Beyond Bitswap: Evaluation Plan

Motivation

This document proposes a set of test plans considered for a full benchmarking of file-sharing protocols in IPFS. The implementation of these test plans can be followed in the following repo: https://github.com/adlrocha/beyond-bitswap.

Testbed Configuration Parameters

We compile here a list of all the configurable parameters from our testbed, what will allow us to easily configure any of the test plans considered in this evaluation plan:

Group	**Parameters**
Use Case	INPUT_DATA={files, dir, random, custom} FILE SIZES (if INPUT_DATA==random) CFG_DATA (if INPUT_DATA==custom) FILES_DIRECTORY (if INPUT_DATA==files) DIRECTORY (if INPUT_DATA==dir)
Data Ingestion	CHUNKER DAG_LAYOUT
Network topology and conditions	N_NODES N_LEECHERS N_SEEDERS N_PASSIVE MAX_CONNECTION_PEERS CHURN_RATE
Node parameters	NODES_BANDWIDTH NODES_LATENCY NODES_JITTER MAX_CPU MAX_RAM
Havoc conditions	ATTACK_TYPE MALICIOUS_NODES (this is a work in progress)

Test plan 0: Baseline

Plan Parameters

We set here a set of parameters used as a baseline for the rest of the tests. Each new test plan may modify some of these metrics in order to deviate from the baseline and test a specific scenario.

PENDING TO IMPLEMENT IN THE TESTBED

- N_NODES: Number of nodes in the network.
- N_LEECHERS: Number of leechers in the network.
- N_SEEDERS: Number of seeders.
- N_PASSIVE: Number of passive nodes. They don't actively participate in the file-sharing protocol.
- MAX_CONNECTION_PEERS: Max connection to peers allowed. A convenient way of specifying these metrics is as a rate (PEER_CONECTIONS/TOTAL_PEERS).
- NODES_BANDWIDTH: Bandwidth of nodes' links.
- NODES_LATENCY: Latency of nodes' links.
- NODES_JITTER: Jitter of nodes.
- NODES_DATASTORE: Type of datastore to be used by the node (filesystem, in-memory, badger).
- CHURN_RATE: Percentage of nodes leaving the network.
- SEED_RATE: Rate of seeders storing the content.

Description

The purpose of this Test Plan is to get benchmark values for the operation of the corresponding file-transfer protocol (Bitswap, Graphsync, etc.) under different use cases. This will set a baseline operation of the protocol for the specific use cases.

Metrics

These are the general metrics that will be collected for every test plan. If additional metrics want to be collected for a test case it will be explicitly specified in its description:

PENDING TO IMPLEMENT IN THE TESTBED
IMPLEMENTED

- LATENCY: Total time required to fetch content from the network.
- THROUGHPUT: Total throughput experienced by leech nodes.
- BANDWIDTH_OVERHEAD: Bandwidth overhead of the file-sharing protocol compared to a plain TCP file-sharing.
- LATENCY_OVERHEAD: Difference between the time to fetch of the protocol compared to a TCP / FTP /Bittorrent download.
- NUM_MESSAGES: Number of messages exchanged in the file-sharing process.
    - MESSAGES_RCVD: Total messages received.
    - BLOCK_RCVD: Number of blocks received by leechers.
    - BLOCK_SENT: Number of blocks sent by seeders.
    - DUP_BLOCK_RCVD: Duplicate blocks received by leechers.
    - RATE_DATA_LOSS: Amount of blocks lost in transit.
- DATA_RCVD: Amount of data received by leechers.

    - CONTROL_DATA_RCVD: Percentage of data due to control messages. Good metric to understand if additional control messages can be included in the protocol as the channel is not being overloaded.
    - TOTAL_DATA_RCVD: Total “useful data” received.
    - DUP_DATA_RCVD: From the above data received, amount due to duplicate blocks.
- CONTENT_DISCOVERY_TIME: Time required to discover all blocks comprising the content to be requested from the network. This metric gives a good sense of the protocol's capability of finding content in the exchange network.
- TRANSMISSION_TIME: Actual time required to transmit the content from seeders to leechers. It signals if improvements in the actual data transmission are required.
- NUM_DHT: Number of times the file-sharing protocol needs to resort to the DHT for content discovery. The rationale behind this metric is that if the exchange interface being tested needs to resort a lot of times to the DHT, it means that no blocks are being found in the exchange network and new ways of augmenting content discovery in the exchange network needs to be devised.
- TIME TO FIRST BYTE / BLOCK / FILE: Time required to receive the beginning of a piece of data. Can be really useful to determine the quality of experience of users. This can be easily implemented by using ipfs.Get() without traversing the DAG. In this case, the IPFS node only requests the first block.

Narrative

- Warm up the network:
    - Spin up N_NODES in the network with N_SEEDERS, N_LEECHERS and N_PASSIVE.
- For each file size specified in FILE_SIZES:
    - Generate the files from seeders and add them to the network following DAG_STRUCTURE and CHUNKER strategies.
    - Notify parent CID of content to leechers (seeders and leechers start with an empty blockstore every run).
    - Start MAX_CONNECTION_PEERS with other random peers in the network.
    - Leechers download FILE_SIZES from seeders

Success Criteria

- We get the behavior of the file-sharing protocol under different use case scenarios for the baseline parameters.
- *<Additional fine-tune parameters goals may be added here according to the file-sharing protocol being tested>*

Test Case 0.1: Large Files

Description

This test case simulates the exchange of large files in the network. The content requested will be conformed by files > 100 GB.

Test Parameters

- FILE_SIZES: Array of file sizes for the content.
- DAG_STRUCTURE: Specific IPLD DAG used to store the content.
- CHUNKER: Chunker used to generate the content.

Test Case 0.2: Small Files

Description

This test case simulates the exchange of small files in the network. The content requested will be conformed by files < 1 GB.

Test Parameters

- FILE_SIZES: Array of file sizes for the content.
- DAG_STRUCTURE: Specific IPLD DAG used to store the content.
- CHUNKER: Chunker used to generate the content.

Test Case 0.3: Large Dataset

Description

This test case simulates the exchange of large datasets in the network. The content requested will be conformed by a DAG structure comprising several small files (< 1GB) and an overall size of the full structure > 100GB.

Test Parameters

- FILE_SIZES: Array of file sizes for the content.
- DAG_STRUCTURE: Specific IPLD DAG used to store the content.
- CHUNKER: Chunker used to generate the content.

Test Case 0.4: Video Stream

Description

This test case simulates the exchange of large chunks from a video stream.

Test Parameters

- VIDEO_QUALITY: Array of file sizes for the content.
- DAG_STRUCTURE: Specific IPLD DAG used to store the content.
- CHUNKER: Chunker used to generate the content.

Metrics

- QOS: Experiences quality of service.
    - Account for orderly delivery of blocks.
    - Account for data loss?

Test Case 0.5: Database operations

Description

This test case simulates the implementation of a database system over IPFS. We will model interactions with the network as write and read operations (see how Textile.io represent databases and schemas in the network).

Test Parameters

- SCHEMA_TYPES: Types of schemas used in the tests. We may think of parameters such as document size or schema depth to represent this metric.
- DATABASE_OPERATIONS: Operation to be performed in the scope of the test.
    - NUMBER_WRITES: Number of write operations.
    - NUMBER_READS: Number of read operations.
- DAG_STRUCTURE: Specific IPLD DAG used to store the content.
- CHUNKER: Chunker used to generate the content.

Test plan 1: Different network conditions

In this test plan we deviate certain parameters from the baseline test plan to account for different network conditions. In all test cases described below, all the test cases 0.1-0.6 are run including these deviations from baseline.

Success Criteria

- We get the behavior of the file-sharing protocol under different network scenarios for every use case.
- *<Additional fine-tune parameters goals may be added here according to the file-sharing protocol being tested>*

Test Case 1.1: Small networks

Description

This test case simulates file-sharing over small networks.

Test Parameters

It specifies the specific parameters to be modified from the baseline for the test:

- N_NODES: Number of nodes in the network. (max: 10)
- N_LEECHERS: Number of leechers in the network.
- N_SEEDERS: Number of seeders.
- N_PASSIVE: Number of passive nodes. They don't actively participate in the file-sharing protocol.

Test Case 1.2: Large networks

Description

This test case simulates file-sharing over large networks.

Test Parameters

It specifies the specific parameters to be modified from the baseline for the test:

- N_NODES: Number of nodes in the network. (min: 100)
- N_LEECHERS: Number of leechers in the network.
- N_SEEDERS: Number of seeders.
- N_PASSIVE: Number of passive nodes. They don't actively participate in the file-sharing protocol.

Test Case 1.3: High churn

Description

This test case simulates file-sharing over a network with high churn.

Test Parameters

It specifies the specific parameters to be modified from the baseline for the test:

- CHURN_RATE: Churn rate of the network (min: 0.85)
- N_NODES: Number of nodes in the network.
- N_LEECHERS: Number of leechers in the network.
- N_SEEDERS: Number of seeders.
- N_PASSIVE: Number of passive nodes. They don't actively participate in the file-sharing protocol.

Test Case 1.4: Network within networks

Description

This test case simulates different network topologies and connection patterns between nodes.

Test Parameters

It specifies the specific parameters to be modified from the baseline for the test:

- MAX_CONNECTION_PEERS: Max connection to peers allowed.
- N_NODES: Number of nodes in the network.
- N_LEECHERS: Number of leechers in the network.
- N_SEEDERS: Number of seeders.
- N_PASSIVE: Number of passive nodes. They don't actively participate in the file-sharing protocol.

Test Case 1.5: Heterogeneous networks

Description

This test case simulates a network with very heterogeneous nodes.

Test Parameters

It specifies the specific parameters to be modified from the baseline for the test. These metrics will be specified as a tuple [rate of nodes with that metric, specific value]. Thus, we can set NODES_BANDWIDTH=[(0.2, 130), (0.8, 100)] where we are saying 20% of the nodes will have a BANDWIDTH of 130MB while the 80% have 100MB.

- NODES_BANDWIDTH: Bandwidth of nodes' links.
- NODES_LATENCY: Latency of nodes' links.
- NODES_JITTER: Jitter of nodes.
- NODES_MAX_CPU: Max CPU allowed for nodes.
- NODES_MAX_RAM: Max RAM allowed for nodes.

Test Case 1.6: Protocol coexistence - Hot experiment

Description

This test case simulates a network where nodes are running different versions of a file-sharing protocol. As part of the test we should include the capability of attaching our testbed to an existing IPFS deployment to test our protocols in the wild.

Test Parameters

- COEXISTING_PROTOCOLS: A list of tuples of the percentage of nodes in the network running a specific version of the protocol (e.g. (0.2, v1), (0.8, v2)).
- IS_HOT_DELOYMENT: If true it means we are running the test connected to a real IPFS deployment.

Test plan 2: Different Request Patterns

In this test plan we introduce specific behaviors or interaction patterns in the network, either at a data structure level, discovery or transmission. In all test cases described below, all the test cases 0.1-0.6 are run including these deviations from baseline.

Success Criteria

- We get the behavior of the file-sharing protocol under different network scenarios for every use case.
- *<Additional fine-tune parameters goals may be added here according to the file-sharing protocol being tested>*

Test Case 2.1: Regularly accessed data

Description

This test case simulates the request of regularly accessed content. It can be simulated forcing seeders to store the content to be requested.

Test Parameters

- STORAGE_RATE: Rate of seeders storing the content (min: 0.8).

[Bonus] Test plan 3: Security and Privacy

In this test plan we will evaluate the security and privacy of file-sharing protocols under different attacks. In this case there is no need for running the test plan for every use case.

Success Criteria

- We understand the impact of different attacks to the file-sharing protcol.

- *<Additional fine-tune parameters goals may be added here according to the file-sharing protocol being tested>*

Test Case 3.1: Block flood attack

Description

Malicious nodes flood honest nodes with useless blocks.

Test Case 3.2: Forged HAVE messages

Description

A node sends HAVE messages when he doesn't have the block.

Test Case 3.3: Eclipse HAVE attack

Description

Force timeouts to WANT-BLOCK requests from peers. Send HAVE messages to appear as an honest provider, wait for the WANT-BLOCK and timeout.

Test Case 3.4: Connection flood

Description

Malicious nodes flood honest peers with new connections.

Stage of implementation in the testbed

This table specifies the stage of implementation and priority of the different use cases over the file-sharing testbed.

Test Case	Priority	Is Implemented?
Baseline Metrics and Parameters	High	❌
Test Case 0.1: Large Files	High / Mid / Low	❌ / ✔️
Test Case 0.2: Small Files		❌
Test Case 0.3: Large Dataset		❌

Test Case 0.4: Video Stream		❌
Test Case 0.5: Database operations		❌
Test Case 1.1: Small networks		❌
Test Case 1.2: Large networks		❌
Test Case 1.3: High churn		❌
Test Case 1.4: Network within networks		❌
Test Case 1.4: Network within networks		❌
Test Case 1.5: Heterogeneous networks		❌
Test Case 1.6: Protocol coexistence - Hot experiment		❌