COSMIC - Coding for Optimized Sending in Multimedia InterPlanetary Communication

Sandra Zimmermann*, Paul Schwenteck*, Christian Vielhaus*, Juan A. Cabrera*, Frank H. P. Fitzek*¶
* Deutsche Telekom Chair of Communication Networks, TU Dresden, Germany
¶ Centre for Tactile Internet with Human-in-the-Loop (CeTI)
E-mails: {firstname.lastname}@tu-dresden.de

***Abstract*—A popular technique for enhancing Content Delivery Networks (CDNs) is leveraging the opportunities provided by Peer to Peer (P2P) networks, of which the Interplanetary File System (IPFS) is a widely used one. While IPFS tends to perform well only in homogeneous networks, performance tends to quickly deteriorate when the channel parameters of individual peers differ, which is the case in real-world scenarios. A crucial technology for facilitating the flexible distribution of data requests is Random Linear Network Coding (RLNC). The RLNC technology is leveraged by the protocol Storing coded Packets In-advance For IPFS (SPIFI).**

Based on SPIFI, we design and implement the Coding for Optimized Sending in Multimedia InterPlanetary Communication (COSMIC) protocol, which uses an adaptive data request scheme to fit all imaginable network configurations. We compare the performance of the three protocols, IPFS, SPIFI, and COSMIC, in tailored and random test cases. To quantify performance, we use the transmission time and transmitted overhead as our metrics. Regarding the transmission time, COSMIC beats the other protocols by at least 30%. The transmitted overhead of COSMIC is slightly worse than SPIFI, but still beats IPFS by a wide margin. For comparison, we also calculate the minimal theoretically possible transmission time in a distribution network with random parameters as a lower bound.

***Index Terms*—Distributed Storage, Multimedia Content Distribution, Multisource Transmission, IPFS, RLNC**

I. Introduction

Efficient content distribution is growing in importance in how we use the internet in the future. For example, video consumption is increasing, as is the quality and, therefore, size of individual files and the quality of experience requirements of users [1]. One possibility is to expand the classic server-client architecture by using the upload capacity of hosts forming a Peer to Peer (P2P) network.

A very prominent example of a P2P network is Interplanetary File System (IPFS) [2]. In IPFS, files are broken down into small blocks, which are requested individually from the available network participants. The requesting method works well in homogeneous networks where all participants have the same channel parameters, and no attention needs to be paid to network characteristics. As soon as network parameters show high variance, blocks are likely requested from the wrong peer node, resulting in higher transmission times. Furthermore, blocks may be requested twice, creating unnecessary overhead. Both issues diminish the experience for the user and the network. Moreover, adding more peer nodes, i.e., potential senders, may increase transmission time in some cases because of the aforementioned issues. Optimally distributing the requests in such inhomogeneous networks to all network participants requires extensive knowledge of the network parameters, which cannot be assumed in practice [3].

A fundamental requirement for being able to distribute requests flexibly is to avoid waiting for specific blocks. One way of achieving this is to use linear codes, where linear combinations of blocks are stored instead of individual blocks [4]. In order to be able to restore the original data, a certain number of coded blocks must be available at the receiver. The increased flexibility of requests comes at the cost of additional decoding time. Due to its ability to generate any amount of redundancy at any time, Random Linear Network Coding (RLNC) [5] as a coding scheme has proven effective. The protocol that combines IPFS with RLNC is the Storing coded Packtes In-advance For IPFS (SPIFI) protocol [6]. Instead of specific blocks, only a number of coded blocks is requested from each peer. The peer nodes are required to have enough coded blocks available.

We also design Coding for Optimized Sending in Multimedia InterPlanetary Communication (COSMIC), which is a refinement of the SPIFI protocol. Instead of sending only one request to each peer node, the requests are now distributed over several individual messages, making a more flexible distribution to all nodes possible. Using a hardware testbed, we first examine the performance of the IPFS, SPIFI, and COSMIC protocols in tailored test networks. The test networks enable us to prove that the existing protocols only achieve good results with homogeneous network parameters. To assess the generalizability of the results, we then examine all protocols in 1500 randomly generated networks. Based on the nodes' channel parameters, we calculate the best block distribution and use the theoretical minimum transmission time as a lower bound. COSMIC shows on average 40% less transmission time and 19% less overhead than IPFS.

The rest of the paper is structured as follows. Section II gives an overview of related work regarding coding in distributed storage networks and optimal load distribution. Section III presents the applied protocols and theoretical computations, while Section IV explains our testbed and experimental setup. Both are brought together in Section V, where we present the results of our experiments before summarizing the paper in Section VI.

II. Related Work

A. Request Strategies Improvements in File Systems

The request strategy in IPFS has undergone many improvements over the years [7]. However, few scientific papers are trying to improve the request strategy of IPFS. Therefore, we also consider request strategy improvements for BitTorrent [8], whose request strategy is similar to that of IPFS. In [9], the authors present a survey of performance studies of BitTorrent. One suggested mechanism is to select nodes by proximity to decrease the transmission time and increase network utilization. The authors in [10] and [11] propose schemes that build an intelligent overlay network by selecting nodes that are close by in the underlying network. These schemes employ techniques like synthetic network coordinates and a probabilistic flooding algorithm for topology construction and maintenance. The general findings are that location proximity decreases transmission times and increases network utilization in homogeneous and heterogeneous networks.

B. RLNC in Distributed Storage Systems

Linear Codes such as Reed-Solomon codes have been utilized to reduce storage expenses in distributed storage systems [12]. Linux Redundant Array of Independent Disks (RAID) systems also employ Reed-Solomon codes, allowing them to withstand the failure of two hard drives [13], [14]. Erasure codes are another effective solution for reducing node failure impacts and safeguarding data [15]. We suggest using RLNC to create redundancy against node failure impacts. RLNC offers distinct advantages over other erasure codes due to its simple code structure: every linear combination of source blocks is considered a valid code word. The decodability in a distributed file system depends only on the number of blocks received, not on the specific combination of blocks. Another noteworthy benefit of RLNC is the ability to perform recoding. Any entity possessing at least two coded blocks can create a new coded block by combining them linearly, thus eliminating the need to gather and decode all the data upfront to introduce more redundancy. An implementation of a network-coding-based distributed file system is presented in [16], demonstrating the performance of RLNC regarding throughput and storage costs. Also, [17] proposes a network coding-based data storage to minimize data retrieval time while maintaining high resilience.

C. Divisible Load Theory

The term *divisible load theory* is of great importance in the field of distributed computing [18], e.g. cloud computing [19]. Here, multiple computers are connected to a distributed computing network. A load that requires enormous time to process is divided into fractions. Each computer receives a fraction for processing, depending on the processing speed and the transmission time to the processor. The optimization problem is to balance the load between processors so that the computation is completed in the shortest possible time. This problem can be mapped to a distributed file system, where the processing time for a processor is mapped to the channel bandwidth, and the transmission time is mapped to the end-to-end delay for the communication nodes. With RLNC, each node can respond with coded blocks that are all useful for decoding. So we only need to distribute the number of blocks each node should send and not specify which node should send which specific block.

In [20], the authors have theoretically described the divisible load theory for a peer-to-peer network. However, they only considered the nodes' uploading bandwidth, not their transmission delay. The resulting formula for the optimized request distribution and, thus, the minimum achievable transmission time does not apply to our scenario.

III. Methodology

A. Transmission and Request Protocols

1) State of the Art: IPFS: In IPFS, a file is split into equal-sized blocks, which are stored individually and can be addressed with their individual Content Identifier (CID). The CID of the file, i.e. the root CID, is the hash of the list of CIDs of the individual child blocks. IPFS uses two types of requests: a want-have request and a want-block request. The want-have request queries the availability of a block with a specific CID. The want-block request prompts a node to transmit a specified block. To download a file in IPFS, a want-have request for the root CID is first sent to all connected nodes. Nodes that respond affirmatively are classified as session nodes that may receive want-block requests in the future. Session nodes are assumed to have all or a subset of the child blocks, because of the availability of the root block. After a node's first affirmative response to the want-have request, the client sends a want-block request for the root block to that node. After receiving the root block, the client knows all CIDs of the child blocks. Each child block is requested and transmitted using the following procedure: A want-block request is first sent to one selected random session node. The probability of selecting a particular node for the want-block request increases with the number of blocks received in the session from the respective node so far. At the same time, all other session nodes receive a want-have request. Depending on the timing of the responses by nodes, IPFS proceeds in two different ways. If the child block is first received from the randomly selected node, the request scheme for that particular block ends. Otherwise, an affirmative response to the want-have request arrives first by some other node, which triggers a want-block request to that node. Additional responses to the want-have requests do not trigger more want-block requests. Hence, in IPFS, two nodes may attempt to send the same child block simultaneously. Consequently, IPFS may cause unnecessary data transmissions as overhead.

Overall, the request strategy of IPFS depends on the transmission timings and may cause overhead in the form of unnecessary data transmissions. In addition, the node that answers a want-have request the fastest always receives a want-block request. For that reason, IPFS favors nodes with low end-to-end delays regardless of the available bandwidths. The bias towards selecting nodes by end-to-end delays may increase the file transmission times and lacks a load balancing mechanism.

2) State of the Art: SPIFI: SPIFI was designed to address some of the limitations of IPFS requests. First, SPIFI adds RLNC to the IPFS storage organisation. Instead of storing uncoded blocks like in IPFS, linear combinations with random coefficients of blocks are stored as coded blocks. All child blocks of a root block are used for the linearly encoding. With RLNC, a group of coded blocks can be decoded if a sufficient number of linearly independent coded blocks are available. Secondly, the request strategy of IPFS was adapted. The objective of the SPIFI request strategy is to keep the overhead as low as possible while at the same time simplifying the practical implementation of multisource transmissions without adding coordination effort to the protocol. In SPIFI, the root block acquisition and the selection of session nodes are performed identically as in IPFS. Child blocks, however, are now distributed as coded blocks instead of uncoded blocks. A request is sent to each session node for $\lceil C/N \rceil$ coded blocks, where C is the number of child blocks and N is the number of session nodes. The nodes generate more coded blocks in advance, when there is little load and can answer any request directly with coded blocks from their storage. The request strategy generates an overhead of at most N blocks. Unlike IPFS, the data is only available at the application layer after C independent linear combinations have been received and the user data has been decoded.

While the design target of SPIFI is to minimize overhead, timing issues can produce additional requests in heterogeneous networks. A node with a high delay may answer the root block request late, while the root block was already received and requests for $\lceil C/(N-1) \rceil$ coded blocks have been sent to other nodes. The latecomer node then receives a request for $\lceil C/N \rceil$ blocks. At this point in time, the client has requested $\lceil C/N \rceil$ too many child blocks.

To summarize, SPIFI reduces the transmission overhead by balancing the load equally among nodes. However, in heterogeneous networks, the transmission time may suffer from slow nodes.

3) Our Contribution: COSMIC: Based on the SPIFI protocol, we have designed the COSMIC protocol to exploit flexible request strategies further leveraging RLNC and adapt to heterogeneous and dynamic networks. Instead of requesting data from each node only once as in SPIFI, the requests are spread over several iterations in COSMIC. Initially, a fixed number of k blocks is requested from all N nodes. The number k should be large enough to provide the necessary time to survey the network and small enough to allow enough further requests to make adjustments. Afterward, we track how many blocks each node has sent and how many requests are still pending. Once the number of blocks expected from a node falls below the threshold value r, a new request for r blocks is sent to this node.

In the worst case, all nodes fall below the threshold simultaneously, and only a single block is missing. Then, the client requests r blocks from each node, so that the amount of requested blocks totals to $(2r-1)N$. Hence, the maximum overhead equals $(2r-1)N-1$ blocks, which can be significantly higher than the maximum overhead of SPIFI depending on C, N, and r. For large r, the total overhead increases. A small r may increase the number of requests, which increases the total transmission time. A suitable choice of parameters makes it possible to balance both aspects. Our preliminary observations have shown that this is the case for $k=5$ and $r=3$.

4) Our Contribution: tinyC: tinyC is a variation of the COSMIC protocol, in which the parameters are set $k=2$ and $r=1$. The choice of parameters tries to minimize the overhead. This means that initially, a request is sent to each node for two blocks. The next request for a single block is only sent when all former requests are fulfilled. The possible overhead is, therefore, a maximum of $N-1$ blocks. Since the request strategy creates idle times at the nodes, we expect longer transmission times compared to COSMIC with other parameters.

TABLE I: Summary of main notations

Symbol	Description
N	Number of nodes
n	Node index
G	Size of payload data [bytes]
C	Number of child blocks
g_n	Payload data transmitted by node n [bytes]
g_n^*	Payload data transmitted by node n in the optimal case [bytes]
t	Overall transmission time [seconds]
t_n	Transmission time of node n [seconds]
t_{min}	Lower bound of transmission time
b_n	Bandwidth of node n's transmission channel [MBit/s]
d_n	Delay of node n's transmission channel [seconds]
r	COSMIC threshold [blocks]
k	COSMIC initial request [blocks]

B. Lower Bound Transmission Time

Multisource transmission can reduce transmission times and, therefore, application delays significantly. However, a suboptimal request distribution induces the problem of waiting for slower nodes, thereby increasing the total transmission time and reducing the user's experience quality. When the protocol-induced overhead is disregarded, the transmission time t_n of a single node n is determined only by its channel parameters, bandwidth b_n and delay d_n, and the amount of data to be transmitted g_n; this is illustrated in Eq. 1. Table I shows the notation used. The total transmission time is determined as the maximum of the transmission times of all nodes. The objective of an optimal request distribution is to minimize the total transmission time. At the same time, the total payload transmitted by all nodes is equal to the original data size. These requirements are shown in Eq. 2 and Eq. 3. The maximum of the individual transmission times t_n is minimal if all transmission times are equal. With the equality of all transmission times, we get N equations for the optimal amount of transmitted data g_n^* of the form Eq. 4. Together with Eq. 3, we obtain a linear system of $N+1$ equations with $N+1$ variables. The solution of the linear equation system results in a closed-form expression for the lower bound of the transmission time t_{min} (Eq. 6).The time t_{min} is the theoretically possible minimum transmission time if we only consider the channel bandwidth and delay as transmission

limitations. More transmission limitations exist, such as computing delays, which are not considered for the theoretical transmission time. Therefore, t_{min} is the theoretical lower bound and not achievable in practice. Our request strategies are designed so that every node can send the optimal number of blocks g_n^*. Furthermore, we base our theoretical calculations on the assumption that we can optimize the overhead and transmission simultaneously (cf. Eq. 2, Eq. 3). However, in practical terms, to minimize the transmission time, creating overhead can not be avoided because we deal with integer block counts and we have to account for network dynamics.

$$t_n = \frac{g_n}{b_n} + d_n \tag{1}$$

$$t = \max_{\forall n}\{t_n\} \to \min \tag{2}$$

$$G = \sum_{n=1}^{N} g_n \tag{3}$$

$$t_{min} = t_n = \frac{g_n^*}{b_n} + d_n \Rightarrow g_n^* = (t_{min} - d_n)\, b_n \tag{4}$$

$$Eq.\ 4\ in\ Eq.\ 3:$$

$$G = \sum_{n=1}^{N} (t_{min} - d_n)\, b_n = t_{min} \sum_{n=1}^{N} b_n - \sum_{n=1}^{N} b_n d_n \tag{5}$$

$$t_{min} = \frac{G + \sum_{i=1}^{N} b_i d_i}{\sum_{i=1}^{N} b_i} \tag{6}$$

C. Performance Metrics

To measure the performance of the four request strategies we use two metrics, for which lower values indicate better performance. The transmission time $t > t_{min}$ is measured as the time duration between starting a request and receiving the last block. We do not consider the time to store the file on the disc, as this depends on the consumer's hardware. As a second metric, we use the transmission overhead. The overhead is the ratio between the additional blocks that have been sent but are no longer needed to decode the file and the number of payload blocks.

IV. Measurement Setup

A. Network Topology

The investigated network topology comprises one consumer and five provider nodes, all using the IPFS framework to communicate with each other. All five provider nodes store the complete data. We implemented IPFS, SPIFI, and COSMIC for our test bed NET Playground [21]. The NET Playground is a multi-functional network testbed comprising interconnected Odroid-XU4 devices orchestrated by a control unit. The nodes are Odroid devices. We set the Odroids' different delay and bandwidth values with the Linux traffic control (tc) tool.

Each measurement is a single request from the consumer to obtain a 10 MB file. We capture the transmission time of the requests by logging the request and response timestamps. In addition, we count the number of blocks sent from each node to calculate the overhead.

B. Tailored Test Cases

In this section, we outline three different test cases tailored to explore the performance of distributed file systems under specific network conditions characterized by varying bandwidth and delay configurations. These test cases create diverse channel conditions while ensuring consistency in the lower bound of the transmission time ($t_{min} = 1.125\,s$) across all scenarios.

The first test case, the EQUAL scenario, establishes a homogeneous network environment where all nodes possess the same bandwidth and delay characteristics. The scenario emulates a network with uniform channel conditions, facilitating equitable data transmissions. The EQUAL scenario provides a benchmark for comparison against scenarios with diverse channel properties. As SPIFI was designed for homogeneous networks, we expect it to perform well in comparison to IPFS and COSMIC.

The FLANK scenario introduces variance in bandwidth and delay configurations among nodes. Nodes 1 and 2 exhibit extreme values, with very low bandwidth and high delay, while nodes 4 and 5 showcase contrasting characteristics, featuring high bandwidth and low delay. Node 3 is between these extremes, maintaining a symmetrical distribution of channel properties. This configuration mimics scenarios where network conditions vary gradually, allowing for a systematic assessment of file system performance. Since SPIFI has to wait for the slowest nodes to send their blocks, we expect it to perform very poorly compared to IPFS.

The SERVER test case simulates a scenario akin to a client-server architecture. So far, we have designed the channel characteristics of the nodes so that nodes with a high bandwidth have a low delay and vice versa. In the server test case, nodes 1 to 4 have a very low bandwidth and delay. These four nodes represent local nodes with short distances and low data transfer rates, akin to neighboring nodes with limited communication capacities. In contrast, node 5 symbolizes a remote server with high bandwidth rates but higher delay. This configuration challenges IPFS to efficiently manage data transmission between nodes, as seen in previous work [6]. Since IPFS distributes its requests to the neighboring nodes and the server randomly, there are cases where IPFS does not use the server effectively and thus achieves poorer time results. The bandwidth and delay values of the nodes in the different test cases is shown in Table II.

TABLE II: Tailored Test Cases

Test Case	Bandwidth $\{b_n\}$ in MBit/s	Delay $\{d_n\}$ in ms
EQUAL	{15, 15, 15, 15, 15}	{27, 27, 27, 27, 27}
FLANK	{2, 2, 14, 28, 28}	{50, 50, 27, 5, 5}
SERVER	{4, 4, 4, 4, 60}	{5, 5, 5, 5, 50}

C. Randomized Test Case

Through the tailored test cases, we aim to comprehensively evaluate the performance of IPFS, SPIFI, and COSMIC under certain network conditions, providing valuable insights into their adaptability and efficiency. However, the goal of

COSMIC is to design a request strategy that works well for every network configuration. Therefore, we carry out a series of measurements in which the channel conditions are chosen at random. We define the nodes' bandwidth capacity $b_n \in [1, 30]$ MBit/s and the delay $d_n \in [5, 50]$ ms. We test all request strategies in 1500 random network configurations.

V. Evaluation

A. Tailored Test Cases

We evaluate the relative transmission time of the four protocols IPFS, SPIFI, COSMIC, and tinyC in the tailored test cases outlined in Section IV. This test sequence uses well-designed network parameters to investigate under which circumstances state-of-the-art protocols perform exceptionally well or exceptionally poorly. For this purpose, we have constructed three networks that all share the same lower bound of the transmission time t_{min}. Hence, the measured values can be compared quantitatively.

Table III shows the mean relative transmission time t/t_{min} and the mean overhead of all request strategies in the tailored test networks. Regarding the relative transmission time, all strategies perform well in the EQUAL test case. SPIFI's performance worsens with an increasing variance of network conditions, which is expected by design. The relative transmission time starts at low values of approximately 1.6 in the EQUAL test case and reaches a value of 4.3 in the SERVER case. IPFS is slower than the other strategies in the EQUAL case and has a medium relative transmission time of 3.2. The performance is slightly better in the FLANK test case, as the nodes with high bandwidths also have low delays and are more frequently queried. Only in the SERVER case IPFS's relative transmission time also rises to more than 4, as the nodes with low delays are queried more often, resulting in longer waiting times for individual blocks. COSMIC and tinyC have outstanding relative transmission times below 2 close to the lower bound in the EQUAL and FLANK test case and just slightly above in the SERVER test case. tinyC is slightly worse in comparison to COSMIC. In the SERVER test case, a more aggressive request strategy with a larger COSMIC threshold r can produce even lower transmission times at the expense of increased overhead.

TABLE III: Results for Tailored Test Cases

Protocol	Rel. trans. time t/t_{min}			Overhead in %		
	EQUAL	FLANK	SERVER	EQUAL	FLANK	SERVER
IPFS	3.2	2.9	4.1	78	64	88
SPIFI	1.6	3.6	4.3	12	46	24
COSMIC	1.7	1.5	2.1	33	36	25
tinyC	1.8	1.7	2.6	12	13	12

The smallest overhead of 12% is reached by tinyC and SPIFI in the EQUAL test case. COSMIC has a larger overhead of 33%. Both COSMIC and tinyC produce a similar overhead across all test cases. SPIFI incurs more overhead when individual nodes have a high delay, and their want-have replies miss the first round of requests. This affects two nodes in the FLANK case, so the extra overhead is higher than in the SERVER case, where only one node has a higher delay. IPFS has the highest overhead of all protocols at around 70%, which increases even further in the SERVER case. Overall, the correlation between overhead and transmission time is very high with IPFS, as duplicated blocks delay the payload transmission.

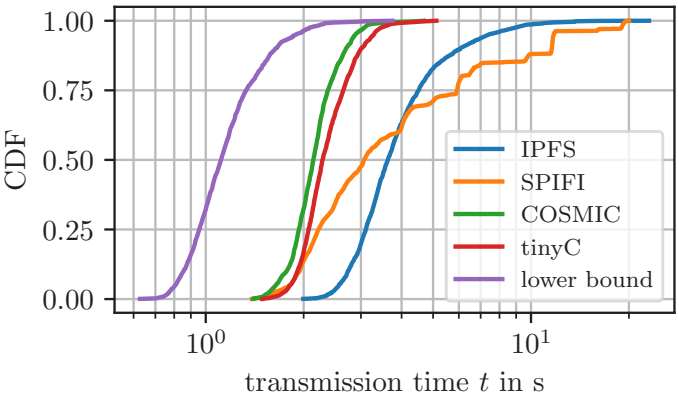

Fig. 1: CDF over transmission time for all request strategies over 1500 random networks and CDF of the theoretical lower bound of the transmission time for comparison.

B. Transmission Time in Random Networks

In the first series of tests, we investigated how the request strategies behave in various special cases. All strategies perform better in networks with a low variance of network conditions. For SPIFI, this matches the design goal. However, the other three strategies are designed for general-purpose applications and should work well in general cases. To verify the fulfillment of the design goal, we tested all strategies in randomly generated networks.

Fig. 1 shows the CDF of the transmission time in 1500 random networks for each request strategy and, for comparison, the lower bound of the transmission time. The curve of IPFS runs parallel to the lower bound of the transmission time but is shifted to the right. While the lower bound varies between 0.6s and 4s, IPFS has a value range of 2s to 24s. COSMIC and tinyC have a much steeper curve. Initially, the two protocols still have a clear distance to the lower bound time of 1.6s, but they tend towards the lower bound for long transmission times. The 95% percentile of COSMIC is 3s. tinyC has an equivalent curve, a constant distance to COSMIC, and a 95% percentile of around 3.5s. SPIFI even manages to outperform tinyC at very low transmission times but has a very flat curve, i.e., a considerable variance of possible results, and is worse than IPFS in 30% of cases. SPIFI has its 95% percentile at 12s and reaches 100% only at 20s. IPFS lags behind the other protocols in good cases and has a transmission time of 2s at best. The 95% percentile is at 9s and thereby well ahead of SPIFI, with the 100% value slightly behind SPIFI's value at 24s. However, the transmission times of both protocols are far from the lower bound in all cases.

C. Overhead in Random Networks

In addition to the transmission time, we also observe the overhead caused by random networks. The overhead is shown

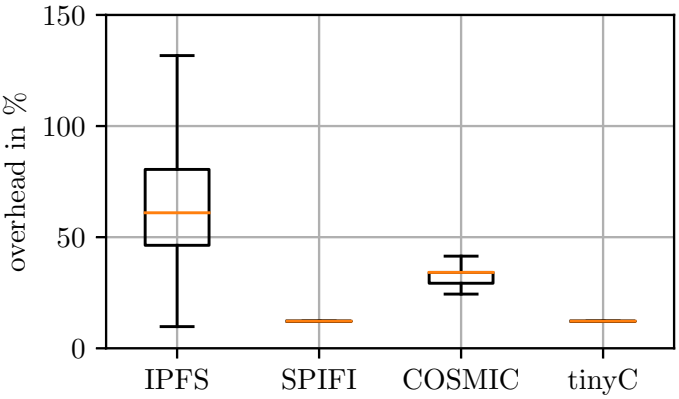

Fig. 2: Mean overhead for all request strategies averaged over 1500 random topologies.

in Fig. 2. SPIFI and tinyC both have a constant value of only 12%, which corresponds to the design specification for both strategies. Both protocols are insensitive to the variance of network conditions. COSMIC has a slightly higher overhead between 25% and 45%. The protocol averages an overhead of 38%. IPFS produces results with the largest variance of values. In the best case, it is even better than SPIFI at around 10% but reaches maximum values of 140%. On average, IPFS has an overhead of 60% and is significantly worse than the other three protocols.

VI. Conclusion

In this work, we analyzed popular protocols for multisource content distribution. Based on the state-of-the-art technologies IPFS and SPIFI, we developed the adaptive request strategy COSMIC. We showed that the established protocols only work well in homogeneous networks and have a wide range of possible results with increasing variance of transmission parameters. We tested all protocols in randomly generated networks with variable transmission bandwidth and delay and calculated the theoretical lower bound of the transmission time for comparison. The protocols we developed show a 40% lower transmission time with 20% less overhead compared to IPFS. tinyC minimizes the generated overhead while COSMIC generates slightly more overhead with a marginally lower transmission time. Both protocols consistently stay very close to the lower bound of the transmission time and are, therefore, preferable to the state-of-the-art protocols.

In order to refine the knowledge gained in this study and test the capabilities of adaptive request strategies, we want to investigate networks in which the transmission parameters can change during transmission in future experiments. A further generalization is the possibility of admitting network participants who do not have complete data but can only contribute parts of the data.

Acknowledgment

Funded in part by the German Research Foundation (DFG, Deutsche Forschungsgemeinschaft) as part of Germany's Excellence Strategy – EXC 2050/1 – Project ID 390696704 – Cluster of Excellence "Centre for Tactile Internet with Human-in-the-Loop" (CeTI) of Technische Universität Dresden and the Federal Ministry of Education and Research (BMBF) of Germany in the programme of "Souverän. Digital. Vernetzt." – Joint project 6G-life (project 16KISK001K) and within the DAAD School of Embedded and Composite AI (SECAI) (project 57616814).

References

[1] "2024 global internet phenomena report," https://www.sandvine.com/global-internet-phenomena-report-2024, accessed: 2024-03-11.
[2] J. Benet, "Ipfs - content addressed, versioned, p2p file system," 2014.
[3] Y. Liu, L. Liu, Z. Yan, and J. Hu, "The algorithm of multi-source to multi-sink traffic scheduling," in *2021 17th International Conference on Mobility, Sensing and Networking (MSN)*, 2021, pp. 638–644.
[4] R. Ahlswede, Ning Cai, S. . R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, July 2000.
[5] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4413–4430, 2006.
[6] P. Schwenteck, S. Zimmermann, C. von Lengerke, J. A. Cabrera, and F. H. Fitzek, "Revisiting MARS: storing coded packets In-Advance for IPFS," in *2024 IEEE International Conference on Communications (ICC): Communication Software and Multimedia Symposium (IEEE ICC'24 - CSM Symposium)*, Denver, USA, Jun. 2024.
[7] "New improvements to ipfs bitswap for faster container image distribution," https://blog.ipfs.tech/2020-02-14-improved-bitswap-for-container-distribution/, accessed: 2024-03-11.
[8] B. Cohen, "Incentives build robustness in bittorrent," *Workshop on Economics of PeertoPeer systems*, vol. 6, 06 2003.
[9] R. L. Xia and J. K. Muppala, "A survey of bittorrent performance," *IEEE Communications Surveys & Tutorials*, vol. 12, no. 2, pp. 140–158, 2010.
[10] A. Qureshi, "Exploring proximity based peer selection in bittorrent-like protocol," *MIT*, vol. 6, p. 824, 2004.
[11] L. Zhang, J. K. Muppala, and W. Tu, "Exploiting proximity in cooperative download of large files in peer-to-peer networks," in *Second International Conference on Internet and Web Applications and Services (ICIW'07)*, 2007, pp. 1–1.
[12] N. Drucker, S. Gueron, and V. Krasnov, "The comeback of reed solomon codes," in *2018 IEEE 25th Symposium on Computer Arithmetic (ARITH)*, 2018, pp. 125–129.
[13] P. Corbett *et al.*, "Row-diagonal parity for double disk failure correction," in *Proceedings of the 3rd USENIX Conference on File and Storage Technologies*. San Francisco, CA, 2004, pp. 1–14.
[14] H. P. Anvin, "The mathematics of raid-6," 2007.
[15] X. Xie *et al.*, "Az-code: An efficient availability zone level erasure code to provide high fault tolerance in cloud storage systems," in *2019 35th Symposium on Mass Storage Systems and Technologies (MSST)*, 2019, pp. 230–243.
[16] Y. Hu *et al.*, "Ncfs: On the practicality and extensibility of a network-coding-based distributed file system," in *2011 International Symposium on Networking Coding*, 2011, pp. 1–6.
[17] A. Marandi, H. Sehat, D. E. Lucani, S. Mousavifar, and R. H. Jacobsen, "Network coding-based data storage and retrieval for kademlia," in *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, 2021, pp. 1–7.
[18] B. Veeravalli, D. Ghose, and T. Robertazzi, "Divisible load theory: A new paradigm for load scheduling in distributed systems," *Cluster Computing*, vol. 6, pp. 7–17, 01 2003.
[19] M. Abdullah and M. Othman, "Cost-based multi-qos job scheduling using divisible load theory in cloud computing," *Procedia Computer Science*, vol. 18, pp. 928–935, 2013, 2013 International Conference on Computational Science. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050913004018
[20] T. Wu, M. Li, and M. Qi, "Optimizing peer selection in bittorrent networks with genetic algorithms," *Future Generation Computer Systems*, vol. 26, no. 8, pp. 1151–1156, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X1000110X
[21] P. Schwenteck, S. Zimmermann, C. von Lengerke, C. Scheunert, and F. H. Fitzek, "NET playground - a Multi-Functional testbed for distributed systems," in *2024 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom) (IEEE BlackSeaCom 2024)*, Tbilisi, Georgia, Jun. 2024, p. 5.98.