In-depth analysis of the IDA-Gossip protocol

Kadir Korkmaz, Joachim Bruneau-Queyreix, Stéphane Delbruel, Sonia Ben Mokhtar, Laurent Réveillère

In-depth analysis of the *IDA-Gossip* protocol

Kadir Korkmaz
Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI
Talence, France
kadir.korkmaz@u-bordeaux.fr

Joachim Bruneau-Queyreix
Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI
Talence, France
joachim.bruneau-queyreix@u-bordeaux.fr

Stéphane Delbruel
Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI
Talence, France
stephane.delbruel@u-bordeaux.fr

Sonia Ben Mokhtar
Liris - CNRS
Lyon, France
sonia.ben-mokhtar@liris.cnrs.fr

Laurent Réveillère
Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI
Talence, France
laurent.reveillere@u-bordeaux.fr

***Abstract*—Gossip-based dissemination protocols are important building blocks of large-scale distributed systems as they may impact both the systems' efficiency and fault tolerance. There exist many flavors of gossip dissemination protocols. *IDA-Gossip* is one of the gossip dissemination protocols proposed in the context of blockchains to efficiently disseminate large messages. It relies on multi-chunk gossip dissemination, erasure coding, and Merkle hash trees. However, despite its claimed efficiency, there is no in-depth analysis of this protocol to understand its behavior under different conditions (e.g., with injected faults). In this work, we evaluate the behavior of *IDA-Gossip* by relying on extensive experiments and simulations. Specifically, we evaluate *IDA-Gossip* both in terms of performance and resilience to faults by varying its configuration parameters and the number of faulty nodes, respectively. This study results in several takeaways. First, *IDA-Gossip* provides excellent dissemination latency compared to classic gossip. Second, it provides excellent coverage even with 40 percent of faulty nodes in the system. Finally, the use of erasure coding provides an important advantage to *IDA-Gossip* compared to classic multi-chunk gossip dissemination protocols.**

I. Introduction

With advances in hardware and software technologies, the size of distributed systems is increasing. Today, there are many distributed systems that contain several thousand nodes distributed around the world and connected via the Internet [1]–[3]. An important characteristic of these systems is a high churn rate which is a high change in the set of participating nodes due to joins, graceful leaves, and failures [4]. Due to the large size and high churn rate, communication primitives such as reliable broadcast are not considered practical for today's large-scale distributed systems. In this context, gossip dissemination protocols fill an important gap by providing an efficient group communication primitive with probabilistic guarantees.

Gossip dissemination protocols are originally proposed in the context of database replication [5], and later on, they are adopted as probabilistic alternatives of reliable broadcast primitives. In a distributed system that depends on gossip dissemination, messages are spread in a manner very similar to epidemic diffusion: upon receiving a new message, a node becomes infected, and it infects fanout nodes by sending the message. Fanout is an important parameter of gossip dissemination protocol that controls the redundancy of dissemination. A message infects a node only once. Gossip dissemination protocols are considered practical alternatives to broadcast primitives for large-scale distributed systems because each node communicates with a few other nodes to disseminate a message.

979-8-3503-9730-7/22/$31.00 ©2022 European Union

Although gossip dissemination mechanisms are practical, they are not the most efficient ones because each message needs to be forwarded fanout times to provide a probabilistic dissemination guarantee. The higher fanout values cause a high resource consumption but provide high delivery guarantees. Also, gossip dissemination of large messages incurs a high latency with store-and-forward mechanism where each message needs to be received completely and stored before being forwarded. For many open systems in which nodes can join or leave the system without any restriction, store-and-forward is the only viable option to protect the system against Denial of Service (DoS) [6] attacks: a correct node needs to validate each message before forwarding it to its neighbors, therefore; it should wait to deliver the full message before forwarding.

There are many gossip dissemination protocols, that aim to improve the efficiency of dissemination. *IDA-Gossip* [7] is one of them. It is a gossip dissemination protocol proposed in the context of blockchains to disseminate large messages like blocks of transactions fast. Specifically, it is designed to disseminate large messages efficiently by removing the incurred high latency caused by the store-and-forward mechanism. *IDA-Gossip* combines Information Dispersal Algorithms(IDA) [8] and gossip dissemination protocols to circumvent the limitations of classic gossip protocols. *IDA-Gossip* makes use of message chunking, erasure coding, and Merkle hash trees. Using chunk-based message dissemination where a big message is chunked into smaller chunks, *IDA-Gossip* alleviates the high latency caused by the store-and-forward mechanism. By using an erasure coding mechanism, *IDA-Gossip* protects chunks against loss. Finally, by using Merkle hash trees, it provides an efficient chunk authentication scheme. *IDA-Gossip* is a promising building block for many distributed systems that needs to disseminate large messages—likes blocks of

transactions [9]. Although the description of IDA-Gossip is straightforward, its properties are not investigated in depth under different conditions. We believe that understanding its properties might help its adoption.

This paper makes the following contributions: (1) we provide a formal description of the IDA-gossip protocol, (2) we conduct a thorough evaluation of IDA-gossip through experiments and simulations, (3) we identify the limitations of IDA-gossip and explain when this protocol is a good candidate to replace classic gossip alternatives in distributed systems.

II. The *IDA-Gossip* Protocol

In this section, first, we provide our system model, and later we provide the details of *IDA-Gossip* protocol. Finally, we communicate our analysis on the adoption of *IDA-Gossip* protocol.

A. System model and assumptions

We consider a static system composed of N nodes, with a fraction $f < 1/2$ of Byzantine nodes, and a fraction $1 - f$ of honest (correct) nodes. Correct nodes follow the protocol while Byzantine nodes may deviate from it in any possible way. Each node is connected to a set of d peers selected uniformly at random. A node only communicates with its peers. Nodes communicate over reliable synchronous channels in which messages are neither lost nor duplicated. Any message sent by one node is received by another within a known time interval.

We assume the presence of an adversary controlling all Byzantine nodes to prevent correct nodes from delivering messages. Byzantine nodes can either drop the messages they receive or alter the content of the message they receive and forward. We assume the availability of cryptographic hash functions and Public-key cryptography mechanisms. Any message sent by nodes is authenticated using signatures. Finally, the adversary cannot break the cryptographic primitives. The source nodes are selected among the correct nodes, and the identities of source nodes are known in advance by other nodes: therefore, Byzantine nodes can not flood the system with irrelevant messages.

B. The details of IDA-Gossip protocol

In the *IDA-Gossip* protocol, each node selects d neighbors to communicate. κ denotes the number of chunks of a message M. ϕ is a security parameter of the protocol that takes values between 0 and 1, and it controls the ratio of data chunks and parity chunks. ϕ is calculated theoretically to protect the source node against up to 10 faulty sampled neighbors out of d neighbors. The source node chunks a large message M into $(1 - \phi)\kappa$ equal sized chunks, $C_1, C_2, C_3, ..., C_{(1-\phi)\kappa}$. Later on, the source node calculates $\phi\kappa$ additional parity chunks using an erasure-coding scheme—Reed-Solomon erasure coding [10]. In total, the source node produces κ chunks.

The use of chunks in *IDA-Gossip* introduces the problem of chunk authentication: without an efficient authentication mechanism, it would be costly to authenticate chunks. *IDA-Gossip* makes use of Merkle hash trees to authenticate chunks of a message M. The source calculates a Merkle tree which is a binary tree. The leaves of the Merkle tree consist of the hashes of chunks in order— $H(C_1), H(C_2), H(C_3), ..., H(C_\kappa)$, and inner nodes are calculated from bottom to up by concatenating and hashing children's values. The source node calculates a Merkle proof for each chunk, it disseminate each chunk with its Merkle proof. Any node can authenticate a received chunk using the attached Merkle proof.

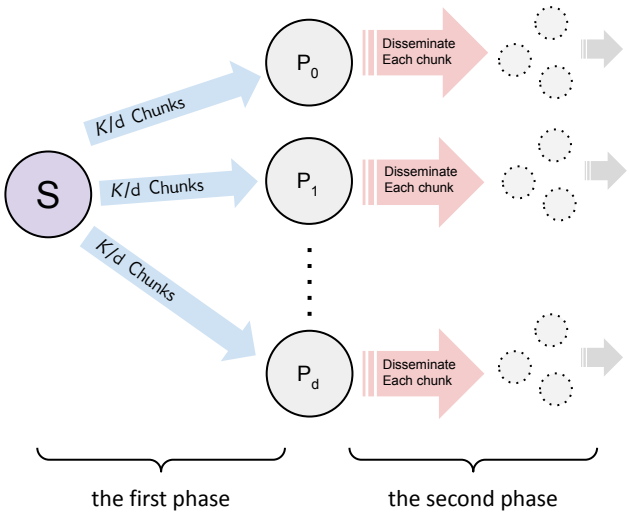

Fig. 1. The two phases of *IDA-Gossip*: in the first phase, the source node sends κ/d distinct chunks to its peers, and in the second phase, its peers disseminate chunks using classic gossip.

The authors of RapidChain have shown that we can derive a threshold ζ for ϕ, such that the probability of having a proportion of corrupted nodes greater than ζ in the neighbor set of the source node is at most 0.1. Using a hypergeometric distribution, under the assumption $f < 1/2$ and $d = 16$, setting $\zeta = \phi = 0.63$ guarantees that a message is not delivered to all honest nodes with probability 0.1. Therefore, although at the end of the first phase none of the source node's neighbors can reconstruct the original message, the system is in a state where at least $(1 - \phi)\kappa$ chunks are possessed by honest nodes with high probability.

For analysis purposes, one can divide *IDA-Gossip* protocol into two distinct phases, as shown in Fig. 1: in the first phase, the source node chunks a message into smaller chunks, calculates parity chunks, and sends a subset of chunks to its peers. In the second phase, the peers of the source disseminate messages using classic gossip dissemination on behalf of the source node. Redundancy is the key element to protect the system against faulty nodes and message loss. Unlike classic gossip dissemination protocols, in the first phase of *IDA-Gossip*, the redundancy comes from parity chunks calculated using erasure coding. In the second phase of *IDA-Gossip* redundancy comes from forwarding message fanout times.

The original evaluation of *IDA-Gossip* conducted in RapidChain [7] considers the following parameter values: messages

of size 2MB, $d = 16$, $\phi = 0.63$, and $\kappa = 128$.

C. On the adoption of IDA-Gossip

Because *IDA-Gossip* disseminates chunks of the message instead of the message itself, a node cannot verify the content of a received chunk according to the upper-layer protocol. This is a crucial property that should be taken into consideration for the adoption of *IDA-Gossip*, e.g., in open systems where the identity and number of nodes are not publicly known, and where the source node is not always expected to behave correctly (for example, permissionless blockchains systems, etc.). Indeed, an adversary in control of the source node could flood the whole system by disseminating irrelevant messages. Further, a node could behave in a faulty manner and disseminate an invalid message. In such scenarios, faulty behaviors would remain hidden until a correct node obtains enough chunks to reconstruct the original message. Although this issue is not specifically related to *IDA-Gossip* but to chunk-based gossip more generally, it highlights the need to develop solutions that allow verification of the content of each chunk as it is disseminated, in relation to the upper-layer protocol employing *IDA-Gossip*. Also, it is possible to mitigate this kind of behavior by employing accountability mechanisms as in [11]–[13].

Conversely, *IDA-Gossip* represents a very good candidate for permissioned systems or semi-permissionless systems where a subset of nodes is identified to conduct a specific task such as in sharded blockchains where the sharding committees are elected (*e.g.*, RapidChain [7], Omniledger [14], Elastico [15]), or in permissioned blockchains such as Hyperledger Fabric [16].

III. Evaluation with Experiments

In this section, the performance and limitations of *IDA-Gossip* are evaluated in a controlled testbed. We aim to address the following questions: (1) How does *IDA-Gossip* perform compared to classic gossip dissemination? (2) How does *IDA-Gossip* perform under different conditions such as varying message size, number of chunks, etc.? (3) Because *IDA-Gossip* employs erasure coding that comes with redundancy in the transmitted data, what is its impact on the utilization of network bandwidth and the performance of the dissemination compared to classical gossip techniques? (4) How are *IDA-Gossip*'s performances affected under faults?

A. Methodology

1) Evaluation environment and implementation: The experiments presented in this section are conducted on the Grid'5000 [17] platform. We consider 32 physical machines with 18 cores, 96 GB of memory, and connected with high-speed links-25 Gbps. In each experiment, we deploy 4096 *IDA-Gossip* nodes (128 nodes per machine) to emulate a large-scale distributed system. The use of 4096 nodes reflects approximately the order of magnitude of today's open system size: Tor network being composed of ∼8,000 relays, and Bitcoin of ∼15,000 nodes. We emulate wide-area network conditions by capping the bandwidth of each node to 20 Mbps and adding a one-way latency of 15 milliseconds to each communication link between processes using cgroups and traffic control subsystem of Linux.

We implemented a prototype of *IDA-Gossip* in Golang, consisting of 3000 lines of code. Our implementation chunks a large message into $(1 - \phi\kappa)$ chunks, and adds $\phi\kappa$ parity chunks using a Reed-Solomon erasure coding library[1]. We implemented a coordination service, and it is used by each node as a rendezvous point at bootstrap.

2) Experimental protocol: An experiment starts with a fresh deployment of an *IDA-Gossip* system with 4096 nodes hosted on 32 machines. Upon start, a node registers its IP address and port number to the coordination service and waits for the registration of other nodes. Later on, the node retrieves the node list from the coordination service. The node list contains the IP addresses and port numbers of all nodes in the system. Each node samples 16 peers uniformly at random using the node list, and it establishes connections to sampled nodes. Each node accepts up to 125 incoming connections. To start dissemination, we select a source node uniformly at random. The source node disseminates a message using *IDA-Gossip*. Other nodes forward each delivered chunk 8 times, therefore we have used the fanout value of 8. An experiment terminates when all nodes have forwarded the chunks that they have delivered. To guarantee this condition, we use a conservative timeout value that we empirically determined in preliminary experiments. At the end of each experiment, we tear down the network and collect measured statistics from nodes. Each experiment is run 30 times to consolidate the measured metrics.

An experimental setup consists of a set of fixed parameters: the number of chunks, the message size, the proportion of Byzantine nodes, and the sequential or concurrent dissemination of chunks from a node to its neighbors—dissemination concurrency. Table I lists the data chunks and parity chunks used in our experiments. These values are calculated according to the target number of chunks, κ, and by setting ϕ to 0.63 to tolerate up to 10 faulty neighbors out of the 16 neighbors of the source node.

In our experiments, we vary the size of messages between 1 and 36 MB, the chunk count between 16 and 256, and the proportion of Byzantine nodes between 0 and 40%. Finally, we vary the concurrent dissemination of chunks—dissemination concurrency—from 1 (sequential) to 128 (high degree of concurrence).

3) Measured Metrics: We measure the following four metrics: 1) first chunk latency, 2) latency, 3) amount of uploaded data, and 4) coverage.

The *first chunk latency* is the time needed for any node in the system to deliver a chunk from the disseminated message. This information is a precursor of how early a node starts contributing to the dissemination process. In classical gossip protocols that employ the store-and-forward mechanism, the

[1]https://github.com/klauspost/reedsolomon

TABLE I
PARAMETER VALUES OF IDA-GOSSIP USED IN OUR EXPERIMENTS

#Chunk = κ	#Data Chunks	#Parity Chunks	d	κ/d
256	96	160	16	16
128	48	80	16	8
64	24	40	16	4
32	12	20	16	2
16	6	10	16	1

majority of nodes do not contribute to the dissemination until the last rounds of dissemination. An efficient dissemination mechanism should employ all available resources at the earliest possible time. Hence this metric is a sound performance indicator for gossip protocols disseminating large messages.

Latency is the time needed to deliver enough chunks to reconstruct the original message disseminated by the source node. *Uploaded data* is the amount of data uploaded by a node. It is directly proportional to the fanout value, and the message size. Uploaded data is a metric that measures the redundancy of gossip dissemination, helping us to quantify the bandwidth usage overhead caused by the use of erasure coding techniques in *IDA-Gossip*. Finally, *coverage* is the percentage of nodes that successfully deliver the initial message.

B. IDA-Gossip vs classic gossip dissemination

We compare *IDA-Gossip* with *classic gossip* dissemination where a message is disseminated in its entirety (without chunking) using the store-and-forward mechanism. More specifically, we aim to understand the performance difference between these two protocols when considering message sizes between 1 and 36 MB. We conducted a set of experiments by varying the message size. In this set of experiments, we have considered a fault-free environment.

For *IDA-Gossip*, we consider the use of $\kappa = 128$ chunks as originally set by the designers of RapidChain. Our classic gossip instances use a fanout value of 8—as *IDA-Gossip* instances. Also, classic gossip instances use a single-threaded message dissemination mechanism in which a node sends chunks one peer at a time. In our preliminary experiments, we have observed that in case of limited bandwidth and large message sizes this is the best strategy that produces minimal message dissemination latency for classic gossip instances. Fig. 2 plots quartiles (first, second, and third quartiles) of the first chunk latency, latency, and uploaded data. Coverage is not represented as both protocols always reach 100% of the nodes.

The first chunk latency of classic gossip dissemination is equal to its latency because the whole message consists of a single chunk. When we compare both protocols in terms of first chunk latency, *IDA-Gossip* provides excellent results for all message sizes: with 36 MB messages, the first chunk is delivered and starts being disseminated to other nodes in the system within 11.5 seconds. Therefore, *IDA-Gossip* employs system resources at a much earlier time compared to classic gossip.

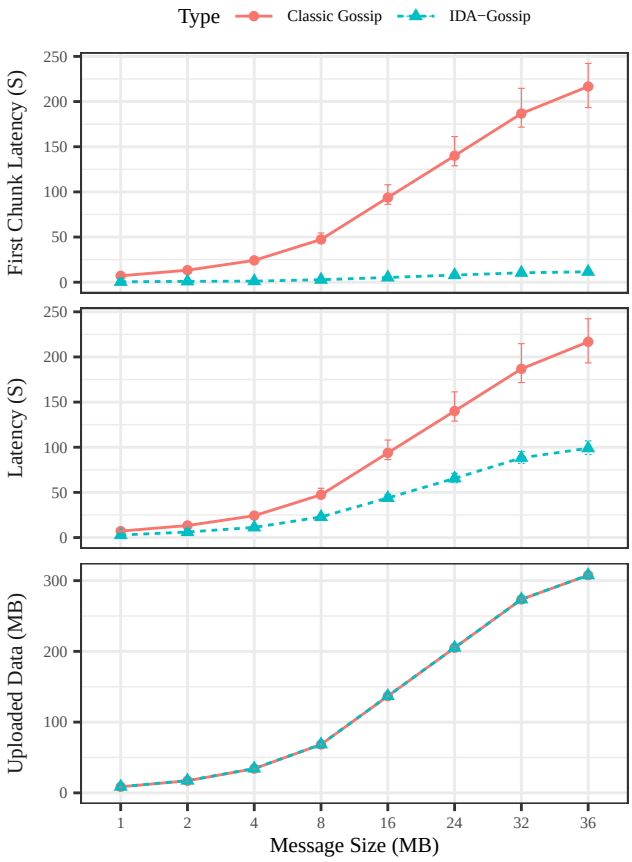

Fig. 2. Classic gossip dissemination compared to *IDA-Gossip* with 128 chunks.

With all message sizes, *IDA-Gossip* provides lower latency compared to classic gossip for all message sizes. With messages size 1, 2, 16, and 36 MB: *IDA-Gossip* provides latency values of 2.7, 6.0, 43.8, and 98.9 seconds while classic gossip instance provides 7.1, 13.3, 93.7, and 216.7 seconds respectively. This highlights that *IDA-Gossip* outperforms the classic gossip technique that we have considered.

In terms of network usage, *IDA-Gossip* and classic gossip provide similar performance: As depicted in Fig. 2, although the uploaded data per node values are very close to each other for specific message size, they are not the same, and the difference is less than 0.5% for all configurations. The small difference in uploaded data statistics stems from two facts: (1) *IDA-Gossip* makes use of Merkle hash trees to authenticate messages, and they have an overhead of O(log(κ)) storage. (2), individual chunks come with a small overhead of metadata storage which is O(1) because each chunk needs to be disseminated with some metadata. In this comparison in the case of *IDA-Gossip*, we did not observe a significant overhead caused by parity chunks because an *IDA-Gossip* node forwards messages until reconstructing the original message for that purpose an *IDA-Gossip* node needs to deliver $(1-\phi\kappa)$ chunks as we stated previously.

C. IDA-Gossip with different chunk counts and message sizes

We investigate the effect of chunk count and message size on the performance of *IDA-Gossip*. In this experiment, we considered *IDA-Gossip* instances with 16, and 256 chunk counts. Also, we considered message sizes between 1 and 36 MB. Fig. 3 depicts the collected statistics from experiments. First of all, the first chunk latency, latency, and uploaded data measurements for all instances of *IDA-Gossip* is increasing with the increase in message size. This is an expected result because nodes have limited bandwidth to disseminate messages, and increased message size causes an increase in latencies.

When we consider a specific message size, the first chunk latency of an *IDA-Gossip* instance with a high number of chunk counts is always lower—better—compared to another *IDA-Gossip* instance with a smaller chunk count. This is another expected result because an increased chunk count results in a smaller chunk size. Smaller chunks are disseminated faster in the network.

We made an interesting observation, *IDA-Gossip* with 32 chunks always provided the lowest latency measurements for all message sizes. The reason for this is that in this set of experiments we have used a dissemination concurrency value of 8 because each node forwards a message 8 times in parallel. This results in the best performance for *IDA-Gossip* with 32 chunks. For higher chunk counts, one needs to increase dissemination concurrency value otherwise the effect of latency added to channels—to emulate WAN conditions—will be overemphasized on the measured latency metric. We study the effect of dissemination concurrency value on the performance of *IDA-Gossip* in subsection III-E

As seen in Fig. 3, the amount of uploaded data per node is increasing when the message size increases, as expected. However, for a given message size, all instances of *IDA-Gossip* nodes upload a similar amount of data. We note a minor difference lower than 1% between instances of *IDA-Gossip* with 16 chunks and 256 chunks due to the size of the Merkle proofs. This implies that a higher chunk count incurs a slightly higher bandwidth cost.

D. IDA-Gossip with Faults

We now evaluate the performance of *IDA-Gossip* under faults by varying the proportion of byzantine nodes in the system, where byzantine nodes drop all the messages they receive. In our experiment, we exclude the byzantine behavior of altering the chunk content or the Merkle proofs as this does not impact the performance of *IDA-Gossip*. We set the message size to 2 MB, and vary the percentage of faulty nodes from 0% to 40%. We also explore the impact of varying the number of chunks (from $\kappa = 16$ to 256) in this faulty environment.

Fig. 4 shows the the evaluation results. In terms of first chunk latency, *IDA-Gossip* instances with high chunk counts (64, 128, and 256) provide stable results when increasing the percentage of byzantine nodes, whereas the other instances (with 16 and 32 chunks) provide slightly degraded first chunk latency.

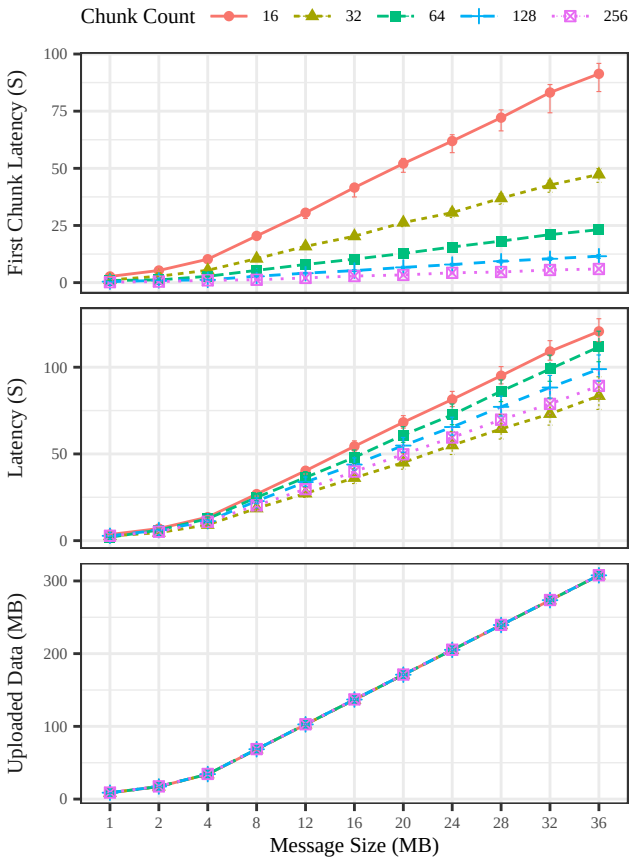

Fig. 3. The behavior of *IDA-Gossip* with different chunk counts and message sizes.

The latency of all *IDA-Gossip* instances is increasing along with the ratio of faulty nodes. This increase is more pronounced in *IDA-Gossip* with $\kappa = 16$ and 32 chunks. In a fault-free system, *IDA-Gossip* with 32 chunks is providing the lowest latency but, under faults, its performance is affected more than other instances.

From these experiments, we conclude that *IDA-Gossip* with high chunk counts, *i.e.*, above 64 chunks, are more resilient to Byzantine behaviors, and provide smooth latency degradation when facing faults.

In terms of coverage, all instances of *IDA-Gossip* provide stable performance: with an increased fraction of faulty nodes we did not observe a significant decrease in coverage, and it is above 99% for all instances.

E. IDA-Gossip with Different Dissemination Concurrency Values

In case of big messages and limited bandwidth, a node should send messages to its neighbors one by one; otherwise, the sending might take longer time because concurrent send events will compete for the same bandwidth resource, and this will increase the latency of dissemination. *IDA-Gossip* chunks a large message into smaller pieces, and this makes the chunk-sending strategy vital for the performance of the system. In

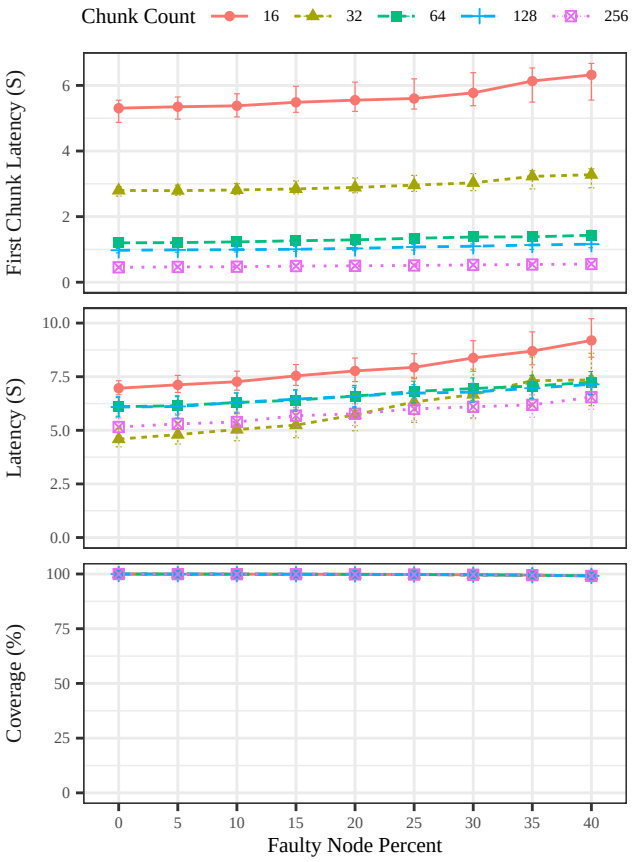

Fig. 4. The behavior of *IDA-Gossip* with different percentages of byzantine nodes.

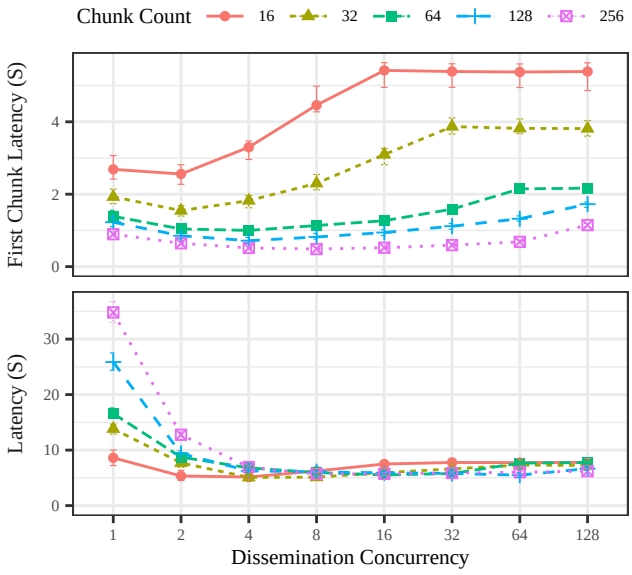

Fig. 5. The behavior of *IDA-Gossip* with different dissemination concurrency values.

this subsection, we investigate the effect of dissemination concurrency on the performance of *IDA-Gossip* because choosing the wrong strategy might result in sub-optimal performance.

We have conducted a set of experiments where we keep the message size constant—2MB—, and we vary the dissemination concurrency value between 1 and 128 so that a node serves a limited number of connections simultaneously. Without any restriction, dissemination concurrency is equal to the fanout value because a node opens a single connection to each sampled peer, and each peer connection is owned by a distinct thread that serves the connection. The value of dissemination concurrency 1 means that a node serves peers one by one, sequentially, and the value of 2 means that a node can serve up to two peers concurrently, and so on. To increase the dissemination concurrency above the fanout value, our implementation opens more than one connection to each peer; for example, with a fanout value of 8 to reach dissemination concurrency of 128, a node needs to open 16 $(128/8)$ connections to each sampled peer.

In this set of experiments, as seen in Fig. 5, we have observed that smaller values of dissemination concurrency, like 1 and 2, provide the best first-chunk latency. When we increase the dissemination concurrency, the first chunk latency increases up to a point and later stays constant for all *IDA-Gossip* instances: This is because there is a limited number of chunks to disseminate. On the other hand, smaller values of dissemination concurrency values produce undesirable latency measurements for some *IDA-Gossip* instances: for example, *IDA-Gossip* with 256 and 128 chunks provides respectively 35 and 25 seconds latency with 1 simultaneously served connection. When we send a high number of chunks sequentially, the latency of the channel piles up, and it affects the final latency of dissemination.

Each *IDA-Gossip* instance provides the best latency with a different dissemination concurrency value: For example, *IDA-Gossip* with 16 chunks provides a latency of 5.153 seconds with dissemination concurrency of 4, and *IDA-Gossip* with 32 chunks provides a latency of 5.119 seconds with dissemination concurrency of 8. *IDA-Gossip* instance with 128 chunks provides the best latency with dissemination concurrency of 16.

In these experiments, we have considered only the message size of 2MB. The choice of chunk count,κ, and message size will change the characteristics because the size of a chunk depends on these parameters; therefore, to obtain optimal performance, one needs to examine the effect of dissemination concurrency according to considered chunk count, message size and considered network bandwidth capacity.

IV. Evaluation with Simulations

In this section, we compare the fault resilience of *IDA-Gossip* to the one of standard chunk-based gossip dissemination. We wish to quantify the resilience improvement brought by the use of erasure-coding in *IDA-Gossip*. To that end, we implement a simulation engine in Golang that simulates the two following gossiping strategies for disseminating a message M of size $|M|$: *IDA-Gossip* as described in this paper, and a

chunk-based gossip dissemination where a message is chunked into n pieces.

A. Methodology

1) Simulations: Our simulation engine is discrete-time based where time is divided into consecutive rounds. In a round, a node can receive multiple chunks from different nodes. In the same round, a node forwards chunks that are received in the previous round. We use a fanout value of 8, therefore; for each message, a node samples 8 other nodes uniformly random and sends the message to these nodes.

The system consists of 4096 nodes, as in our experimental evaluation. A simulation run starts by having a source node selected uniformly at random sending chunks to its neighbors, and ends when there is no node with chunks to forward. We configure the simulations with a fixed proportion f of faulty nodes, that we vary from 0 to 99 with steps of 1. For each configuration, we run 1000 simulations. At the end of the 1000 runs, the simulator aggregates the results and computes the measured metrics.

2) Simulated strategies: In the *IDA-Gossip* strategy, the source node chunks the message M into $n = (1-\phi)\kappa = 48$ data chunks and adds $\phi\kappa = 80$ parity chunks. Any set of $(1-\phi)\kappa$ different chunks is enough to reconstruct the original message. Then, the source node samples $d = 16$ nodes from the system and sends them each $\kappa/d = 8$ chunks.

In the classic chunk-based gossip strategy, a source node splits M into $n = 48$ chunks and sends each chunk to 8 other nodes. To reconstruct the original message, a node requires all n chunks. In both strategies, each node stores and forwards each received chunk until being able to reconstruct the original message. Then, nodes may only receive other chunks but will ignore them.

3) Measured Metrics: To evaluate the resilience of both protocols, we measured the following metrics: 1) coverage, 2) dissemination failure ratio, 3) received chunk count, and 4) delivered chunk count. *Coverage* is the percentage of nodes that delivers the full message at the end of a simulation run. The *dissemination failure ratio* is the percentage of simulation runs in which none of the nodes, except the source, managed to reconstruct the original message. The *received chunk count* is the number of chunks received by a node in a simulation run. Note that a node can receive multiple copies of the same chunk because our gossip engine implements a push-based gossip where no communication happens between nodes to identify chunks that should be sent/receive according to already received chunks. The *delivered chunk count* is the number of distinct chunks delivered by a node to reconstruct the original message. This must be equal to the chunk count of the original message which is 48 for both simulated strategies.

B. IDA-Gossip vs Classic Multi-chunk Gossip Dissemination

For all measured metrics, we display their mean values in Fig. 6. *IDA-Gossip* and chunk-based gossip protocols behave similarly when considering the numbers of received and delivered chunks. Indeed, we employ the same push-based gossip

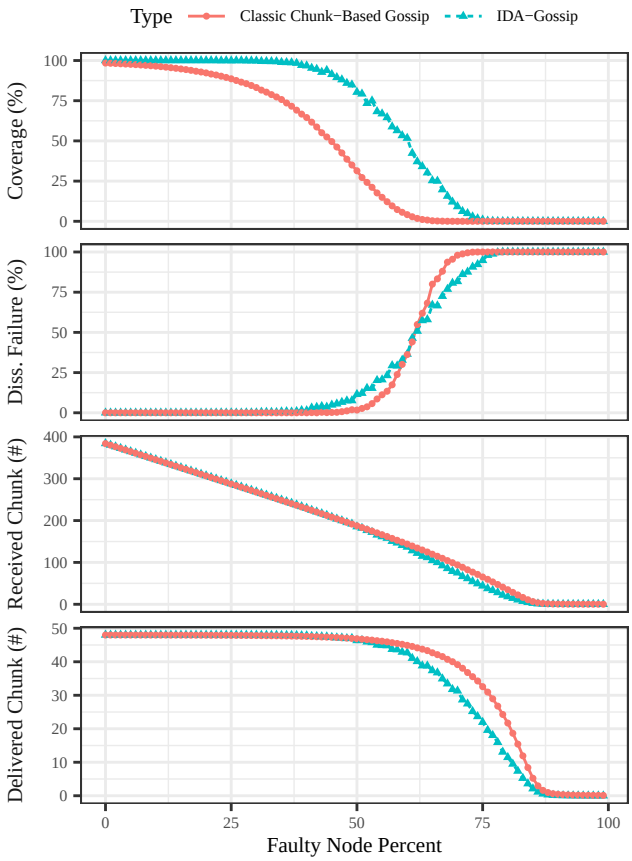

Fig. 6. Comparison of *IDA-Gossip* with classic multi-chunk gossip under byzantine behaviors.

technique in both strategies and use the same data chunk count $n = 48$. When the system consists of correct nodes only, the mean number of received chunks per node is 384. With $n = 48$ data chunks, and each node forwarding each delivered chunk to its 8 neighbors, it implies that each node sends $48 \times 8 = 384$ chunks in a simulation run. Therefore, each node is expected to receive 384 chunks on average. Also, each node is expected to deliver 48 chunks to finish dissemination.

Regarding coverage, *IDA-Gossip* provides 100% coverage with up to $f = 40\%$ of byzantine nodes in the system. On the other hand, the classic chunk-based gossip suffers from low coverage even with a low fraction of faulty nodes because the probability of not delivering a single chunk of a message with an increased number of data chunks is increasing. This is not the case for *IDA-Gossip* because different nodes forward different subsets of chunks, and any subset with cardinality 48 is enough to reconstruct the original message.

We observe that *IDA-Gossip* starts suffering from dissemination failures slightly earlier than classic chunk-based gossip. This is because the dissemination fails if a source node samples more than 10 faulty nodes out of 16; in this case, an insufficient number of chunks is disseminated, and none of the nodes can reconstruct the original message except the source itself. The classic gossip approach suffers from this problem

later than *IDA-Gossip* because each chunk is forwarded 8 times by the source node, diminishing the likeliness of losing a chunk because of faulty nodes.

V. Related Work

The use of multiple chunks and erasure coding has been studied extensively in the context of gossip dissemination in different settings. In this section, we compare *IDA-Gossip* with state-of-the-art propositions and we highlight similarities and differences.

One of the earliest examples of multi-chunk gossip dissemination is Splitstream [18]. It aims for efficient dissemination of messages, and fair distribution of the dissemination cost among contributing nodes. Splitstream considers a structured peer-to-peer network where the communication pattern of processes is scheduled in advance to obtain optimum latency and optimum communication cost. Splitstream chunks a large message into multiple pieces, and it constructs disjoint dissemination trees in a deterministic manner for each chunk. In a dissemination tree, only inner nodes disseminate the message. A node has different roles in different trees. Unlike *IDA-Gossip*, Splitstream's evaluation considers a structured peer-to-peer network, and it does not consider byzantine faulty nodes. Also, Splitstream does not consider parity chunks. Therefore, evaluation of Splitstream does not help to understand the properties of the *IDA-Gossip* protocol.

Sanghavi et al. [19] investigate the cost of gossip dissemination in an unstructured setting where nodes randomly contact other nodes to send or receive messages. They propose a gossip dissemination protocol, INTERLEAVE, that relies on multi-chunk gossip dissemination. They compare classic gossip with multi-chunk gossip dissemination from a theoretical point of view, and they provide an analysis of the optimum gain that can be achieved from splitting a message—multi-chunk—compared to sending a single large message. Their analysis shows the benefit of multi-chunk gossip dissemination theoretically. Although they mention the possible benefits of using erasure coding in the context of push and pull gossip dissemination they do not investigate it. Later on, Cigno et al. [20] investigate the performance of INTERLEAVE protocol by using simulations. They aim to quantify the properties of the INTERLEAVE protocol. Both works provide important information about multi-chunk gossip dissemination but they do not consider erasure coding, therefore, one can not gain information about *IDA-Gossip* by looking at these studies.

An important use case for multi-chunk gossip dissemination is live streaming. Multi-chunk gossip dissemination is indispensable for live streaming because of the size of the messages. Bar gossip [13], LiFtinG [12], and AcTinG [11] are protocols designed to handle rational nodes in streaming systems. Rational nodes do not want to contribute to the dissemination of messages to decrease resource consumption, and they are risk avers. All these protocols depend on multi-chunk gossip dissemination. Although they consider authenticated messages, none of these protocols use erasure coding or an efficient chunk authentication mechanism. Also, their fault model only considers rational behaviors that are a subset of Byzantine behaviors.

In the context of live streaming, another important proposition is Gossip++ [21]. Gossip++ uses push-and-pull gossip mechanisms together to implement a hybrid gossip protocol. It uses erasure coding to improve dissemination performance. In Gossip++, the source node chunks a large message into 100 chunks and adds 5 parity chunks. Gossip++ considers way fewer parity chunks than *IDA-Gossip*. Unlike *IDA-Gossip*, the source node forwards each chunk multiple times: Gossip++ uses different fanout values for source nodes and other nodes. These fanout values are respectively 5 and 8. The evaluation of Gossip++ considers only 200 nodes. Finally, the evaluation considers only rational nodes. Although Gossip++ shows the utility of erasure coding in the context of gossip dissemination, its evaluation can not help to understand the behavior of *IDA-Gossip* because of the differences in protocol design and experimental setup.

All of the mentioned works have similarities with *IDA-Gossip*: many use chunk-based gossip dissemination for efficient dissemination of large messages, and few consider erasure coding. None of them handle the problem of efficient chunk authentication. Although the majority of them have an extensive experimental study of the considered protocol, because of the specific differences in the protocol design and considered systems, it is hard to interpolate the results of these studies to understand the properties of the *IDA-Gossip* protocol. The majority of experimental studies consider only a few hundred nodes, like Gossip++, which can be considered a small size for a distributed system of today. Therefore, our work fills a gap in the literature, and provides an extensive experimental study of *IDA-Gossip* according to today's standards to quantify its benefits and to understand its properties under different conditions.

VI. Conclusion

Our experimental evaluation revealed that *IDA-Gossip* provides significant latency improvement compared to classic store-and-forward gossip dissemination: for all considered message sizes, even with a message size of 1 MB, *IDA-Gossip* provides the lowest latency. Also, *IDA-Gossip* better utilizes system resources compared to classic gossip dissemination because nodes start contributing earlier to the dissemination. Parity chunks and chunk authentication mechanisms of *IDA-Gossip* do not incur a significant bandwidth usage overhead compared to classic gossip dissemination.

We have observed, all instances of *IDA-Gossip* with different chunk counts provide similar bandwidth usage when the message size is kept constant. In a fault-free setting, *IDA-Gossip* with 32 chunks provides the lowest latency. On the contrary, *IDA-Gossip* instances with a high number of chunk counts provide better system utilization because of lower first chunk latency. When we injected faults into the system, we observed that *IDA-Gossip* instances with a high chunks count provide more resilience and graceful performance degradation. Also, we have observed that the choice of chunk-sending

strategy is vital to obtain optimum performance from *IDA-Gossip* instances.

Our simulations revealed that, in the presence of faulty nodes in the system, *IDA-Gossip* provides excellent coverage—above 99%—compared to classic multi-chunk gossip dissemination. *IDA-Gossip* starts to suffer from dissemination failures earlier than classic gossip dissemination where none of the nodes delivers the message except the source. Finally, our simulations showed that *IDA-Gossip* and classic multi-chunk gossip dissemination provide similar bandwidth usage characteristics in terms of received and send chunk counts. Therefore, the use of parity chunks does not incur an overhead on bandwidth usage.

ACKNOWLEDGMENT

This work was partially funded by Région Nouvelle-Aquitaine, under grant 2018-1R50117 (project B4IOT), and the French Agence Nationale de la Recherche (ANR), under grant ANR-21-CE25-0021 (project GenBlock)

REFERENCES

[1] Y. Gao, J. Shi, X. Wang, Q. Tan, C. Zhao, and Z. Yin, "Topology Measurement and Analysis on Ethereum P2P Network," in *2019 IEEE Symposium on Computers and Communications (ISCC)*, Jun. 2019, pp. 1–7, iSSN: 2642-7389.

[2] R.-G. Simion and M.-L. Pura, "A BitTorrent DHT Crawler," in *2019 IEEE 13th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, May 2019, pp. 155–160.

[3] S. Park, S. Im, Y. Seol, and J. Paek, "Nodes in the Bitcoin Network: Comparative Measurement Study and Survey," *IEEE Access*, vol. 7, pp. 57 009–57 022, 2019.

[4] P. B. Godfrey, S. Shenker, and I. Stoica, "Minimizing churn in distributed systems," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 147–158, Aug. 2006. [Online]. Available: https://doi.org/10.1145/1151659.1159931

[5] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, "Epidemic algorithms for replicated database maintenance," in *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, ser. PODC '87. New York, NY, USA: Association for Computing Machinery, Dec. 1987, pp. 1–12. [Online]. Available: https://doi.org/10.1145/41840.41841

[6] F. Lau, S. Rubin, M. Smith, and L. Trajkovic, "Distributed denial of service attacks," in *2000 IEEE international conference on systems, man and cybernetics*, vol. 3, Oct. 2000, pp. 2275–2280 vol.3, iSSN: 1062-922X.

[7] M. Zamani, M. Movahedi, and M. Raykova, "RapidChain: Scaling Blockchain via Full Sharding," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: Association for Computing Machinery, Oct. 2018, pp. 931–948. [Online]. Available: https://doi.org/10.1145/3243734.3243853

[8] N. Alon, H. Kaplan, M. Krivelevich, D. Malkhi, and J. Stern, "Addendum to "Scalable Secure Storage when Half the System is Faulty"," *Information and Computation*, vol. 2004, 2003.

[9] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," *Decentralized Business Review*, p. 9, 2008.

[10] I. S. Reed and G. Solomon, "Polynomial Codes Over Certain Finite Fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, Jun. 1960, publisher: Society for Industrial and Applied Mathematics. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/0108018

[11] S. B. Mokhtar, J. Decouchant, and V. Quéma, "AcTinG: Accurate Freerider Tracking in Gossip," in *2014 IEEE 33rd International Symposium on Reliable Distributed Systems*, Oct. 2014, pp. 291–300, iSSN: 1060-9857.

[12] R. Guerraoui, K. Huguenin, A.-M. Kermarrec, M. Maxime Monod, and S. Prusty, "LiFTinG: Lightweight Freerider-Tracking Protocol in Gossip," INRIA, Research Report RR-6913, 2010. [Online]. Available: https://hal.inria.fr/inria-00379408

[13] H. C. Li, A. Clement, E. L. Wong, J. Napper, I. Roy, L. Alvisi, and M. Dahlin, "BAR Gossip," 2006. [Online]. Available: https://www.usenix.org/conference/osdi-06/bar-gossip

[14] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding," in *2018 IEEE Symposium on Security and Privacy (SP)*, May 2018, pp. 583–598, iSSN: 2375-1207.

[15] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A Secure Sharding Protocol For Open Blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, Oct. 2016, pp. 17–30. [Online]. Available: https://doi.org/10.1145/2976749.2978389

[16] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. W. Cocco, and J. Yellick, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, ser. EuroSys '18. New York, NY, USA: Association for Computing Machinery, Apr. 2018, pp. 1–15. [Online]. Available: https://doi.org/10.1145/3190508.3190538

[17] D. Balouek, A. C. Amarie, G. Charrier, F. Desprez, E. Jeannot, E. Jeanvoine, A. Lèbre, D. Margery, N. Niclausse, L. Nussbaum, O. Richard, C. Perez, F. Quesnel, C. Rohr, and L. Sarzyniec, "Adding Virtualization Capabilities to the Grid'5000 Testbed," in *Cloud Computing and Services Science*, ser. Communications in Computer and Information Science, I. I. Ivanov, M. van Sinderen, F. Leymann, and T. Shan, Eds. Cham: Springer International Publishing, 2013, pp. 3–20.

[18] M. Castro, P. Druschel, A.-m. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: High-bandwidth multicast in a cooperative environment," in *In SOSP'03*, 2003.

[19] S. Sanghavi, B. Hajek, and L. Massoulie, "Gossiping With Multiple Messages," *IEEE Transactions on Information Theory*, vol. 53, no. 12, pp. 4640–4654, Dec. 2007, conference Name: IEEE Transactions on Information Theory.

[20] R. L. Cigno, A. Russo, D. Carra, and S. Antipolis, "On Some Fundamental Properties of P2P Push/Pull Protocols," p. 7.

[21] D. Frey, R. Guerraoui, A.-M. Kermarrec, and M. Monod, "Boosting Gossip for Live Streaming," in *2010 IEEE Tenth International Conference on Peer-to-Peer Computing (P2P)*, Aug. 2010, pp. 1–10, iSSN: 2161-3567.