Clients behaviour report

Introduction

In order to study differences in behaviour of ETH-2 clients, the BSC team has been running Teku, Nimbus, Lighthouse, Prysm and Lodestar for several hours, letting the clients sync to the ETH-2 chain from scratch. The objective of this study has been to monitor specific metrics in order to understand the behaviour and performance of the clients when initialized to sync to the ETH-2 network.

Clients have not always been run in the exact same period of time, as Prysm and Nimbus have been launched at the same moment and run together, Teku and Lighthouse were also launched together, but Lodestar and a second run of Teku have been run during at different periods of time. In addition to some differences in clients' behaviour that may have been caused by conditions of the Medalla Test Network at a given time, we have found out some patterns and gathered some insights about the clients.

The metrics that have been monitored are:

- Syncing time
- Peer connections
- Disk Usage
- CPU
- Memory Usage
- Network incoming traffic
- Network outcoming traffic

And all the figures can be found at our Github repository

Considerations:

- Tests have been run on two nodes, each node with an Intel(R) Xeon(R) CPU E5-2620 0 @ 2.00GHz with 1 core and two threads.
- Each client has been run on a single node with the following resources:
    - Total memory available: 4.8 GB
    - Total Storage: 34 GB (40GB total, of which 6GB used for the system)
- Teku has been run allocating 2 GB to the JVM, as allocating more resources would make the server crash.
- Clients have been running on the Medalla Test Network using the basic configuration. Non additional flags or configuration have been used with the exception of the flag on teku that limits the JVM memory and rising the heap size of Lodestar on the package.json.

Analysis

Client Syncing

In *Figure 1* we can see that Lighthouse appears to be the fastest client when it comes to syncing, though its syncing process stops (around hour 24) due to reaching the maximum storage capacity of the server. This behaviour is analyzed more in depth in the section “Disk Usage”. After Lighthouse stopped its syncing process, it was unable to catch up. Prysm was able to sync faster than the other clients in the long run with a steady progress throughout the whole test. Nimbus and Lodestar seem to be the ones that take more time to sync.

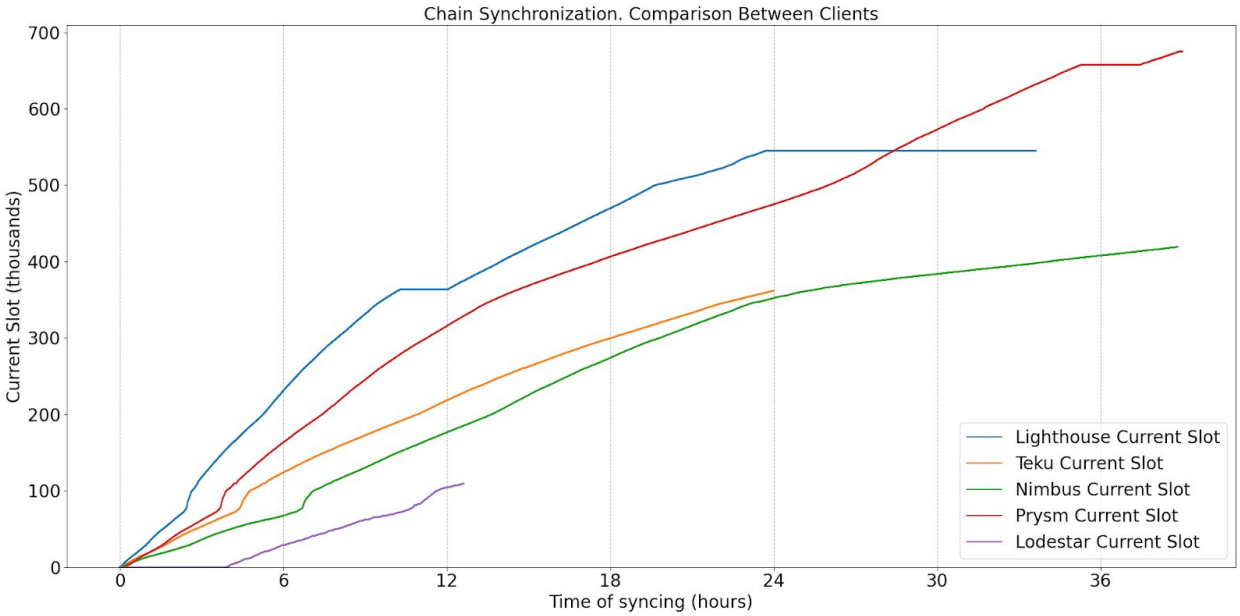

Figure 1 - Current slot of Eth2 clients

Peer connections

We also keep track of the number of peer connections for each client during the experiment. We can see in *Figure 2* that Teku has a stable number of peer connections at more than 70 peers. Lighthouse is in general the client that connects with more peers after Teku: it is stable with around 50 peer connections until a period (hour 20) where the number of connected peers drastically decreases (oscillating around 25 peers), and later rises its peer connection back to 50 peers (hour 23). After hour 23 we can see sharp drops and recoveries in peer connections. Nimbus peers connections oscillate significantly, usually connecting with over 15 peers and arriving up to 42 peers. Prysm distinguishes itself for its stability of peer connections, as it succeeds in achieving 30 peer connections without almost no variation. When it comes to Lodestar, the client does not report the number of peers until the moment in which Lodestar

retrieves the genesis block (hour 4), and from this point we can notice that Lodestar's peer connections oscillate in the range of 18-30 peers.

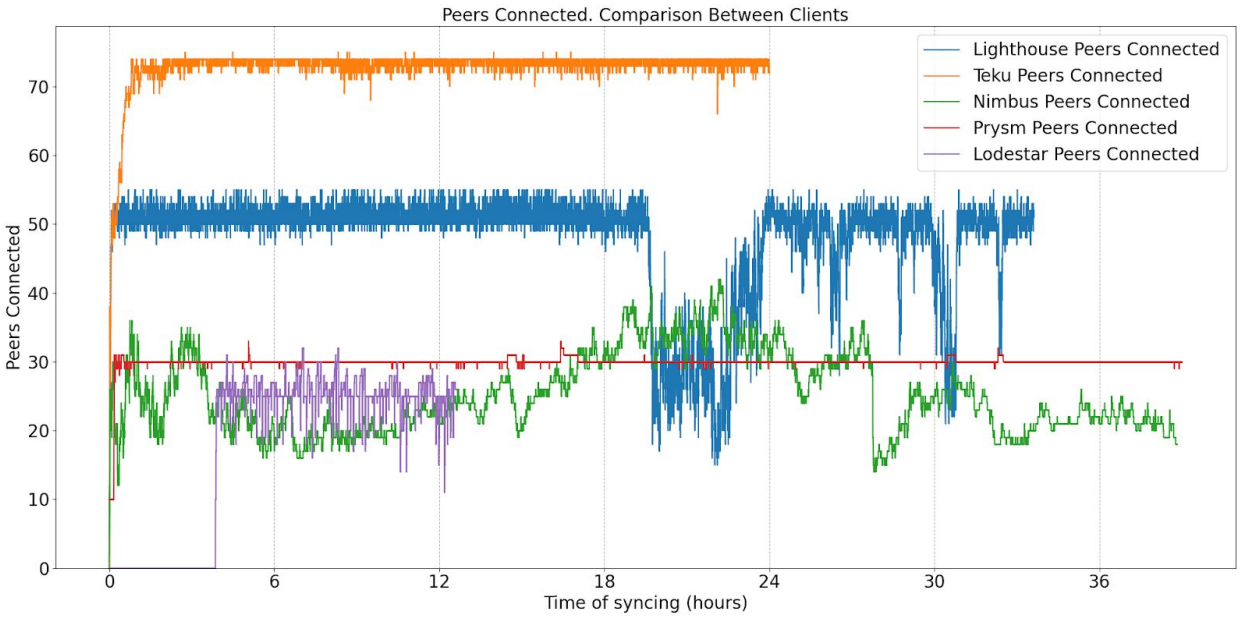

Figure 2 - Peer connections of Eth2 clients

Disk usage

When it comes to Disk Usage, we can see in *Figure 3* how Nimbus maintains a low storage usage, followed by Prysm and Teku. Differently from other clients, Teku shows an oscillating behaviour in disk usage, showing that it records on its storage more data than other clients, and performs periodic cleaning. We also witness some interesting behaviour for Lighthouse and Teku as we can see (hours 3-7) a sharp increase in Disk usage for Teku and Lighthouse, succeeded by a similarly sharp drop. Interestingly, this behaviour is witnessed from the clients around slot 100000, where we can see that the speed of syncing rises rapidly. Nimbus also witnesses a relatively sharp rise in syncing and disk storage around the same slot, but it does not reduce its storage afterwards (in contrast to Teku and Lighthouse). At the same slot, Lodestar appears to sync faster while its storage remains constant. Prysm also increases its syncing speeds, and it's disk storage continues its trend without any variations. The increase of disk usage for different clients for a specific slot can be seen even more clearly in *Figure 4,* where it can be seen the different behaviours of the clients when handling disk usage around slot 100000, and witness a sharp increase in the disk usage of Teku, Lighthouse and Nimbus while the behaviour of Prysm and Lodestar seems to be unaffected. It is interesting to notice also how Nimbus rises its disk usage around slot 100000 more conservatively than Teku and Lighthouse, and does not witness any storage reduction afterwards. Considering the relatively similar behaviour of noticeably increasing and decreasing disk usage (hours 3-7, *Figure 3*) it is interesting to notice how Teku minimized the time of higher disk usage, while Lighthouse has run several hours with additional data before dumping it.

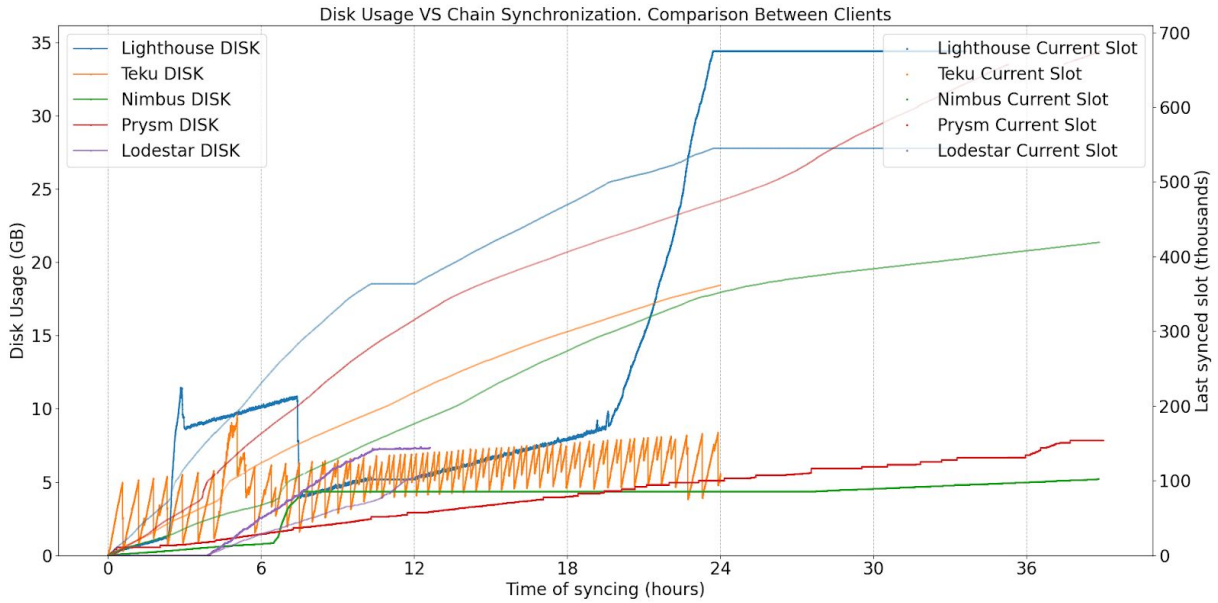

Figure 3 - Disk Usage of Eth2 clients

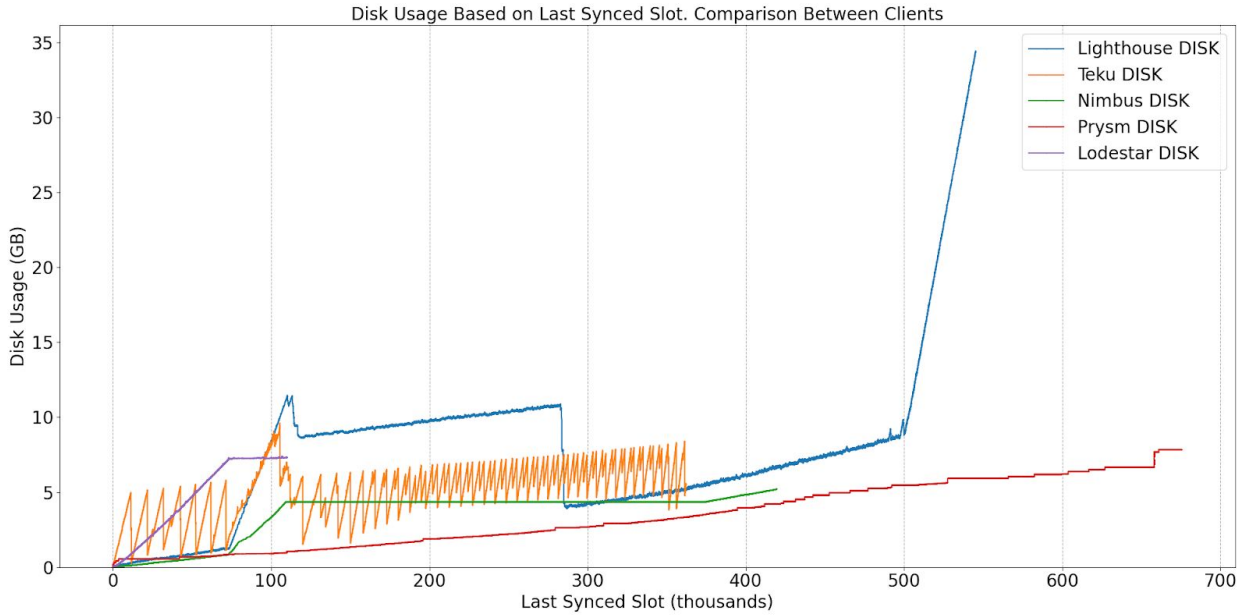

Figure 4 - Disk Usage of Eth2 clients for slots

Interestingly, there is a time (hour 20, *Figure 3*) from which Lighthouse sharply accumulates data, while at the same time slightly reducing it's synchronization speed. After this high increase in disk usage, Lighthouse reaches the disk limit of our server, and its syncing stops. We can notice this behaviour around slot 500000 *(Figure 4*), and considering that no increase in syncing speed has been detected, Lighthouse's behaviour in this slot does not seem to be correlated to the behaviour occurring in slot 100000. Even if Prysm's syncing speed increases around slot 500000, its disk usage remains unaffected (as what happened around slot 100000). We assume

that in a non-finalization period, Lighthouse has been downloading all the forks and consumed more storage than needed. With the exception of this strange Lighthouse behaviour, Lodestar seems to use more storage than the other clients. It can be noticed also that Lodestar starts gathering blocks after a period of time (hour 4, *Figure 3*), and this behaviour seems to depend on the time spent researching the genesis block.

CPU

From monitoring the CPU usage across the different clients, we noticed in *Figure 5* that CPU usage of Prysm and Lodestar concentrates around 50%, while CPU usage of Teku usually oscillates up to 100% and Nimbus' CPU does not surpass 50%. Lighthouse is usually above 50% until a point (hour 20-24) where the client can rely on available storage to gather blocks and its CPU stabilizes around 100%. After this period (hour 24), Lighthouse is not syncing because the system reaches out its storage capacity, and its CPU consequently drops.

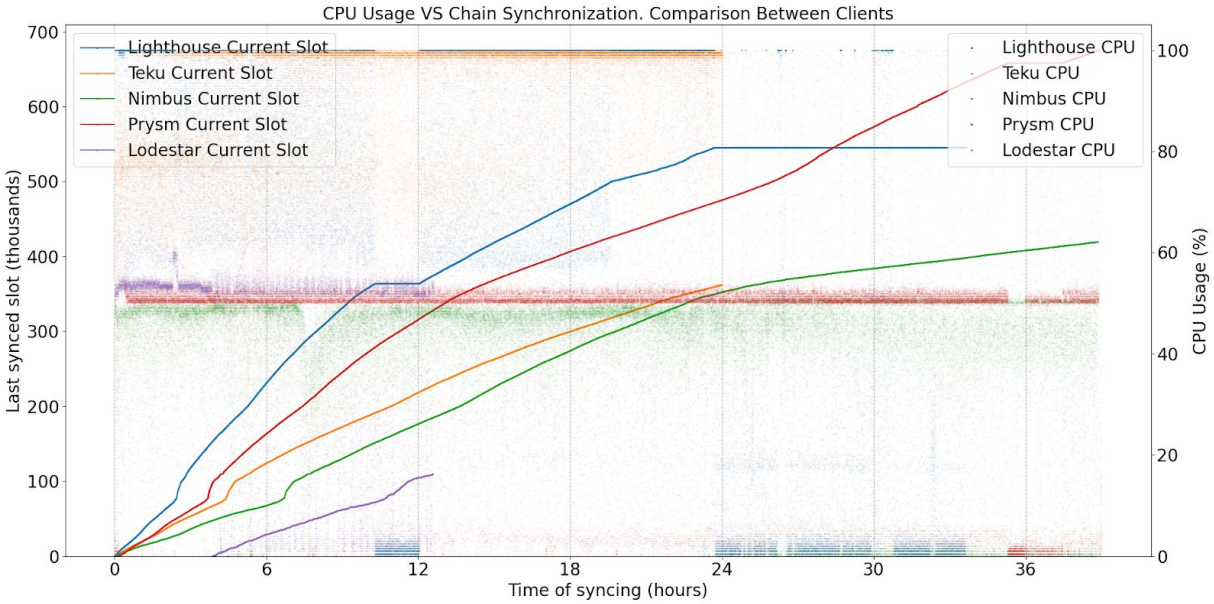

Figure 5 - CPU usage of Eth2 clients

Also we can notice at hours 10-12 how the CPU usage of Lighthouse is reduced drastically in correspondence of periods of inactivity where the client is not syncing nor gathering blocks (*Figure 6*), while its number of peer connections is stable (*Figure 7*) and incoming traffic does not witness changes *(Figure 8)*. We can witness a similar behaviour for Prysm (hour 35, *Figure 10*), with the difference that in the second part of the period where the syncing stops (hours 36-37, *Figure 10*) the CPU recovers back to 50% and the disk usage increments again (while network incoming traffic is reduced (*Figure 11*). It would be interesting to understand the reason why these clients have halted their syncing process for this period while still having stable peer connections. Nimbus witnessed a quite sharp drop in CPU usage (hours 7-8, *Figure 9*), and we can see that this drop comes just after a steep increase in disk usage and blocks synced.

In the other clients, such an increase in synced blocks in a period of time is not followed by a CPU drop.

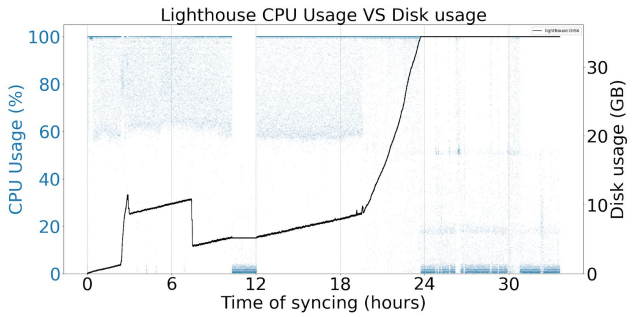

Figure 6 - Lighthouse CPU and disk storage

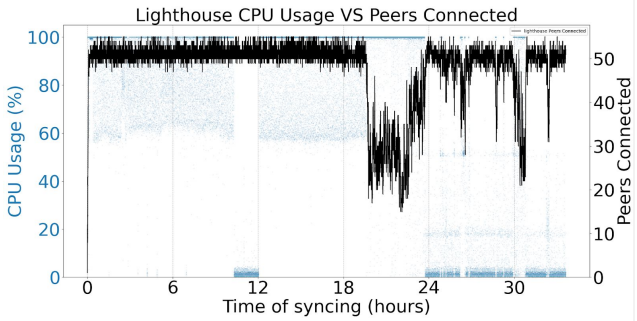

Figure 7 - Lighthouse CPU and peers connections

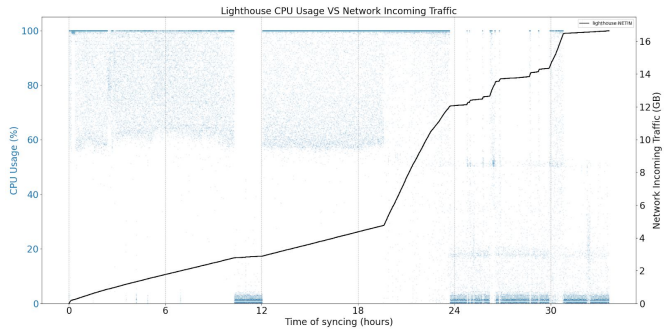

Figure 8 Lighthouse CPU and incoming traffic

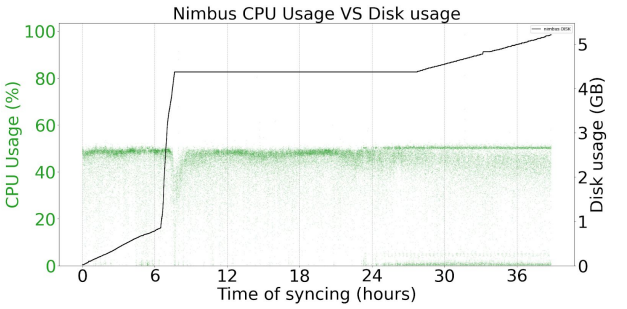

Figure 9 -Nimbus CPU and disk usage

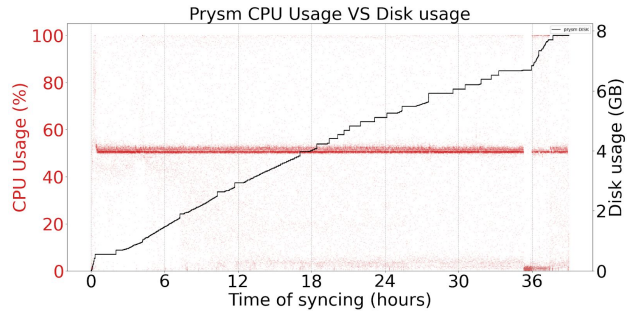

Figure 10 - Prysm CPU and disk storage

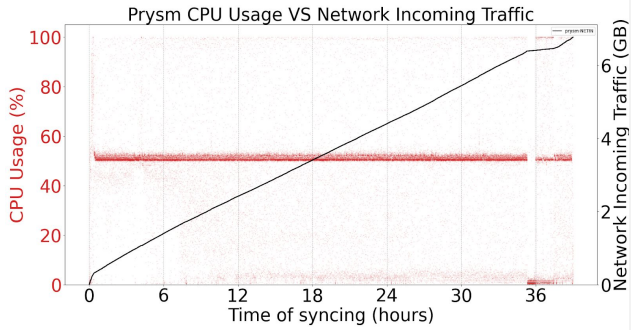

Figure 11 - Prysm CPU and incoming traffic

Memory

When it comes to memory consumption, we can see in *Figure 12* how Teku and Prysm tend to have a similar behaviour, by starting with a steep usage of memory consumption and settling with similar performances in the long run. We can notice sharp drops of memory usage of Prysm indicating some periodic cleaning process, while the behaviour of Teku is more constant. Nimbus distinguishes itself with a very low memory consumption with minimal changes.

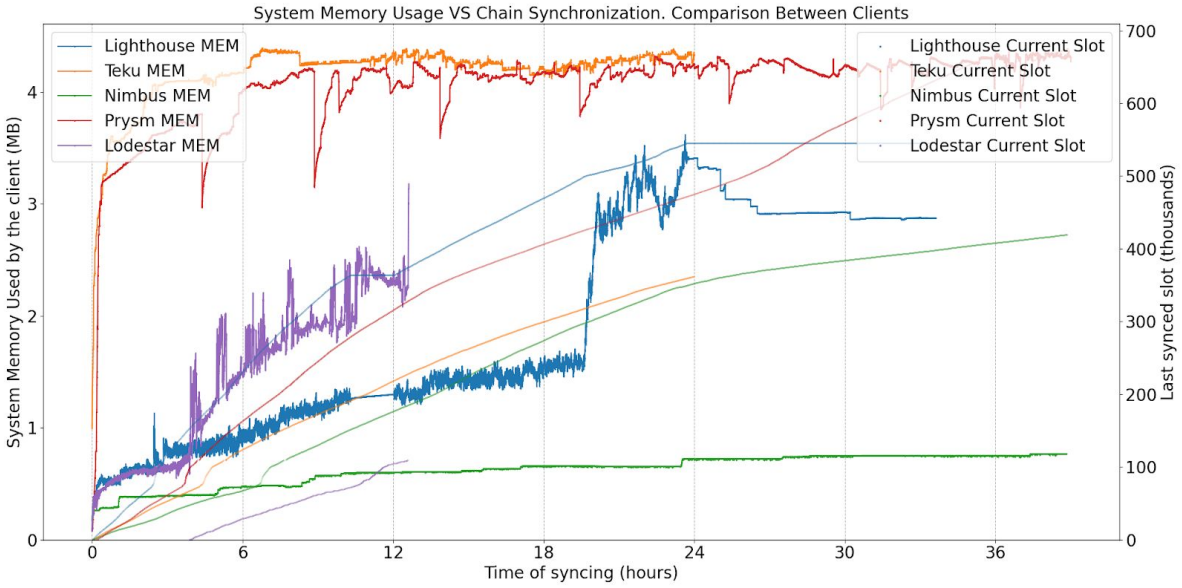

Figure 12 - Memory consumption of Eth2 clients

On the other side, in *Figure 13* we can witness a sharp rise in memory usage of Lighthouse (hour 20) that corresponds to the time in which Lighthouse began to sharply increase its storage usage (probably due to non-finalization period).

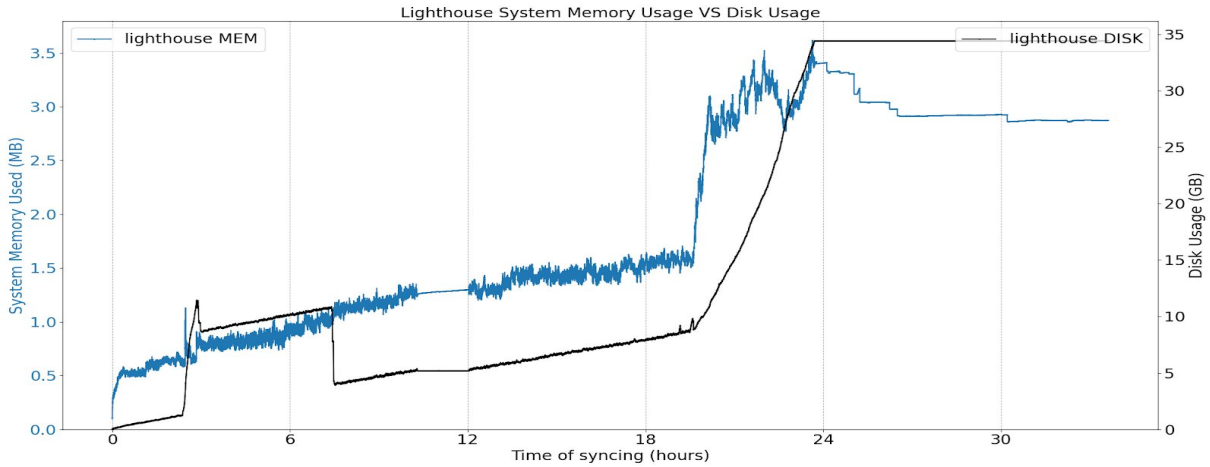

Figure 13 - Lighthouse memory and disk storage

After this period, Lighthouse reaches the disk storage limit of the server, and it can be seen that Lighthouse's memory usage begins to diminish while its synchronization stops. Lodestar memory usage is characterized by periodic spikes, and appears to be less demanding than the memory usage of Teku and Prysm. It can also be noticed that the first relevant spike corresponds to the moment in which the client actually starts the syncing process.

Network incoming traffic

When it comes to network incoming traffic, we can see in Figure 14 that Nimbus is the lightest client followed by Prysm. Teku is the client that has more network traffic for the first period (until hour 21), but then Lighthouse sharply increases network incoming data, memory usage and storage while its connection to peers drastically drops (*Figure 15)*.

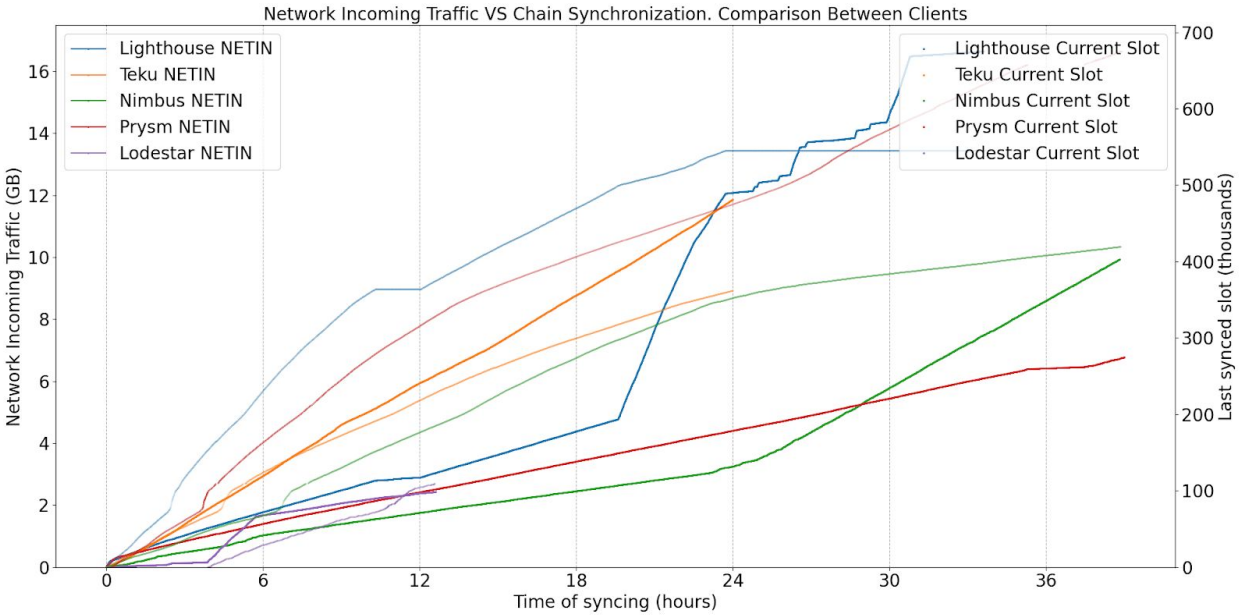

Figure 14 - Clients network incoming traffic

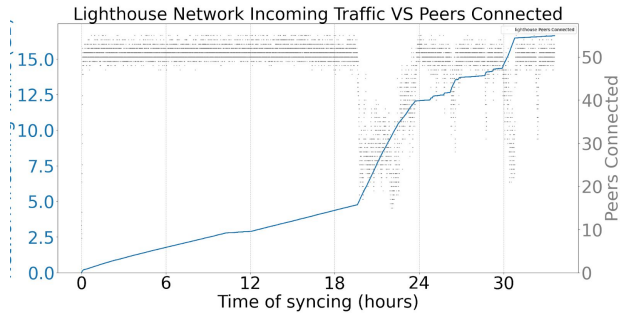

Figure 15 - Lighthouse peers connections and network incoming traffic

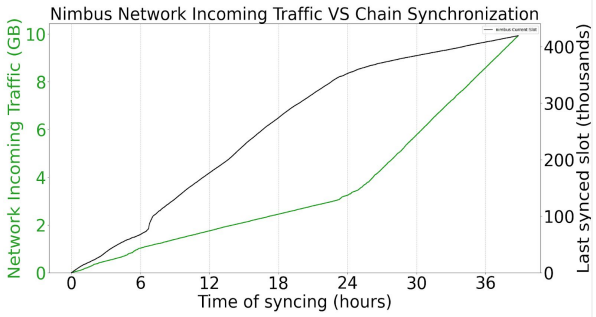

Figure 16 - Nimbus syncing and incoming network traffic

The rising of network incoming traffic and the decreasing of peer connections can also be noticed in other periods (hours 27 and 30-32, *Figure 15*). For Nimbus, It is interesting to notice in *Figure 16* that there is a point in which incoming network traffic increases, while at the same time the curve that represents its synchronization smoothens. It is not clear why network incoming traffic has been rising while the syncing does not appear to have accelerated accordingly (but has actually slowed). Prysm and Teku show a normal steady behaviour.

Network outcoming traffic

In general, we can see in *Figure 17* normal behaviour across the clients for network outcoming traffic. Prysm and Nimbus are the more conservative in outcoming network traffic and their behaviour is quite similar (as their network outcoming traffic almost overlaps).

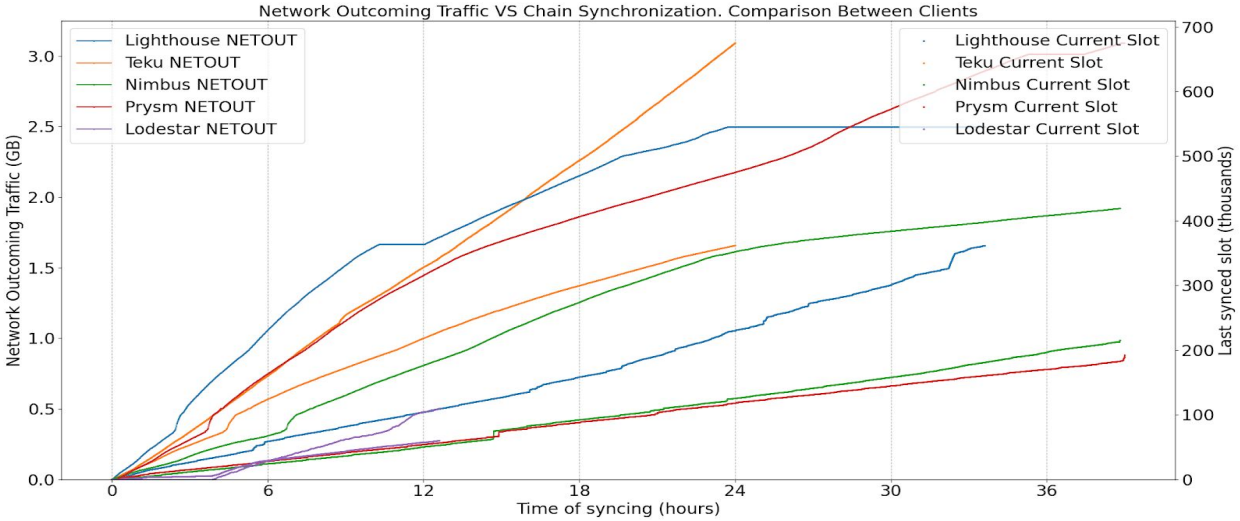

Figure 17- Clients network outcoming traffic

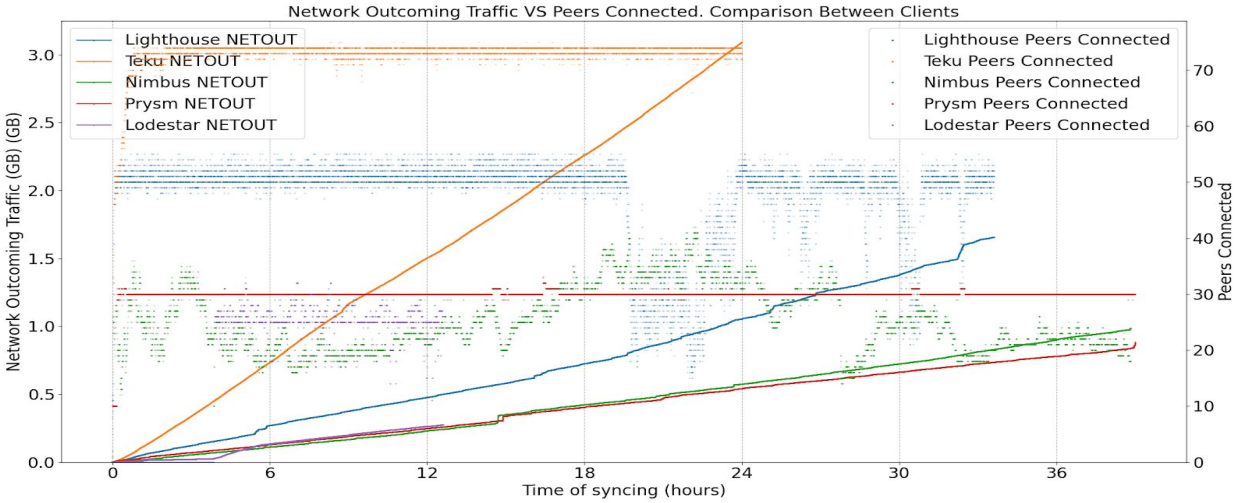

Figure 18- Clients network outcoming traffic and connected peers

Teku seems to share more data to the peers, as we can see from a steep increasing line. The amount of data shared by clients can be contextualized by comparing it to clients' peer connections: those clients with more peer connections have more outcoming traffic as it can be seen in Figure 18.

Conclusions

For our test we assumed that the majority of users would run the clients with their default configuration. For this reason, clients have been launched with default configuration (with the exception of the Teku flag to limit the JVM memory, and rising the Lodestar heap size in the package.json). The use of this default behaviour needs to be taken into account when reading these conclusions, as specific customizations could be performed on the clients in order to obtain different results and conclusions.

Nimbus

From the study we carried out, Nimbus appears to be the lightest client for memory usage, disk storage and CPU requirements. The focus of the Nimbus team to optimize this client to run on mobiles is reflected by the low consumption of resources. When it comes to network traffic (both incoming and outcoming) we witness that it is second just to Prysm, and it would be interesting to understand what made the Nimbus client have more incoming traffic and slower speed of syncing (from hour 23, Figure 18). Nimbus' CPU usage is the lowest of the clients, not rising more than 50%. If the objective of the team is to reduce further the incoming and outcoming traffic, Prysm could be an interesting benchmark to rely on. At the same time, the reduced network traffic makes Nimbus the slowest client to sync the chain. As the process of syncing the whole chain is just a one time job, it is interesting to see the achievements in running a functional ETH-2 client with such minimal use of resources.

Teku

Teku seems to require more resources to be run than the rest of the clients (i.e., Memory, network and CPU). The objective of targeting commercial institutions is reflected by prioritizing robustness over resource optimization. In the study we performed, we allocated 2GB to the JVM, as the server would crash if more resources were allocated. We know that the team is aware and working on optimizing this issue. Teku seems to be quite resilient, as we could compare Teku's behaviour with Lighthouse in the same period of time, and witnessed a faster management of disk storage (hours 3-7) in moments of non-finalization. Seeing the disk storage behaviour, it seems Teku stores a higher amount of data and dump it regularly. We assume this behaviour is obtained on purpose in order to let the client manage fork decisions faster. As for higher network outcoming and incoming traffic, we notice that Teku can rely on more resources to propagate and gather blocks, which can be an advantage from the network perspective, both on security and efficiency.

Lighthouse

Lighthouse appears to be the fastest client when it comes to chain syncing. On the other side, it shows some unpredictability in its behaviour, noted for example by the sharp rise in disk storage, memory and network traffic (hour 20, Figures 6-7-13). Even after the sharp rise in memory consumption, it still is lower than the one of Teku and Prysm. Disk storage is in general higher than the rest of the other clients and it also sharply increases (hour 20), surpassing by far the storage allocated by any other client (reaching the server limit and preventing further syncing). Incoming network traffic also increases steeply, higher than every other client, while outcoming network traffic is lower than Teku. It is reasonable to think that in our test Lighthouse has not been able to finalize and kept downloading all the possible forks, reaching the maximum capacity of the server. It would be interesting to understand the reasons for this behaviour, as well as exploring ways to manage it, as in periods of non-finalization resource consumption increases drastically.

Prysm

Prysm client seems to be quite optimized when it comes to the relatively low disk storage, and CPU generally stable around 50%. Also, incoming and outcoming network traffic is lower than any other client. On the other side, memory consumption seems to be at the same level as Teku. Prysm seems to behave quite predictably in terms or resources usage, except for periodically sharp drops in memory consumption. Its connection with peers is also very stable.

Lodestar

While memory usage is lower than Teku's and Prysm's, and shows that Lodestar can be run with relatively moderate resources. Behaviours such as the steep outcoming network traffic and the time waited in order to get the genesis block seem to leave space for optimization.

Acknowledgement

This work has been done by the BSC team (Mikel Cortes, Luca Franceschini and Leonardo Bautista-Gomez) supported with grants from the Ethereum Foundation.