

Multi-hop Byzantine Reliable Broadcast with Honest Dealer Made Practical

Silvia Bonomi, Giovanni Farina, Sébastien Tixeuil

▶ **To cite this version:**

Silvia Bonomi, Giovanni Farina, Sébastien Tixeuil. Multi-hop Byzantine Reliable Broadcast with Honest Dealer Made Practical. [Technical Report] Sorbonne Université, CNRS, Laboratoire d'Informatique de Paris 6, LIP6, F-75005 Paris, France; Dipartimento di Ingegneria Informatica Automatica e Gestionale "Antonio Ruberti", Sapienza Università di Roma, Rome, Italy. 2019. hal-01826865v3

HAL Id: hal-01826865
https://hal.science/hal-01826865v3

Submitted on 4 Sep 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Multi-hop Byzantine Reliable Broadcast with Honest Dealer Made Practical

Silvia Bonomi[⋆], Giovanni Farina[†⋆] ✉ , Sébastien Tixeuil[†]

[⋆]Dipartimento di Ingegneria Informatica Automatica e Gestionale
"Antonio Ruberti" (DIAG),
Sapienza Università di Roma, Rome, Italy
bonomi@diag.uniroma1.it

[†]Sorbonne Université,
CNRS, LIP6,
F-75005 Paris, France
Giovanni.Farina@lip6.fr, Sebastien.Tixeuil@lip6.fr

Abstract

We revisit Byzantine tolerant reliable broadcast with honest dealer algorithms in multi-hop networks. To tolerate Byzantine faulty nodes arbitrarily spread over the network, previous solutions require a factorial number of messages to be sent over the network if the messages are not authenticated (e.g. digital signatures are not available). We propose modifications that preserve the safety and liveness properties of the original unauthenticated protocols, while highly decreasing their observed message complexity when simulated on several classes of graph topologies, potentially opening to their employment.

Introduction

Designing dependable and secure distributed systems and networks, that are able to cope with various types of adversaries (ranging from simple errors to internal or external attackers), requires to integrate those risks from the very early design stages. The most general attack model in a distributed setting is the Byzantine model, where a subset of system participants may behave arbitrarily (including malicious), while the rest of participants remains correct. Also, reliable communication primitives are a core building block of any distributed system.

We consider the *reliable broadcast with honest dealer* problem in presence of Byzantine failures *i.e.*, the problem of distributing information from a *source* to every other process considering that a subset of nodes may act

arbitrarily. The reliable broadcast primitive is expected to provide two guarantees: *(i) safety i.e.*, every message m delivered by a correct node has been previously sent by the source and *(ii) liveness i.e.*, every message m sent by a correct source is eventually delivered by every correct node.

We are interested in solving this problem in a *multi-hop* network, in which nodes are not directly connected to every other (*i.e.* the network is not complete). In particular, nodes may have to rely on intermediate ones (hops) in order to communicate, forwarding messages till the final destination. In case the entire system is correct the solution to reliable broadcast is trivial: every node has just to forward the received messages to all of its other neighbors (or it has to question a routing table to know which is the next node to route a specific message) and if the network is connected then it is possible for every node to communicate with every other. Contrarily, if just one single node is faulty, specifically Byzantine faulty, two problems may arise: (i) messages can be modified or generated by faulty nodes that pretend the messages were sent from another node (ii) and messages can be blocked preventing nodes to communicate. It follows that a more sophisticated protocol has to be put in place to ensure the correct communication between the parties.

Lastly, we are interested in unauthenticated solutions, namely in protocols where messages cannot be directly authenticated (*e.g.* employing digital signatures) and thus the nodes cannot immediately verify that a specific received message has been previously sent by a specific other node.

The reliable broadcast with honest dealer enables to simulate a completely connected distributed system equipped with reliable and authenticated channels. It follows that all the solutions designed for completely connected distributed system (Byzantine Agreement, Byzantine Reliable Broadcast etc.) can be directly deployed on top of a multi-hop network once the reliable broadcast service has been deployed.

Related Works

The necessary and sufficient condition to solve the reliable broadcast with honest dealer problem on general networks has been identified by Dolev [7], demonstrating that it can be solved if and only if the network is $2f + 1$-connected, where f is the maximum number of Byzantine faulty nodes.

Subsequently, research efforts followed three paths: *(i)* replacing global conditions with local conditions, *(ii)* employing cryptographic primitives, or *(iii)* considering weaker broadcast specifications.

The Certified Propagation Algorithm (CPA) [24] is a protocol that solves reliable broadcast in networks where the number of Byzantine nodes is *locally bounded*, *i.e.*, in any given neighborhood, at most f processes can be Byzantine. This algorithm has been later extended [23] along several directions: (i) considering different thresholds for each neighborhood, (ii) considering

additional knowledge about the network topology, and (*iii*) considering the general adversary model.

The Byzantine tolerant reliable broadcast with honest dealer can also be addressed employing cryptography (*e.g.*, digital signatures) [5, 9] that enables all nodes to exchange messages guaranteeing authentication and integrity (authenticated protocols). The main advantage is that the problem can be solved with simpler solutions and weaker conditions (in terms of connectivity requirements). However, on the negative side, most of those solutions rely on a third party that handles and guarantees the cryptographic keys, thus the safety of those protocols is bounded to the crypto-system (a potential single point of failure).

Lastly, the broadcast problem has been considered weakening safety and/or liveness property *e.g.*, allowing to a (small) part of correct processes to either deliver fake messages, or to never deliver a valid message [17–19].

Let us note that a common assumption considered by Byzantine tolerant reliable broadcast protocols is to use authenticated point-to-point channels, which prevents a process from impersonating several others (Sybil attack) [8]. The real difference between cryptographic (authenticated) and non-cryptographic (unauthenticated) protocols for reliable broadcast is how the cryptography is employed: non-cryptographic protocols, in fact, may use digital signatures just within neighbors for authentication purposes, whereas the cryptographic protocols make use of cryptographic primitives to enable the message verification even between non-directly connected nodes. Let us finally remark that an authenticated channel not necessarily requires the use of cryptography [27].

Although the Byzantine tolerant reliable broadcast problem with honest dealer has been extensively studied considering alternative and additional assumptions, the solution provided by Dolev [7] is the only one for general settings and it has never been revisited from a performance perspective. Indeed, this solution hints at poor scalability since it requires a factorial number of copies (with respect the size of the network) of the same message to be spread and potentially verified in order to be accepted. This suggest that solving reliable broadcast in the weakest system model (*i.e.*, Dolev's solution [7]) is practically infeasible.

Contributions

We review and improve previous solutions for reliable broadcast in multi-hop networks, where at most f nodes can be Byzantine faulty, making no further assumption with respect to the original setting [7]. In more details, (*i*) we propose and evaluate modifications to the state of art protocols that preserve both safety and liveness properties of the original algorithms, and (*ii*) we define message selection policies in order to prevent Byzantine faulty nodes from flooding the network and to reduce the total number of messages

exchanged.

In a preliminary work [3], we focused on random network topologies, we defined two modifications to the state of art protocols, we proposed one preliminary message selection technique, and we carried out a performance analysis in the scenario where all processes are correct. In this work, by extensive simulations for variously shaped networks and considering active Byzantine processes spreading spurious messages over the network, we show that our modifications enable to keep the message complexity close to quadratic (in the size of the network). Our work thus paves the way for the practical use of Byzantine tolerant reliable broadcast solutions in realistic-size networks.

System Model and Problem Statement

System Model

We consider a distributed system composed by a set of n processes $\Pi = \{p_1, p_2, \dots p_n\}$, each one having a unique integer identifier. Processes are arranged in a communication network. This network can be seen as an undirected graph $G = (V, E)$ where each node represents a process $p_i \in \Pi$ (*i.e.* $V = \Pi$) and each edge represents a communication channel connecting two of them $p_i, p_j \in \Pi$ (*i.e.* $E \subset \Pi \times \Pi$), such that p_i and p_j can communicate. In the following, we interchangeably use terms *process* and *node* and we refer to *edges, links* and *communication channels* similarly.

We assume an omniscient adversary able to control up to f processes allowing them to behave arbitrarily. We call them *Byzantine* processes. All the processes that are not Byzantine faulty are *correct*. Correct processes do not a priori know the subset of Byzantine processes. Processes have *no global knowledge about the system* (*i.e.* the size or the topology of the network) except the value of f.

Communication channels allow connected processes to exchange messages, providing two interfaces: $SEND(p_{dest}, m)$ and $RECEIVE(p_{source}, m)$. The former requests to send a message m to process p_{dest}, the latter delivers the message m sent by process p_{source}. Processes that are not linked with a communication channel have to rely on others that relay their messages in order to communicate, in a *multi-hop* fashion. We assume *reliable* and *authenticated* communication channels, which provide the following guarantees: *(i) reliable delivery*, namely if a correct process sends a message m to a correct process q, then q eventually receives m; *(ii) authentication*, namely if a correct process q receives a message m with sender p, then m was previously sent to q by p.

We consider a *synchronous system*, namely we assume that *(i)* there is a known upper bound on the message transmission delays and *(ii)* a known upper bound on the processing delays. We assume a computation that evolves

in sequential synchronous *rounds*. Every round is divided in three phases: *(1)* send, where processes send all the messages for the current round, *(2)* receive, where processes receive all the messages sent at the beginning of the current round and *(3)* computation, where processes execute the computation required by the specific protocol. In a single round, any message can traverse exactly one hop, namely the message exchange occurs only between neighbor processes. We measure the time in terms of the number of rounds.

Problem Statement

We consider the *reliable broadcast with honest dealer* problem from a *source* s assuming f Byzantine failures arbitrary spread in the network [7]. A protocol solves the Byzantine tolerant Reliable Broadcast (BRB) with honest dealer problem if the following conditions are met:

- *(Safety)* if a *correct* process delivers a message m, then it has been previously sent by the source;
- *(Liveness)* if a *correct* source broadcast a message m, then m is be eventually delivered by every correct process.

Notice that in the case of a correct source, all correct processes deliver the broadcast message. Instead, if the source is Byzantine faulty, then not all correct processes necessarily deliver the broadcast message and/or they may deliver different messages.

We referred (following the literature) with message m to a content (*i.e.* a value) that has been broadcast. Every information spreading protocol place a content inside a message with a specific format, adding protocol related overhead. Therefore, to ease of explanation, we refer with *content* to the value that has been broadcast and with *message* to the one exchanged by a protocol. Therefore, a message refers to the union of a content and the overhead added by the employed protocols.

Background

We start by presenting and remarking some definitions and theoretical results to lead the reader in an easier understanding. Subsequently, we present the state-of-the-art protocols solving the BRB problem and we provide an analysis of them.

Basic Definitions and Fundamental Results

For all the definitions and results that follow, let us consider the cube graph $\hat{G}$ depicted in Figure 1 as example.

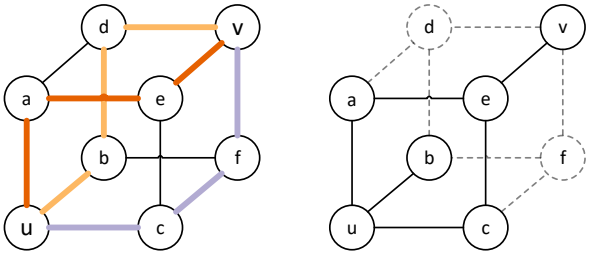


Figure 1: **Disjoint Paths (a) and Vertex Cut (b)** of a cube graph.

Definition 1 (neighbors)**.** *Given an undirected graph $G = (V, E)$, two nodes u, v are* adjacent *or* neighbors *if there is an edge connecting them (*i.e. *$\{u, v\} \in E$).*

In graph $\hat{G}$, the neighbors of node u are nodes a, b and c.

Definition 2 (path)**.** *Given an undirected graph $G = (V, E)$, a* path *is a sequence of adjacent nodes without repetitions (*i.e. *path $:= (v_1, v_2, \ldots, v_x)$: $\forall i \in [1, x-1]$, $v_i \in V$, $\{v_i, v_{i+1}\} \in E$). The two extreme nodes of a path are called* ends*.*

Definition 3 (connected nodes and connected graph)**.** *Given an undirected graph $G = (V, E)$, two nodes u, v are* connected *if there exist at least one path with ends u, v, they are* disconnected *otherwise. The graph G is connected if it exist at least one path between every pair of nodes.*

In graph $\hat{G}$, the sequence (u, a, e, v) is a path with ends u and v, thus nodes u and v are connected.

Definition 4 (independent/disjoint paths)**.** *Given an undirected graph $G = (V, E)$, two or more of its paths are* independent *or* disjoint *if they share no node except their ends.*

In graph $\hat{G}$, the sequence (u, c, f, v) is another path that is disjoint with respect to (u, a, e, v).

Definition 5 (vertex cut)**.** *Given an undirected graph $G = (V, E)$, the removal of a set of nodes $C \subset V$ from G results in a subgraph $G_C = (V_C, E_C)$, where $V_C = V - C$ and $E_C \subset E : \forall \{v_i, v_j\} \in E, \{v_i, v_j\} \in E_C \iff v_i, v_j \notin C$. Given two not adjacent nodes $u, v \in V$, a* vertex cut *$C \subset V - \{u, v\}$ for u, v is a set of nodes whose removal from the graph G disconnects u from v, namely u, v are disconnected in G_C.*

In graph $\hat{G}$, the set of nodes $\{d, e, f\}$ is a vertex cut for (u, v), because its removal will disconnects nodes u from v.

Given an undirected graph $G = (V, E)$ and two not adjacent nodes u, v, the maximum number of mutually disjoint paths with ends u and v is referred with $\kappa'(u, v)$, and the size of the smallest vertex cut C separating u from v is referred with $\kappa(u, v)$.

Remark 1 (Global Menger Theorem)**.** *A graph is k-connected (or it has* vertex connectivity *equals to k) if and only if it contains k independent paths between any two vertices.*

Remark 2 (Vertex Cut VS Disjoint Paths)**.** *Let $G = (V, E)$ be a graph and $u, v \in V$. Then the minimum number of vertexes that disconnects u from v in G is equal to the maximum number of disjoint $u - v$ paths in G, namely $\kappa(u, v) = \kappa'(u, v)$.*

In graph $\hat{G}$, the maximum number of disjoint paths between nodes u, v ($\kappa'(u, v)$) is 3, as depicted in Figure 1a. Furthermore, at least 3 nodes have to be removed from the network in order to disconnect nodes u, v (Figure 1b), thus showing the equivalence $\kappa'(u, v) = \kappa(u, v)$.

The reader can refer to [6] for addition details.

The Byzantine Reliable Broadcast problem can be solved under the assumptions we considered in the system model when the following condition is met:

Remark 3 (Condition for Byzantine Reliable Broadcast [7])**.** *Given a network G composed of n processes where at most f can be Byzantine faulty, the Byzantine Reliable Broadcast can be achieved if and only if the vertex connectivity of G is at least $2f + 1$.*

Byzantine Reliable Broadcast Protocols

There exists two solutions addressing the Byzantine Reliable Broadcast with honest dealer problem in the system model we considered, that are the Dolev [7] and the Maurer et al. [20] algorithms. Any other solutions for the BRB problem makes extra or different assumptions (*e.g.* digital signatures, higher density networks, weaker versions of safety or liveness, etc.).

The protocols that follow are defined by:

- a *propagation algorithm*, which rules how messages are spread over the network;
- a *verification algorithm*, that decides if a content can be accepted by a process guaranteeing the safety of reliable broadcast.

The basic idea behind the following protocols is to leverage the authenticated channels to collect the labels of processes traversed by a content, in order to compute the maximum disjoint paths, in the case of Dolev, or the minimum vertex cut, in the case of Maurer et al., of all the paths traversed

by the content. Those two methodologies are theoretically equivalent due to the Menger Theorem (Remark 2), *i.e.* if a message can traverse $f + 1$ disjoint paths in a network, then it can traverse paths such that their minimum vertex cut is $f + 1$ and vice versa.

Dolev Reliable Broadcast Protocol *(D-BRB)*

Dolev [7] defined the seminal algorithm addressing the BRB problem. The messages exchanged by his protocol have the format $m := \langle s, content, path \rangle$, where s is the label of the process asserting to be the source, *content* is the content to broadcast and *path* is a sequence of nodes.

Propagation Algorithm:

1. the source process s sends the message $m = \langle s, content, \varnothing \rangle$ to all of its neighbors;
2. a correct process p saves and relays any message $m = \langle s, content, path_i \rangle$ sent by a neighbor q to all of other neighbors not included in $path_i$, appending to $path_i$ the label of the sender q, namely process p stores and multicasts $m = \langle s, content, path_i \cup \{q\} \rangle$.

   The messages carrying $path_i$ with loops or $path_i$ that includes the label of the receiving process p are discarded at reception.

Verification Algorithm:

1. given a set of messages $m_i = \langle s, content, path_i \rangle$ received by process p and carrying the same values for s and *content*, the *content* is delivered by p if there exist $f + 1$ disjoint paths among of all the related $path_i$.

Maurer et al. Reliable Broadcast Protocol *(MTD-BRB)*

Maurer et al. [20] extended and improved the algorithm defined by Dolev to deal with dynamic distributed systems, where the communication network changes over the time. As a matter of facts, a static communication network (our case) can be seen as a dynamic network that never changes, making their solution employable on the system model we are considering.

The format of messages exchanged by their protocol is $m := \langle s, content, pathset \rangle$, again carrying the information about the process s asserting to be the source and the *content* of broadcast. The difference with respect the previous algorithm is the data structure employed to collect the labels of traversed nodes: *pathset*, that it discards the traversing order.

Furthermore, *MTD-BRB* verifies the minimum vertex cut of the pathset traversed by a content instead of the maximum disjoint paths. The reason is that on dynamic networks the Menger theorem (Remark 3) does not hold, specifically the minimum vertex cut may be greater than or equal to the maximum disjoint paths between two nodes [15].

Propagation Algorithm:

1. the source process s sends the $m = \langle s, content, \varnothing \rangle$ to all of its neighbors;

2. a correct process p saves and relays any message $m = \langle s, content, pathset_i \rangle$ sent by a neighbor q to all of other neighbors not included in $pathset_i$, appending to $pathset_i$ the label of the sender q, namely process p stores and multicasts $m = \langle s, content, pathset_i \cup \{q\} \rangle$.

   The messages carrying $pathset_i$ with duplicates or $pathset_i$ that includes the label of the receiving process p are discarded at reception.

Verification Algorithm:

1. given a set of messages $m_i = \langle s, content, pathset_i \rangle$ received by process p and carrying the same values of s and *content*, the *content* is delivered by p if the minimum vertex cut of the related $pathset_i$ is at least $f + 1$.

Discussion

The two protocols we presented are *quiescent*, namely at a certain point all correct processes stop sending messages. To be precise, this is true only if all processes are correct or at least the additional knowledge about the size of the system is given to the processes to guarantee the termination of message spreading. Contrarily, a Byzantine process may continuously introduce spurious messages carrying $path_i / \tilde{pathset}_i = \{random_label\}$ that are forwarded to all processes. We temporarily assume, for ease of evaluation, that all processes are correct in the analysis that follows. In order to evaluate and compare the protocols reported and the solution we are going to design, we analyze the following metrics:

1. *message complexity i.e.*, the total number of messages exchanged in a single broadcast (the amount of messages exchanged from the beginning of the broadcast till the moment when all processes stop sending messages);

2. *delivery computational complexity i.e.*, the complexity of the procedure executed by a process during the computation phase to decide if a content can be accepted.

3. *broadcast latency i.e.*, the time between the beginning of the broadcast and the time when all correct processes deliver the content.

The message complexity of *D-BRB* protocol is factorial in the size of the network. The reason is that for every path connecting the source with any

other node (*i.e.* that are order of the permutations over the full set of nodes) a message with related $path_i$ is generated. This potentially results in an factorial number of $path_i$ to elaborate by every process in order to deliver a single content. For sake of explaination, let us consider the cube graph depicted in Figure 1 and let us assume that process u starts a broacast, thus spreading a content with an empty *path* that will traverse the paths (u, a), (u, b) and (u, c) on the communication network. The neighbors of the source will receive the content and its related (empty) path, they will attach the label of the sender and they will forward it to all of their neighbors not already included, e.g. process a will forwards the content with the path (u) to processes d and e and thus a message related to the paths (u, a, d) and (u, a, e) will be generated (and the same will be done also by processes b and c). The process d will receive the path (u) from a and b (the same happens for processes e and f from different processes). Consequently, a message carring (u, a) will be forwarded by d to b and v and a message carrying (u, b) will be sent by d to a and v. The messages continue to be generated as long as all possible paths are traversed, one message for each path.

Furthermore, to the best of our knowledge, the only method available to identify $f + 1$ disjoint $path_i$ is the reduction to a NP-Complete problem, *Set Packing* [11]. We refer to this method with *DP* (Disjoint Paths), namely to the reduction and solution of the associated Set Packing instance. This implies that the delivery complexity of the algorithm is exponential.

The *D-BRB* guarantees the safety and liveness properties of BRB when the strict enabling condition is met (Remark 3), respectively because the Byzantine processes $b_1, b_2, \ldots b_f$ cannot propagate a different content $\tilde{content} \neq content$ with source s through no more than f disjoint paths, and assuming a vertex cut of size f made by the faulty processes, $f + 1$ disjoint paths are still available between any pairs of correct processes.

The *MTD-BRB* protocol is equivalent with respect the message complexity and delivery complexity to *D-BRB*. Specifically, even if all the $paths_i$ over the same set of nodes are all collapsed in a single *pathset*, they are still factorial in the number of nodes (*i.e.* they are order of the combinations over the full set of nodes), and messages carrying any possible $pathset_i$ are generated, potentially leading to an input of factorial size for the verification algorithm.

Again, to the best of our knowledge, the only method available to identify a vertex cut of size less than or equal to f is the reduction to a NP-Complete problem, *Hitting Set* [11]. We refer to this method as *VC* (Vertex Cut), namely to the reduction and solution of the associated Hitting Set instance. This implies that the delivery complexity of this algorithm is exponential too.

The safety and liveness properties of BRB are guaranteed by *MTD-BRB* due to the same argumentation made for *D-BRB*: the Byzantine processes $b_1, b_2, \ldots b_f$ cannot propagate a different content $\tilde{content} \neq content$ with

	Dolev	Maurer et al.
Message Complexity	Factorial (Permutations of nodes)	Factorial (Combinations of nodes)
Delivery Complexity	NP	NP
Broadcast Latency	$\leqslant n-k$	$\leqslant n-k$

Table 1: Analysis of state of art protocols

source s through pathsets with a minimum vertex cut greater than f and they cannot make a vertex cut on the communication network greater than f.

The broadcast latency of both protocols is bounded by the graph metric called *wide diameter* [12]. Given a k-connected graph G, the wide diameter is the maximum number l such that there exist k internally disjoint (u, v)-paths in G of length at most l between any pair of vertices u and v. This value depends on the graph topology. In the worst case, the wide diameter of a graph is $n-k$ [13]. It follows that the broadcast latency of both protocols is upper bounded by $n-k$, because in at most $n-k$ rounds k disjoint path are traversed between every pair of nodes. As a clarifying example, let us consider a k-connected generlized wheel graph with n nodes, that is composed by the disjoint union between a cycle and a $k-2$ clique (a grafical example is depicted in Figure 5b) and let us chose as source a node on the cycle and let us focus on one of its neighbors on the cycle at distance two. It is possible to verify that in order to interconnect the pair of nodes we are considering through k disjoint paths, one path of length $n-k$ has to be traversed.

The Table 1 summarizes the presented analysis.

Practical Reliable Broadcast Protocol

Due to the high message complexity and delivery computational complexity of the reviewed protocols, they do no scale and they cannot be successfully employed.

We further analyze some deeper details of the aforementioned protocols and we define simple modifications that result in drastically reducing the message complexity. Specifically, we start by arguing that *pathsets* and *VC* should be preferred respectively as message format and verification algorithm. Subsequently, we propose modifications that aim in reducing the amount of messages spread by preventing from forwarding useless messages, thus redefining a protocol solving BRB.

Paths VS Pathsets

It is possible to note that, given the solutions available, there is no reason to prefer path over pathset while collecting the label of the traversed processes, indeed: *(i)* due to the reduction to set related problems, paths are converted into sets to be analyzed; *(ii)* two paths over the same set of nodes are not disjoint and have a cut of size equal to 1, and *(iii)* the pathsets interconnecting two endpoints are fewer than the relative paths. For those reasons we adopt the pathset data structure as message format to collect the labels of traversed processes in designing an improved protocol.

Minimum Vertex Cut VS Maximum Disjoint Paths

We remark that both verification algorithms solve a NP-Complete problems and, considering the Menger theorem in Remark 2, one may conclude that there is no tangible reason to prefer one among VC and DP. As a matter of fact, the equivalence between the two metrics in Remark 2 occurs when no restriction on the length of the paths is assumed. In fact, when the path lenght is bounded, the minimum vertex cut between two nodes may be higher than or equal to the maximum number of disjoint paths interconnecting them [16]. Let us take the example proposed in Figure 2 [16], let us focus on nodes u and v as endpoints and consider only the paths of length at most 5. It can be verified that at least two nodes have to be removed from the graph in order to disconnect u from v considering only the paths of length at most 5. Nevertheless, no two disjoint paths exist considering only the paths with the same constraint. In other words, given a graph G of n nodes and considering only the paths of length at most $l < n$, the size of the minimum vertex cut of those interconnecting two nodes may be greater than or equal to the maximum number of disjoint paths interconnecting them. This implies that, whenever a synchronous system is assumed and the paths are all traversed synchronously like in our system model (*i.e.* the path of length 1 are all traversed in 1 instant (round), the paths of length 2 are all traversed in two instant (round) and so on), it may be possible to interconnect two endpoints with a minimum cut equal to k in fewer hops (*i.e.* rounds) with respect k disjoint paths. This results also in saving in message complexity if a halting condition is embedded inside the protocol, namely if the message propagation stops when all correct processes delivered the content. For this reason we adopt VC as verification methodology.

Improvements

Practical Reliable Broadcast Protocol (BFT-BRB)

We redefine a protocol for the Byzantine Reliable Broadcast with honest dealer. This protocol employs the same message format and verification

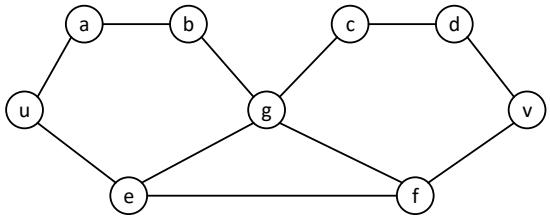


Figure 2: **Graph example to compare minimum vertex cut and maximum disjoint paths.**

algorithm of *MTD-BRB*, namely the label of the processes traversed by a message are collected in *pathsets* and the contents are verified through the VC methodology. We introduce four modifications in the propagation algorithm and one in the verification algorithm that aim to reduce the total number of messages exchanged and we prove their correctness, namely that their employment do not prevent the original algorithms of Dolev and Maurer et al. from enforcing safety and liveness of Byzantine Reliable Broadcast with honest dealer when the strict enabling condition is met (Remark 3), because they prevent from forwarding messages that are not useful for the delivery of a content.

Modification 1. *If a process p receives a content directly from the source s (*i.e. *the source and the sender coincides), then it is directly delivered by p.*

Modification 2. *If a process p has delivered a content, then it can discard all the related pathsets and relay the content only with an empty pathset to all of its neighbors.*

Modification 3. *A process p relays pathsets related to a content only to the neighbors that have not yet delivered it.*

Modification 4. *If a process p receives a content with an empty pathset from a neighbor q, then p can discard from relaying and analyzing any further pathset related to the content that contains the label of q.*

Modification 5. *A process p stops relaying further pathsets related to a content after it has been delivered and the empty pathset has been forwarded.*

The modification 1 follows from the defitions of disjoint paths and vertex cut, indeed a path of lenght two is disjoint with respect every other one with the same ends, and the vertex cut is defined between not adjacent nodes, thus there is no vertex cut between neighbors..

The purpose of modifications 2, 3, and 4 is to reduce the amount of messages exchanged by the protocol and to be analyzed by processes. The modification 2 also provides a transparent way to get the neighbors q of a

process p know that a specific content has been delivered by p. This one has already been employed [22] for the purpose of topology reconstruction.

The modification 5 introduces a halting condition in the protocol with respect the state of art, indeed all correct processes stop from relaying further messages at the round subsequent the last delivery of a process. Furthermore, this modifications makes the original solutions quiescent without assuming that processes know the size of system.

Let us consider the network topology depicted in Figure 1a as an example to detail the advantages introduced by the presented modifications. Let us select node u as source process and let us consider the all the paths of lenght two starting from u, namely $(u, c, f), (u, c, e), (u, b, f), (u, b, d), (u, a, d), (u, a, e)$.

Processes d, e and f, following modification 1 will relay only an empty path instead of extending the paths they received, namely avoiding to generate (u, c, f, v), (u, c, e, v), (u, b, f, v), (u, b, d, v), (u, a, d, v), (u, a, e, v).

Processes d, e and f, leveraging modification 1, know that the nodes a, b and c have already delivered the content associated to the paths. Applying modification 2, processes d, e and f do not relay further paths to a, b and c, namely they do not generate paths (u, c, f, b), (u, c, e, a), (u, b, f, c), (u, b, d, a), (u, a, d, b), (u, a, e, c).

The modification 3 applies in cases a process p receives paths in round r_i but it delivers the associated content in a round $r_j > r_i$. A neighbor q of p that has not yet delivered the content will get the extension of paths received by p in r_i and potentially the empty path in r_j+1. The modification 3 enables q to discard from the analysis in delivering the associated content all paths previously received from p and to consider only the empty pathset.

The pseudo code of our protocol is presented in Figure 3. For the ease of explanation and notation, we show the procedure and variables only related to the broadcast of a single *content* spread by s.

Initially, every process is not aware about the nodes in its neighborhood but it can easily retrieve them with authenticated channels. For every not delivered content, a process stores *(i)* the received pathsets related to the content (*Pathsets* variable), *(ii)* the pathsets not yet relayed (*To_Forward* variable) and *(iii)* the labels of neighbors that have delivered the content (*Neigh_Del* variable).

Every process starts the round with the send phase, namely selecting the messages to forward and transmitting them. In particular, it extracts part or all of the message related to a content to relay (*select* function), and it forwards them to all of its neighbors that have not yet delivered the content, thus applying modification 3 in line 8.

During the receive phase, for every received message related to a content not yet delivered, the label of the sender is attached to the received *pathset* and the resulting collection is stored in order to be considered for the delivery and to be forwarded (we assume an implicit mechanism avoiding duplicate *pathsets*). The modification 2 enables a process p to know that a sender q

has delivered the content (line 15). Then, the modification 3 allows p to discard part of the *pathset* previously received (lines $17-22$) and that may arrive (line 13).

Finally, in the computation phase, all received *pathsets* related to the content are analyzed. Specifically, in case a process has received the content directly from the source s (*i.e.* the sender and the source coincides, the receiver has received the *pathset* $\{s\}$, line 26, modification 1) or the minimum vertex cut computed on *Pathsets* is greater than f, it delivers the content, it discards all the *pathsets* not yet forwarded and it enqueues to relay the empty *pathset*, namely applying modification 2 in lines $30, 31$.

The implementation of modification 5 can be found in line 12. Indeed, once that a process has delivered the content, it discards all the residual *pathsets* to forward (line 30). In the receive phase all the messages related to the content already delivered are discarded (line 12) due to modification 4, thus the select function in line 6 only extracts the empty pathset in the round subsequent the delivery.

We prove the correctness of the proposed modifications thought the following theorems (assuming the system model we presented and under the assumption of the strict condition in Remark 3).

Theorem 1. *Let p be a process executing either the Dolev or the Maurer* et al. *algorithm to broadcast a content. If p delivers a content received directly from the source, then the safety property continues to be satisfied (*i.e. *employing modification 1).*

Proof. It follows directly from the property of the channels (reliable and authenticated), indeed the channels guarantee that every received message has been previously sent by the sender, that coincides with the safety property of reliable broadcast. □

Theorem 2. *Let p be a process executing either the Dolev or the Maurer* et al. *algorithm to broadcast a content. If p delivered a content, then p can relay that content with an empty path/pathset and the safety property continues to be satisfied (*i.e. *employing modification 2).*

Proof. The aim of the information about the nodes traversed by a content is to enable a process p to decide whether it can be safely accepted. Once it has been delivered, the information about the nodes traversed before reaching p is not useful, because the content has been already verified as safe by p. □

Theorem 3. *Let p be a process executing either the Dolev or the Maurer* et al. *algorithm to broadcast a content and let us assume that the modification 2 is employed. Even if p does not relay messages carrying the content to its neighbors that already delivered it, the liveness property continues to be satisfied (*i.e. *employing modification 3).*

```
1   Neigh = [p1, p2,...pj]
2   Neigh_Del = []
3   To_Forward = []
4   Pathsets = []

5   For each round r :

        # Send phase of round r
6       Next_To_Forward = select( To_Forward )
7       ForAll pathset_tf in Next_To_Forward :
8           ForAll neight not in Neigh – Neigh_Del :
9               If neigth not in pathset_tf :
10                  SEND( neight , m = < s, content, pathset_tf > )

        # Receive phase of round r
11      For each q, m = < s, content, pathset > from RECEIVE ( q, m ) :
12          If < s, content > not in DELIVERED :
13              If pathset ∩ Neigh_Del == ∅ :
14                  received_pathset = pathset + { q }
15                  If received_pathset == { q }:
16                      Neigth_Del.add( q )
17                      For each pathset_x in Pathsets:
18                          If q in Pathsets:
19                              Pathsets.remove (pathset_x)
20                      For each pathset_x in To_Forward:
21                          If q in To_Forward:
22                              To_Forward.remove (pathset_x)
23                  To_Forward.add( received_pathset )
24                  Pathsets.add( received_pathset )

        # Computation phase of round r
25      If < s, content > not in DELIVERED :
26          If {s} in Pathsets or not VC( Pathsets, f ) :
27              DELIVER( < s, content > )
28              DELIVERED.add ( {< s, content > } )
29              Pathsets.add ()
30              To_Forward.clear()
31              To_Forward.add( {})
```

Figure 3: **Byzantine Tolerant Reliable Broadcast with honest dealer Protocol.**

Proof. Let us assume that there exists three processes p, q, r such that only q has already delivered the content and that, among others, the following communication channels are available: (p, q), (q, r). From Theorem 2 we know that process q can safely relay the content with an empty path/pathset (*i.e.* employing modification 2). Thus, any further path/pathset containing p and q, after the delivery of q, does not affect the results of DP and VC verifying the content on r. It follows that any further transmission related to the content from p to q can be avoided after that q has delivered without compromising liveness. □

Theorem 4. *Let p be a process executing either the Dolev or the Maurer* et al. *algorithm to broadcast a content and let us assume that the modification 2 is employed. If process p receives an empty path/pathset relative to a content from a neighbor q, than p can discard from its analysis and from relaying further path/pathset containing the label of q and the liveness property continues to be satisfied (*i.e. *employing modification 4).*

Proof. Let us assume that there exists three processes p, q, r such that only p has already delivered a content and that, among others, the following communication channels are available: (p, q), (p, r). We have to prove that process p can discard, verifying the associated content, further path/pathset containing the label of q but $\{q\}$ without affecting the liveness property. This follows from the fact that path/pathset of unit lenght are included in every solution of the VC and DP and that any path/pathset containing more labels does not increase the value computed by VC and DP. We have to prove that this reasoning extends also for process r, so that process p can avoid relaying further path/pathset over $\{q\}$. On process r, any path/pathset that extends $\{q, p\}$ does not increase the value obtained by VC and DP. It follows that any other path/pathset over $\{q\}$ has not to be relayed. □

Theorem 5. *Let p be a process executing either Dolev or Maurer* et al. *algorithm to broadcast a content and let us assume that the modification 2 is employed. If p has delivered and relayed the content with an empty path/pathset to all of its neighbors, then p can stop from relaying further related paths/pathsets and the liveness property continues to be satisfied (*i.e. *employing modification 5).*

Proof. It follows from the fact that any further path/pathset related to the content received and relayed by p does not increase the minimum cut/the maximum disjoint paths computed on other processes with respect the empty path/pathset relayed by p. Said differently, all the neighbors of p receive the paths/pathset $\{p\}$ and any further path/pathset relayed by p becomes $\{\ldots, p\}$, increasing neither the minimum vertex cut nor the maximum disjoint paths. □

Preventing Flooding and Forwarding Policies.

We highlighted the fact that the verification algorithm has potentially to analyze a factorial, in the size of the network, amount of pathsets even only considering all processes to be correct. Nevertheless, a Byzantine process b can potentially flood the network with spurious messages (*i.e.*, $\tilde{m} := \langle \tilde{s}, \tilde{content}, \tilde{pathset} \rangle$ where $\tilde{s}$, $\tilde{content}$ and $\tilde{pathset}$ can be invented by the faulty process) that are also diffused by the correct ones. Considering that the amount of messages plays a crucial impact on the employment of the protocol we defined, a countermeasure must be researched.

A common way to limit the flooding capability of Byzantine processes is to constraint the channel capacity of every process, namely limiting the amount of messages that every process is allowed to send in a time window.

Noticed that, by introducing such a constraint, we are limiting the relaying capability of every process, while the Byzantine processes can continuously generate spurious messages potentially preventing the liveness property to be satisfied. It follows that a selection policy among all the messages to relay is demanded.

Every process has to relay *pathsets* to all of its neighbors that have not yet delivered the content. A *pathset* that may lead a neighbor q to the delivery of the associated content has not to contain the label of q (because it would be directly discarded), namely a process p has to select among the *pathsets* to forward the ones that do not include the label of q. There may be many pathsets that do not include q. Thus, we consider and evaluate two selection policies: *(i) Multi-Random* and *(ii) Multi-Shortest* . The Multi-Random is an extension of the forwarding policy proposed in [3].

The algorithms for the pathsets selection implementing the Multi-Random and Multi-Shortest policies are presented in Figure 4. The selection iteratively picks one *pathset* and checks if it is "useful" for any neighbor (*i.e.* if any neighbor to contact is not included in the *pathset*). This selection continues till *(i)* all the neighbors to contact receives at least one *pathset* where they are not included or *(ii)* the bound on channel capacity has been reached. The Multi-Random policy iteratively picks randomly a possible *pathset* to forward, the Multi-Shortest gives priority to the shorter ones.

We compare and analyzed them both in the following.

```
If MULTI_SHORTEST_POLICY :
        Pathsets.sort()
Else if MULTI_RANDOM_POLICY:
        Pathsets.shuffle()
Node_to_Contact = Neighbors − Neighbors_Del
Next_To_Forward = []
For pathset_i in Pathsets :
        Num_Missing = | Node_To_Contact |
        If Num_Missing == 0 or | Next_To_Forward | == CHANNEL_BOUND :
                Break
        Node_to_Contact = Node_to_Contact ∩ pathset_i
        If | Node_to_Contact | != Num_Missing :
                Next_to_Forward.add(pathset_i)
                Pathsets.remove(pathset_i)
Return Next_To_Forward
```

Figure 4: **Multi-Shortest and Multi-Random Policies.**

Practical Reliable Broadcast Evaluation

We simulate the protocol and the policies we proposed in order to evaluate their effectiveness and to compare our protocol with the state-of-art solutions.

According to the system model we defined, we simulate single broadcasts that evolves in rounds. Therefore, the passage of time is measured in number of rounds.

We made use of the implementation provided by Gainer-Dewar and Vera-Licona [10] for the algorithm defined by Murakami and Uno [21] to solve the VC reduction to the hitting set problem.

We consider the following parameters in our simulation:

- n, *i.e.* the size of the network considered;
- k, *i.e.* the vertex connectivity of the network considered;
- *topology*, *i.e.* the topology of network considered;
- *channel capacity*, *i.e.* the maximum number of messages that a process can send in a link per round;
- *kind of failure*, *i.e.* how faulty process behave;
- *forwarding policy*, *i.e.* one among *Multi-Shortest* and *Multi-Random* .

In order to carry an analysis as complete as possible, we consider the following network topologies:

- k-regular k-connected random graph [25];
- k-pasted-tree [1];
- k-diamond [1];
- multipartite wheel;
- generalized wheel [26];
- Barabási-Albert graph [2].

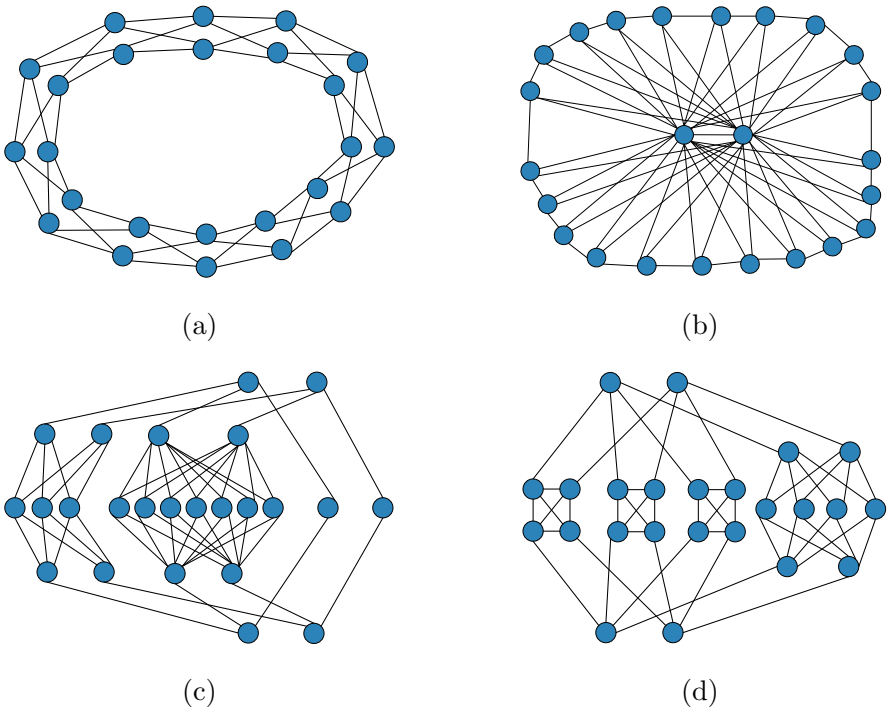


Figure 5: **Graph topologies**. $n = 24$, $k = 4$. (a) multipartite wheel, (b) generalized wheel, (c) k-pasted-tree, (d) k-diamond.

A graph is regular if every node is connected to the same number of neighbors, namely in a k-regular graph every node is connected exactly to k neighbors. The k-regular k-connected graphs have vertex connectivity equals to k with the minimum necessary number of edges. The k-regular k-connected random graphs are the ones uniformly sampled among all possible regular graphs employing the sampling methodology defined in [25].

The k-pasted-trees and k-diamond graphs are *Logarithmic Harary Graph* [14], namely topologies designed to be robust to failures and suited for distributed systems where the information spreading occurs by message flooding. Indeed, they are k-connected graphs with a logarithmic diameter and

with minimal edges guaranteeing the node-connectivity (*i.e.* the removal of an edge decreases vertex connectivity of the network). For specific values of network size n and vertex connectivity k they are k-regular. A graphical example of k-pasted-trees and k-diamond are respectively presented in Figures 5c and 5d.

We refer with *multipartite wheel* to a regular graph composed by the concatenation of disjoint groups of $k/2$ nodes such that every node in a group is connected to exactly all the $k/2$ nodes in other 2 groups and no node inside a group is connected with others of the same group. A graphical example is provided in Figure 5a.

Notice that k-regular k-connected graphs can be constructed in several ways, indeed we are considering four different constructions that are either always regular or regular for specific settings. The sequel demonstrates that the specific construction impacts protocol performance.

We considered also the Barabási-Albert graphs that models complex and social networks with scale-free power law degree distribution. The aim is to evaluate our protocol also on topologies not designed for distributed systems.

Finally, we consider the generalized wheel, *i.e.* the topology generated by the disjoint union between a cycle and a $k-2$ clique. An example can be found in Figure 5b. It has been considered as a worst case scenario.

We carry our simulations either considering the maximum number of tolerable faulty processes, thus for every k-connected network we assume $f = \lfloor (k-1)/2 \rfloor$ failures (Remark 3), or we tested all possible values for f between 0 and $\lfloor (k-1)/2 \rfloor$. In any case, processes deliver a content only when the related pathsets have a minimum vertex cut greater than $\lfloor (k-1)/2 \rfloor$.

We consider two configurations for the channel capacity: *bounded* and *unbounded*. The former constrains processes to send a limited number of messages per link in every round, the latter imposes no restriction. For the bounded case we assume a bound for the channel capacity equal to $f+1$ messages.

The source code of the presented simulations can be found online [4].

Simulating Byzantine Behaviours

We move from the scenario where all processes are correct, to the case where the f Byzantine processes act as crash failures, (thus not relaying any message, we refer to them as *passive* Byzantines), till the case they spread spurious messages (we refer to them as *active* Byzantines). Specifically to this last scenario, we have to notice that spurious contents (*i.e.* contents generated by Byzantine processes $b_i \neq s$ sent inside a message with source s) are never accepted by correct processes (if the BRB enabling condition in Remark 2 is met) and their spreading and verification is disjoint with respect the content broadcast by the source (because they are related to a different s - *content*). For this reason we impose to Byzantine processes to

spread only spurious *pathsets* in our simulations (thus relaying the content broadcast by the source). The purpose is to flood the correct processes with spurious pathsets trying to not facilitate the achievement of the delivery condition. In detail, the Byzantine processes diffuse *pathsets* containing the label of one correct neighbor of the receiver in the first round, and *pathseths* containing one of the correct neighbor of the receiver with a random label in the subsequent rounds. Every Byzantine process sends $f + 1$ messages (the maximum amount allowed by the channel capacity) containing different *pathsets* on every of its link per round.

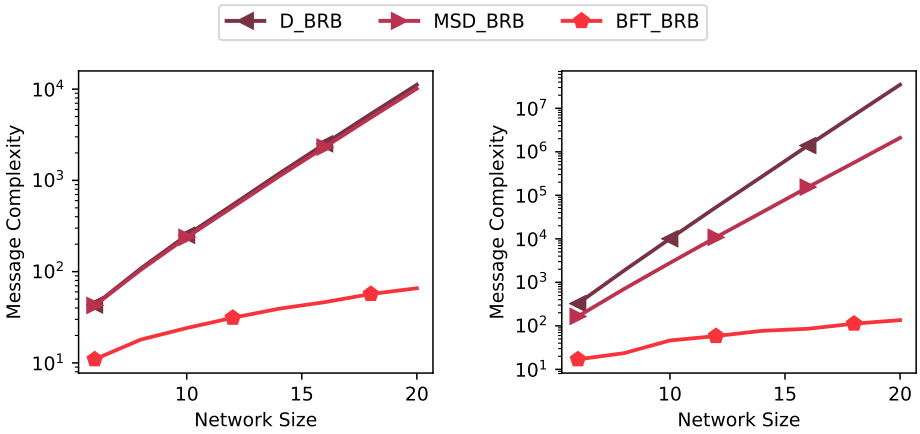
Additionally, we consider two kinds of active Byzantine processes: *omniscient* and *general*. Omniscient active faulty processes know the content that the source is going to spread before receiving it through a message, thus they start flooding correct processes with spourious pathsets from the beginning of broadcast. General active faulty processes, instead, spreads spurious pathsets in the round subsequent the first reception of a message containing the content, namely as soon as they get knowledge about the content through the network. Notice that there are other strategies that Byzantine processes may adopt generating spourious pathsets, especially if such Byzantines are omniscient about the state of all other processes. The Byzantine strategy that we adopted has been choosen to allows faulty processes to generate pathsests that may be selected by the correct process due to their lenght.

In every simulation, the source and the Byzantine processes are randomly placed.

For all the results we are going to show we directly plot all the measures we got as points (except for Figures 6,15,16 where the mean of the measures is depicted) in order to show their distributions, and we accordingly increase the size of the points with higher density.

Comparison with the state of art

We start comparing the message complexity of the state of art solutions with our protocol. We consider k-regular k-connected random graphs, we assume the vertex connectivity k equal to 3 and 5 and we simulate *D-BRB*, *MTD-BRB* and *BFT-MTB* considering unbounded channels and all correct processes, and we vary the size of the network from $n = 6$ to $n = 20$. We previously remarked about the lack in the state of art protocols of a halting condition, indeed they generate all source-to-other paths/pathsets in every execution. It can be noticed in Figure 6 that the modifications we defined have a remarkable impact on the message complexity even in a small and all-correct scenario. It can also be noticed the advantage gained by choosing *pathsets* over *paths*, as expected.



(a) (b)

Figure 6: **State of art VS our protocol, message complexity.** Random regular network, unbounded channels, $f = \lfloor (k-1)/2 \rfloor$ all correct. (a) $k = 3$, (b) $k = 5$.

Multi-Random VS Multi-Shortest

We proposed as a countermeasure against the capability of Byzantine processes to flood the network a constraint on the channel capacity, namely limiting the amount of messages that a process can send over a link per round, and we set this bound equal to $f + 1$. Then, we proposed two forwarding policies to select which pathsets relay in the actual round. Assuming bounded channels, we compare the presented policies, *Multi-Random* and *Multi-Shortest* , considering networks of size $n = 100$, topologies random regular, multipartite wheel, k-diamond and k-pasted-tree, passive Byzantine failures. The results are presented in Figures 7 and 8 (notice that scale of the graphics in Figure 7 are logarithmic). Starting with the Multi-Random policy, it can be seen in Figures 7 that, while for some graphs the Multi-Random policy acts smoothly, the random regular (confirming the results achieved in our preliminary work [3]) the multipartite wheel graphs and the k-diamond, there exist topologies where the broadcast latency and message complexity may conspicuously increase (k-pasted-tree). It follows that on some kind of graphs the selections of paths that the Multi-Random policy may take are not equivalment with respect the protocol progression and that additional criterion have to be considered in the selection. This lead us to discard such a policy to be one generally employable. Contrarily, the performance achieved employing the Multi-Shortest policy appears not affected by this misbehaviour (Figures 8). Therefore, we further investigate the Multi-Shortest policy while increasing the size of the network.

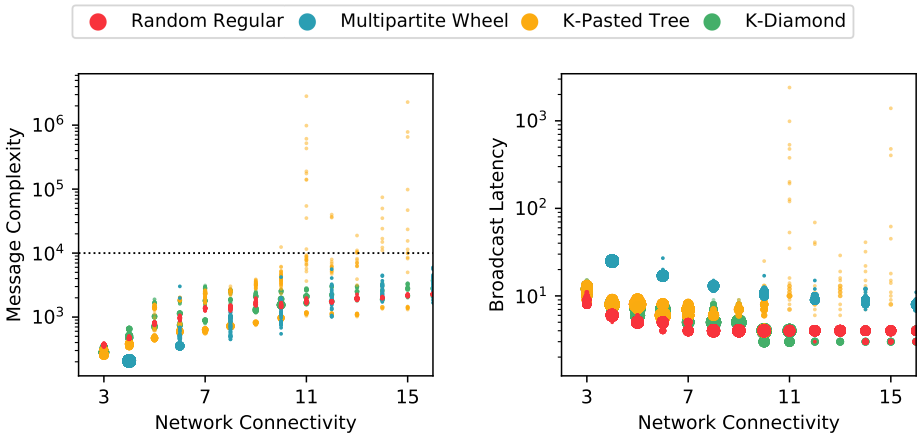


Figure 7: **Multi-Random policy** $n = 100$, $f = \lfloor (k-1)/2 \rfloor$ passive Byzantines, bounded channels, message complexity and broadcast latency.

Multi-shortest policy detailed evaluation

We assume bounded channels and the Multi-Shortest policy, considering networks of size $n = 150$ and $n = 200$, topologies random regular, multipartite wheel, k-diamond and k-pasted-tree, passive and active Byzantine failures. First results are presented in Figures 9 and 10. It is possible to see that the trends of the message complexity and broadcast latency keep defined employing our protocol joined with the Multi-Shortest policy while increasing the size of the network and considering passive Byzantine failures. Specifically, the message complexity keeps always close or below the n^2 boundary. It can also be deduced that a regular network not necessarily results in optimal performances employing our protocol, indeed there are notable differences in the results obtained considering different topologies.

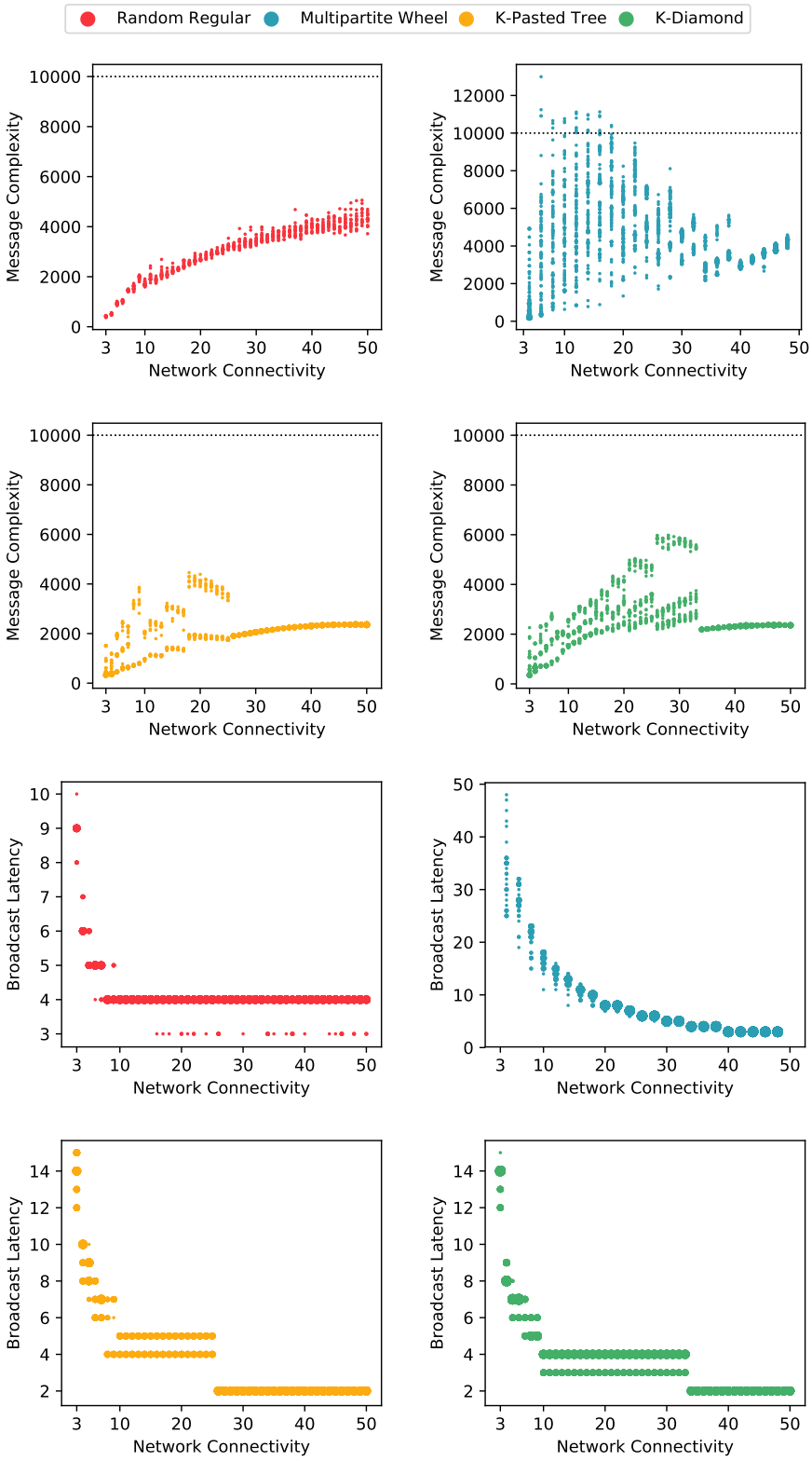


Figure 8: **Multi-Shortest policy, message complexity and broadcast latency, passive Byzantines, bounded channels,** $n = 100$, $f = \lfloor (k-1)/2 \rfloor$.

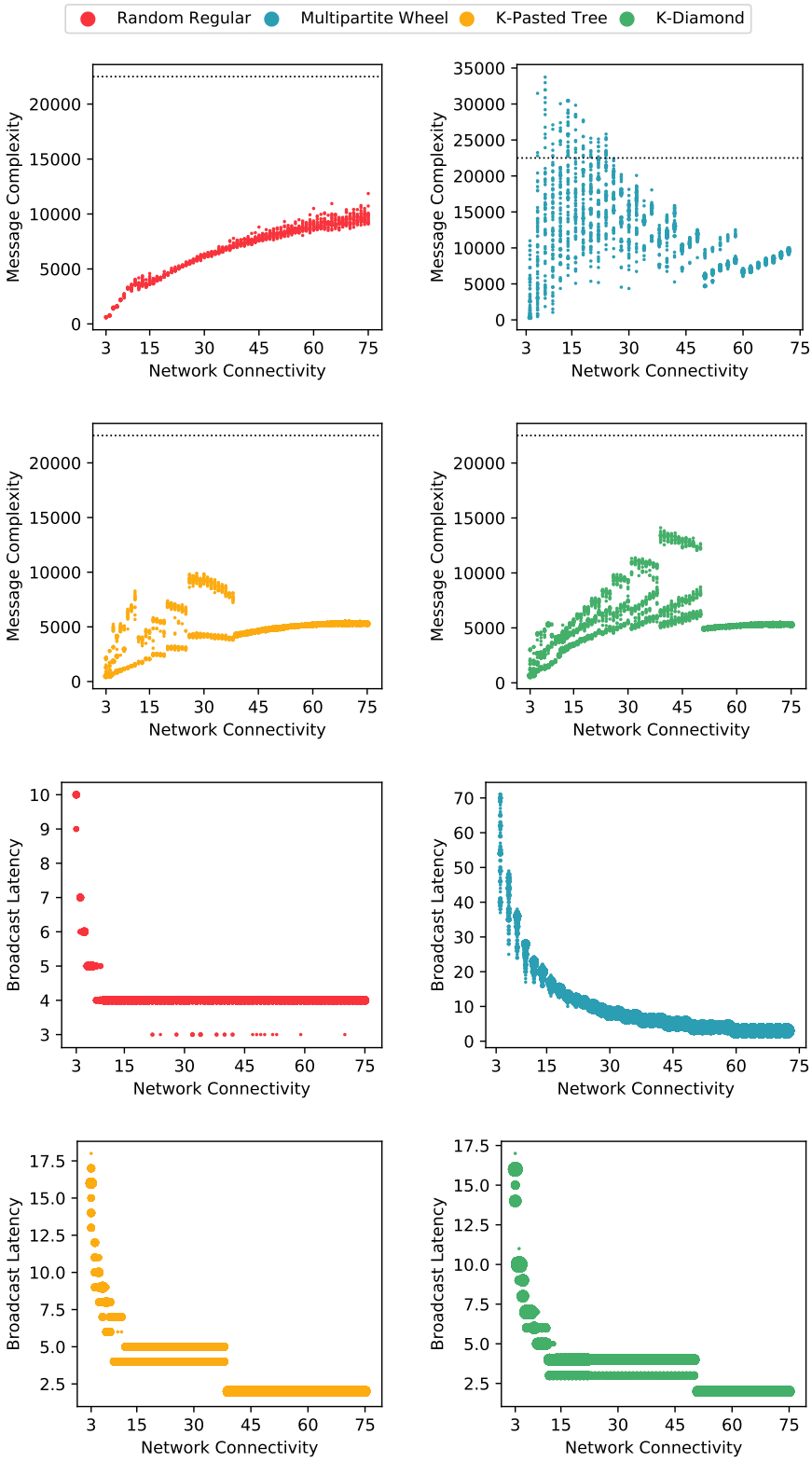


Figure 9: **Multi-Shortest policy, message complexity and broadcast latency, passive Byzantines, bounded channels**, $n = 150$, $f = \lfloor (k-1)/2 \rfloor$.

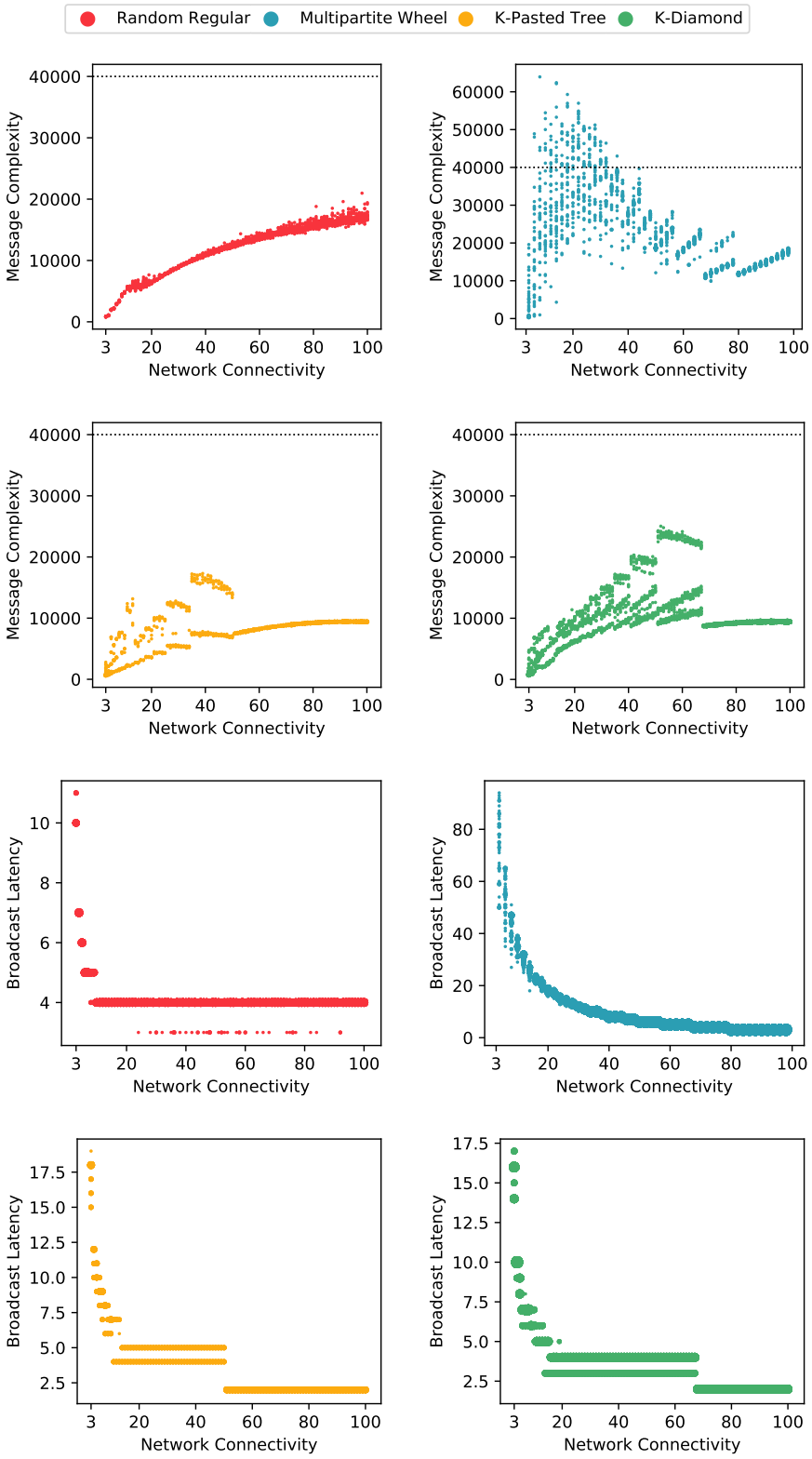


Figure 10: **Multi-Shortest policy, message complexity and broadcast latency, passive Byzantines, bounded channels**, $n = 200$, $f = \lfloor (k-1)/2 \rfloor$.

It can also be noticed from the distribution of the measures that there are several topologies (K-Pasted-Tree, K-Diamond and expecially Multipartite wheel) where the placement of the source and the Byzantine failures plays a remarkable impact on the message complexity. Additional details will be later provided.

To evaluate the effects of the Multi-Shortest policy on the broadcast latency, we simulate the *BFT-BRB* protocol employing either the Multi-Shortest policy or unbounded channels, considering passive Byzantine processes and networks of size $n = 100$. It can be deduced (Figure 11) that the policy we defined introduces negligible delays.

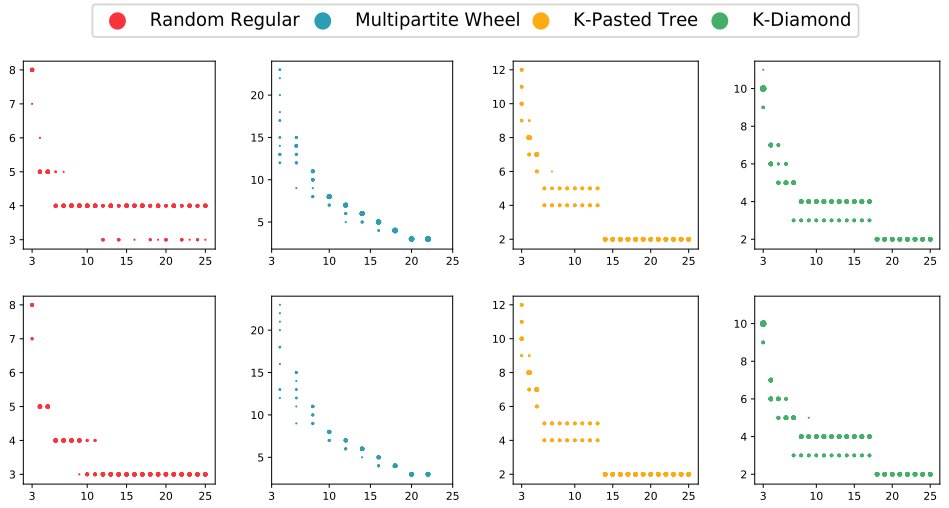


Figure 11: **Delay forwarding policy.** X-axis: network connectivity, Y-axis: broadcast latency, fist row: Multi-Shortest bounded channel, second row: unbounded channel, $n = 50$, $f = \lfloor (k-1)/2 \rfloor$ passive Byzantine.

We move to consider the case of active Byzatine processes, specifically in Figure 12 general active (non omniscient) Byzantine faults are assumed. It can be noticed that, spreading spourious pathsets (using the strategy we defined) once they get knowledge about a content, the Byzantine processes have no negative impact on the message complexity. As a matter of facts they may even help correct processes achieving reliable broadcast (because they relay the content even if they try not to increase the VC on the receiving processes).

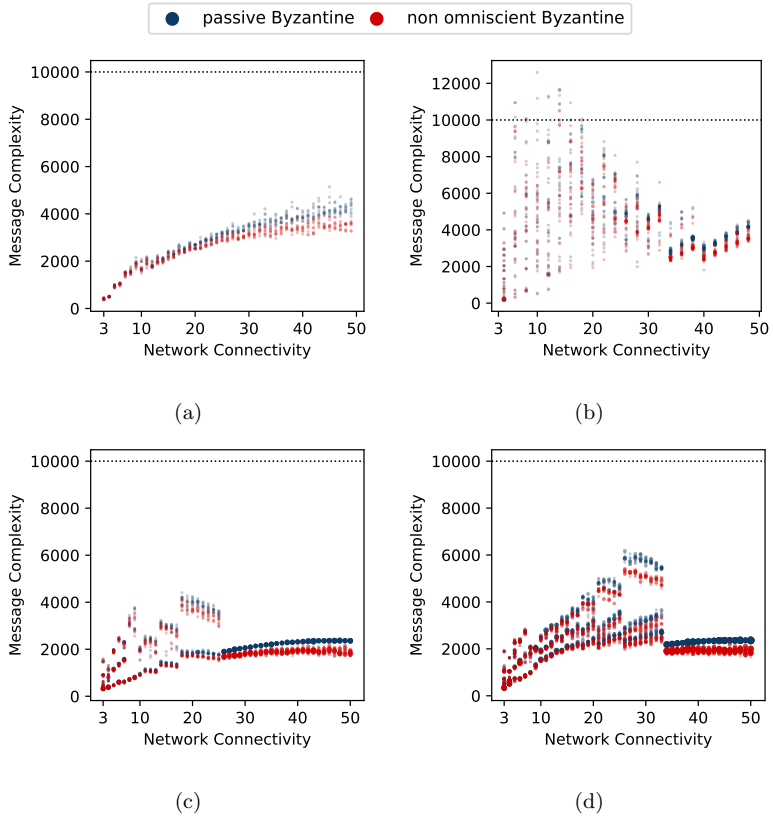


Figure 12: **Multi-Shortest policy, message complexity with either passive or non omniscient (general) Byzantine.** $n = 100$, $f = \lfloor (k-1)/2 \rfloor$, bounded channels. (a) random regular, (b) multipartite wheel (c) k-pasted-tree, (d) k-diamond.

We consider the case of omniscient Byzantine faulty processes, which start spreading spurious pathset about the content from the beginning of broadcast. The results we obtained are presented in Figures 13, 14. It is possible to see that such stronger Byzantine faults are able to remarkably increase the message complexity, nonetheless it keeps close to the n^2 threshold.

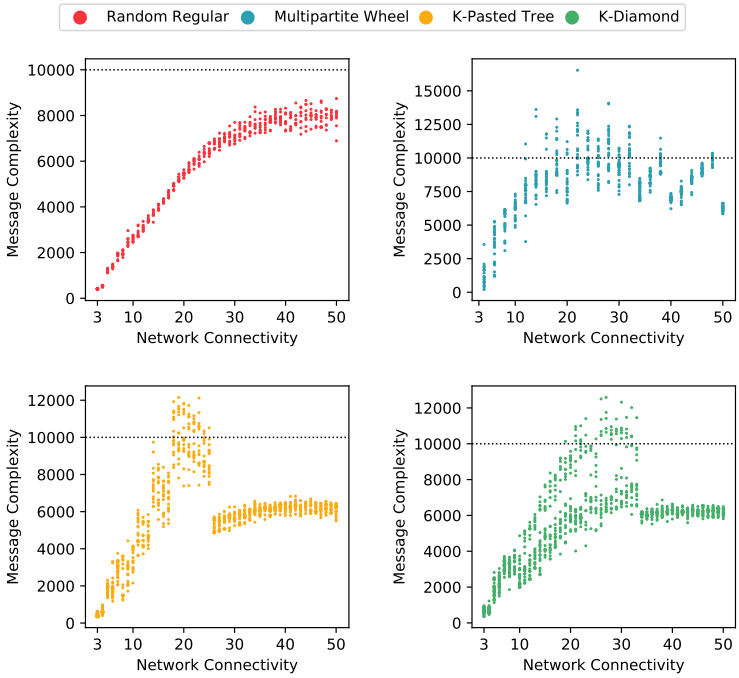


Figure 13: **Multi-Shortest policy, omniscient Byzantine faults.** $n = 100$, $f = \lfloor (k-1)/2 \rfloor$, bounded channels.

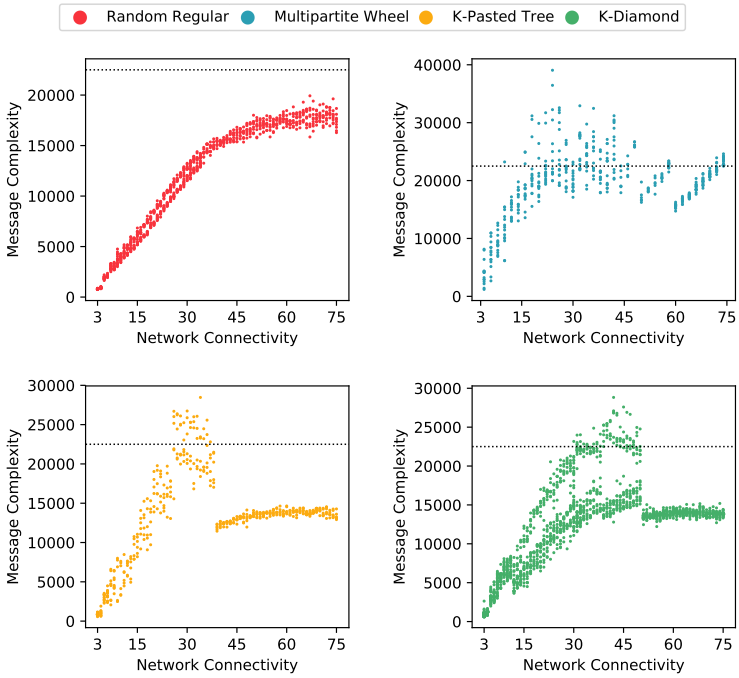
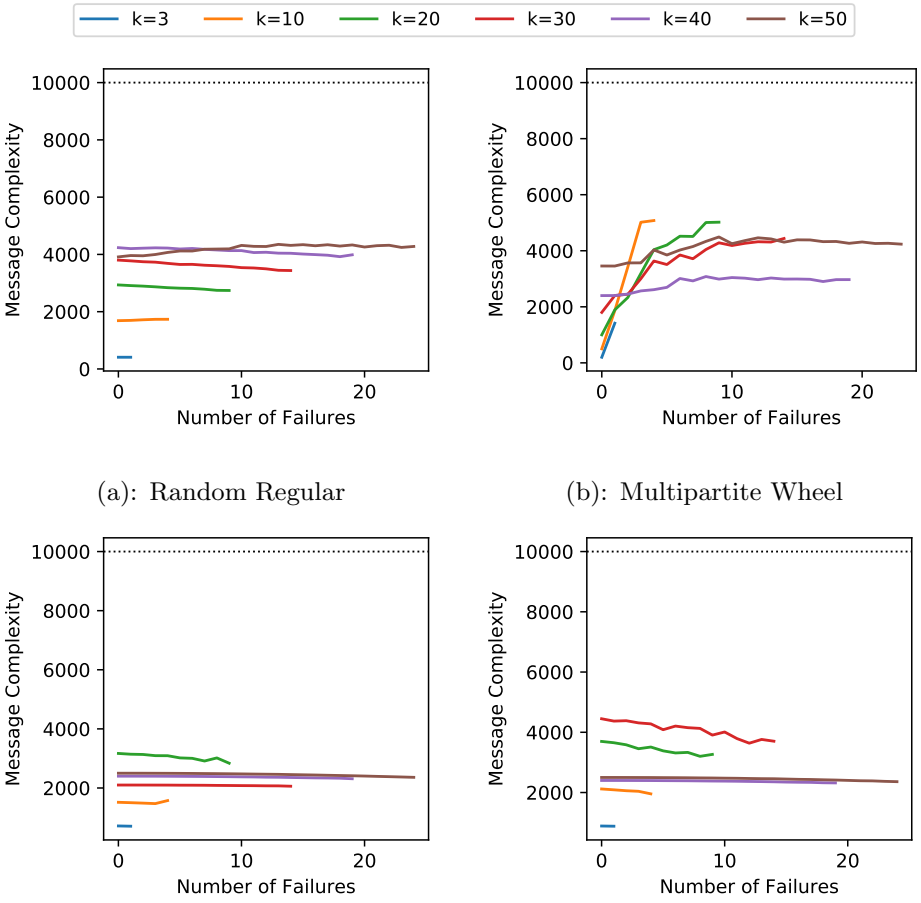


Figure 14: **Multi-Shortest policy, omniscient active Byzantine faults.** $n = 150$, $f = \lfloor (k-1)/2 \rfloor$, bounded channels.

Varying the number of failures

We evaluate how the message complexity evolves when the number of faulty processes is not maximized. We plotted the results we obtained in Figures 15,16. Whatever is the amout of failures, processes deliver a content only if the associated minimum cut is greater than $\lfloor (k-1)/2 \rfloor$.

It is possible to deduce that the resulting message complexity depends on the specific topology considered and on the degree of connectivity. Specifically, both in case of passive and omniscient active Byzantine faults, there are settings where the message complexity remains constant independently from the number of effective failures and others where the message complexity increases exponentially with the number of failures.



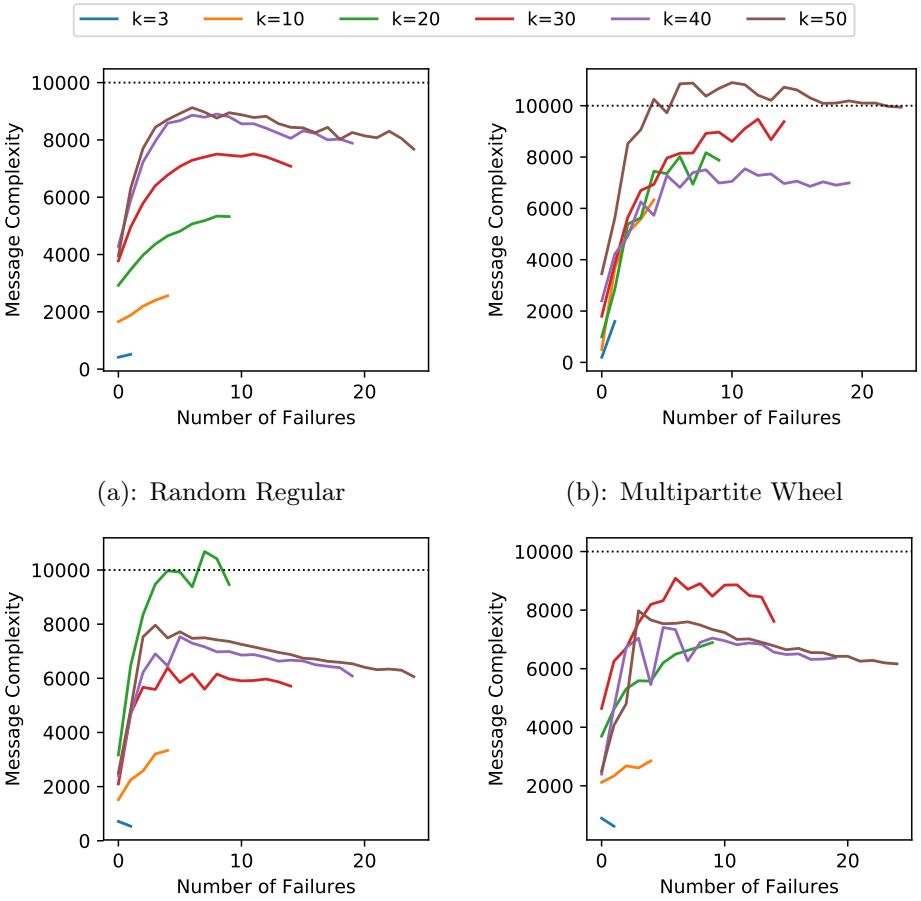
(a): Random Regular

(b): Multipartite Wheel

(c): K-Pasted-Tree

(d): K-Diamond

Figure 15: **Multi-Shortest policy, varying the number of faults, message complexity, passive Byzantines.** $n = 100$.



(a): Random Regular

(b): Multipartite Wheel

(c): K-Pasted-Tree

(d): K-Diamond

Figure 16: **Multi-Shortest policy, varying the number of faults, message complexity, active omniscient Byzantines.** $n = 100$.

Barabási-Albert graph

We separately evaluated in Figure 17 our algorithm in a Barabási-Albert graph while varying the attachment parameter m, in order analyze our protocol on a topology with different degree distribution with respect the previous analyzed. The *BFT-BRB* protocol and the Multi-Shortest forwarding policy shown to keep performing in the same manner. To allow the reader to make a comparison with the other topologies, we plot in Figure 18 the relation between the attachment parameter m and the network connectivity.

These simulations allow us to conclude that a Byzantine tolerable reliable broadcast protocol practically employable in synchronous systems without considering further assumptions with respect the state of art is achievable.

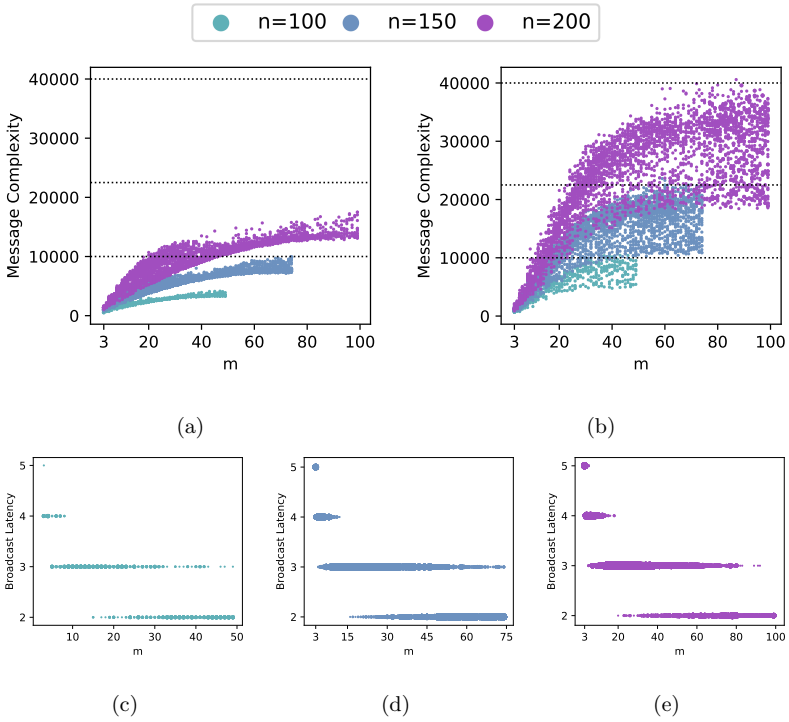


Figure 17: **Multi-Shortest policy, Barabasi-Albert Network, message complexity.** $f = \lfloor (k-1)/2 \rfloor$, bounded channels, $n = 100, 150, 200$ (a) passive Byzantines, (b) active omniscient Byzantines.

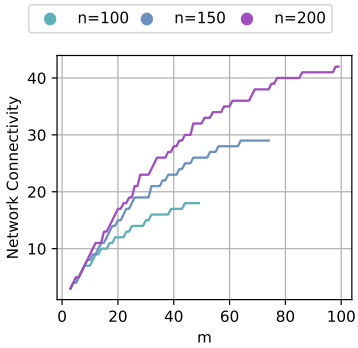
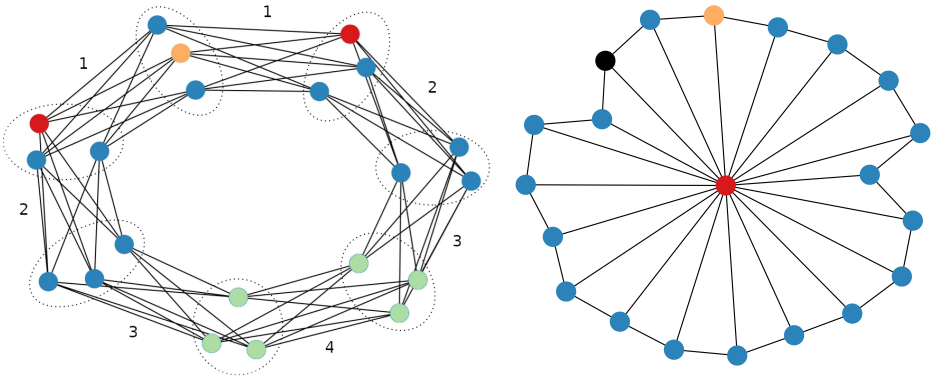


Figure 18: **Barabasi-Albert Network, relation between the attachment parameter m and the network connectivity**

Worst Case Scenarios

For the ease of completeness, we briefly survey two worst case scenarios: the multipartite wheel and the generalized wheel. In Figure 19a is summarized one of executions we are going to present. Let us consider the multipartite

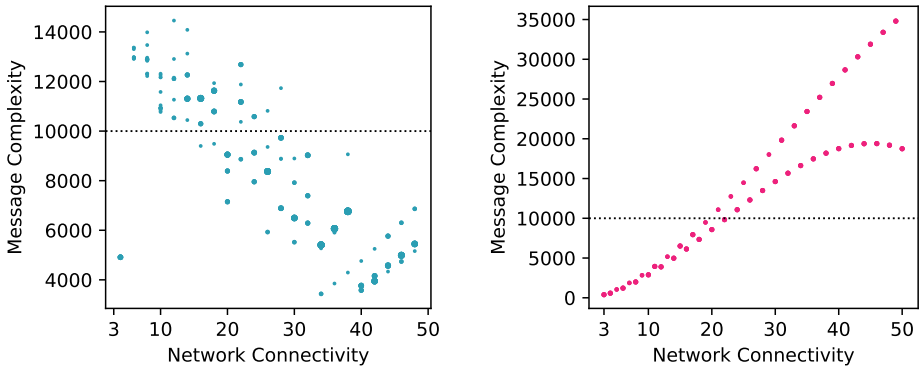
wheel of size $n = 21$ and $k = 6$, choose a node as source (in Figure 19a depicted in orange) and place two faulty processes (in red) in its neighborhood in distinct groups (i.e. those neighbor will have different neighbors). It results that only two correct processes per group delivers the content during the first round. Subsequently, they relay the message to all the nodes in the consecutive group. But, none of this node is able to deliver the message: the minimum cut of the generated paths is 2 and processes demand paths with minimum cut at least 3. The nodes succeed in delivering the message only when "the propagation on the two sides met", achieving a minimum cut of 4. It can be noticed that a considerable amount of paths may be generated in this specific worst case scenario while the values of n and k increases. Nonetheless, the *BFT-BRB* protocol and the Multi-Shortest policy reduced such a message complexity case as shown in Figures 8, 9 and 10. We additionally simulate in Figure 20a our protocol with the Multi-Shortest policy on a multipartite wheel of size $n = 100$ with passive Byzantine in the worst placement.



(a) (b)

Figure 19: **Worst case scenarios.** Multi-Partite Wheel (a) and Generalized Wheel (b).

Another worst case scenario is depicted in Figure 19b. Let us assume a generalized wheel, pick a source on the cycle and the Byzantine processes always located on the clique. The Figure 20b show that in this specific case our algorithm and the Multi-Shortest policy are less effective in reducing the message complexity while Byzantine processes are located in the clique.



(a) (b)

Figure 20: **Multi-Shortest policy, generalized wheel, worst Byzantine placementm passive Byzantine.** (a) multipartite wheel, (b) generalized wheel.

Conclusion

We revisited available solutions for the reliable broadcast in general network hit by up to f arbitrarily distributed Byzantine failures, and proposed modifications following performance related observations. Although the delivery complexity of our protocol remains unchanged with respect the state-of-art solutions, our experiments show that it is possible to drastically reduce the message complexity (from factorial to polynomial in the size of the network), practically enabling reliable broadcast in larger systems and networks with authenticated channels. There are several open problems that may follows: is it possible to define a solution to the hitting set problem suited for the specific input generated by our protocol? Is it possible to remove from the system the contents generated by Byzantine processes? And under which assumption? Which are the graph parameters that govern the message complexity of our protocol? Our results open to the possibility of identifying a polynomial theoretical bound on message complexity solving the reliable broadcast problem with honest dealer. Finally, the Bizantine Reliable Broadcast problem should be analyzed also on dynamic networks. Even if the protocol we proposed can directly be employed on asynchronous and/or dynamic systems, the achieved gain in message complexity is not guaranteed due to the weaker synchrony assumptions, and probably specific assumption on the evolution of the system must be guaranteed in searching a practical employable solution.

References

[1] Baldoni, R., Bonomi, S., Querzoni, L., Tucci Piergiovanni, S.: Investigating the existence and the regularity of logarithmic harary graphs. Theor. Comput. Sci. **410**(21-23), 2110–2121 (2009). DOI 10.1016/j.tcs.2009.01.041

[2] Barabási, A.L., Albert, R.: Emergence of scaling in random networks. science **286**(5439), 509–512 (1999). DOI 10.1126/science.286.5439.509

[3] Bonomi, S., Farina, G., Tixeuil, S.: Multi-hop byzantine reliable broadcast made practical. In: 8th Latin-American Symposium on Dependable Computing, LADC 2018, Foz do Iguaçu, Brazil, October 8-10, 2018, pp. 155–160 (2018). DOI 10.1109/LADC.2018.00026

[4] Bonomi, S., Farina, G., Tixeuil, S.: Multi-hop byzantine reliable broadcast made practical simulation code (2019). DOI 10.5281/zenodo.3314359. URL `https://github.com/giovannifarina/BFT-BRB`

[5] Castro, M., Liskov, B.: Practical byzantine fault tolerance and proactive recovery. ACM Trans. Comput. Syst. **20**(4), 398–461 (2002). DOI 10.1145/571637.571640

[6] Diestel, R.: Graph Theory. Springer Berlin Heidelberg (2017). DOI 10.1007/978-3-662-53622-3

[7] Dolev, D.: Unanimity in an unknown and unreliable environment. In: 22nd Annual Symposium on Foundations of Computer Science, Nashville, Tennessee, USA, 28-30 October 1981, pp. 159–168 (1981). DOI 10.1109/SFCS.1981.53

[8] Douceur, J.R.: The sybil attack. In: Peer-to-Peer Systems, First International Workshop, IPTPS 2002, Cambridge, MA, USA, March 7-8, 2002, Revised Papers, pp. 251–260 (2002). DOI 10.1007/3-540-45748-8_24

[9] Drabkin, V., Friedman, R., Segal, M.: Efficient byzantine broadcast in wireless ad-hoc networks. In: 2005 International Conference on Dependable Systems and Networks (DSN 2005), 28 June - 1 July 2005, Yokohama, Japan, Proceedings, pp. 160–169 (2005). DOI 10.1109/DSN.2005.42

[10] Gainer-Dewar, A., Vera-Licona, P.: The minimal hitting set generation problem: Algorithms and computation. SIAM J. Discrete Math. **31**(1), 63–100 (2017). DOI 10.1137/15M1055024

[11] Garey, M.R., Johnson, D.S.: Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA (1990)

[12] Hsu, D.F.: On container width and length in graphs, groups, and networks–dedicated to professor paul erdös on the occasion of his 80th birthday–. IEICE transactions on fundamentals of electronics, communications and computer sciences **77**(4), 668–680 (1994)

[13] Hsu, D.F., Luczak, T.: On the k-diameter of k-regular k-connected graphs. Discrete Mathematics **133**(1-3), 291–296 (1994). DOI 10.1016/0012-365X(94)90036-1

[14] Jenkins, K., Demers, A.J.: Logarithmic harary graphs. In: 21st International Conference on Distributed Computing Systems Workshops (ICDCS 2001 Workshops), 16-19 April 2001, Phoenix, AZ, USA, Proceedings, pp. 43–50 (2001). DOI 10.1109/CDCS.2001.918685

[15] Kempe, D., Kleinberg, J.M., Kumar, A.: Connectivity and inference problems for temporal networks. J. Comput. Syst. Sci. **64**(4), 820–842 (2002). DOI 10.1006/jcss.2002.1829

[16] Lovász, L., Neumann-Lara, V., Plummer, M.: Mengerian theorems for paths of bounded length. Periodica Mathematica Hungarica **9**(4), 269–276 (1978). DOI 10.1007/bf02019432

[17] Maurer, A., Tixeuil, S.: Byzantine broadcast with fixed disjoint paths. J. Parallel Distrib. Comput. **74**(11), 3153–3160 (2014). DOI 10.1016/j.jpdc.2014.07.010

[18] Maurer, A., Tixeuil, S.: Containing byzantine failures with control zones. IEEE Trans. Parallel Distrib. Syst. **26**(2), 362–370 (2015). DOI 10.1109/TPDS.2014.2308190

[19] Maurer, A., Tixeuil, S.: Tolerating random byzantine failures in an unbounded network. Parallel Processing Letters **26**(1) (2016). DOI 10.1142/S0129626416500031

[20] Maurer, A., Tixeuil, S., Défago, X.: Communicating reliably in multihop dynamic networks despite byzantine failures. In: 34th IEEE Symposium on Reliable Distributed Systems, SRDS 2015, Montreal, QC, Canada, September 28 - October 1, 2015, pp. 238–245 (2015). DOI 10.1109/SRDS.2015.10

[21] Murakami, K., Uno, T.: Efficient algorithms for dualizing large-scale hypergraphs. Discrete Applied Mathematics **170**, 83–94 (2014). DOI 10.1016/j.dam.2014.01.012

[22] Nesterenko, M., Tixeuil, S.: Discovering network topology in the presence of byzantine faults. IEEE Trans. Parallel Distrib. Syst. **20**(12), 1777–1789 (2009). DOI 10.1109/TPDS.2009.25

[23] Pagourtzis, A., Panagiotakos, G., Sakavalas, D.: Reliable broadcast with respect to topology knowledge. Distributed Computing **30**(2), 87–102 (2017). DOI 10.1007/s00446-016-0279-6

[24] Pelc, A., Peleg, D.: Broadcasting with locally bounded byzantine faults. Inf. Process. Lett. **93**(3), 109–115 (2005). DOI 10.1016/j.ipl.2004.10.007

[25] Steger, A., Wormald, N.C.: Generating random regular graphs quickly. Combinatorics, Probability & Computing **8**(4), 377–396 (1999). DOI 10.1017/S0963548399003867

[26] Xu, J.: Topological Structure and Analysis of Interconnection Networks, 1st edn., p. 257. Springer Publishing Company, Incorporated (2010)

[27] Zeng, K., Govindan, K., Mohapatra, P.: Non-cryptographic authentication and identification in wireless networks. IEEE Wireless Commun. **17**(5), 56–62 (2010). DOI 10.1109/MWC.2010.5601959