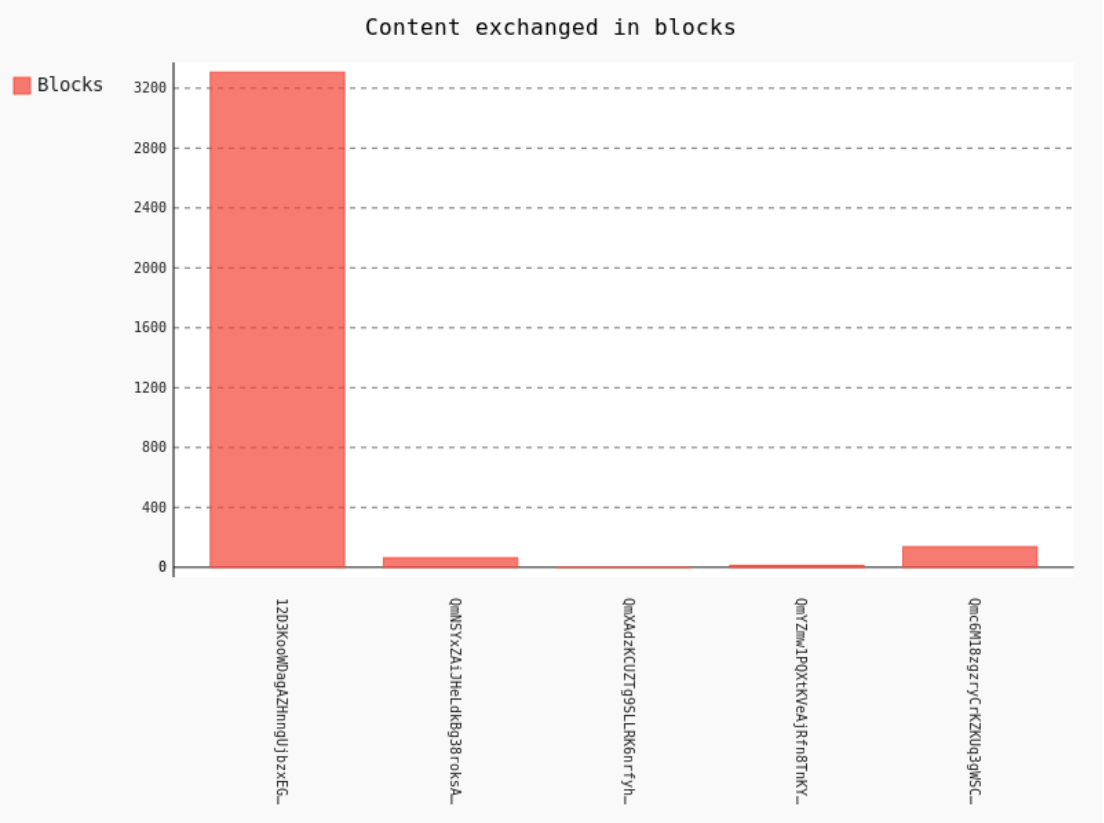
	IP	Country	Region	City
QmNSYxZAiJHeLdkBg38roksAR9So7Y5eojks1yjEcUtZ7i	138.201.67.220	Germany	Bavaria	Gunzenhausen
QmQEzth7bvxo3bDNYhZ3ihQJ8tCpLvnFPjsWHFK2TCXugN	167.99.132.247	Germany	Hesse	Frankfurt am Main
QmUEMvxS2e7iDrereVYc5SWPauXPyNwxcy9BXZrC1QTcHE	94.130.135.167	Germany	Saxony	Falkenstein
Qmdex5yxScjFL8QVaLXWCjYntvy1gfFXYZmAYBANZeasiz	139.178.89.157	United States	California	Sunnyvale

Presidential Daily Briefs

The Presidential Daily Briefs have a total size of 458.54 MB and the download lasted about 6 minutes. After an initial phase in which the download took a long time to start, the bandwidth in input skyrocketed and, even considering this deadlock period, the average rate in input was 1387 KBps.



In practice just one peer from Germany shared the content. The last peer CID specified in the table didn't have an IPv4 address so it was not tracked.

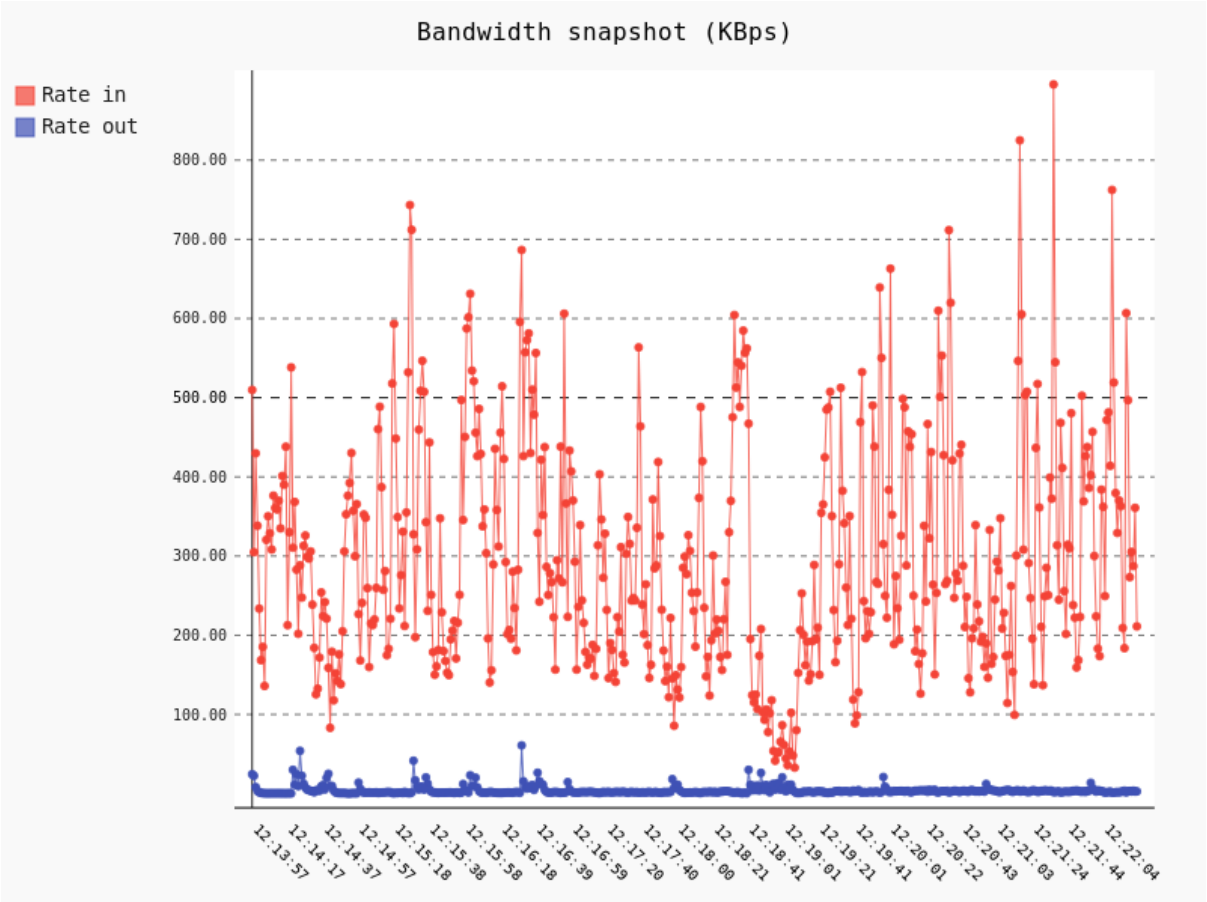


	Agent version	Protocol version	Latency	Received	Sent	Blocks exchanged
12D3KooWDagAZHnngUjbzxEGKR8k5mfaYzaWEke8As22ndrAsBYx	go-ipfs/0.7.0/ea77213	ipfs/0.1.0	50.58ms	449.45 MB	0 B	3307
QmNSYxZAiJHeLdkBg38roksAR9So7Y5eojks1yjEcUtZ7i	go-ipfs/0.8.0/	ipfs/0.1.0	56.92ms	166.05 KB	0 B	63
QmXAdzKCUZTg9SLLRK6nrfyhJmK1S43n2y5EBeGQ7zqhfW	go-ipfs/0.8.0/48f94e2	ipfs/0.1.0	77.15ms	102.0 B	0 B	1
QmYZmw1PQXtKVeAjRfn8TnKYAurSdezRd1kdDjenzxTF6y	go-ipfs/0.8.0/48f94e2	ipfs/0.1.0	90.86ms	1.31 KB	0 B	13
Qmc6M18zgzryCrKZKUq3gWSCSbc5UhMve1YbCUGYock7DL	go-ipfs/0.7.0/ea77213e3	ipfs/0.1.0	55.48ms	8.21 MB	0 B	137

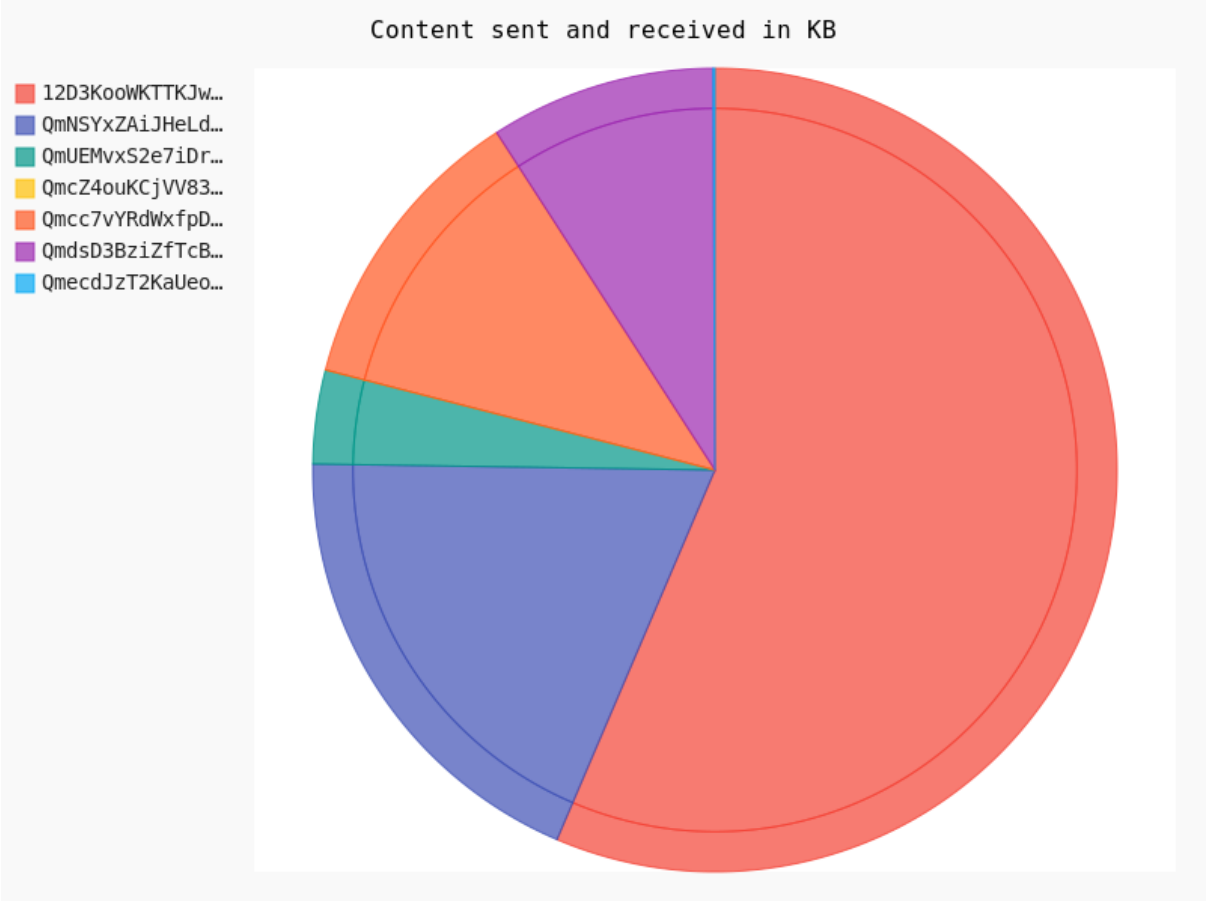
	IP	Country	Region	City
12D3KooWDagAZHnngUjbzxEGKR8k5mfaYzaWEke8As22ndrAsBYx	45.132.245.6	Germany	Baden-Württemberg	Karlsruhe
QmNSYxZAiJHeLdkBg38roksAR9So7Y5eojks1yjEcUtZ7i	138.201.67.220	Germany	Bavaria	Gunzenhausen
QmXAdzKCUZTg9SLLRK6nrfyhJmK1S43n2y5EBeGQ7zqhfW	147.75.100.149	Netherlands	North Holland	Schiphol
QmYZmw1PQXtKVeAjRfn8TnKYAurSdezRd1kdDjenzxTF6y	147.75.45.185	France	Provence-Alpes-Côte d'Azur	Marseille

IETF RFC Archive

The IETF RFC Archive has a total size of 507.49 MB and the download lasted about 1 hour. Like the xkcd comics, this download was problematic and highly irregular, with a lot of periods in which it completely stopped. The average download speed in the period considered in this chart was about 298 KBps.



The resulting pie chart is more interesting than the previous one because more peers shared with us the content. The peer that contributed the most is located in the United States. Other peers from France, Germany and Japan contributed as well.



	Agent version	Protocol version	Latency	Received	Sent	Blocks exchanged
12D3KooWKTTKJwNB5B1fYwGTp6AHYpZDw1t1ajvBHUjiFFRLQWwr	go-ipfs/0.8.0/	ipfs/0.1.0	158.34ms	289.07 MB	0 B	5898
QmNSYxZAiJHeLdkBg38roksAR9So7Y5eojks1yjEcUtZ7i	go-ipfs/0.8.0/	ipfs/0.1.0	62.90ms	96.4 MB	0 B	1578
QmUEMvxS2e7iDrereVYc5SWPauXPyNwxcy9BXZrC1QTcHE	go-ipfs/0.8.0/	ipfs/0.1.0	73.78ms	19.31 MB	0 B	329
QmcZ4ouKCjVV83dz6VD1yTWDM7t5NZBQarEvVM21CMvoH3	go-ipfs/0.7.0/	ipfs/0.1.0	96.79ms	5.09 KB	0 B	12
Qmcc7vYRdWxfpDHSRLW27kMsuPsBUthdjxk4qyfFup8YMR	go-ipfs/0.6.0/	ipfs/0.1.0	57.72ms	60.75 MB	0 B	981
QmdsD3BziZfTcBeqgAUjoMbPHtBCUFQo534RpMALzgEkMR	go-ipfs/0.8.0/	ipfs/0.1.0	224.32ms	46.41 MB	0 B	458
QmecdJzT2KaUeoDBFv6r7nua7A8fnEQeBqUmdsJtxCBESg	go-ipfs/0.8.0-dev/c675d324a	ipfs/0.1.0	81.13ms	391.62 KB	0 B	11

	IP	Country	Region	City
12D3KooWKTTKJwNB5B1fYwGTp6AHYpZDw1t1ajvBHUjiFFRLQWwr	73.165.85.199	United States	Pennsylvania	Norristown
QmNSYxZAiJHeLdkBg38roksAR9So7Y5eojks1yjEcUtZ7i	138.201.67.220	Germany	Bavaria	Gunzenhausen
QmUEMvxS2e7iDrereVYc5SWPauXPyNwxcy9BXZrC1QTcHE	94.130.135.167	Germany	Saxony	Falkenstein
QmcZ4ouKCjVV83dz6VD1yTWDM7t5NZBQarEvVM21CMvoH3	87.122.224.252			
Qmcc7vYRdWxfpDHSRLW27kMsuPsBUthdjxk4qyfFup8YMR	62.138.18.240	France	Grand Est	Strasbourg
QmdsD3BziZfTcBeqgAUjoMbPHtBCUFQo534RpMALzgEkMR	36.14.5.168	Japan	Tokyo	Nerima
QmecdJzT2KaUeoDBFv6r7nua7A8fnEQeBqUmdsJtxCBESg	84.249.26.98	Finland	Pirkanmaa	Tampere

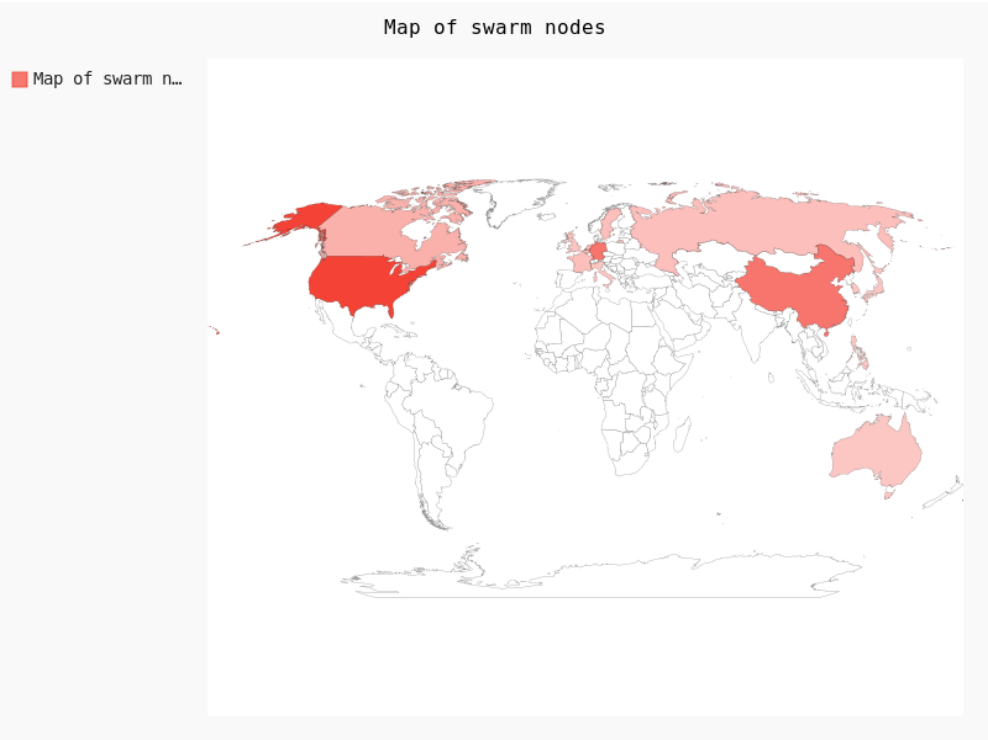
Bootstrap nodes

The bootstrap nodes taken in consideration are the standard ones maintained by the developers of IPFS. No further bootstrap nodes were added into the bootstrap list during all these experiments.

	IP	Country	Region	City
QmQCU2EcMqAqQPR2i9bChDtGNJchTbq5TbXJJ16u19uLTa	147.75.77.187	United States	New Jersey	Parsippany
QmbLHAnMoJPWSCR5Zhtx6BHJX9KiKNN6tpvbUcqanj75Nb	147.75.83.83	Netherlands	North Holland	Schiphol
QmNnooDu7bfjPFoTZYxMNLWUQJyrVwtbZg5gBMjTezGAJN	147.75.109.213	United States	California	Sunnyvale
QmaCpDMGvV2BGHeYERUEnRQAwe3N8SzbUtfsmvsqQLuvuJ	104.131.131.82	United States	California	San Francisco
QmcZf59bWwK5XFi76CZX8cbJ4BhTzzA3gU1ZjYZcYW3dwt	147.75.94.115	Japan	Tokyo	Tokyo

Swarm nodes

For this part only a subset of swarm CIDs were taken in consideration in order to not incur in requests error from the external geolocation API, from this subset the peers are mainly from the United States, China and Germany. Some other peers are scattered in Canada, Sweden, UK, Russia and Australia. The only IP that was found in Italy was mine. One interesting fact that resulted from the swarm analysis is the possibility to run the IPFS node in different machines (with different IPs) but with the same peer CID



12D3KooWJe54VShngB16UngYvcmTB4VfnMPzKoxibP9d3cu17o6U	110.185.219.2	China	Sichuan	Chengdu
12D3KooWJe54VShngB16UngYvcmTB4VfnMPzKoxibP9d3cu17o6U	171.88.41.22	China	Sichuan	Chengdu
12D3KooWJe54VShngB16UngYvcmTB4VfnMPzKoxibP9d3cu17o6U	218.89.239.169	China	Sichuan	Chengdu
12D3KooWJefeaXhgJGmxzcqFq9twQASbt9Fuc2Ez7MRgrcpt3aLu	88.99.98.148	Germany	Saxony	Falkenstein
12D3KooWK23cGwK5ph1VucSSqjD2AhL8vu2vVCaJ7xGRiGSYbNaf	134.122.89.142	Germany	Hesse	Frankfurt am Main
12D3KooWKSrcfXFU13WYXVrEjLC8QkprFfeqd2fBibpSTP1DiJ1W	155.4.243.109	Sweden	Värmland County	Hagfors

Questions

Describe at least two differences existing between the classical Kademlia protocol and the version of the protocol used by IPFS.

The classical Kademlia protocol is enhanced in IPFS by taking ideas from S/Kademlia and the Coral sloppy DHT to enhance security and performance.
S/Kademlia is a secure key-based routing protocol that mainly implements Proof of Works through crypto-puzzles and public key cryptography to limit the free node identifier generation. There are also other features like parallel lookups over multiple disjoint paths used to avoid attackers that can reroute packets into its identifiers. To avoid eclipse attacks a static crypto-puzzle is used to make it difficult to generate a specific non-random node identifier because, if the attacker wants an identifier for a particular zone, it has to generate a public key such that its double hash has a prefix that starts with a given number of zeros. To avoid Sybil attacks a dynamic crypto-puzzle is used to require the attacker to use a lot of computational effort to generate many identities (through the computation of an hash function that takes in input the concatenation of the public key and a generated number X, the output must have a prefix that starts with a number of zeros that can be incremented to increase the difficulty).

The Coral DHT was originally used to develop a content distribution network, it solves some issues of classical DHTs like hot spots (a single node responsible for a popular content) and poor data locality (mainly because the last hops of the path are random) that are core problems for a P2P file distribution system.
To solve these issues, each node stores only a maximum number of references (list of values) to peers that manage a particular key. If the maximum is reached, the exceeding values are spread across multiple nodes. This is implemented in the content insertion algorithm by first trying to insert a pair (key, reference) using the classical Kademlia routing (forward phase), if the node that is responsible to store the pair is full then the algorithm backtrack on the lookup path and stores it on the first free node (backward phase). If a particular content is popular, a lot of nodes will become proxies and so the content may be found along the path to the original destination.

Describe the main advantages of the distance metric used by Kademlia with respect to other distances, like the ring distance of Chord.

The distance between 2 objects in Kademlia is computed as the bitwise exclusive OR of their identifiers. The longer is the common prefix of two identifiers, the smaller is the distance computed by the exclusive OR, so closer nodes are characterized by a long common prefix. This does not imply that the numeric values of the identifiers are closer. The XOR distance is a metric because it has several properties: it is symmetrical, transitive, it respects the triangular inequality and it is unidirectional (this property is really useful because it allows starting parallel lookups that converge to the same path and it is possible to store values along this path to avoid hotspots).

The symmetric property guarantees that for each pair of identifiers `x` and `y`, the distance between `x` and `y` is equal to the distance between `y` and `x`. This allows, differently from Chord, to build the routing table by acquiring knowledge of the network using queries that are originated by other peers. There are also other differences between Kademlia and Chord like the managing of a node join/leave and the managing of the routing tables that allows having much more flexibility and fault tolerance.