Byzantine Reliable Broadcast on partially connected networks with signatures

Rahim Klabér[1] , **Jérémie Decouchant**[1]
[1]TU Delft

Abstract

In this paper, we consider Byzantine reliable broadcast on partially connected networks using signatures. Byzantine reliable broadcast in partially connected and authenticated networks can be achieved by combining two algorithms, Gabriel Bracha's double-echo broadcast protocol and Danny Dolev's reliable communication protocol. Bracha's algorithm allows for Byzantine reliable broadcast in fully connected networks, while Dolev's algorithm can be used to provide reliable communication in networks which are atleast $2f+1$-connected, where f is the number of Byzantine nodes. For Byzantine reliable broadcast in partially connected networks, Bracha's algorithm can be used with Dolev's algorithm providing an abstraction of a fully connected network. We show how signatures can be used to lower the connectivity requirement to $f + 1$ and lower the message complexity. We also show how aggregate or multi-signatures can be used to lower bandwidth used by the algorithm. When compared to the state-of-the-art Bracha-Dolev without signatures, our protocol has a message complexity which 20 times lower (N=60,f=6).

1 Introduction

Distributed systems consist of autonomous computing entities that communicate between each other to solve a problem. Lately, distributed systems have been popularized by cryptocurrencies, such as Bitcoin, but many other applications of distributed systems exist. Examples include distributed databases, peer-to-peer file sharing and distributed rendering.

Some of these distributed systems rely on algorithms to broadcast data even in the presence of faulty processes. Two well known broadcasting algorithms are Gabriel Bracha's double echo authenticated broadcast [6] and Danny Dolev's reliable communication algorithm [9]. Dolev's and Bracha's algorithms both work in synchronous and asynchronous networks and assume authenticated links. The difference between them is that Dolev's algorithm works in a partially connected network with a minimum connectivity of $2f+1$, while Bracha's algorithm works in a fully connected network with $f < N/3$, where f is the number of Byzantine processes and N is the total number of processes. Additionally, Dolev's algorithm is restricted in that the sender cannot be faulty, while Bracha's algorithm does not have this restriction.

Dolev's algorithm provides an abstraction known as reliable communication, which guarantees that a message broadcast by a correct process is delivered by all correct processes. While Bracha's algorithm provides an abstraction known as reliable broadcast, which is also known as Byzantine Reliable Broadcast (BRB), which adds the additional case where the sender of a message might be Byzantine.

Bracha's and Dolev's algorithms can be combined to provide BRB in partially connected networks [12]. This is achieved by using Dolev's algorithm to provide the abstraction of a fully connected network and running Bracha's algorithms on top. Unfortunately, the Bracha-Dolev algorithm has a high algorithmic complexity.

Recently, some optimizations on Bracha-Dolev have been described [4; 5]. These optimizations reduce the number of messages sent and improve the latency and bandwidth consumption of Bracha-Dolev.

Digital signatures have been used to lower the amount of messages sent by Bracha's algorithm [8; 10] or to reduce the connectivity requirement of Dolev's algorithm from $2f + 1$ to $f + 1$ [1].

In this paper, we explore the possibility of improving the message complexity of Bracha-Dolev. Concretely, (1) we describe 4 modifications to Bracha-Dolev, which use digital signatures to reduce the amount of messages sent. (2) We evaluate each modification in a simulation written in kotlin in a number of different scenarios. (3) We give recommendations as to when a particular modification should be used.

The remained of the paper is organized as follows. Section 2 discusses related work. Section 3 describes the system model. Section 4 describes Bracha's and Dolev's algorithms and how they can be combined. In section 5 we describe our modifications to Bracha-Dolev. We evaluate our improvements in section 6. Section 7 explores the ethics of this paper. Finally, we conclude the paper in section 8.

2 Related Work

Danny Dolev introduced an algorithm for achieving reliable communication in partially connected networks [9], in both synchronous and asynchronous networks. When a message is broadcast using this protocol, the nodes forward the messages and keep track through which nodes a message was received. When a message is received through enough node-disjoint paths, it is accepted. Dolev's algorithm is not practical due its high message complexity.

Using signatures, it is possible to reduce the number of messages exchanged by Dolev's algorithm [11]. Because of the signatures, nodes no longer need to keep track of the path through which a message has traveled. Instead, a node can immediately deliver upon receiving one message with a correct signature, which drastically reduces the complexity of Dolevs' algorithm.

Byzantine reliable broadcast on fully-connected networks can be achieved by using Gabriel Bracha's authenticated double echo broadcast algorithm [6]. This protocol works in stages 3 stages. When a node broadcasts a message, all the nodes broadcast messages to each other to indicate that they acknowledge the broadcast. Nodes broadcast different types of messages depending on which stage they are at. Nodes accept the broadcast message when they are in the last stage and when they have received enough messages.

Just like Dolev, Bracha's algorithm can be modified to use digital signatures such that the amount of messages exchanged is lowered. One such algorithm is the Signed Echo Broadcast [7]. This algorithm is like Bracha's algorithm, but in this case nodes do not broadcast messages to each other. Rather, all nodes send signatures of the broadcast message to the broadcaster, who then sends the signatures to every other node. Unlike Bracha's algorithm, this protocol does not guarantee Byzantine reliable broadcast, since the broadcaster could decide to not send the signatures to certain nodes.

Astro II [8], a distributed payment protocol which allows for payments by only broadcasting and not by reaching consensus, uses a Signed Echo Broadcast like algorithm for its broadcast layer. In this case the algorithm is modified such that nodes not only send back signatures to the broadcaster, but also to the recipient of the payment. This prevents partial payment attacks. While Astro II does not guarantee Byzantine reliable broadcast, it demonstrates that algorithms can be tailored to specific application where not all guarantees are required.

Bracha's and Dolev's algorithms can be combined to achieve Byzantine reliable broadcast in $2f+1$-connected networks [12]. This can be achieved by modifying Bracha to use Dolev's algorithm as an abstraction of a fully-connected network. Every time Bracha's algorithm needs to broadcast messages network, it uses Dolev's algorithm to broadcast the message to the network. Recently, Bracha-Dolev was improved [4]. Even the improved Bracha-Dolev is not practical due to its high message complexity, Every Bracha message is broadcast using Dolev's algorithm which already has a high message complexity.

3 Problem Description and System model

We a assume a partially connected network consisting of N processes, which can each be uniquely identified by an id. We assume that $f < N/3$ processes can be Byzantine, i.e., that they can deviate arbitrarily from the underlying protocol. Additionally, we assume that Byzantine nodes are computationally bounded, i.e., that they cannot break the signature schemes used. We assume that processes know about the number of nodes N, the maximum number of Byzantine nodes f and about the ids of each process. Additionally, we assume that N, f, and the ids do not change over the lifetime of the network.

The network can be represented by an undirected graph with each vertex representing a process and each edge representing a communication-channel. We assume communication-channels are reliable; messages cannot be lost and always arrive at their destination. However, we do not assume authenticated communication channels. Nodes can directly communicate with other nodes which are connected by a communication-channel. Communication between nodes which are not directly connected is possible by relying on other nodes in the network forwarding data. We assume that processes are able to digitally sign data and verify digital signatures. We assume that the network is at least $f + 1$ node-connected and that the network topology is unknown to the individual nodes. Additionally, we assume a network is static; nodes cannot enter or leave the network after it has been constructed.

4 Background

In this section, we first describe Dolev's and Bracha's algorithms in detail and then we outline how Dolev's and Bracha's algorithms can be combined to achieve Byzantine reliable broadcast in partially connected networks.

4.1 Dolev's algorithm for reliable communication in partially connected networks

Dolev's algorithm can guarantee reliable communication in the presence of f faulty nodes if the network is $2f + 1$-connected (i.e., a node can reach another node through $2f+1$ node-disjoint paths), and if nodes communicate through authenticated and reliable channels [9]. Dolev presented two versions of the algorithm, one which relies on the network topology being known to the nodes and one which works when the network topology is unknown. We consider the case where the topology is unknown.

Aside from the data which is being broadcast, a Dolev-message also contains the identity of the broadcaster and the path through which the message has traveled. When a node broadcasts data using Dolev's algorithm, every node which receives the message forwards it while also changing the message to keep track of its path. When a node receives a message through $f + 1$ node-disjoint paths, it can accept the message.

It should be noted that Dolev's algorithm is not practical due to its high message complexity.

4.2 Bracha's algorithm for Byzantine reliable broadcast on fully connected networks

Byzantine reliable broadcast on fully-connected networks can be achieved by using Gabriel Bracha's authenticated double echo broadcast algorithm [6]. This algorithm can tolerate up to $f < N/3$ Byzantine nodes, where N is the number of nodes in the network and requires authenticated and reliable links.

The algorithm has multiple stages:

1. the broadcaster broadcasts an **INIT** message to the network.
2. Upon receiving an either an **INIT**, $(N + f)/2$ **ECHO** messages or $f + 1$ **READY** messages, broadcast an **ECHO** message.
3. $(N+f)/2$ **ECHO** messages or $f+1$ **READY** messages, broadcast a **READY** message.
4. Upon receiving $2f + 1$ **READY** messages, deliver the message.

When counting messages, messages received in the previous stages are also included. Because of the stages, one can be sure that either all nodes accept the message or none at all.

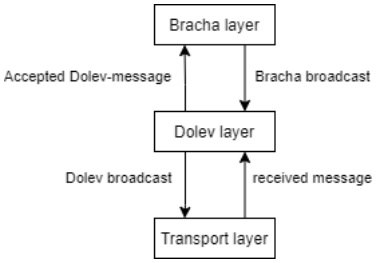

Figure 1: Layered arhitecture of Bracha-Dolev, adopted from [4]

4.3 Byzantine reliable broadcast in partially connected networks

Byzantine reliable broadcast in $2f + 1$-connected networks can be achieved by combining Dolev's and Bracha's algorithms. Bracha's algorithm is used as the means to actually achieve BRB, while Dolev's algorithm is used as a way to send messages to the entire network. Each message sent by Bracha is broadcast by Dolev's algorithm.

Figure 1 illustrates how Bracha-Dolev works. Bracha-Dolev can be seen as a protocol consisting of multiple layers. Bracha-messages are sent to the Dolev layer to be broadcaster using Dolevs algorithm. When a message is accepted by the Dolev layer, it sends this message to the Bracha layer.

While Bracha-Dolev does provide BRB in partially connected networks, its message complexity means that it is not practical.

5 Bracha-Dolev modifications

In this section, we describe how Bracha-Dolev can be modified to improve the message complexity. We also reflect on the difficulty of modifying the Bracha layer.

5.1 [M.1] Signing Dolev messages

Dolev's algorithm requires the network to be $2f + 1$-connected, i.e., there should exist $2f + 1$ node disjoint paths between any two nodes. This is necessary due to the possibility of Byzantine nodes in the network. For a process to deliver a message, it would need to receive the message via $f + 1$ node disjoint paths. Since there can be a maximum of f Byzantine processes this ensures that the majority of messages it receives is the correct message, which can then be delivered.

The connectivity requirement can be reduced to $f + 1$ with the use of signatures [11]. To achieve this the following modifications are be made. (1) Each process generates a key pair and uses the public key as their identity. (2) Each message sent is accompanied by a digital signature of the message. (3) When a process receives at least one message which was not tampered with, it can be delivered. Since each message contains a digital signature, the integrity of the message can be verified and the message can be authenticated. The modification is described in Algorithm 1. With this modification, instead of receiving a message via $f + 1$ node disjoint paths, a node only needs to receive one copy of the message. Additionally, since paths do not matter anymore, every node needs to forward the message only once, which reduces the message complexity.

This modification is not novel [1], but it was never properly described before due to the cost of using digital signatures. Nowadays, computers have gotten faster and this is no longer a large concern.

After a node has forwarded a message, forwarding it again has no merit, since unlike with the normal Bracha-Dolev, forwarding a message again does not add any extra information. In the normal Bracha-Dolev forwarding a message again might give you extra information in the form of the paths.

Using signatures turns Dolev into just flooding. Nodes need to receive a message once to deliver. After delivering a message nodes then forward the message to its neighbors.

5.2 [M.2] Stop participating after Bracha-delivering

If digital signatures are used with Bracha-Dolev, it is possible to further decrease the number of messages sent. A node can stop participating in the protocol after it has Bracha-deliverd. If a node has Bracha-delivered it means it has forwarded enough messages such that its neighbors can also Bracha-deliver. Since we don't deal with paths, there is no merit in forwarding more messages.

5.3 [M.3] Combining messages

In Bracha-Dolev without signatures, when a node receives a message it needs to forward that message. Depending on whether it has delivered the message, it also needs to create its own ECHO or READY message and broadcast it. Bonomi et al [4] described how to combine the two messages instead of sending both individually by creating new message types, e.g., echo_echo, echo_ready etc.

In our case three things are needed to validate a Dolev-message, the broadcaster, Bracha-message and signature.

Algorithm 1 Dolev with signatures

```
1: function DOLEVBROADCAST(data)
2:     msg ← create empty message
3:     data ← sign data
4:     send message to neighbours
5: end function
6: function ONDOLEVMESSAGE(msg)
7:     if msg has been accepted then
8:         return
9:     end if
10:    if VERIFYMESSAGE(msg) then
11:        accept msg
12:        forward msg
13:    end if
14: end function
15: function VERIFYMESSAGE(msg)
16:    if msg signature is valid then
17:        return true
18:    end if
19:    return false
20: end function
```

Therefore, we can combine multiple messages by appending the broadcaster of the second message, its signature and the type of Bracha-message being sent.

Due to using signatures the number of messages combined together can grow quickly. For example, Figure 2 shows a sub-graph of a network, with the directed edges signifying to direction of messages. if node p_1 Bracha-broadcasts a message to node p_2, p_2 will immediately deliver the message and broadcast another. Node p_2 combines the two messages and sends them to node p_3 and so on. Normally this "chain" happens so easily since nodes who receive the initial message from the broadcaster can immediately broadcast their own echo. Compared to Bracha-Dolev without signatures, combining messages make more sense since there are more opportunities to combine messages.

Combining messages also reduces the bandwidth used by the protocol, since the payload is sent only once in a combined messages, instead of n times, where n is the number of messages combined.

[M.4] Aggregate- and Multi-signatures

An aggregate signatures scheme is a digital signature scheme which allows n signatures to be combined into one signature [3]. Aggregate signatures can contain signatures from n different signers, each signing a distinct message.

On the other hand, a multi-signature scheme is a digital signature scheme which allows n signatures of the same data to be combined into one signature [2]. Unlike aggregate signatures schemes, multi-signature schemes cannot be used to combine signatures of different data.

We can reduce the size of the message by using aggregate- or multi-signature schemes. In this case, instead of having multiple signatures, the message would contain one or signatures depending on whether a aggregate- or multi-signature scheme is used. If a multi-signature scheme is used, there might be one signature for the echo messages and one signature for the ready messages.

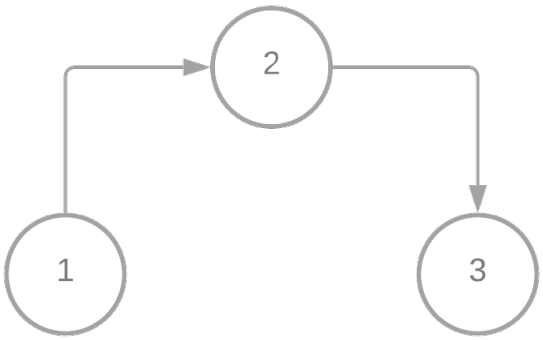

Figure 2: sub-graph of a network

While aggregate and multi-signatures can reduce the bandwidth usage, they also increase the latency. They require more computation compared to normal digital signatures.

Combining signatures without aggregate or multi-signatures allows nodes to decide which signatures to forward. With aggregate or multi-signatures, this is not possible. This might counteract the reduced bandwidth usage.

5.4 Problems with modifying the Bracha layer

Except for M.2, all modifications are for the Dolev layer. This does not mean that it is hard to modify Bracha's algorithm. The problem stems from the fact that the Bracha-layer thinks that it is in a fully connected network, while in reality it isn't. This means that some modifications which could normally be applied to Bracha's algorithm either cannot be applied or are harder to apply when Bracha's algorithm is used in Bracha-Dolev.

6 Evaluation

In this section, we evaluate our modifications and give recommendation on when they should be used.

6.1 Setup

We have implemented a simulation of the improvements made in kotlin, a jvm language. The simulation is publicly available[1]. Each node runs concurrently using coroutines, since hundreds of coroutines can be created without a large cost. The nodes communicate using queues. Each node has its own queue where other nodes can send messages to. In the simulation, Byzantine nodes do not participate in any way in a broadcast. We use a computer with a ryzen 7 3700x processor and 16gb of ram.

We neither limit the bandwidth nor the latency. We first compare our modifications to the state-of-the-art Bracha-Dolev without signatures [4]. Afterwards, we evaluate the impact of each individual optimization while varying the num-

[1] https://gitlab.tudelft.nl/jdecouchant/rp21-group31-5-klaber

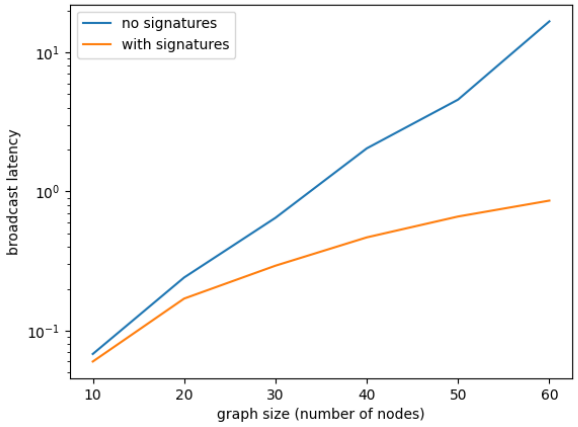

Figure 3: broadcast latency comparison between Bracha-Dolev with signatures and without on random graphs. f = number of nodes / 10, degree = 2*f+1. For the Bracha-Dolev with signatures, M.1, M.2 and M.3 are used.

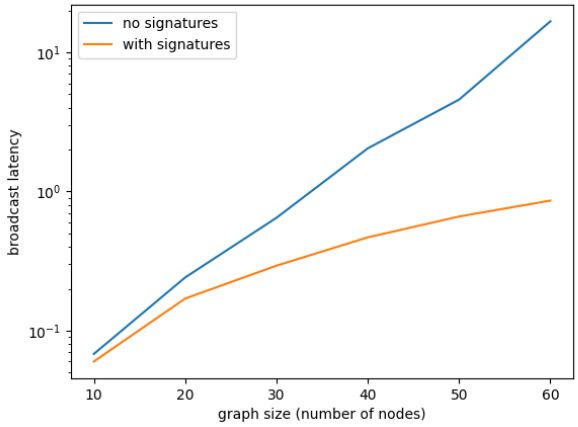

Figure 4: broadcast latency comparison between Bracha-Dolev with signatures and without on random graphs. f = number of nodes / 5, degree = 2*f+1. For the Bracha-Dolev with signatures, M.1, M.2 and M.3 are used.

ber of Nodes and the number of Byzantine nodes. We measure the number of messages exchanged and the time it takes for all correct nodes to deliver the broadcast message. The protocol is non-deterministic, therefore we run experiments 5 times and use the average in our evaluations.

6.2 Results and comparisons

Comparison with state of the art

In this section we compare our modified Bracha-Dolev to the state-of-the-art Bracha-Dolev without signatures [4]. It is important to note that these two protocols have different assumptions, we assume that cryptography is available while Bracha-Dolev without signatures assumes authenticated links.

We compare the message complexity of the two protocols on random graphs while varying the number of nodes in the network and the percentage of nodes that are Byzantine.

We first compare the message complexity of the two protocols with the number of Byzantine nodes set to 10% and 20%. The results can be seen in Figure 5 and Figure 6. We notice that in networks of size 10 where the percentage of Byzantine nodes is 10%, the difference is small. However, in networks of size 10 where the percentage of Byzantine nodes is 20%, there is already a large difference. Starting in networks of size 20, the protocol without signatures already has a message complexity which is more than two times larger than the protocol with signatures. As the network grows, this disparity becomes larger. Eventually, at a network size of 60 the protocol without signatures has a message complexity about 20 times that of the protocol with signatures.

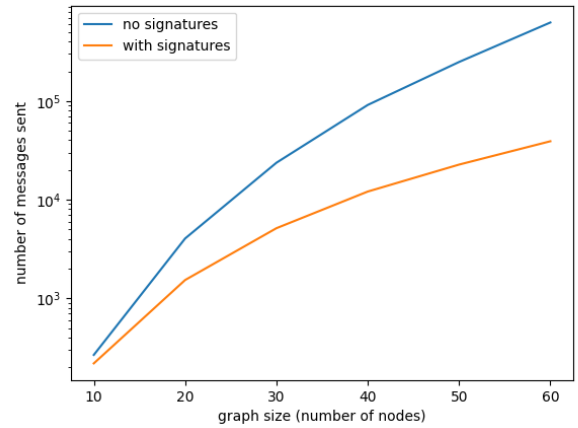

Figure 5: message complexity comparison between Bracha-Dolev with signatures and without on random graphs. f = number of nodes / 10, degree = 2*f+1. For the Bracha-Dolev with signatures, M.1, M.2 and M.3 are used.

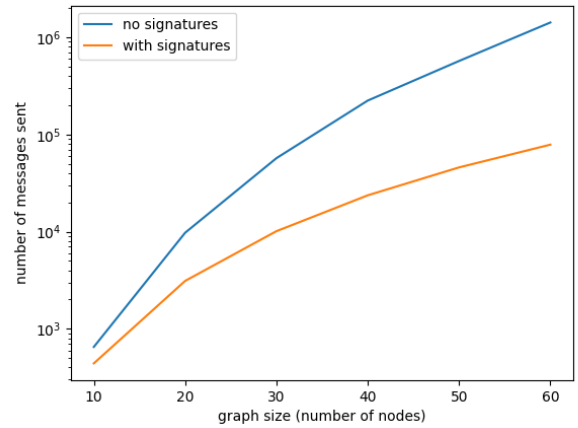

Figure 6: message complexity comparison between Bracha-Dolev with signatures and without on random graphs. f = number of nodes / 5, degree = 2*f+1. For the Bracha-Dolev with signatures, M.1, M.2 and M.3 are used.

We also compare the broadcast latencies of the two protocols. The results can be seen in Figure 3 and Figure 4. In small networks, the two have similar broadcast latencies. However, as the network size grows so does the difference in latencies. In a network of 60 nodes, the broadcast latency of the protocol without signatures is about 15 times that of the protocol with signatures. This disparity mirrors that of the message complexity, therefore broadcast latency is likely due to the differing message complexity of the two protocols.

Evaluation of individual modifications

First, we evaluated the message complexity of each optimization except for M.2 (stop participating after Bracha-deliver). The results are summarized in Table 1. Tables are used due to the small difference in values. We observed that combining messages reduces the message complexity by around between 10 and 27% compared to only signatures. Compared to just combining signatures, using aggregate signatures does not seem to have an impact, results show that it is sometimes much worse and occasionally a tiny bit better than just combining messages. This is not expected since using aggregate signatures is almost the same as just combining messages. When using aggregate signatures nodes might forward a message even if the recipient has signed the message, since aggregate signatures need to know all signers to verify the signature. So it is not possible to split up a message. However even this would not cause the poor message complexity of aggregate signatures.

nodes	signature	combine	combine_aggregate
10	449	362	325
20	2625	2227	2025
30	7420	6668	7203
40	15932	14892	17018
50	30554	27628	29534
60	50989	46799	51399
70	81766	73606	78194
80	124086	108457	107449
90	168598	150752	150405
100	226833	203130	205704

Table 1: number of messages exchanged during a Bracha-Dolev broadcasts without M.2. Each row is 3 separate broadcasts which have different modifications enabled. The signature column is for M.1, the combine column is for M.3 and the combine_aggregate column is for M.4. f = n / 5, connectivity = f+1.

Next, we evaluated the message complexity of each optimization including M.2. The results are summarized in Table 2. Compared to not using M.2, using M.2 reduces the message complexity by 6 and 12 percent. This improvement is larger for combining message than just signatures. This is likely due to the fact that combining messages make some nodes deliver faster since there is more information in one message.We observed that combining messages with M.2 reduces the message complexity by around between 10 and 25% compared to only signatures. This is about the same as without M.2. Compared to without M.2, The results for aggregate signatures are much better and are always lower than just combining message. This is unexpected and might indicate that the broadcast latency has a large impact on using aggregate signatures.

nodes	signature	combine	combine_aggregate
10	397	336	298
20	2340	2022	1926
30	6992	5894	5986
40	15492	13666	13303
50	29000	25428	25106
60	48652	41969	41809
70	75620	67366	66373
80	111044	99068	97984
90	156092	138020	136901
100	211872	189512	188174

Table 2: number of messages exchanged during Bracha-Dolev broadcasts with M.2. Each row is 3 separate broadcasts which have different modifications enabled. The signature column is for M.1, the combine column is for M.3 and the combine_aggregate column is for M.4. f = n / 5, connectivity = f+1

nodes	signature	combine	combine_aggregate
10	0.017	0.0154	1.62
20	0.075	0.059	4.54
30	0.16	0.14	11.8
40	0.25	0.26	22
50	0.38	0.42	32.82
60	0.56	0.61	45.41
70	0.84	0.87	60.65
80	1.15	1.16	75.14
90	1.43	1.52	91.32
100	1.71	1.9	117.37

Table 3: Broadcast latencies for Bracha-Dolev broadcasts. Each row is 3 separate broadcasts which have different modifications enabled. The signature column is for M.1, the combine column is for M.3 and the combine_aggregate column is for M.4. f = n / 5, connectivity = f+1

To investigate effect of M.2 on aggregate signatures, we also evaluated the broadcast latency of the protocol, i.e., how long it takes for all correct nodes in the network to Bracha-deliver. Table 1 summarizes the broadcast latencies without M.2 and Table 2 summarizes the broadcast latencies with M.2. We observe that latencies of signatures and the normal combine decrease by around 10% when comparing M.2 and no M.2. We also observe that the latency improvements of aggregate signatures are much more varied, from around 0 to around 13%. Additionally, the latencies for aggregate signatures are much larger compared to signatures and combine. This is likely due to the poor performance of creating and verifying aggregate signatures compared to ”normal” signatures. The combination of the concurrency of the simulation, the poor performance of aggregate signatures and the fact that aggregate signatures cannot be easily split, is likely the cause behind the difference between M.2 and no M.2. Presumably some nodes are taking longer to Bracha-deliver, due to not having enough computing power.

nodes	signature	combine	combine_aggregate
10	0.019	0.016	1.6152
20	0.063	0.055	4.7092
30	0.14	0.12	10.02
40	0.24	0.26	18.55
50	0.36	0.39	26.2
60	0.54	0.58	37.1
70	0.73	0.81	50.14
80	0.95	1.12	66.8
90	1.3	1.42	88.49
100	1.52	1.77	102.82

Table 4: Broadcast latencies for Bracha-Dolev broadcasts. Each row is 3 separate broadcasts which have different modifications enabled. The signature column is for M.1, the combine column is for M.3 and the combine_aggregate column is for M.4. f = n / 5, connectivity = f+1

6.3 Recommendations

Using signatures dramatically reduces the message complexity and the broadcast latency, therefore signatures should always be used unless unavailable. Modification M.2 should also always be used. It has zero downsides while giving a modest improvement. Modification M.3 can increase the individual message sizes. Therefore, depending on how messages are handled, this may decrease performance in low bandwidth environments. Modification M.4 reduces the message sizes compared to M.3 in exchange for less flexibility and increasing the computational resources needed. Unless bandwidth is extremely important, M.4 should not be used. Even when bandwidth is important, M.3 should be first optimized and bench-marked before considering M.4.

7 Responsible Research

This paper does not have many ethical problems. However, problems can arise when the algorithm described in this paper is used in other applications, such as Blockchain. In that case problems with the algorithm can have financial implications.

All results obtained in this paper should be simple to reproduce, as the simulation code in publicly available online. Protocols used to benchmark our solution are were reimplemented in the environment used by our protocol. These protocols are also publicly available. Only the tests that used random networks cannot be perfectly replicated, however the difference in the result should be small.

8 Conclusion

In this paper, we discussed Byzantine reliable broadcast on partially connected networks when nodes can sign messages. We described how signatures can be used by Bracha-Dolev to reduce its message complexity. We describe a total of 4 modifications which reduce the message complexity or the bandwidth usage. We evaluate each modification, discussed the results and gave recommendations on when to use each modification. Our modifications reduce the broadcast latency by 15 times and the message complexity by 20 times when $N = 60$ and $f = 6$, compared to the state-of-the-art Bracha-Dolev without signatures. We also reflected on the ethical aspects of the research. Future works include combining Trusted Execution Environments, signatures and sharding to further improve Bracha-Dolev.

References

[1] Amos Beimell and Matthew Franklin. Reliable communication over partially authenticated networks. In Marios Mavronicolas and Philippas Tsigas, editors, *Distributed Algorithms*, pages 245–259, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[2] Mihir Bellare and Gregory Neven. Identity-based multi-signatures from rsa. In Masayuki Abe, editor, *Topics in Cryptology – CT-RSA 2007*, pages 145–162, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[3] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *Advances in Cryptology — EUROCRYPT 2003*, pages 416–432, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[4] Silvia Bonomi, Jérémie Decouchant, Giovanni Farina, Vincent Rahli, and Sébastien Tixeuil. Practical byzantine reliable broadcast on partially connected networks, 2021.

[5] Silvia Bonomi, Giovanni Farina, and Sébastien Tixeuil. Multi-hop byzantine reliable broadcast made practical. *CoRR*, abs/1903.08988, 2019.

[6] Gabriel Bracha. Asynchronous byzantine agreement protocols. *Inf. Comput.*, 75(2):130–143, November 1987.

[7] Christian Cachin, Rachid Guerraoui, and Luis Rodrigues. *Introduction to Reliable and Secure Distributed Programming (2. ed.).* 01 2011.

[8] Daniel Collins, Rachid Guerraoui, Jovan Komatovic, Petr Kuznetsov, Matteo Monti, Matej Pavlovic, Yvonne-Anne Pignolet, Dragos-Adrian Seredinschi, Andrei Tonkikh, and Athanasios Xygkis. Online payments by merely broadcasting messages. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 26–38, 2020.

[9] D. Dolev. Unanimity in an unknown and unreliable environment. In *22nd Annual Symposium on Foundations of Computer Science (sfcs 1981)*, pages 159–168, 1981.

[10] Zijiang Hao, Raymond Ji, and Qun Li. Fastpay: A secure fast payment method for edge-iot platforms using blockchain. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 410–415, 2018.

[11] Alexandre Maurer, Sébastien Tixeuil, and Xavier Défago. Reliable communication in a dynamic network in the presence of byzantine faults, 2015.

[12] Ye Wang and Roger Wattenhofer. Asynchronous byzantine agreement in incomplete networks. In *Proceedings of the 2nd ACM Conference on Advances in Financial*

Technologies, AFT '20, page 178–188, New York, NY, USA, 2020. Association for Computing Machinery.