Decreasing message complexity in Byzantine Fault Tolerant communications using Consistent-Broadcast

Daniel Prinsze
Supervisor(s): Bart Cox, Jérémie Decouchant
EEMCS, Delft University of Technology, The Netherlands

23-6-2022

A Dissertation Submitted to EEMCS faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering

***Abstract*— During this research we have replaced Bracha's layer in the state-of-the-art Bracha-Dolev protocol to improve the performance by decreasing the message complexity of the protocol running on top of a given network topology so long as the requirements stated by Bracha and Dolev are met. Bracha-Dolev is an algorithm that is used to establish a Byzantine fault tolerant communication in a network but it requires a lot of messages to reach consensus. This improvement has been achieved by utilizing a Byzantine Consistent Broadcast algorithm in place of Bracha's layer: Authenticated Echo Broadcast in order to reduce the message complexity and reduce the network consumption compared to the original optimized Bracha-Dolev algorithm. Some of the optimizations applied to optimized Bracha-Dolev have also been applied to this new protocol under the hard assumption that the sender is always reliable. As a result, this new protocol is an optimal choice in such instances where these constraints hold and where multiple faulty nodes are present in the system.**

I. Introduction

In the communication between multiple processes in a distributed network, it might be that some nodes perform in an adversarial manner that will inhibit the functionality of the whole system. Behaving in such a manner is usually caused by internal faults, simple errors in communication over a network (interference) or malicious intent. Such processes are labeled Byzantine processes, referring to the Byzantine generals problem which is a famous communication problem that is used as an analogy to describe the issues with achieving consensus in a distributed network [8].

Reliable communication in a distributed network by means of achieving consensus is of great importance for the process of verification in transactions or distributing workload between remote systems amongst other use-cases. As such a number of different protocols have been developed to achieve this consensus taking into account these types of faults. The goal of these protocols is to achieve a Byzantine fault tolerant (BFT) network where consensus can be reached even though Byzantine processes are present. Previous works have proven that independent of network topology or model, in order to reach consensus there exist two upper bounds on the number of faulty processes f that may be present in a network where $f < \frac{n}{3}$ and where the connectivity of the network is at least $2f + 1$ when utilizing Dolev's algorithm [6] which also hold for asynchronous randomized network configurations [14].

Realizing a consensus in a network can be done by combining the protocol proposed by Gabriel Bracha [4] and the protocol proposed by Danny Dolev [6] where each protocol will fulfil a specific function in their respective layer.

Previous works have optimized or proposed optimizations for these protocols since Dolev's and Bracha's algorithms need to send a lot of messages in order to achieve this consensus which can prove to be problematic regarding the amount of time it takes when considering a large scale distributed network and many other optimizations have been proposed to improve distributed communication in the presence of byzantine processes [13][9][7][1][3][10][2]. The larger a distributed network of systems becomes, the more computational power is demanded. As a result, the energy consumption will increase which makes this a costly endeavor for long and computationally intense tasks.

As of today and to our knowledge there are no available works yet that have tried replacing Bracha's algorithm with an algorithm of the consistent broadcast paradigm while imposing hard constraints in order to reduce message complexity of a system that utilizes Bracha-Dolev to communicate.

This paper will try to extend the research done by Bonomi et al. [2] while imposing some constraints that will contain the research within its intended scope. *Byzantine consistent broadcast* (BCB) poses a very likely group of candidates to replace Bracha's layer. The algorithm of particular interest is the *Authenticated Echo Broadcast* (AEB) algorithm [5] since it drops the 'Ready' message type in its entirety and is not as computationally intensive as other options within the same group (i.e. due to generating digital signatures). Using this algorithm, it is theoretically possible to achieve a message complexity of $O(n^2)$ messages required given that the sender is always reliable, thereby providing a significant reduction on required resources where the only hard requirement made in such a set-up would be that the sender always has to be reliable [5].

This study has set-out to provide concrete data regarding whether or not using Dolev in conjunction with BCB-Dolev in a network with a reliable sender provides a significant enough speed-up and decrease in message complexity in a real life simulation and to provide conclusive results that warrant further study and examination. Furthermore this paper has sought to answer the following questions:

i. Is it possible to reduce the message complexity in Bracha-Dolev utilizing consistent broadcast given that the sender is always reliable?
ii. Is this strategy valid and is it correct? Does it ensure consensus?

In section II we will talk more about the related work and the ideas that led to researching and implemention this new form of a BFT protocol; BCB-Dolev.

In section III we will explain the system model that has been used, the necessary interface, the required constraints and the properties resulting of these settings.

In section IV we will discuss the methodology of this research in which a deeper look will be taken into the hardware that has been used, the experimental setup that is required in order to execute the experiment properly, what choices have been made regarding the experimental setup, how these choices were validated and how the experiment can be recreated.

In section V we will discuss the results, findings and contributions of this research, whether or not these results match our theoretical expections and evaluate and argue why our results are correct. In section VI we will discuss how the research was conducted in a responsible manner and subsequently in section VII we will delve into the conclusions and possible future works and-or improvements that could be made.

II. Related work

The optimized Bracha-Dolev protocol consists of two protocols of different paradigms to create an algorithm that is able to establish a BFT communication. The first

group of protocols that includes Bracha's algorithm is the group of Byzantine Reliable Broadcast (BRB) algorithms and the second is the group of Reliable Communication (RC) protocols. Typically, BRB protocols allow for correct processes to all decide whether or not to accept a message, even if the sender is faulty. Whereas RC protocols allow processes to authenticate messages in a partially connected protocol, and provides to each process the illusion that it is directly connected to all other processes.

Byzantine Reliable Broadcast: Bracha's work [4] has proposed a solution to the problem of achieving consensus in a network where Byzantine processes are present and has done so by creating a protocol that makes use of three different message types: `INIT`, `ECHO` and `READY`, that will propagate in three different phases throughout the entire network as seen in Figure 1. It is important to note however that reliable nodes only relay a message if and only if it has received one of the `INIT` messages, $\frac{n+f}{2}$ of the `ECHO` messages and $f+1$ messages of the `READY` type. A hard bound set by Bracha is that there should be at most $f < \frac{n}{3}$ Byzantine processes in the system.

Reliable Communication: Whereas Bracha's protocol only works in fully connected networks, the protocol proposed by Dolev should be able to establish a reliable connection link in both synchronous and asynchronous networks as long as the graph has a connectivity of at least $2f+1$ [6][14]. It manages to do so by looking for $f+1$ disjoint paths as depicted in Figure 2.

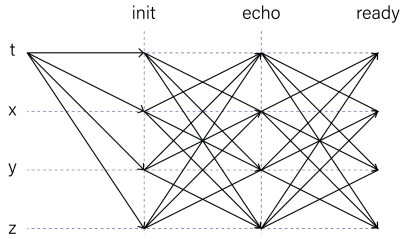

Fig. 1: Bracha's communication protocol utilizing three phases to distribute a message sent by t

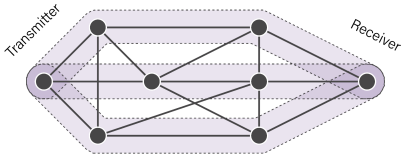

Fig. 2: Dolev's protocol establishing $f+1$ disjoint paths in the presence of 2 faulty nodes

The problem with both of these protocols is that they have a high message complexity and in Dolev's case there is also a high computational complexity required to calculate the disjoint paths it needs to find. In order to mitigate this problem, we have have proposed a different implementation under the constraint that the original transmitter is always reliable through the means of the consistent broadcast paradigm. The work done by Ramasamy et al. implemented a more efficient *atomic broadcast* algorithm that reduced the message complexity from $O(n^2)$ to amortized $O(n)$ utilizing a consistent broadcast algorithm [10]. This has proven the potential and the feasibility for the algorithm to be implemented in conjunction with Dolev's algorithm instead of Bracha's.

The algorithm that Ramasay et al. [10] has employed in their work is known as *Authenticated Echo Broadcast* (AEB) which has been developed by Srikanth et al. [12]. This algorithm is known to implement consistent broadcast in quadratic message complexity $O(n^2)$ whilst only requiring two of the three message types that Bracha's original algorithm utilizes. Much like Bracha's algorithm, it first *c-broadcasts* the message m to all nodes and then when the amount of faulty nodes in the system is $f < \frac{n}{3}$, every correct process waits until it has received $\frac{n+f+1}{2}$ `ECHO` messages (in other words, a Byzantine quorum has been reached) from other nodes before it finally will *c-deliver* the original message m as depicted in Figure 3.

A different algorithm within the same consistent broadcast paradigm is know as *Signed Echo Broadcast* (SEB) or simply *Echo Broadcast* which has been developed by M. K. Reiter [11]. Unlike AEB, SEB utilizes digital signatures to establish a secure connection in a network and manages to do so in linear message complexity $O(n)$. Just as with AEB, SEB also first sends a message m to all network participants and waits until a Byzantine quorum is reached. However, instead of propagating this `ECHO` message, each process digitally signs this message and returns it to the sender t where t then awaits a Byzantine quorum of these signed statements before it relays them to all processes in the network in the final step as depicted in Figure 4.

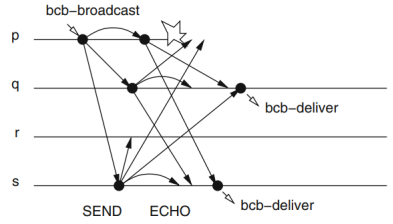

Fig. 3: Authenticated echo broadcast establishing a connection [5]

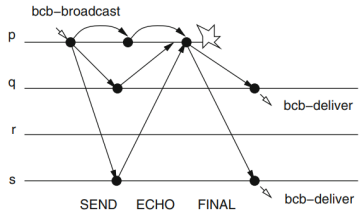

Fig. 4: Signed echo broadcast establishing a connection [5]

The work done by Aguilera et al. [1] has provided even further information about whether or not the consistent broadcast paradigm is useful compared to the group of BRB protocols when considering the utilization and implementing of digital signatures. They have shown that digital signatures are more costly than simple network communication and

that the same holds for the sending of a lot of replicas, a characteristic of Bracha-Dolev's protocol.
This research has considered all the previous works and both possible implementations and has determined that it would be optimal to combine AEB with Dolev instead of SEB since SEB will require more computational resources which, depending on the systems that utilize it, will increase the latency due to the computational complexity resulting from establishing these digital signatures. Further more, the strategy of implementing AEB with Dolev should reduce the message complexity substantially already, making the increased computational load and latency an undesirable side-effect.

III. System model and problem statement

The current state of the art Bracha-Dolev communication protocol is the one provided by Bonomi et al. [2]. In their work they provide an optimized version of Bracha-Dolev that, based on certain circumstances such as network topology and connectedness, can provide an improvement in performance compared to the regular version of the protocol. During this experiment the focus will be on comparing Optimized Bracha-Dolev versus an implementation of BCB-Dolev that utilizes AEB with regards to throughput and latency as a result of reduced message complexity.
In order to implement AEB in the form of BCB-Dolev, some constraints must be applied to the system model for the protocol to function as desired. We will adhere to a similar way of declaring this model as Bonomi et al. [2].
Nodes: Each graph will consist of n nodes (or processes) that will be denoted or referenced using the following notation: $\mathbb{P} = \{t, p_1, p_2, ..., p_{n-1}\}$ where p_i is the unique identifier of each subsequent non-initial transmitter node, p_1 through p_{n-1} are the elements (processes) of the set of nodes ($\mathbb{P}$) in the graph and t the initial transmitter of a given message (or payload). For the nodes in the system model it is assumed that t is always reliable as a transmitter and that the amount of faulty nodes in the system f is always less than a third of the amount of nodes in the system: $f < \lfloor \frac{n}{3} \rfloor$.
Connections: Each graph will be denoted as $G = (V, E)$ where V includes the set $\mathbb{P}$ as to indicate that all nodes in $\mathbb{P}$ are part of the given presented graph such that $\mathbb{P} \subseteq V$ and $\forall 0 < i < n : t \in V \wedge p_i \in V$. A hard requirement in order to utilize AEB states that we will assume all connections in E are established using *authenticated perfect point-to-point links* [5]. Topologies can either be known or unknown but we will assume an unknown topology where there is a hard requirement that each $v \in V$ has a connectivity of at least $2f + 1$. We will also assume the rest of the network edges (or connections) to be reliable such that messages will always be delivered, whether Byzantine or not. Figure 5 provides an example for a topology with eight nodes and a connectivity of at least three.
Byzantine Consistent Broadcast (BCB): Utilizing consistent broadcast we construct an interface that can guarantee a set of properties inherent to the protocol [5]. This interface contains two mains *Events*:

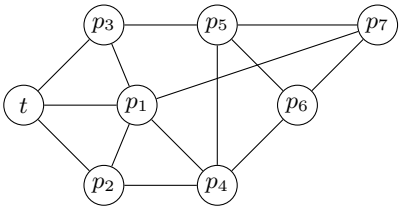

Fig. 5: An example graph with $n = 8$ can $f = 1$ where transmitter t will be able to reliable deliver the message since the connectivity is larger or equal to $2f + 1$

1) *Request:* a transmitter t and only transmitter t broadcasts to all other nodes p_i
2) *Indication:* whenever a node $v \in V$ broadcasts a payload m, it is delivered.

The following properties are guaranteed when using this interface:
[BCB1]: *Validity:* Whenever a correct process $v \in V$ broadcasts a message m then eventually every correct process delivers m.
[BCB2]: *No duplication:* Every correct process delivers at most one message.
[BCB3]: *Integrity:* If some correct process delivers a message m with sender p_i and process p_i is correct, then m was previously broadcast by p_i.
[BCB4]: *Consistency:* If some correct process delivers a message m and another correct process delivers a message m', then we can conclude that $m = m'$.
Reliable Communication: Following the BCB interface we need another interface that will represent Dolev's layer in the BCB-Dolev implementation. For this an *Reliable Communication* (RC) interface has been assumed with the same interface. Given the initial assumption that $t \in V$ where t is always a reliable sender, every property of the BCB interface is still valid and as such a correctly defined system model has been constructed.

IV. Methodology

In order to execute the experiment it should be possible to keep track of important metrics from which we can extract a definitive conclusion regarding the optimality of the implementation compared to optimized Bracha-Dolev. Such metrics include the amount of messages sent, the amount of bytes a payload contains, the amount of broadcasts executed, the times at which each broadcast was executed and completed and the amount of deliveries made. The amount of messages sent will show whether or not the implementation has a better message complexity than optimized Bracha-Dolev, the size of the payload will give more insight into the network consumption and load on the network as well as the throughput in conjunction with the amount of messages sent and the broadcast times should provide insight into the latency of the algorithm as a whole.
During the experiment we measure these metrics in an existing network. Such a network has to be instantiated, either by using different nodes over vast distances or by using isolated processes on a local network. This research has opted to use isolated instances of Docker processes in

order to emulate different nodes on a local network. The experiment could be performed using seperate systems as well but this would require additional measures to ensure integrity. However, it will probably still have the same ratios as the setup that uses the Docker instances.

In order to run the experiment we make use of Salticidae which is a small and compact network library that offers handlers and bindings that we can utilize to implement the different phases of the communication protocol we have implemented as well as basic initiators and terminators for the communication in our simulated network. The implementations have been written in C++ and the scripts that execute them and compile the date have been written in Python. For C++, version 9.4.0 of the GCC compiler has been used and for Python version 3.8.10 of the interpreter. For Docker we have used version 4.9.0. No compiler flags have been set in the Makefile other than the linking to the Salticidae libraries. The experiment has been executed over a wireless local network using a computer that accomodates 16GB of random access memory and 4 cores clocked at 1.7GHz. This has influenced the results because Salticidae makes use of multithreading.

The algorithm that has been implemented is a combination of both Dolev's algorithm as well as *Authenticated Echo Broadcast* (AEB). The pseudocode for the AEB algorithm that has been used to implement BCB-Dolev is provided in Algorithm 1. The setup makes sure that the initial broadcast performed by the sender t is performed by sending a Dolev message to a node p_i in the network. Then if this node p_i is reliable, that node will execute the AEB algorithm in order to deliver the message m to it's neighbours. If the node p_i is not reliable then we request to relay the message using Dolev's algorithm to a node p_j instead. This process is repeated until a reliable node is found. The pseudocode for Dolev's protocol can be found in Algorithm 2.

In order to recreate the BCB-Dolev implementation Algorithm 1 and 2 can be combined through replacing the forall triggers in Algorithm 1 on lines 14-15 and 19-20 with the broadcast operation $\langle dolev, \texttt{BROADCAST} \mid [msgType, m]\rangle$ in Algorithm 2 on line 9-13. It is important to note that the message type in the Dolev broadcast event should also be passed where $msgType \in \{\texttt{INIT}, \texttt{ECHO}\}$. Furthermore, the $\langle al, \texttt{DELIVER} \mid q, [msgType, m]\rangle$ in Algorithm 1 should be replaced by $\langle dolev, \texttt{DELIVER} \mid [msgType, m]\rangle$. The resulting protocol should be a valid and correct implementation of BCB-Dolev.

The work done by Bonomi et al. [2] also provides some optimizations for their algorithm. However, some are applicable to BCB-Dolev as well and as such have been applied during the execution of this experiment. These optimizations include:

[MD.1]: If a process p_i receives a message m directly from the transmitter t, then p_i directly delivers it.

[MD.2]: If a process p_i has delivered a message m, then it can discard all the related paths and relay the message only with an empty path to all of its neighbours.

[MD.3]: A process p_i relays path related to a message only to the neighbours that have not delivered it.

Algorithm 1: *Authenticated Echo Broadcast (AEB) in the presence of a reliable transmitter*

```
1  Parameters:
2        Π: the set of all nodes in the network
3        n: the amount of nodes in the network
4        f < ⌊n/3⌋: maximum amount of Byzantine nodes
5        t: the transmitter such that t ∈ Π
6  Uses: Auth, perfect point-to-point links, instance al
7
8  upon event ⟨bcb,INIT⟩ do
9        sentEcho := FALSE
10       delivered := FALSE
11       echos := ∅
12
13 upon event ⟨bcb, BROADCAST | m⟩ do
14       forall e ∈ Π do
15             trigger ⟨al, SEND | e, [SEND,m]⟩
16
17 upon event ⟨al, DELIVER | q, [SEND,m]⟩ such that
     q = t ∧ sentecho =FALSE do
18       sentecho := TRUE
19       forall e ∈ Π do
20             trigger ⟨al, SEND | e, [ECHO, m]⟩
21
22 upon event ⟨al, DELIVER | q, [ECHO, m] do
23       if echos[q] = ∅ then
24             echos[q] := m
25
26 upon exists m ≠ ∅ such that
     #({∀e ∈ Π|echos[e] = m}) > (n+f)/2 ∧ delivered =FALSE do
27       delivered := TRUE
28       trigger ⟨bcb,DELIVER | t, m⟩
```

Algorithm 2: *Dolev's protocol for a 2f+1 connected network at process e_i*

```
1  Parameters:
2        f: maximum amount of Byzantine nodes
3  Uses: Auth, perfect point-to-point links, instance al
4
5  upon event ⟨dolev,INIT⟩ do
6        delivered := FALSE
7        paths := ∅
8
9  upon event ⟨dolev, BROADCAST | m⟩ do
10       forall e_j ∈ neightbours(e_i) do
11             trigger ⟨al, SEND | e_j, [m, []]⟩
12       delivered := TRUE
13       trigger ⟨dolev, DELIVER | m⟩
14
15 upon event ⟨al, DELIVER | e_j, [m, path]⟩ do
16       paths.INSERT(path + [e_j])
17       forall e_k ∈ neighbours(e_i) \ (path ∪ {e_j}) do
18             trigger ⟨al, SEND | e_k, [m, path + [e_j]]⟩
19
20 upon event (e_i is connected to the source through f + 1 node-disjoint paths
     contained in paths) ∧ delivered = FALSE do
21       trigger ⟨dolev, DELIVER | m⟩
22       delivered = TRUE
```

[MD.5]: A process p_i stops relaying further paths related to a message after it has been delivered and the empty path has been forwarded.

A set of newly provided modifications has also been accomodated when running the experiment. Notably:

[MBD.3]: *Echo to Echo transitions:* When a process p_i receives an ECHO message from a neighbour p_j, p_i will then Dolev-deliver the message in line with [**MD.2**] and will forward the message to its neighbours whilst excluding the paths. As a result of delivering the ECHO message, process p_i might also send another ECHO message to all of its neighbours after having delivered the SEND message of the source, or received $f + 1$ ECHOS.

[MBD.7]: Ignore all received ECHO messages related to the contents of a message that has been delivered.
[MBD.9]: Do not send anything related to content to neighbours that have successfully delivered that content.
[MBD.10]: If a node p_i receives a message m_1 (e.g., an *ECHO* from t) with path $path_1$, for which a previous message m_0 with $path_0$ was received (and which only differs from m_1 by its path) and such that $path_0$ is a subpath of $path_1$, then p_i can ignore m_1.
[MBD.12]: If the transmitter t has more than $2f + 1$ neighbours, it can transmit its SEND message to $2f + 1$ of its neighbours instead of to all of them.
The experiment itself has been executed with a message payload of 10 Bytes for a set of different connectivities $[4, 6, 8, 10, 12, 14, 16]$, while having a set of faulty nodes present $[1, 3, 5]$ where the number of nodes in the system has been chosen to be constant $n = 31$. The experiment has been performed whilst always having [**MD.1**] through [**MD.5**] enabled and both with and without applicable optimizations provided by Bonomi et al. [2].

V. Contributions, Findings and Results

This research has implemented a fully functional and correct version of the protocol proposed in Section IV by combining the *Authenticated Echo Broadcast* protocol with Dolev's protocol in conjunction with an expandable code base. Furthermore, this research has shown the correctness and validity of this combination and that this protocol adheres to all necessary properties in order for it to be Byzantine fault tolerant in Section III. The full implementation and codebase can be found on Gitlab.
During our experimentation we found that our implementation of BCB-Dolev greatly reduces the message complexity compared to the original unoptimized Bracha-Dolev protocol. When applying all proposed optimizations by Bonomi et al. [2] to Bracha-Dolev and BCB-Dolev where applicable we found that our implementation of BCB-Dolev has a more efficient message complexity in systems with more faulty nodes.
First we will evaluate the results from the first experimental phase. Figure 6 depicts the results required after executing the first phase of the experiment for the unoptimized BCB-Dolev protocol. In this figure, we can see the average amount of messages sent per broadcast cycle for connectivities $[4, 6, 8, 10, 12, 14, 16]$ with $f = 1$, $f = 2$ and $f = 3$ faulty nodes in a system of 31 nodes with 75 broadcast cycles. It becomes clear that there is a correlation between the connectivity of a network, the amount of faulty nodes and the amount of messages sent during a broadcast cycle. This correlation is not surprising because of how the algorithm works. Whenever the transmitter sends a Dolev message to a reliable node, that node then starts a BCB-Dolev broadcast. This BCB-Dolev broadcast will then send messages to all its connected neighbours. As such it is logical that the higher the connectivity, the more messages will be sent and as such these results are in line with our expectations.
Figure 7 shows the average amount of messages with the same parameters but now for the unoptimized Bracha-Dolev protocol. As with BCB-Dolev it becomes clear that again there is a correlation between the connectivity of a network, the amount of faulty nodes and the amount of messages sent during a broadcast cycle and for the same reasons this matches our expectations. However, the average amount of messages sent per cycle is far greater than we saw in BCB-Dolev. This can be mostly attributed to the fact that we have three message phases instead of two where, again depending on the connecitivity, messages will be sent to each other connected node. As a result these message complexities will be substantially higher and as such these results were in line with our expectations.

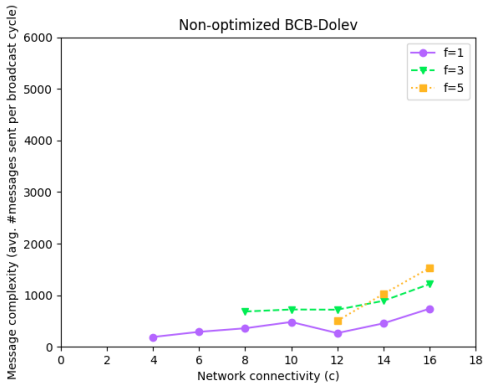

Fig. 6: *Message complexity BCB-Dolev, n=31, 75 broadcasts executed*

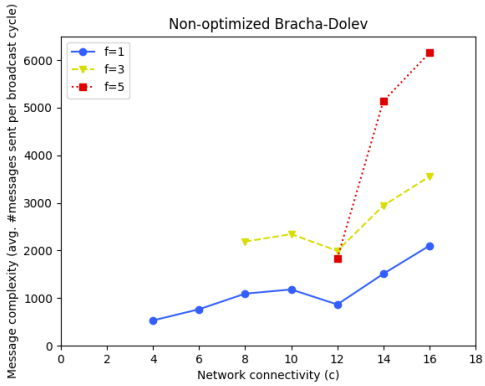

Fig. 7: *Message complexity BRB-Dolev, n=31, 75 broadcasts executed*

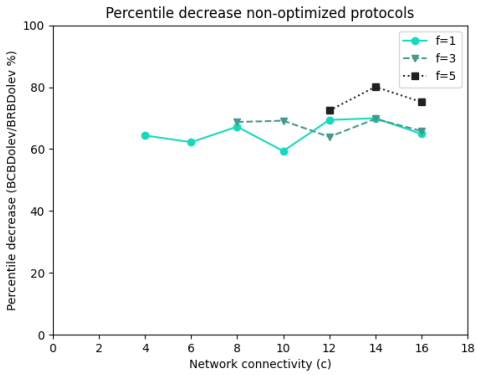

Fig. 8: *Percentile decrease of BCB-Dolev v.s. BRB-Dolev*

During the evaluation of the experimental results of the unoptimized BCB-Dolev and BRB-Dolev protocols it becomes clear that unoptimized BCB-Dolev is, quite strictly, the better option in terms of lowest message complexity, boasting an average message complexity reduction of $\sim 65\%$ for $f = 1$, about the same for $f = 2$ and an average of about $\sim 75\%$ for $f = 3$ as seen in Figure 6, 7 and 8. Whilst these reductions are impressive, these results are not very surprising given the fact that original Bracha-Dolev utilizes all three phases of Bracha's communication algorithm which causes a factor of about $O(n^2)$ more messages to be sent on top of what our implementation requires. Furthermore, the initial broadcast in our implementation propogates an instance of a Dolev message throughout the network until it has found a reliable node that can propogate the BCB-Dolev broadcast. As a result, the two phases of the BCB broadcast do not have to propogate throughout every node in the network unlike with BRB-Dolev. These factors both play a big role in the size of the reduction in message complexity and were expected based on the theory presented in Section II and Section III. In the second phase of our experiment, we applied all proposed optimizations to Bracha-Dolev and compared it to our partially optimized BCB-Dolev algorithm, using a part of the same optimizations and using the same settings as in the first phase.

Just as during the first phase we can deduce that the results are in line with our expectations due to having applied all optimizations **MBD.1** through **MBD.12** to Bracha-Dolev which has significantly reduced the amount of messages sent. While the optimizations applied to BCB-Dolev have not posed a decrease as severe as with Bracha-Dolev, it is still in line with our expectations due to the various applied optimizations. It becomes quite clear that the reduction becomes less severe and in some cases our implementation becomes less efficient. From Figure 9, 10 and 11 we can derive that for $f = 1$ our BCB-Dolev implementation requires up to $\sim 55\%$ more messages than the optimized Bracha-Dolev protocol. However, we notice that for higher amounts of faulty nodes in the system, our implementation does perform better with regards to the message complexity where for $f = 2$ we achieve an average decrease of about $\sim 30\%$ and for $f = 3$ an average decrease of about $\sim 40\%$ over all connectivities. The fact that our implementation has a higher message complexity compared to the optimized Bracha-Dolev protocol can be largely attributed to the other optimizations applied to it that were not applicable to our implementation of BCB-Dolev. More interesting are the results regarding networks with higher amounts of faulty processes which is when our implementation seems to again provide a higher degree of efficiency over optimized Bracha-Dolev.

As can be seen in the figures that present that data there seem to be dips in message complexity for a set of differing connectivities per version on the protocol that was being tested. It was expected that the higher the connectivity the higher the message complexity. However, during testing it became clear that the optimized and non-optimized versions of Bracha-Dolev weren't able to execute all 75 broadcasts effectively. The same holds for the optimized and non-optimized versions of BCB-Dolev. However, the performance of our implementation

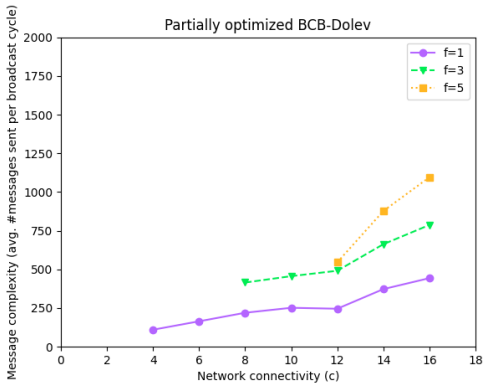

Fig. 9: *Message complexity optimized BCB-Dolev, n=31, 75 broadcasts executed*

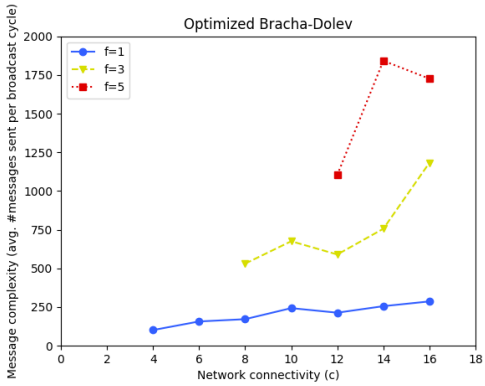

Fig. 10: *Message complexity optimized BRB-Dolev, n=31, 75 broadcasts executed*

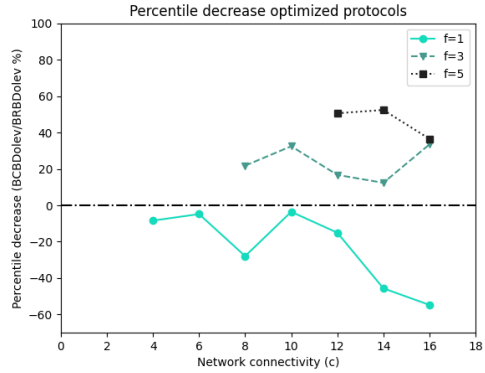

Fig. 11: *Percentile decrease of Opt BCB-Dolev v.s. Opt BRB-Dolev*

was less affected overall. These anomalies were happening consistently across all test phases and were mostly present when the connectivities were set to $c = 12$, $c = 14$ and $c = 16$ for optimized Bracha-Dolev and $c = 12$ for non-optimized and optimized BCB-Dolev as well as non-optimized Bracha-Dolev. The failure to execute all broadcasts effectively causes the message complexity on average to be lower than it should be. Currently, it is not clear what the cause of this failure to execute all broadcasts for these settings is. However, since the failure was consistent in all testing, the acquired data was still deemed usable. In fact, since the experiment was mainly designed to determine whether or not our implementation

provides a significant decrease in message complexity, the fact that our optimization was able to perform all 75 broadcasts with more consistency and still achieved a lower average message complexity compared to the optimized Bracha-Dolev protocol despite finding lower numbers because of the failure to execute all broadcasts, shows that our new implementation is indeed more efficient by a secure margin. These findings make BCB-Dolev a good candidate for future research and use.

VI. Responsible research

The experimental setup has been briefly discussed in section IV. In this section we will dive deeper into how the experiment was executed in a responsible manner and how to recreate the results using the implementation that has been created for this project. First and foremost, all code and files that have been used to draw the conclusions made in this paper are available on Gitlab.

During this research the only concern was to get results from performing the experiment with the presented implementation and the implementation derived from the works by Bonomi et al. [2] regardless of whether it matches our expections or not to ensure integrity of the research. All the algorithms have been implemented in this codebase to the best of our abilities. However, it was found that for some connectivities, network protocols were unable to successfully complete all broadcast cycles. This however is probably a rather subtle bug in the code base as the problem was consistent throughout all testing and as such has not impacted our ability to interpret the results. In order to acquire results one should run the `testDockerLocal.py` file with the settings that one wants to test. The averages as well as the minimum and maximum values are printed to the console for all metrics mentioned before. All the log files will be stored in `code3/log` and can be accessed after having completed the experiment with the desired settings. Due to hardware limitations, all experimentation data has been logged manually after each execution of the main test file.

It is important to note that the experiment has some limitations. First of all, the experiment was performed on commercial hardware due to limitations that could not be solved in the short amount of time available for this project. While the chosen experimental setup should in theory provide a close enough analogy for a real network with multiple independent nodes, it would be optimal to actually perform the experiment in such fashion. Because automated test scripts would cause the hardware it was performed on to crash, all tests were done manually, logging the results to ensure integrity of the acquired data.

VII. Conclusion

This paper has provided a functional implementation for BCB-Dolev with a set of optimizations that were applicable from the work this research has extended. We have accumulated, presented and argued our results in Section V where we have shown that by replacing Bracha's layer with *Authenticated Echo Broadcast* and thus creating a new protocol; BCB-Dolev, we can significantly reduce the message complexity in a network compared to using optimized Bracha-Dolev. Our partially optimized implementation of BCB-Dolev is more efficient regarding message complexity compared to the fully optimized Bracha-Dolev protocol provided by Bonomi et al. [2] when the amount of faulty nodes in the system is larger than $f = 1$ even when not all broadcasts performed by optimized Bracha-Dolev were effective. We have shown that this implementation, under the strong assumption that the sender is always reliable, is correct and ensures consensus and we have shown that the network consumption can be reduced by up to $\sim 50\%$ for $f = 5$ when using our implementation compared to optimized Bracha-Dolev. This research has shown that BCB-Dolev is a good candidate for future research given the results that have been acquired. Future work could look into finding and implementing optimizations for this protocol in known and unknown topologies. Other research topics might include finding weaker assumptions under which the protocol would still be functional.

References

[1] Marcos K Aguilera, Naama Ben-David, Rachid Guerraoui, Dalia Papuc, Athanasios Xygkis, and Igor Zablotchi. Frugal byzantine computing. *arXiv preprint arXiv:2108.01330*, 2021.

[2] Silvia Bonomi, Jérémie Decouchant, Giovanni Farina, Vincent Rahli, and Sébastien Tixeuil. Practical byzantine reliable broadcast on partially connected networks. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, pages 506–516. IEEE, 2021.

[3] Silvia Bonomi, Giovanni Farina, and Sébastien Tixeuil. Multi-hop byzantine reliable broadcast with honest dealer made practical. *Journal of the Brazilian Computer Society*, 25(1):1–23, 2019.

[4] Gabriel Bracha. Asynchronous byzantine agreement protocols. *Information and Computation*, 75(2):130–143, 1987.

[5] Christian Cachin, Rachid Guerraoui, and Luís Rodrigues. *Introduction to reliable and secure distributed programming*. Springer Science & Business Media, 2011.

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

[7] Chiu-Yuen Koo. Broadcast in radio networks tolerating byzantine adversarial behavior. In *Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, pages 275–282, 2004.

[8] Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM (JACM)*, 27(2):228–234, 1980.

[9] Andrzej Pelc and David Peleg. Broadcasting with locally bounded byzantine faults. *Information Processing Letters*, 93(3):109–115, 2005.

[10] HariGovind V Ramasamy and Christian Cachin. Parsimonious asynchronous byzantine-fault-tolerant atomic broadcast. In *International Conference On Principles Of Distributed Systems*, pages 88–102. Springer, 2005.

[11] Michael K Reiter. Secure agreement protocols: Reliable and atomic group multicast in rampart. In *Proceedings of the 2nd ACM Conference on Computer and Communications Security*, pages 68–80, 1994.

[12] TK Srikanth and Sam Toueg. Simulating authenticated broadcasts to derive simple fault-tolerant algorithms. *Distributed Computing*, 2(2):80–94, 1987.

[13] Lewis Tseng, Nitin Vaidya, and Vartika Bhandari. Broadcast using certified propagation algorithm in presence of byzantine faults. *Information Processing Letters*, 115(4):512–514, 2015.

[14] Ye Wang and Roger Wattenhofer. Asynchronous byzantine agreement in incomplete networks. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 178–188, 2020.