Monitoring and Workload Characterization in the IPFS Network

Master Semester Project by Simon Jacob

Advisor: Prof. Dr. Bryan Ford

Supervisors: Dr. Vero Estrada-Galiñanes, Pasindu Tennage

Motivation

- Independent projects are increasingly depending on decentral networks

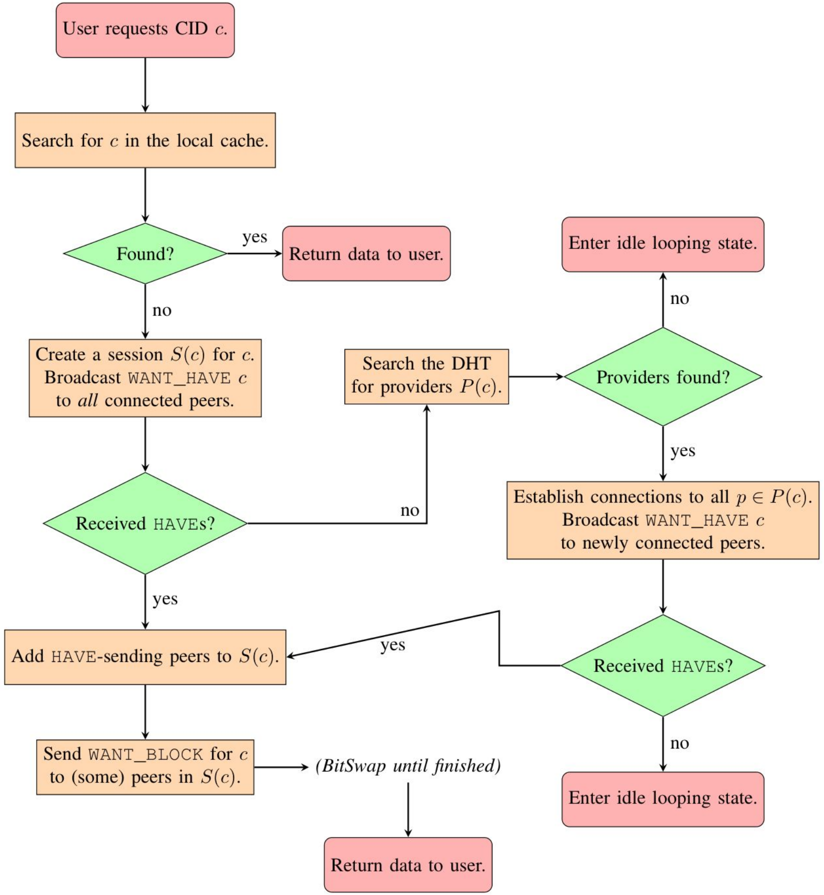
- Information on how these networks are used can be helpful to identify potential problems

- Observing behaviours on IPFS can be useful to optimize the ongoing development of IPFS and research of decentral storage applications in general

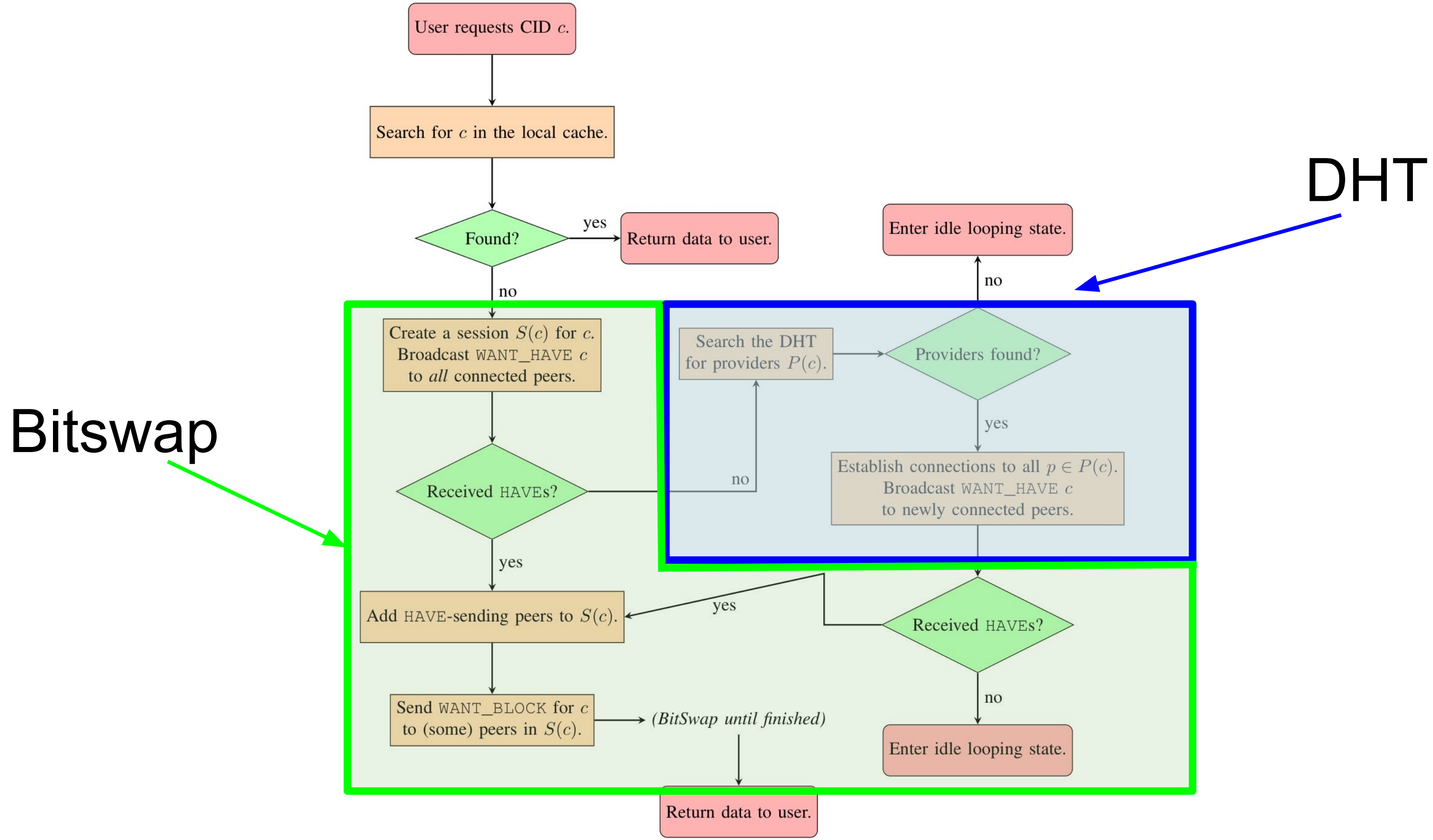
Recap: IPFS (**I**nter**P**lanetary **F**ile **S**ystem)

- **CID** (**C**ontent **Id**entifier): hash corresponding to some content
- **Peer**: Computer that is connected to the IPFS network
- **Provider** (of a CID): Peer that has this CID stored and can share
- **Gateway**: Web service that enables access to CIDs for everyone (instead of just IPFS peers)
- **DHT** (**D**istributed **H**ash **T**able): “Huge table that stores who has what data”
- **Bitswap**: message-based protocol to direct requesting and sending of CIDs between peers

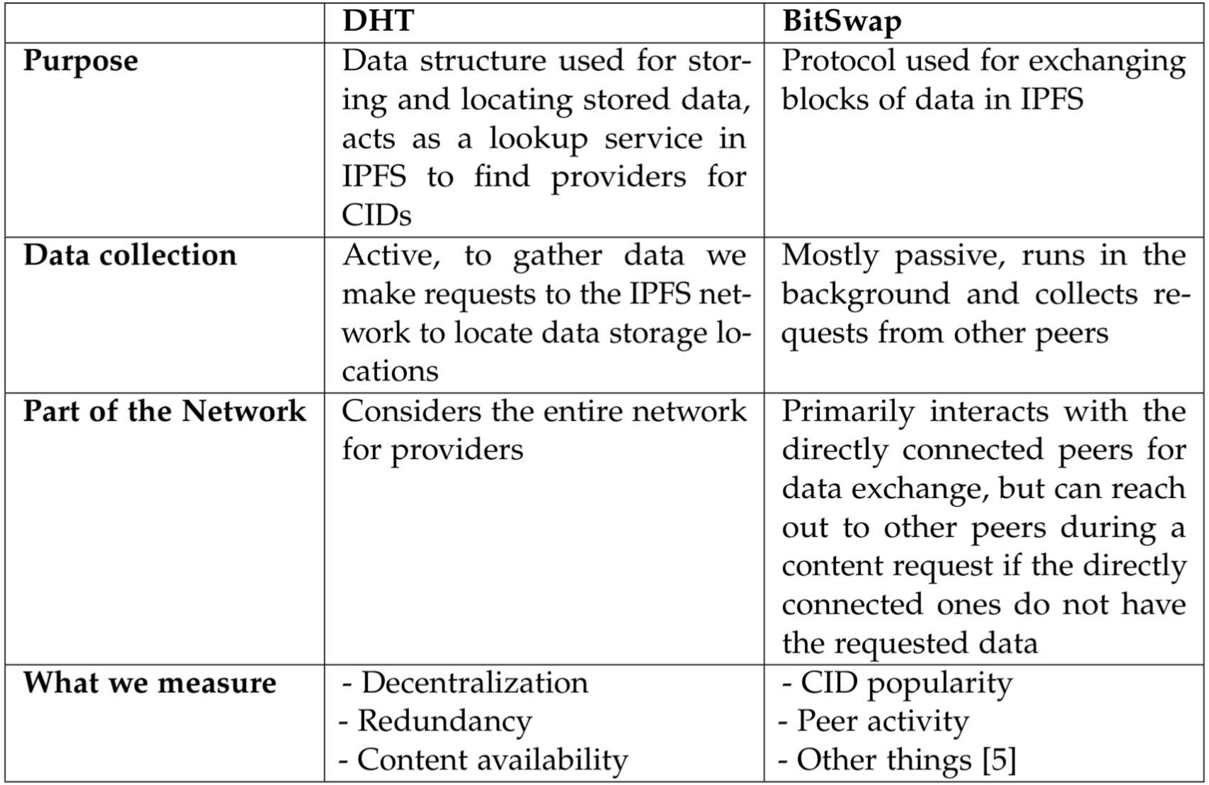
Recap: IPFS Content retrieval


User requests CID c.
Search for c in the local cache.
Found?
yes
Return data to user.
no
Create a session $S(c)$ for c. Broadcast WANT_HAVE c to *all* connected peers.
Received HAVEs?
no
Search the DHT for providers $P(c)$.
Providers found?
no
Enter idle looping state.
yes
Establish connections to all $p \in P(c)$. Broadcast WANT_HAVE c to newly connected peers.
Received HAVEs?
yes
no
Enter idle looping state.
yes
Add HAVE-sending peers to $S(c)$.
Send WANT_BLOCK for c to (some) peers in $S(c)$.
(BitSwap until finished)
Return data to user.

Recap: IPFS Content retrieval


User requests CID c.
Search for c in the local cache.
Found?
yes
Return data to user.
no
Enter idle looping state.
no
DHT
Bitswap
Create a session $S(c)$ for c. Broadcast WANT_HAVE c to *all* connected peers.
Search the DHT for providers $P(c)$.
Providers found?
yes
Establish connections to all $p \in P(c)$. Broadcast WANT_HAVE c to newly connected peers.
Received HAVEs?
no
yes
Add HAVE-sending peers to $S(c)$.
yes
Received HAVEs?
no
Send WANT_BLOCK for c to (some) peers in $S(c)$.
(BitSwap until finished)
Enter idle looping state.
Return data to user.

Recap: DHT vs. BitSwap



	DHT	**BitSwap**
Purpose	Data structure used for storing and locating stored data, acts as a lookup service in IPFS to find providers for CIDs	Protocol used for exchanging blocks of data in IPFS
Data collection	Active, to gather data we make requests to the IPFS network to locate data storage locations	Mostly passive, runs in the background and collects requests from other peers
Part of the Network	Considers the entire network for providers	Primarily interacts with the directly connected peers for data exchange, but can reach out to other peers during a content request if the directly connected ones do not have the requested data
What we measure	- Decentralization - Redundancy - Content availability	- CID popularity - Peer activity - Other things [5]

Table 2.1: Comparison between DHT and BitSwap in IPFS.

Three Goals of this work

1. Continue Public Gateway Dataset Analysis (based on previous work)

2. Measure accessibility, decentralization and redundancy of a popular website that uses IPFS (Wikipedia on IPFS)

3. Monitor Data requests to characterize IPFS workload (approach adapted from a paper)

Three Goals of this work

1. Continue Public Gateway Dataset Analysis (based on previous work)

2. Measure accessibility, decentralization and redundancy of a popular website that uses IPFS (Wikipedia on IPFS)

3. Monitor Data requests to characterize IPFS workload (approach adapted from a paper)

Wikipedia on IPFS

- Read-only mirror of Wikipedia
- Available for 8 different languages

How many peers participate?

How is the article availability on IPFS?

How distributed is it?

How much redundancy?

Wikipedia on IPFS

Strategy:

- For each language, every hour, sample 2.5% of articles
- Perform IPFS DHT queries to find any providers for these articles
- If providers are found, use another IPFS command to check if providers are reachable, i.e. if we can establish a connection to them

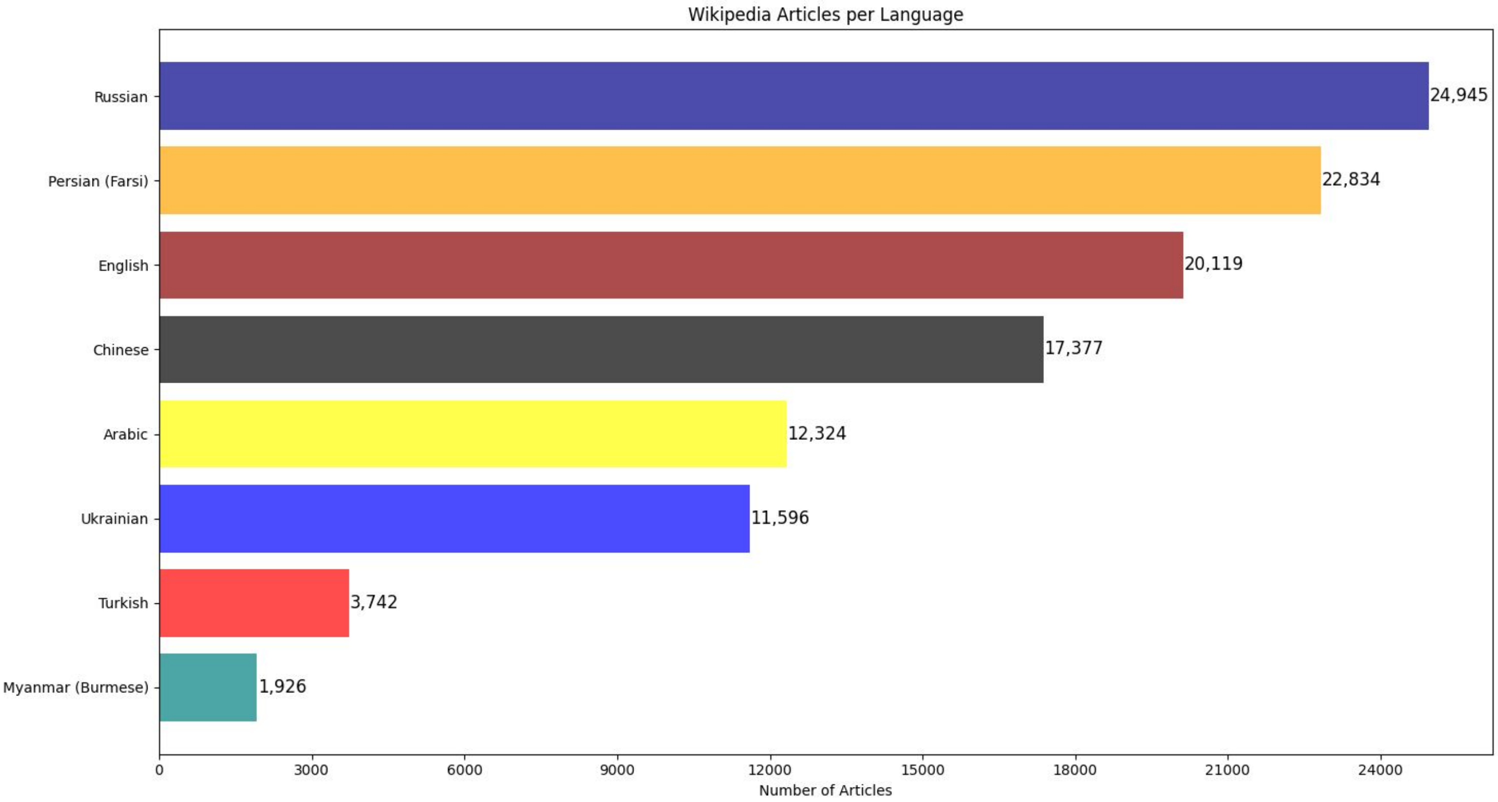
Wikipedia on IPFS

Articles found by a web scraper using a recursive search to get articles from the main page and one level below

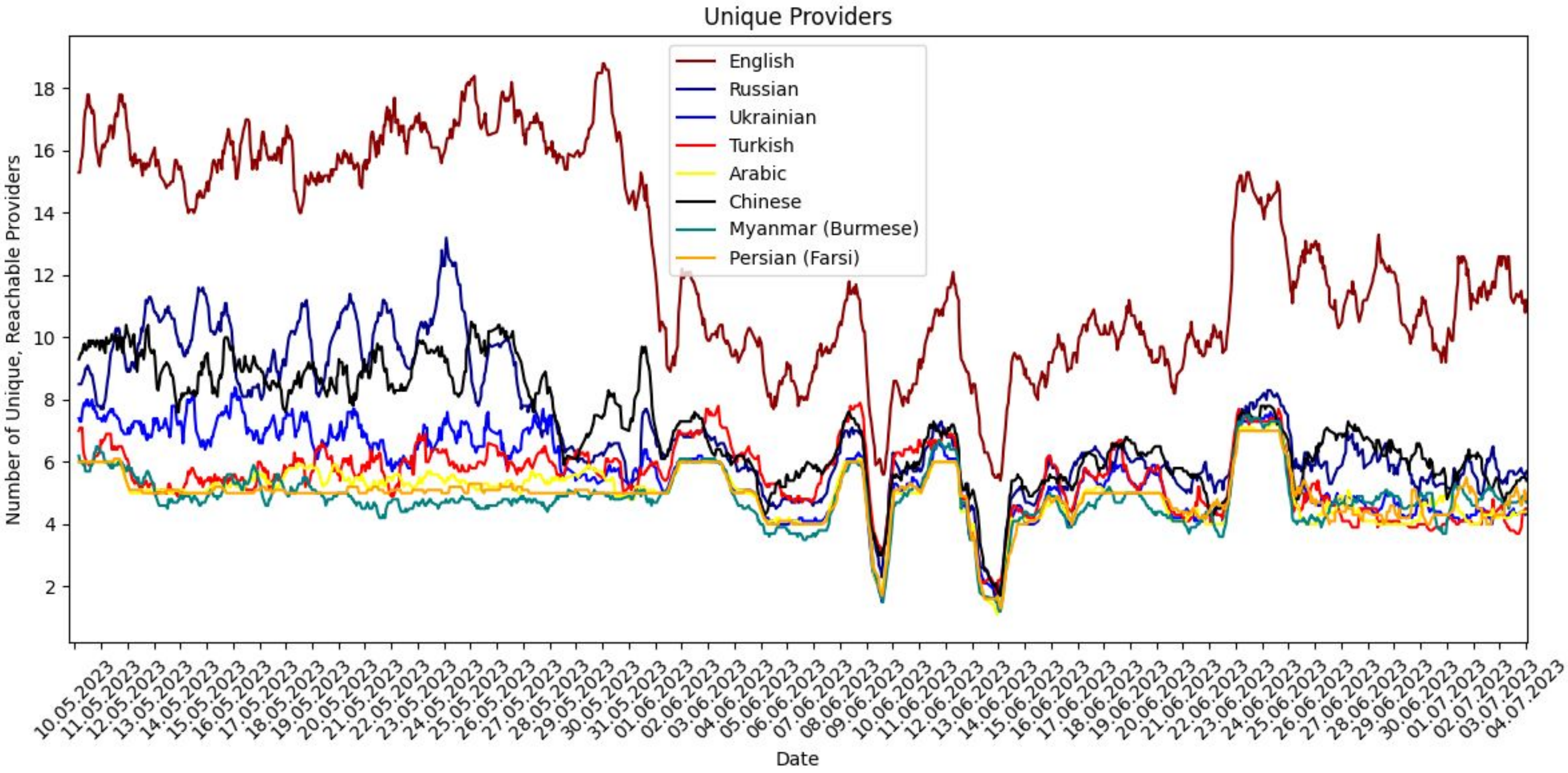
Strategy:

- For each language, every hour, sample 2.5% of articles
- Perform IPFS DHT queries to find any providers for these articles
- If providers are found, use another IPFS command to check if providers are reachable, i.e. if we can establish a connection to them

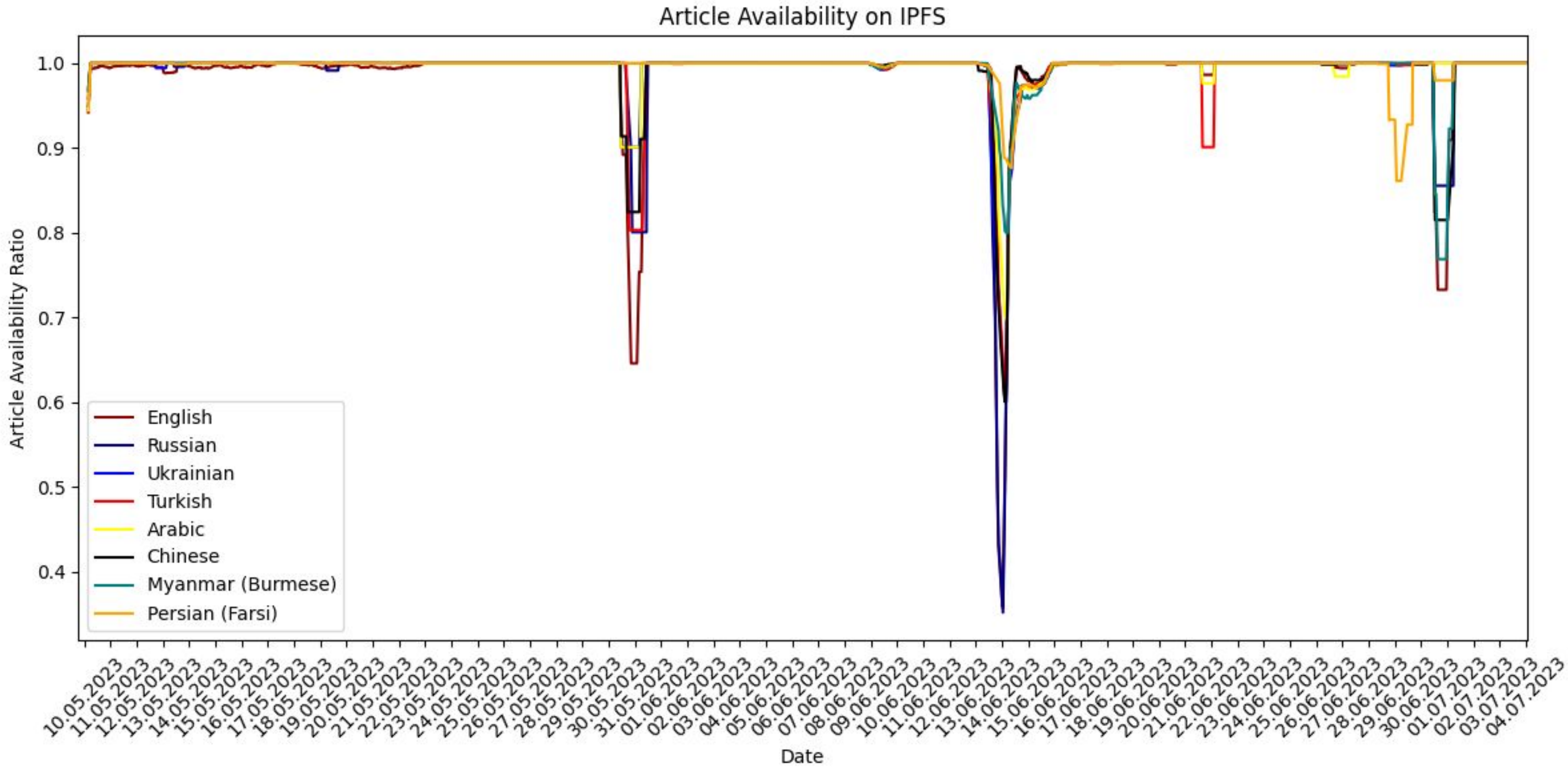
Wikipedia on IPFS



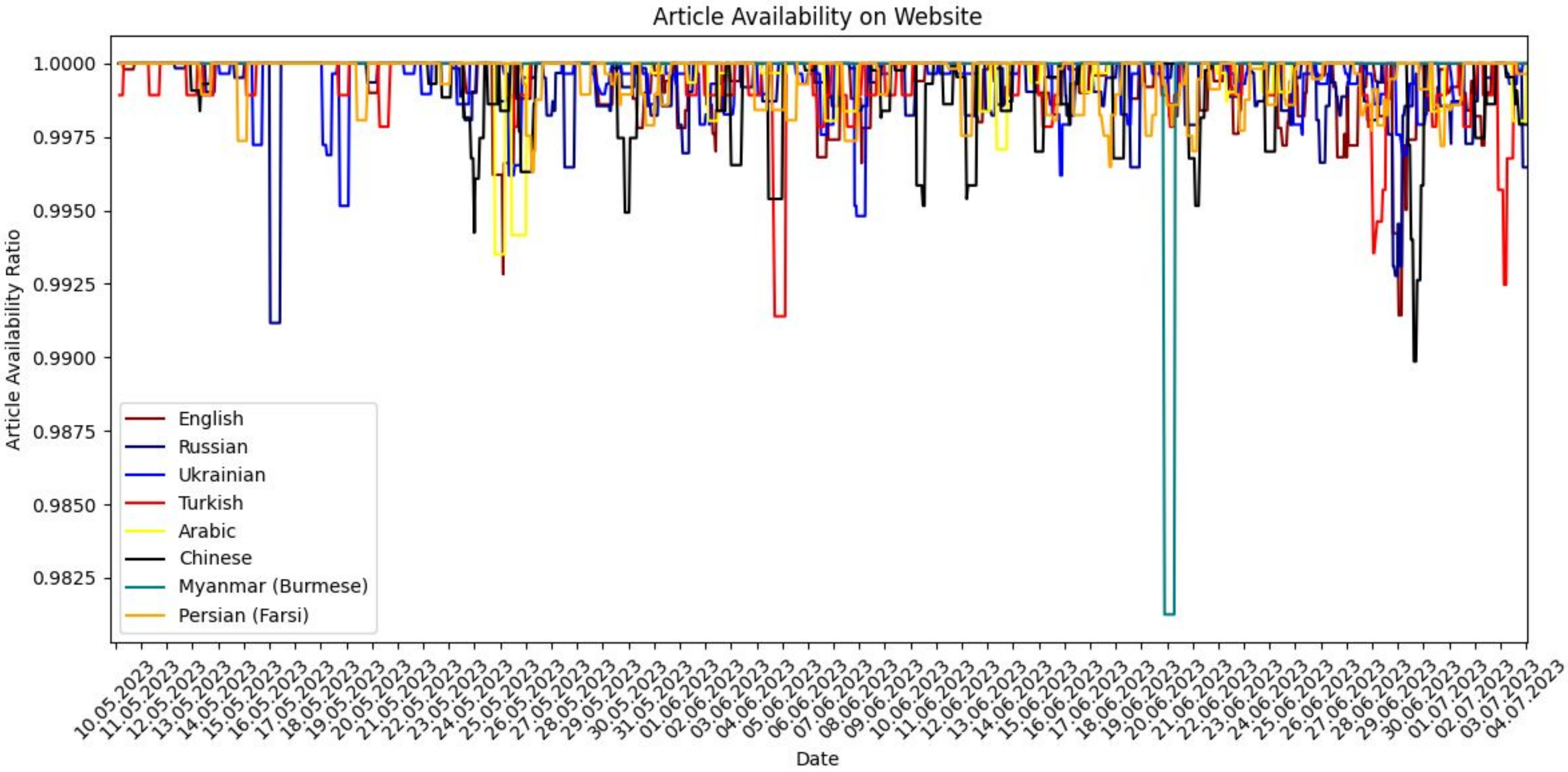
Wikipedia on IPFS - Number of participating peers



Wikipedia on IPFS - Article Availability on IPFS



Wikipedia on IPFS - Article Availability on Website



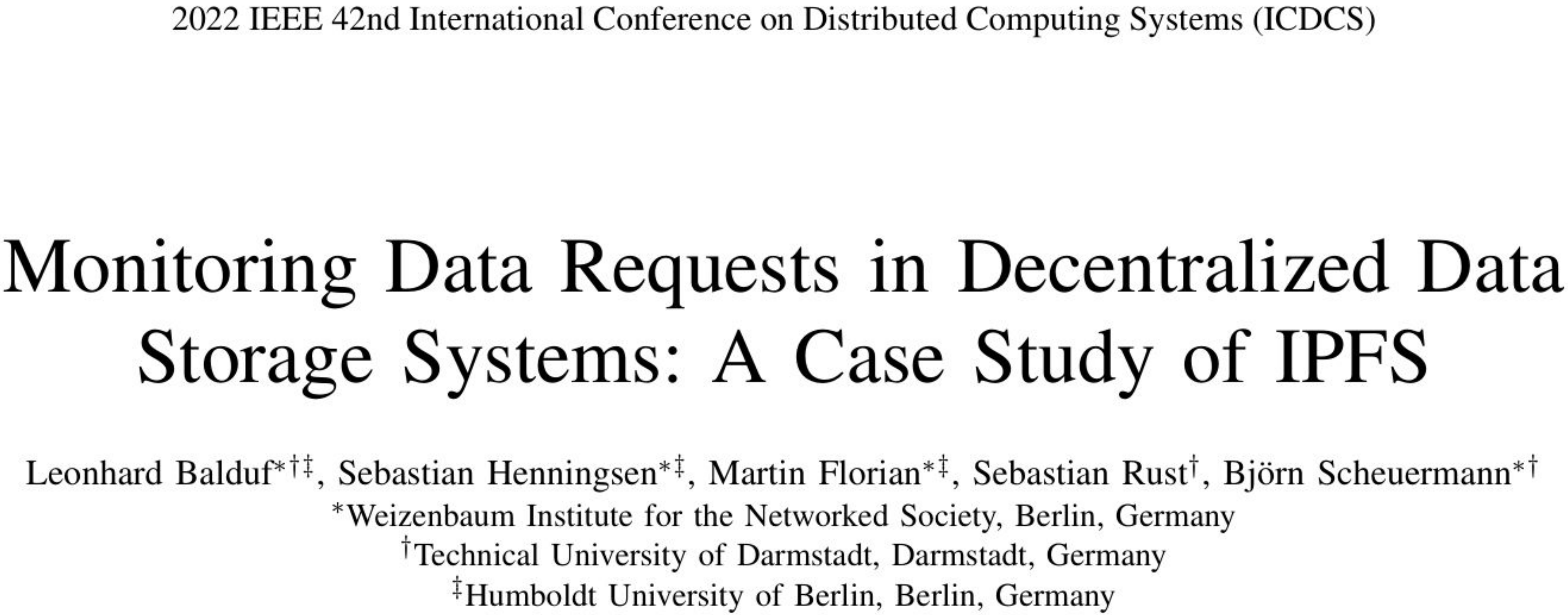
Three Goals of this work

1. Continue Public Gateway Dataset Analysis (based on previous work)

2. Measure accessibility, decentralization and redundancy of a popular website that uses IPFS (Wikipedia on IPFS)

3. Monitor Data requests to characterize IPFS workload (approach adapted from a paper, see next slide)

Bitswap Monitoring

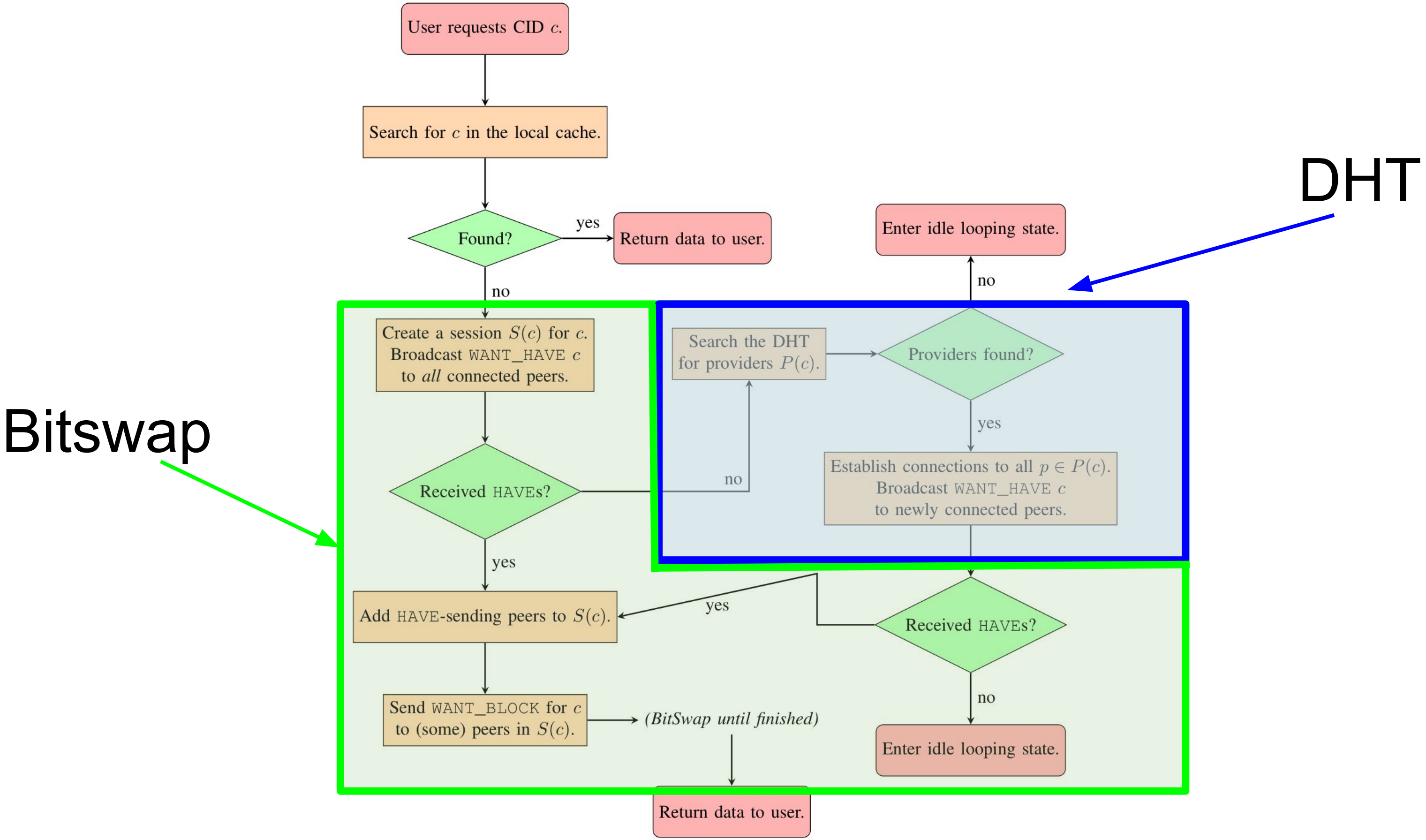


2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)

Monitoring Data Requests in Decentralized Data Storage Systems: A Case Study of IPFS

Leonhard Balduf[*†‡], Sebastian Henningsen[*‡], Martin Florian[*‡], Sebastian Rust[†], Björn Scheuermann[*†]
[*]Weizenbaum Institute for the Networked Society, Berlin, Germany
[†]Technical University of Darmstadt, Darmstadt, Germany
[‡]Humboldt University of Berlin, Berlin, Germany

Bitswap Monitoring


User requests CID c.
Search for c in the local cache.
Found?
yes
Return data to user.
no
Create a session $S(c)$ for c. Broadcast WANT_HAVE c to *all* connected peers.
Received HAVEs?
yes
Add HAVE-sending peers to $S(c)$.
Send WANT_BLOCK for c to (some) peers in $S(c)$.
(BitSwap until finished)
Return data to user.
no
Search the DHT for providers $P(c)$.
Providers found?
no
Enter idle looping state.
yes
Establish connections to all $p \in P(c)$. Broadcast WANT_HAVE c to newly connected peers.
Received HAVEs?
yes
no
Enter idle looping state.
DHT
Bitswap

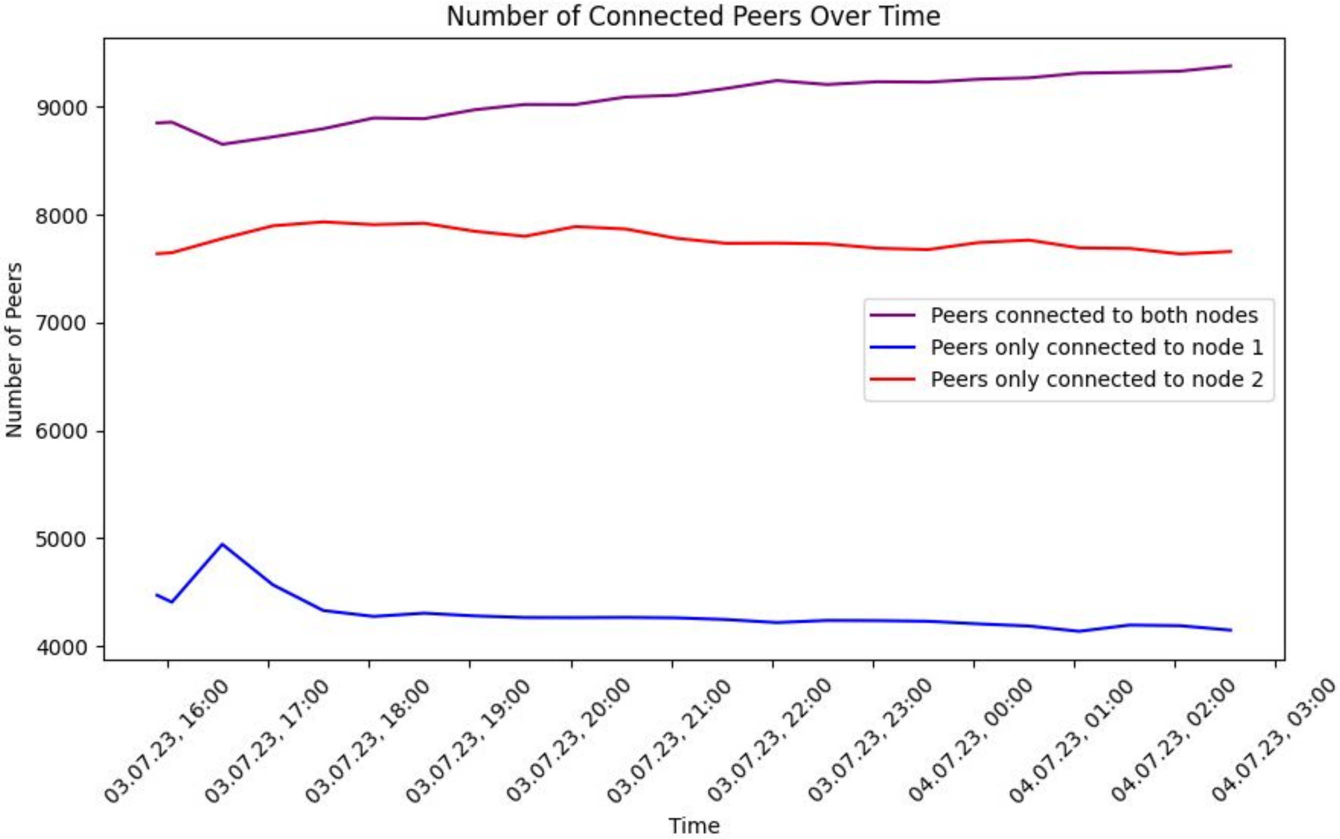
Bitswap Monitoring

- Docker setup: two passive IPFS nodes (“monitors”) collect and save all received BitSwap messages (in compressed JSON format)

- NAT / Firewall / Port forwarding absolutely crucial as we need to be connected to thousands of peers simultaneously (didn’t work at all with dedis-* VMs)

- After collecting the data, we can unify & deduplicate the collected data

Bitswap Monitoring - Connected Peers



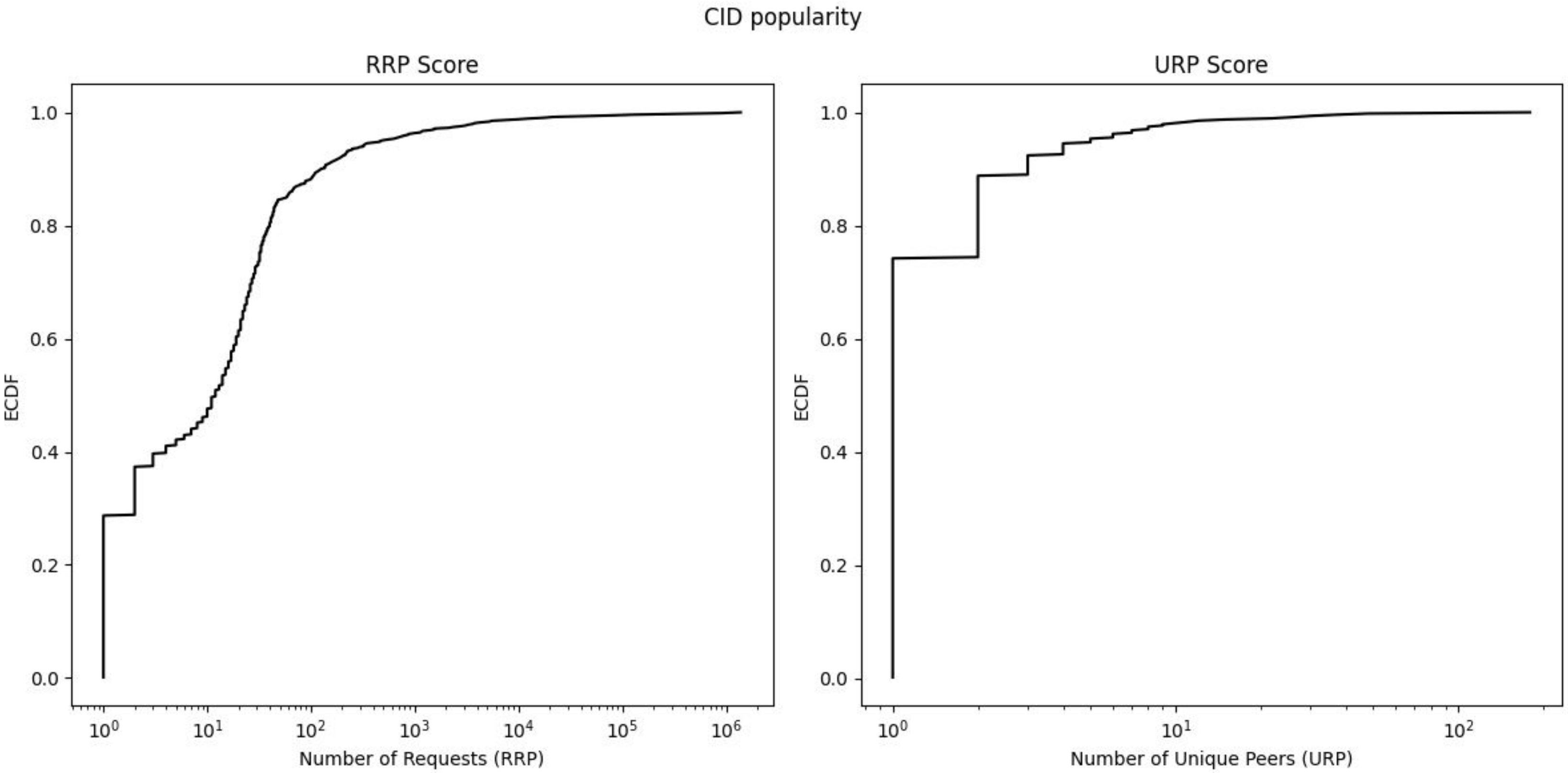
Bitswap Monitoring - CID Popularity

- **RRP** (**R**aw **R**equest **P**opularity): Total number of requests received for this particular CID

- **URP** (**U**nique **R**equest **P**opularity): Number of unique peers that requested this particular CID

Note: for the rest of the plots, only data collected from ~a single hour on 02.07.2023

Bitswap Monitoring - CID Popularity



Bitswap Monitoring - Identifying Gateway Peers

1. Create a block of random data, resulting in CID **c**
2. Add monitoring nodes as providers for **c**
3. Request **c** from gateway via HTTP
4. Wait for BitSwap messages that request **c**
5. ???
6. Profit (= find gateway peers)

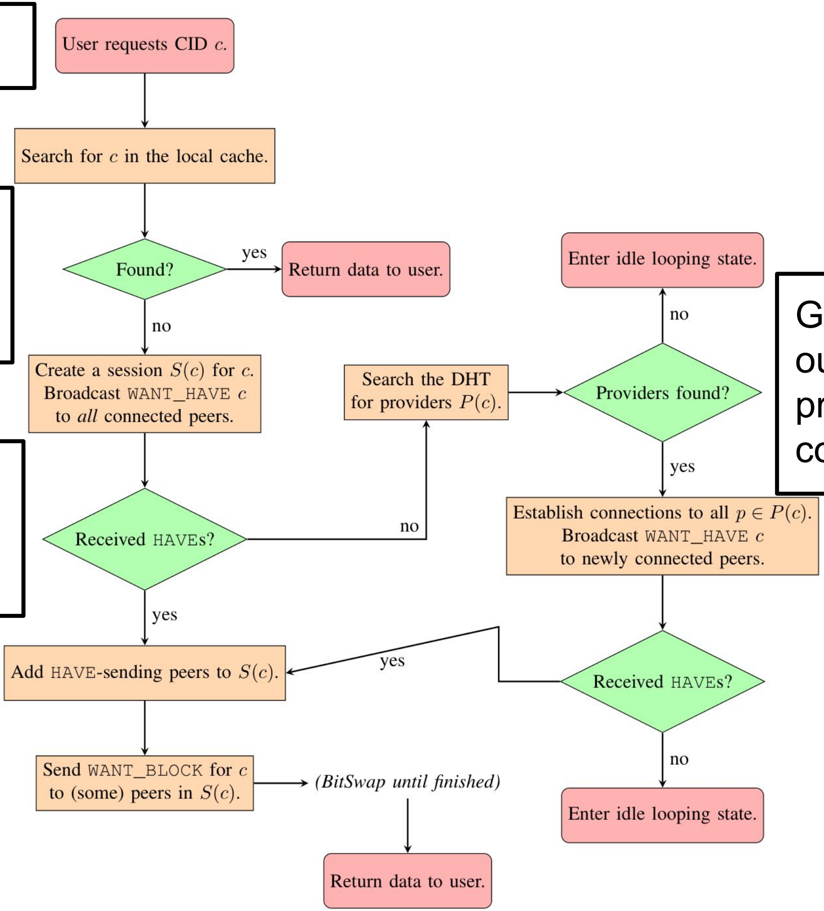
Why does this work?

Bitswap Monitoring - Identifying Gateway Peers

User = Gateway peer

Not in cache: **c** is random data that the gateway peer has never seen before

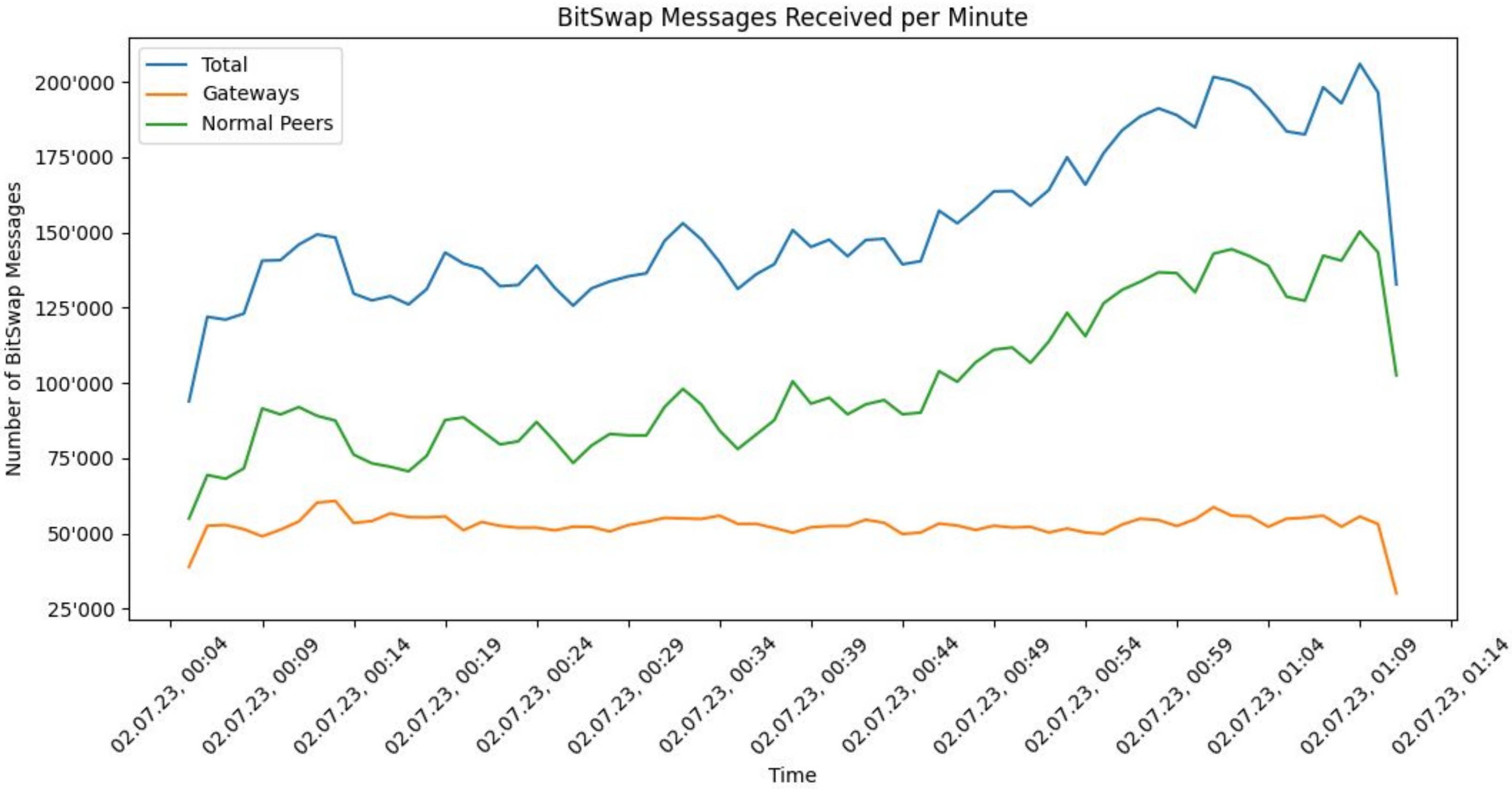
Therefore, gateway peer will seek **c** from other peers



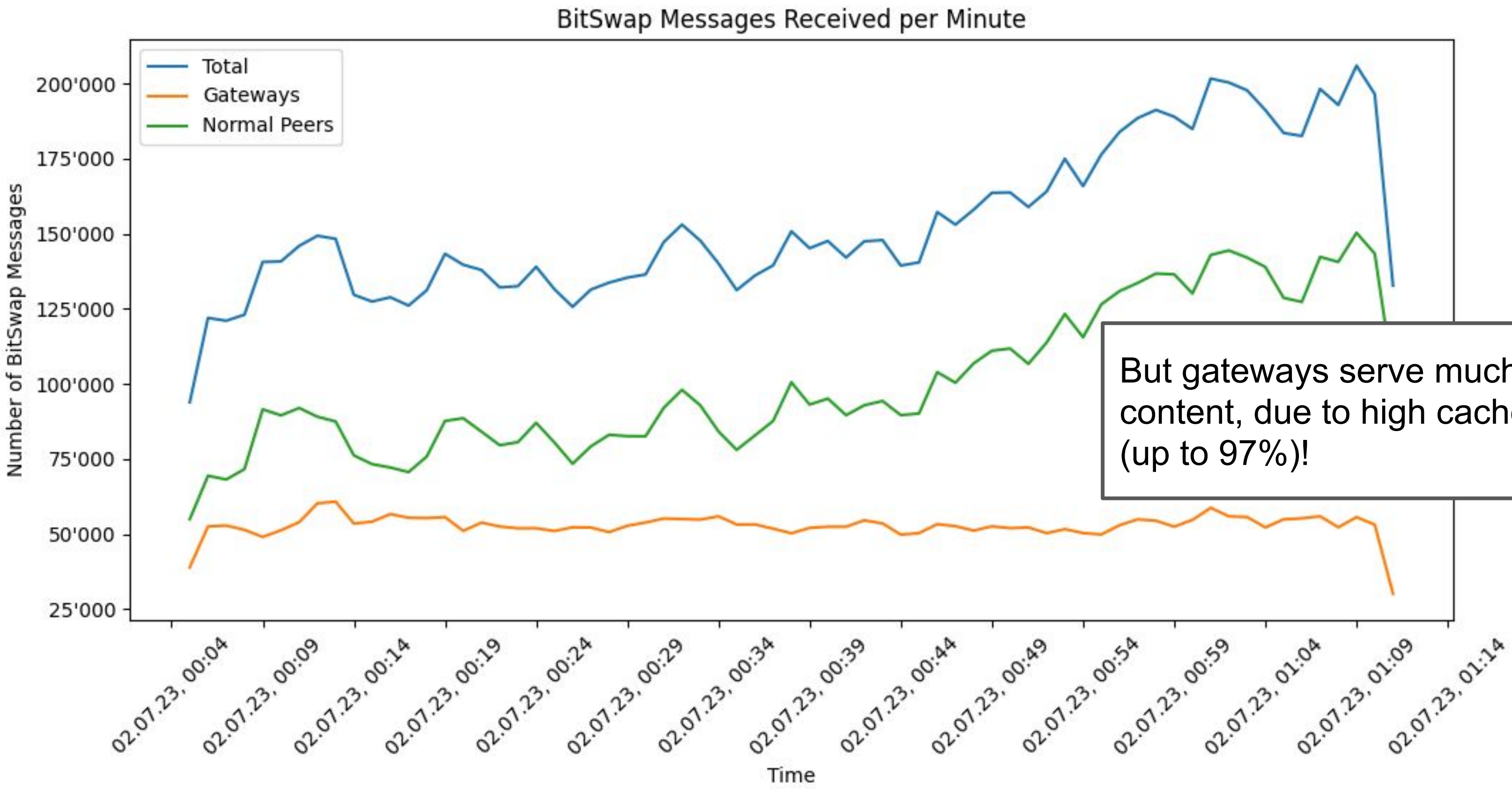
User requests CID c.
Search for c in the local cache.
Found?
yes
Return data to user.
no
Create a session $S(c)$ for c. Broadcast WANT_HAVE c to *all* connected peers.
Received HAVEs?
no
Search the DHT for providers $P(c)$.
Providers found?
no
Enter idle looping state.
yes
Establish connections to all $p \in P(c)$. Broadcast WANT_HAVE c to newly connected peers.
Received HAVEs?
yes
no
Enter idle looping state.
yes
Add HAVE-sending peers to $S(c)$.
Send WANT_BLOCK for c to (some) peers in $S(c)$.
(BitSwap until finished)
Return data to user.

Gateway peer will find our monitoring nodes as providers and request content

Bitswap Monitoring - Identifying Gateway Peers



Bitswap Monitoring - Identifying Gateway Peers



But gateways serve much more content, due to high cache hit ratio (up to 97%)!

ToDo’s

1. Upload code and documentation to Github (probably on Thursday)

2. Code adjustments made to the BitSwap setup repository need to be finalized and turned into a PR (probably next week)

Conclusions

- With the **DHT** we can learn about providers of content - but not who makes requests
- With **BitSwap** we can learn who makes requests - but not who serves the content
- IPFS is bad for privacy: with the **DHT**, we can see for any CID which user has it stored
- IPFS is bad for privacy: with **BitSwap**, if we are connected to a user, we can track all requests coming from this user and e.g. learn about their content preferences
- Using gateways can alleviate the second problem - especially if combined via Tor (there is also a public gateway available as a .onion hidden service)

Questions for future work:

- What if we have more than two monitors? How much more peers will we be connected to?
- What if the monitors are more active, e.g. providing or downloading data? Will they be connected to more/different peers?
- With a similar approach as shown before, can we also identify *restricted* Gateways?
  Example: Gateway for Wikipedia on IPFS website