Mid Term Peer to Peer Systems and Blockchains

Fausto F. Frasca 559482

April 2021

Contents

1 Application Architecture

The application is written in Java and exploits the HTTP API offered by the go-ipfs client. Thanks to that, the application monitor can be executed also in a different machine with respect to where is running the ipfs node.

The architecture consist of six tasks execute by a cached thread pool. At the bootstrap of the application, the task are initialized and passed to the thread pool for their execution. The tasks are instances of the following classes:

- getObject
- outputStat
- activePeersStatistics
- bandwidthStatistics
- streamsStatistics
- agentVersionStatistics

What they do is explained in more detail in following paragraphs. The only one that return a value is *getObjectTask*. After received the value, the others task are stopped and the execution finish.

The execution finish before the initialization of the tasks if the IPFS node is not available.

1.1 GetObject

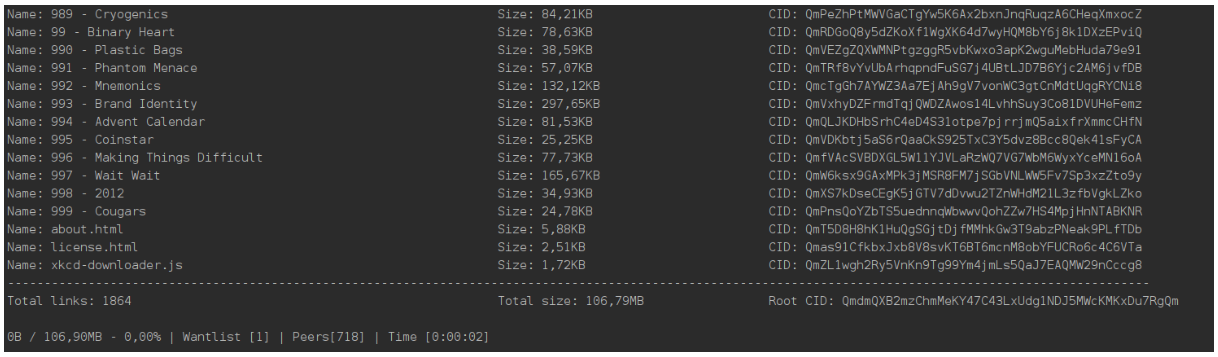
This class take care of sending the HTTP get IPFS request to the node for retrieve the object associated to the CID passed. When the download is finished, return the time required for the download in the form of string.

1.2 OutputStat

This is the only class that prints some statistic to the output. First of all it prints some information about the links retrieved from the dag. For each link are shown name, size and CID associated to the link. At the end of the list, there is a recap with the number of the links, the total bytes that will be downloaded and the root CID.

After that, it is printed a line with some statistics retrieved using the API */bitswap/stat*. From the right we have, the bytes already downloaded and the respective percentage, the number of keys in the wantlist list, the number of peers connected with the node and the elapsed time since the start of the download. An example is showed by the Figure 1. The bytes downloaded and the percentage are calculated indirectly because the API doesn't provide similar parameters, then there is an error of more or less 5%. The line is updated every 3.5 seconds.

The class can be found under the *Statistics* folder.



```
Name: 989 - Cryogenics                                             Size: 84,21KB                        CID: QmPeZhPtMWVGaCTgYw5K6Ax2bxnJnqRuqzA6CHeqXmxocZ
Name: 99 - Binary Heart                                            Size: 78,63KB                        CID: QmRDGoQ8y5dZKoXf1WgXK64d7wyHQM8bY6j8k1DXzEPviQ
Name: 990 - Plastic Bags                                           Size: 38,59KB                        CID: QmVEZgZQXWMNPtgzggR5vbKwxo3apK2wguMebHuda79e91
Name: 991 - Phantom Menace                                         Size: 57,07KB                        CID: QmTRf8vYvUbArhqpndFuSG7j4UBtLJD7B6Yjc2AM6jvfDB
Name: 992 - Mnemonics                                              Size: 132,12KB                       CID: QmcTgGh7AYWZ3Aa7EjAh9gV7vonWC3gtCnMdtUqgRYCNi8
Name: 993 - Brand Identity                                         Size: 297,65KB                       CID: QmVxhyDZFrmdTqjQWDZAwos14LvhhSuy3Co81DVUHeFemz
Name: 994 - Advent Calendar                                        Size: 81,53KB                        CID: QmQLJKDHbSrhC4eD4S31otpe7pjrrjmQ5aixfrXmmcCHfN
Name: 995 - Coinstar                                               Size: 25,25KB                        CID: QmVDKbtj5aS6rQaaCkS925TxC3Y5dvz8Bcc8Qek41sFyCA
Name: 996 - Making Things Difficult                                Size: 77,73KB                        CID: QmfVAcSVBDXGL5W11YJVLaRzWQ7VG7WbM6WyxYceMN16oA
Name: 997 - Wait Wait                                              Size: 165,67KB                       CID: QmW6ksx9GAxMPk3jMSR8FM7jSGbVNLWW5Fv7Sp3xzZto9y
Name: 998 - 2012                                                   Size: 34,93KB                        CID: QmXS7kDseCEgK5jGTV7dDvwu2TZnWHdM21L3zfbVgkLZko
Name: 999 - Cougars                                                Size: 24,78KB                        CID: QmPnsQoYZbTS5uednnqWbwwvQohZZw7HS4MpjHnNTABKNR
Name: about.html                                                   Size: 5,88KB                         CID: QmT5D8H8hK1HuQgSGjtDjfMMhkGw3T9abzPNeak9PLfTDb
Name: license.html                                                 Size: 2,51KB                         CID: Qmas91CfkbxJxb8V8svKT6BT6mcnM8obYFUCRo6c4C6VTa
Name: xkcd-downloader.js                                           Size: 1,72KB                         CID: QmZL1wgh2Ry5VnKn9Tg99Ym4jmLs5QaJ7EAQMW29nCccg8
------------------------------------------------------------------------------------------------------------------------------------------------------------
Total links: 1864                                                  Total size: 106,79MB                 Root CID: QmdmQXB2mzChmMeKY47C43LxUdg1NDJ5MWcKMKxDu7RgQm

0B / 106,90MB - 0,00% | Wantlist [1] | Peers[718] | Time [0:00:02]
```

Figure 1: Output of the class OutputStat

1.3 ActivePeersStatistics

This class is the only one that create two file for two different statistics, both regarding the peers that send data to the node, called active peers. The first statistic is about some specific information of the active peers. Every active peer is represented by an instance of the class *Peer*, that can be found to the folder *Classes*, present in the directory *Peers*. The Peer class contains the following data:

- recv: the data received from the peer
- sent: the data sent to the peer
- exchanged: number of blocks exchanged
- streams: list of streams used by the peer
- continent
- country
- latency
- region
- city
- IP

For retrieve the information about IP, city, region, country and continent is used the API offered by the service *api.ipstack.com*. The call to the IPFS node for retrieve the IP associated to the peer return a list of IP, so it is chosen the first one that generates not null information when it is sent to the ipstack service. The ipv6 and the localhost ip are discarded regardless. The first one because they are not supported by the ipstack, the second one for obvious reasons.

All the instances of the Peer class are collected to the map *mapActivePeers*. This map is written to the file *active_peers_statistics.json* before that the task terminate, when the stop method is invoked.

The second statistic monitored is more simple. It is created a ActiveAndTotalPeers instance that contain three parameters:

- totalPeers: total number of peers to which the node is connected.
- activePeers: total number of peers that have sent some data to the node.
- time: time when the instance of the class is generated.

The instance is immediately written to the file *peers_statistics.json* using the auxiliary class StreamFileWriter. Thanks to that the monitoring guarantees the use of less RAM, especially when the it lasts several hours.

The class ActiveAndTotalPeers can be found to the same folder of the class Peer, while the class ActivePeersStatistics can be found following the path */Statistics/Peers.*

Both the monitored statistics are updated every 15 seconds.

1.4 BandwidthStatistics

This class has the duty to control the bandwidth during the download of the CID. Every 15 seconds it sends a request to the IPFS node, retrieves and clean up the response and writes it in the file *bandwidth_statistics.json* using the auxiliary class StreamFileWriter. This approach allows to not have this data in the RAM, making the monitoring more lightweight when it lasts several hours. Regarding the data, are considered both rate in and rate out.

The class in accessible to the following path *Statistics/Bandiwdth.*

1.5 StreamsStatistics

This class has two map, one containing all the peers seen called *sawSwarmPeers*, and the other one containing the couples (stream name, #occurrences) called *streamsProtocols.* It every 15 seconds retrieves the peer present in the swarm and, for each peer, it checks if it was already seen. If not, the ID of the peer is added to the sawSwarmPeers and the list of streams used by it is retrieved. For each stream the associated occurrence is updated in the streamsProtocols.

When the stop method is invoked, before that the task terminates, the statistics are written to the file *streams_statistics.json.*

The class in accessible to the following path *Statistics/Streams.*

1.6 AgentVersionStatistics

This class has two map, one containing all the peers seen called *sawPeers*, and the other one containing the couples (agent version, #occurrences) called *agentVersions.* It every 15 seconds retrieve all the peers present in the buckets of the distributed hash table. For each peer, check if it is present in sawPeers.

If not, the peer is added to the map and the agent version's occurrence into the agentVersions is update.

When the stop method is invoked, before that the task terminates, the agentVersions map is written to the file *agent_versions_statistics.json*

The class in accessible to the following path *Statistics/AgentVersions*.

1.7 Others Classes

The following list contains the auxiliary classes used by the program:

- *IPFSNode*: this class has the duty to interface with the IPNS node providing the appropriate HTTP APIs. All the tasks that send an HTTP request use this class.
- *BufferedFileWriter*: this class, using the BufferedWriter Java's class, create a file, write the passed statistics and close the file. All the tasks that write their statistics before their termination use this class. The class is in the folder *Aux_Classes*.
- *StreamFileWriter*: this class is used by all the tasks that have to write streams of data to the their file. The class uses the DataOutputStream Java's class. It is in the folder *Aux_Classes*.

2 Statistics

The files downloaded from the IPFS network are:

- **MDSConnect**
  - *CID*: QmcvfB6pAqUfTnuAK8zFKVxbdhopnBPveJrDcy1JAA7HX5
  - *Size*: 18,02GB
  - *Download time*: 8:38:24
- **textfiles.com**
  - *CID*: QmNoscE3kNc83dM5rZNUC5UDXChiTdDcgf16RVtFCRWYuU
  - *Size*: 1,62GB
  - *Download time*: 6:25:28
- **World Wide Web History Project**
  - *CID*: QmRTSA1UFHSx3z7taNRwUVM8AjB2EQwKvyZu3BfJg9QRtZ
  - *Size*: 91,98MB
  - *Download time*: 1:08:44
- **XKCD**

  - *CID*: QmdmQXB2mzChmMeKY47C43LxUdg1NDJ5MWcKMKxDu7RgQm
  - *Size*: 106,79MB
  - *Download time*: 0:59:42

- **yarchive.net**
  - *CID*: QmdA5WkDNALetBn4iFeSepHjdLGJdxPBwZyY47ir1bZGAK
  - *Size*: 195,71MB
  - *Download time*: 0:29:14

The statistics are collected in appropriate folders, denominated with the pattern name_CID, in the directory *StatisticsFiles*. In each of them there are the json files generated during the monitoring plus a directory called *dist*. Inside it there is a html file that shows in more details all the statistics collected using interactive tables and charts.

2.1 Explanation of table and charts

2.1.1 Active peers information

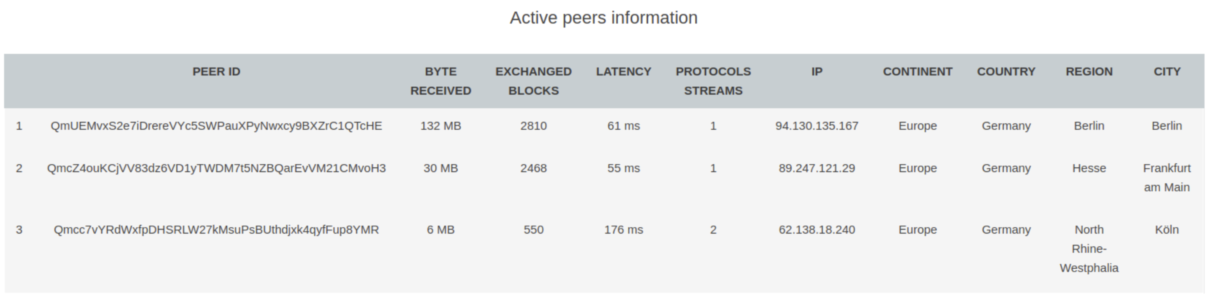
This table shows in detail the information about the active peers:

- PEER ID
- BYTE RECEIVED
- EXCHANGED BLOCKS
- LATENCY
- PROTOCOLS STREAMS
- IP
- CONTINENT
- COUNTRY
- REGION
- CITY

An example is shows by the Figure 2.

2.1.2 Active peers per continent

This column chart shows the numbers of active peers per continent. It is followed by a variable number of column carts that show number of the active peers per country in each continent. The Figure 3 shows an example, followed by the chart in Figure 4 in the corresponding html file.



Active peers information

	PEER ID	BYTE RECEIVED	EXCHANGED BLOCKS	LATENCY	PROTOCOLS STREAMS	IP	CONTINENT	COUNTRY	REGION	CITY
1	QmUEMvxS2e7iDrereVYc5SWPauXPyNwxcy9BXZrC1QTcHE	132 MB	2810	61 ms	1	94.130.135.167	Europe	Germany	Berlin	Berlin
2	QmcZ4ouKCjVV83dz6VD1yTWDM7t5NZBQarEvVM21CMvoH3	30 MB	2468	55 ms	1	89.247.121.29	Europe	Germany	Hesse	Frankfurt am Main
3	Qmcc7vYRdWxfpDHSRLW27kMsuPsBUthdjxk4qyfFup8YMR	6 MB	550	176 ms	2	62.138.18.240	Europe	Germany	North Rhine-Westphalia	Köln

Figure 2: Active peers information's table of yarchive.net

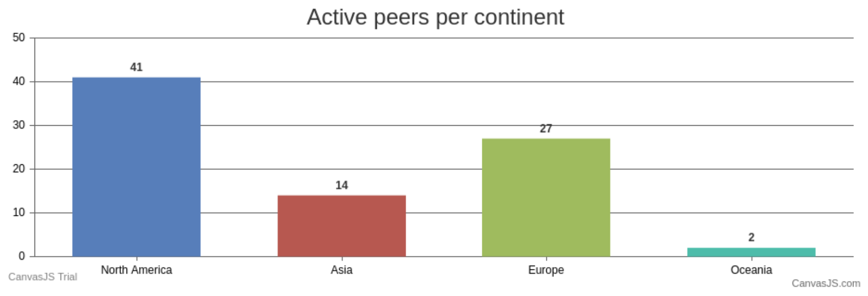


Figure 3: Active peers per continent's chart of XKCD

2.1.3 Top 10 agent versions used

This column chart shows the top 10 agents used by the peers present in the buckets of the dht in the IPFS node. In Figure 5 can be seen an example.

2.1.4 Bandwidth kb/s

This chart show the bandwidth trend for both rate in and out during the download of the files. The data are represented in kb/s. There is also a constant line that show the average rate in, as can be seen in Figure 6

2.1.5 Connected peers and active peers

This chart show the trend of the connected and active peers during the download of the files. You can find an example in Figure 7.

2.1.6 Streams used by peers in the swarm

This column chart shows the most used streams by the peers present in the swarm peers, as you can see in Figure 8.

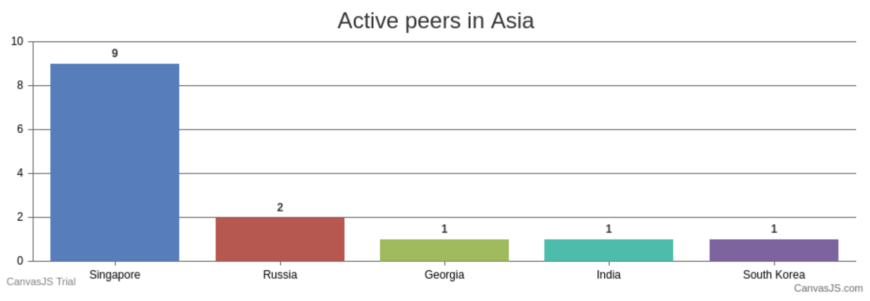


Figure 4: Active peers per continent's chart of XKCD

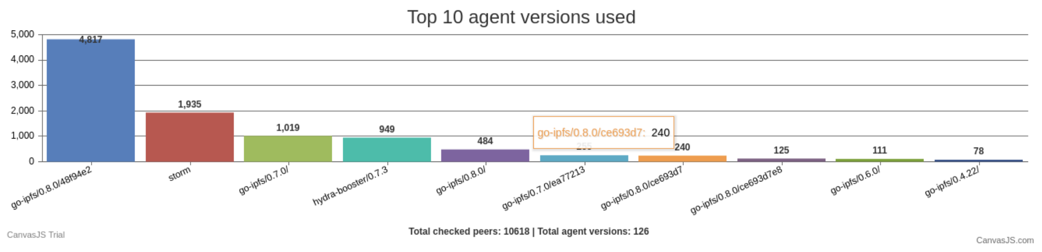


Figure 5: Top 10 agent versions chart of MDSConnect

2.2 Discussion about the statistics

All the statistics have been collected using always the same steps for have always the same starting point: the garbage collector is invoked for the repo, the IPFS node is restarted and after that the download of the object start.

The first evidence about the network is the extreme slowness for download the files, even if of few MB, as shown in the chart of the bandwidth. And the cause is not the home network, because it was use the fiber, but the number of active peers respect the number of peers connected with the IPFS node. An example can be *yarchive.net*, where the average rate in is 181 kb/s and the number of connected peers fluctuates between 400/700 peers while the active peers are only 3. This trend can be saw also for the others files. The maximum number of active peers was 84 during the donwload of the *XKCD*, but also in this case the average rate in was very slow, with only 110 kb/s

About the nationality of the peers the continents that contribute more in the network are the Europe and the North America. About the Europe the major contributor is the Germany. This country is a constant among the active European peers. In some cases it is the only one for the active European peers, in some others cases is flanked by countries like France, United Kingdom, Netherlands, Sweden and others. About the Italy only once has been discovered an active Italian peer, during the download of the *textfiles.com*. It was located in Belpasso, Sicily. Instead for the North America the major contributor are the

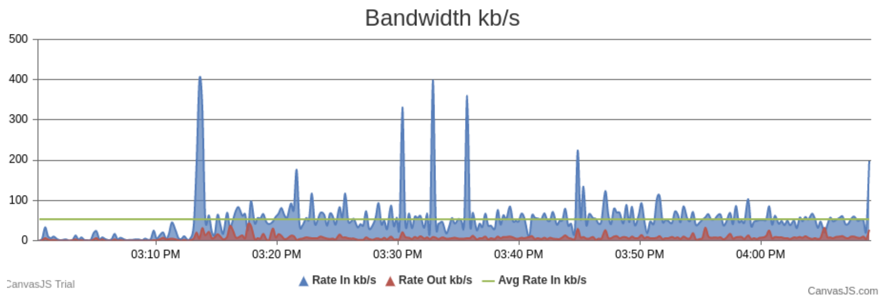


Figure 6: Bandwidth's chart of World Wide Web History Project

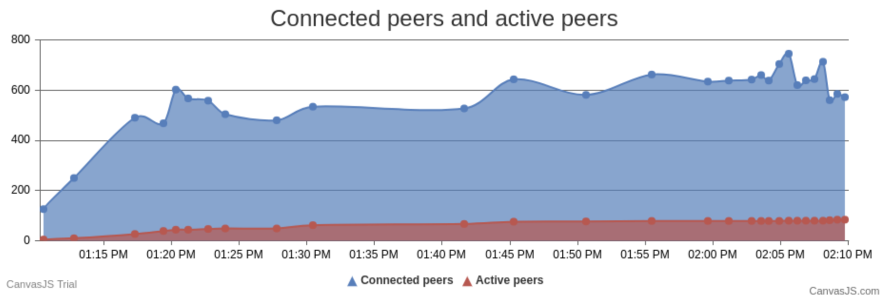


Figure 7: Connected and active peers chart of XKCD

United States. Only in one case was it was flanked by active Canadian peers, during the download of *XKCD*. Always during the the download of XKCD between the active peers have been found peers coming from Asia and Oceania, two rare continents for this test.

About the agents the most used is go-ipfs, version 0.8.0, release 48f94e2, followed by storm and go-ipfs version 0.7.0, the previous version of the official ipfs client. This statistic shows that the users are careful to the developing of the project, updating immediately their go-ipfs client, since the adoption rate of the version 0.8.0 is very high and it was released only in the middle of February. This guarantee a less fragmentation of the client and a less support that should be provided by the developers for the old versions. The storm agent instead is a malware that use the IPFS network to obscure the malicious traffic and inject virus to the peers. Initially only Windows was affected, but now it attacks also Linux and Mac OS.

Instead, about the stream most used by the peers is kademlia version 1.0.0, followed by bitswap version 1.2.0.

For see all the statistics of all downloaded files, you can see the html files. How visualize them is explained in the following chapter. Viewing is highly recommended. The number of tables and charts are too much for put them all

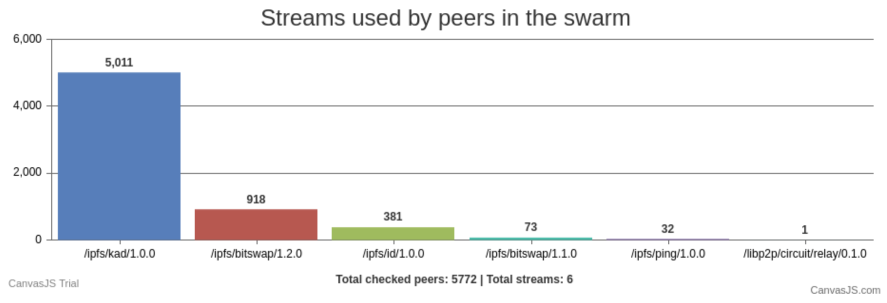


Figure 8: Streams used by peers in the swarm chart of XKCD

together in this PDF file.

3 How to

In this section it will be described how to launch the monitor application and how visualize the html file containing the statistics of the downloaded files. All the tests have been executed in a Linux distro using openjdk 15.0.2 and Apache Maven 3.6.3.

3.1 How to launch the application

The application has been build using Maven, so the dependencies required for run it are Java 15 or more and Maven. First of all, you need to setup the parameters IP and CID present at the beginning of the Main class. By default there is the configuration used during the tests. You can access the Main class following the path *src/main/java/*. After that, for compile and exec the program you can run the dedicated script typing the command ./compile&run.sh in the terminal. First you should see the compilation output and after the output of OutputStat class.

3.2 How to visualize the html statistics file

You must have chrome or chromium, the derived browsers like brave don't work. First of all you need to launch the browser typing in the terminal the command *chromium --allow-file-access-from-files*, after that the browser will be opened and you can drag & drop the html file that you want visualize. At the end you should see a web page containing interactive tables and charts.

4 Q&A

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

- **A:** IPFS use a modified Kademlia protocol called S/Kademliad together with Coral. This protocol improvement provide an high resilience against common attacks to a P2P network, thanks to the restriction applied to the node identifier generation using cryptopuzzles (PoW) and public key criptograhy, the use of routing tables by a sibling list and parallel look-ups over multiple disjoint paths. To avoid the eclipse attack it is used a static cryptopuzzle. For generate a particular ID you need to find a public key such that its double hash n leading zero-bits. Instead against sybil attack is used a dynamic cryptopuzzle. When you want generate an ID you need to compute the hash of the bitwise xor of the public key and a random number X such that the output has m leading zero-bits. These cryptopuzzles are easy to very but hard to solve. The parallel look-ups over disjoint paths instead is used against the attackers that reroutes the packet into their own subset of nodes. Coral is a Sloppy Distributed Hash Table that solves classical DHTs issues like hot spots, when a single node become responsible of popular URL, and poor data locality, the DHT has poor data locality.

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

- **A:** The main advantage of Kademlia with respect to other distances, like Chord, is the symmetry thanks to the bitwise xor. This means that in Kademlia the distance from node A to node B is equal to the distance from node B to node A and their distance will be between 2^{i-1} and 2^i, where i is the number of differents bits between ID_A and ID_B.