Reliable Broadcast in Dynamic Networks with Locally Bounded Byzantine Failures

Silvia Bonomi, Giovanni Farina, Sébastien Tixeuil

Reliable Broadcast in Dynamic Networks with Locally Bounded Byzantine Failures

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

[⋆]Dipartimento di Ingegneria Informatica Automatica e Gestionale Antonio Ruberti,
Università degli Studi di Roma La Sapienza, Rome, Italy
bonomi@diag.uniroma1.it
[†]Sorbonne Université,
CNRS, Laboratoire d'Informatique de Paris 6, LIP6, F-75005 Paris, France
Giovanni.Farina@lip6.fr, Sebastien.Tixeuil@lip6.fr

Abstract

Ensuring reliable communication despite possibly malicious participants is a primary objective in any distributed system or network. In this paper, we investigate the possibility of reliable broadcast in a dynamic network whose topology may evolve while the broadcast is in progress. In particular, we adapt the Certified Propagation Algorithm (CPA) to make it work on dynamic networks and we present conditions (on the underlying dynamic graph) to enable safety and liveness properties of the reliable broadcast. We furthermore explore the complexity of assessing these conditions for various classes of dynamic networks.

1 Introduction

Designing dependable and secure 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 nodes participating in the system may behave arbitrarily (including in a malicious manner), while the rest of processes remain correct. Also, reliable communication primitives are a core building block of any distributed software. Finally, as current applications are run for extended periods of time with expected high availability, it becomes mandatory to integrate dynamic changes in the underlying network while the application is running. In this paper, we address the reliable *broadcast* problem (where a *source* node must send data to every other node) in the context of dynamic networks (whose topology may change while the broadcast is in progress) that are subject to Byzantine failures (a subset of the nodes may act arbitrarily). The reliable broadcast primitive is expected to provide two guarantees: *(i) safety*, namely if a message m is delivered by a correct process, then m was sent by the source and *(ii) liveness*, namely if a message m is sent by the source, it is eventually delivered by every correct process.

This problem can be solved using cryptographic digital signatures [8, 12]. Yet, cryptographic infrastructures make the assumption that some components cannot be compromised (*e.g.* secret key distributor, public key provider, etc.), weakening the generality of the Byzantine model. For this reason, we focus in this paper on non-cryptographic solutions. As a matter of facts, let us note that a common assumption of Byzantine tolerant reliable broadcast protocols is to use authenticated point-to-point channels, which prevent a node from impersonating several ones (Sybil attack [11]). The real difference between the cryptographic and non-cryptographic reliable broadcast protocols is how the cryptography is employed: the non-cryptographic protocols, in fact, may use digital signatures just within neighbors for authentication purposes, whereas the cryptographic protocols employs cryptographic primitives to enable the message verification even between non-directly connected nodes. Moreover, let us note that an implementation of an authenticated channel not necessarily requires the use of cryptography [23].

Related Works. In static networks (that is, in networks whose topology remains fixed during the entire execution of the protocol), a solution for the reliable broadcast problem has been initially provided for complete networks [16], requiring that no more than one third of the nodes are Byzantine (that is, $n > 3f$, where n denotes the size of the network ad f the maximum number of Byzantine nodes). In multi-hop networks, a necessary and sufficient condition is due to Dolev [10], and states that reliable broadcast can be solved if and only if the network is $2f + 1$-connected. This global condition was replaced by a local condition on the number of Byzantine neighbors a node may have [15, 22]. All aforementioned works require high network connectivity.

Extending a reliable broadcast service to sparse networks required to weaken the achieved guarantees [20, 19, 18]: *(i)* accepting that a small minority of correct nodes may accept invalid messages (thus compromising safety), or accepting that a small minority of correct nodes may not deliver genuine messages (thus compromising liveness).

Adapting to dynamic networks proved difficult, as the topology assumptions made by the mentioned proposals may no longer hold as the topology of the network evolves during execution. Without consideration of Byzantine failures, the problem of broadcasting in dynamic networks was extensively studied [4, 6, 5, 3, 1]. Some core problems of distributed computing have been considered in the context of dynamic networks subject to Byzantine failures [13, 2] but, to the best of our knowledge, there exists a single contribution for the reliable communication problem, due to Maurer *et al.* [21]. This reliable broadcast can be seen as the dynamic network extension of the Dolev [10] solution for static networks, and assumes that no more than f Byzantine

processes are present in the network. Also, the protocol to be executed spreads an exponential number of messages with respect to the size of the network and requires to each node to compute the minimal cut over the set of paths traversed by each received message, making the protocol unpractical for real applications.

Contributions. In this paper, we investigate the possibility of reliable broadcast in a dynamic network that is subject to Byzantine faults. More precisely, we address the possibility of a local criterion on the number of Byzantine (as opposed to a global criterion as in Maurer *et al.* [21]) in the hope that a practically efficient protocol can be derived in case the criterion is satisfied. Our starting point is the CPA protocol [15, 22], that was originally designed for static networks. In particular, our contributions can be summarized as follows: *(i)* we extend the CPA algorithm to make it work in dynamic networks; *(ii)* we prove that the original safety property of CPA naturally extends to dynamic networks and we define new liveness conditions specifically suited for the dynamic networks and *(iii)* we investigate the impact of nodes awareness about the dynamic network on reliable broadcast possibility and efficiency.

2 System Model & Problem Statement

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. The passage of time is measured according to a fictional global clock spanning over natural numbers $\mathbb{N}$. The processes are arranged in a multi-hop communication network. The network can be seen as an undirected graph where each node represents a process $p_i \in \Pi$ and each edge represents a communication channel between two elements $p_i, p_j \in \Pi$ such that p_i and p_j can communicate.

Dynamic Network Model. The communication network is *dynamic* i.e., the set of edges (or available communication channels) changes along time. More formally, we model the network as a *Time Varying Graph* (TVG) [7] i.e., a graph $\mathcal{G} = (V, E, \rho, \zeta)$ where:

- V is the set of processes (in our case $V = \Pi$);
- $E \subseteq V \times V$ is the set of edges (i.e., communication channels).
- $\rho : E \times \mathbb{N} \to \{0, 1\}$ is the *presence* function. Given an edge $e_{i,j}$ between two nodes p_i and p_j, $\rho(e_{i,j}, t) = 1$ indicates that edge $e_{i,j}$ is present at time t;
- $\zeta : E \times \mathbb{N} \to \mathbb{N}$ is the *latency* function that indicates how many time is needed to cross an edge starting from a given time t. In particular, $\zeta(e_{i,j}, t) = \delta_{i,j,t}$ indicates that a message m sent at time t from p_i to p_j takes $\delta_{i,j,t}$ time units to cross edge $e_{i,j}$.

The evolution of $\mathcal{G}$ can also be described as a sequence of static graphs $\mathcal{S}_\mathcal{G} = G_0, G_1, \dots G_T$ where G_i corresponds to the *snapshot* of $\mathcal{G}$ at time t_i (i.e. $G_i = (V, E_i)$ where $E_i = \{e \in E \mid \rho(e, t_i) = 1\}$). No further assumption on the evolution of the dynamic network are done.

The static graph $G = (V, E)$ that considers all the processes and all the possible existing edges is called *underlying graph* of $\mathcal{G}$ and it flattens the time dimension indicating only the pairs of nodes that have been connected at some time t'.

In the following, we interchangeably use terms *process* and *node* and we will refer to *edges* and *communication channels* interchangeably.

Let us note that the TVG model is one among the most general available and it is able to abstract and model several real dynamic networks [7].

Communication model and Timing assumption. Processes communicate through message exchanges. Every message has (i) a *source*, which is the id of the process that has created the message and it is

encapsulated in the message content, and (ii) a *sender*, that is the id of the process that is relaying the message. The source and the sender may be the same. The sender of a message is always the process itself or a neighbor in the communication network. We refer with m_s to a message m with p_s as source.

We assume *authenticated* and *reliable* point-to-point channels where (a) *authenticated* ensures that the identity of the sender cannot be forged; (b) *reliable* guarantees that the channel delivers a message m if and only if (i) m was previously sent by p and (ii) the channel has been up long enough to allow the reception (i.e. given a message m sent at time t from p_i to p_j and having latency $\delta_{i,j,t}$, we will have reliable delivery if $\rho(e_{i,j}, \tau) = 1$ for each $\tau \in [t, t + \delta_{i,j,t}]$). Notice that these channel assumptions are implicitly made also on analysis of CPA on static networks and that they are both essential to guarantees the reliable broadcast properties.

At every time unit t each process takes the following actions: (i) *send* where processes send all the messages for the current time unit (potentially none), (ii) *receive* where processes receive and store all the messages for the current time unit (potentially none) and (iii) *computation* where processes process the buffer of received messages and compute messages to be sent during the next time unit according to the deterministic distributed protocol $\mathcal{P}$ that they are executing.

Thus, the system is assumed to be synchronous in the sense that (i) every channel has a latency function that is bounded and the overall message delivery time is bounded by the maximum channel latency and (ii) computation steps are bounded by a constant that is negligible with respect to the overall message delivery time and we consider it equal to 0. We discuss the implication and consequences of lack of synchrony in Section 6.

Failure model. We assume an omniscient adversary able to control several processes of the network allowing them to behave arbitrarily (including corrupting/dropping messages or simply crashing). We call them *Byzantine* processes. Processes that are not Byzantine faulty are said to be *correct*. Correct processes do not a priori know which processes are Byzantine.

We considered the *f-locally bounded* failure model [15] as all CPA related works, i.e., along time every process p_i can be connected with at most f Byzantine processes. In other words, given the underlying static graph $G = (V, E)$, every process $p_i \in V$ has at most f Byzantine neighbors in G.

Specifically to reliable broadcast protocols, a Byzantine process p_b can spread messages $\tilde{m}_s$ with a fake source p_s or it can drop any received message preventing its propagation.

Problem Statement. In this paper, we consider the problem of *Reliable Broadcast over dynamic networks* assuming a f-locally bounded Byzantine failure model from a given correct source p_s. We say that a protocol $\mathcal{P}$ satisfies *reliable broadcast*, if a message m broadcast by a *correct* [1] process $p_s \in \Pi$ (also called *source* or *author*) is eventually delivered (i.e., accepted as a valid message) by every correct process $p_j \in \Pi$. Said differently, a protocol $\mathcal{P}$ satisfies *reliable broadcast*, if the following conditions are met:

- **Safety** if a message m is delivered by a correct process, then such message has been sent by the source p_s;
- **Liveness**: if a message m is broadcast by the source p_s, it is eventually delivered by every correct process.

3 The Certified Propagation Algorithm (CPA)

The Certified Propagation Algorithm (CPA) [15, 22] is a protocol enforcing reliable broadcast, from a correct source p_s, in *static* multi-hop networks with a f-locally bounded Byzantine adversary model, where nodes

[1] note the assumption of a possibly faulty source leads to a more general problem, the *Byzantine Agreement* [9].

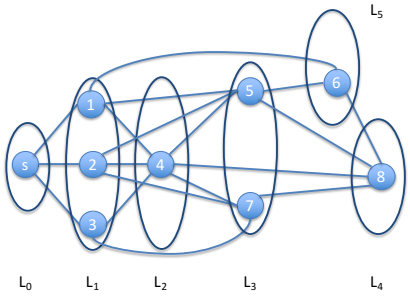


Figure 1: A graph $G = (V, E)$ allowing a k-level ordering with $k = 3$.

have no knowledge on the global network topology. Given a message m to be broadcast, CPA starts the propagation of m_s from p_s and applies three acceptance policies (denoted by *AC*) to decide if m_s should be delivered and forwarded (*i.e.*, transmitted also by nodes different from the source) by a process p_j. Specifically:

- p_s delivers m_s **(AC1)**, forwards it to all of its neighbors, and stops;
- when receiving m_s from p_i, if p_i it the source then p_j delivers m_s **(AC2)**, forwards m_s to all of its neighbors and stops; otherwise the message is buffered.
- upon receiving $f + 1$ copies of m_s from distinct neighbors, p_j delivers m_s **(AC3)**, then forwards it to all its neighbors and stops.

CPA correctness on static networks has been proved to be dependent on the network topology. In particular, Litsas *et al.* [17] provided topological conditions based on the concept of k-*level ordering*. Informally, given a graph $G = (V, E)$ and considering a node p_s as the source, we can define a k-level ordering as a partition of nodes into *ordered levels* such that: *(i)* p_s belongs to level L_0, *(ii)* all the neighbors of p_s belong to level L_1, and *iii)* each node in a level L_i has at least k neighbors over levels L_j, with $j < i$. A k-level ordering is *minimum* if every node appears in the minimum level possible, otherwise it is *relaxed*.

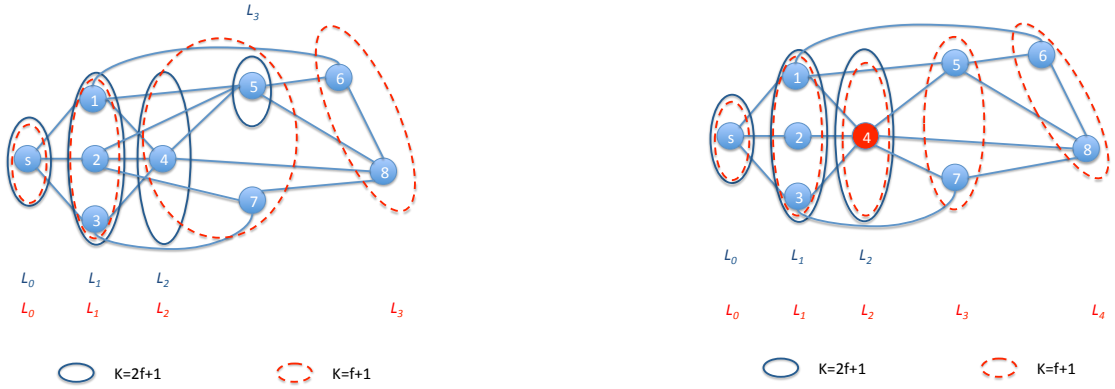
Definition 1 (MKLO) *Let $G = (V, E)$ be a graph and let p_s be a node of G called* source. *A* minimum k-level ordering *(*MKLO*) of G is a partition P_k of nodes into disjoint subsets called levels L_i defined as follows:*

$$\begin{cases} p \in L_0 & \text{if } p = p_s \\ p \in L_1 & \text{if } p \in N_s \\ p \in L_i & \text{if } p \in V \setminus (\bigcup_{j=0}^{i-1} L_j) \text{ and } |N_p \cap (\bigcup_{j=0}^{i-1} L_j)| \geq k \end{cases}$$

An example minimum k-level ordering (with $k = 3$) is shown in Figure 1. A relaxed k-level ordering for the same graph can be obtained by placing p_5 in a level and p_7 in another. Let us note that multiple relaxed k-level ordering may exist while the minimum k-level ordering is unique. Also, once a relaxed k-level ordering is defined, it can be reduced to a minimum k-level ordering.

For CPA to ensure reliable broadcast from p_s, a sufficient condition is that a k-level ordering exists, with $k \geq 2f + 1$. Conversely, the necessary condition demands a k-level ordering with $k \geq f + 1$ (see [17]). Considering Figure 1 and assuming $f = 1$, CPA ensures reliable broadcast from p_s as the sufficient condition

is satisfied. Those conditions can be verified with an algorithm whose time complexity is polynomial in the size of the network, specifically with a modified Breadth-first search. When a graph $G = (V, E)$ satisfies only the necessary condition but not the sufficient one, then further analysis must be carried out. In particular, in order to verify whether G enables reliable broadcast from p_s, one should check whether a k-level ordering exists (with $k = f + 1$) in every sub-graph G' obtained from G by removing all nodes corresponding to possible Byzantine position in the f-locally bounded assumption. The verification of the strict condition has been proven to be NP-Hard [14].



(a) $G = (V, E)$ (b) $G' = (V, E')$

Figure 2: Example of graphs where a k-level ordering exists for $k = f + 1$ but not for $k = 2f + 1$ (with $f = 1$).

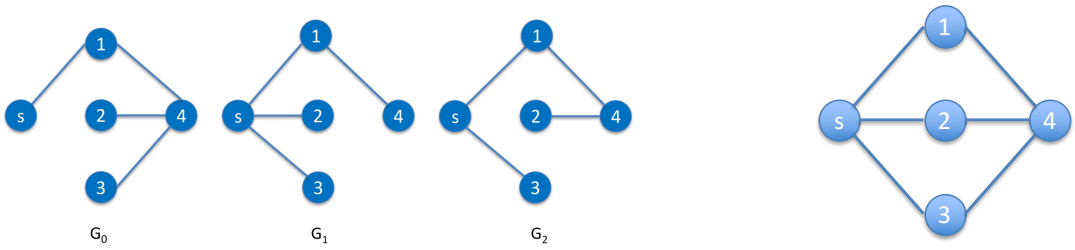
The graph $G = (V, E)$ depicted in Figure 2a does not match the sufficient condition but it satisfies the necessary one for CPA allowing the definition a k-level ordering with $k = f + 1$. Computing all the possible sub-graphs obtained by G by removing all the possible placements of a 1-locally bounded adversary, it is possible to deduce that a k-level ordering, with $k = f + 1$, still exists on every subgraph. Thus, CPA ensures reliable broadcast from p_s on G. Contrarily, the graph $G' = (V, E')$ depicted in Figure 2b has a sub-graph (the one obtained by removing p_4 as a possible faulty process) that does not admit a k-level ordering with $k = f + 1$ anymore. Thus, in this case, CPA does not ensure reliable broadcast from p_s.

4 The Certified Propagation Algorithm on Dynamic Networks

In this section, we consider how CPA behaves on dynamic networks, *i.e.* networks whose topology may evolve in time, and how it needs to be extended to work in such settings.

Let us consider the TVG shown in Figure 3 and suppose process p_2 is Byzantine. If we consider the static underlying graph $G = (V, E)$ shown in Figure 3b, it is easy to verify that running CPA from the source node p_s is possible to achieve reliable broadcast in a 1-locally bounded adversary. However, if we consider snapshots of the TVG at different times[2] as shown in Figure 3a, one can verify that nodes p_3 and p_4 remain unable to deliver the message forever. In fact, p_3 is not a neighbor of the source p_s when the message is broadcast by p_s (*i.e.*, at time t_0), and even if it had happened ($e_{s,3}$ at time t_0) the edge connecting p_4 with its correct neighbor p_3 appears only before the message would have been delivered and accepted by p_3, and thus it is not available for the retransmission. From this simple example its easy to see that the temporal

[2]For the sake of simplicity, we consider the channel delay always equal to 1 in the example.



(a) A Time Varying Graph $\mathcal{G} = (V, E, \rho, \zeta)$

(b) Underlying graph $G = (V, E)$

Figure 3: Example of a simple TVG and its underlying static graph.

dimension plays a fundamental role in the definition of topological constraints that a TVG must satisfy to enable reliable broadcast.

4.1 CPA Safety in Dynamic Networks

In the following, we show that the authenticated and reliable channels are necessary to ensure the reliable broadcast through CPA.

Lemma 1 *The CPA algorithm does not ensure safety of reliable broadcast when channels are not both authenticated and reliable (even in static graphs).*

Proof An authenticated channel guarantees that the identity of the sender of a message cannot be forged. Without this assumption a Byzantine process can impersonate an arbitrary number of processes and invalidate the *f-locally bounded* assumption.

A reliable channel guarantees that a message is received as it was sent. Without this assumption, an unreliable channel can simulate a Byzantine process that is not bound to authenticated channels, also invalidating the *f-locally bounded* assumption. $\square_{Lemma\ 1}$

The same channel assumptions are sufficient for ensuring safety also on dynamic networks.

Theorem 1 *Let $\mathcal{G} = (V, E, \rho, \zeta)$ be the TVG of a network with f-locally bounded Byzantine adversary. If every correct process p_i runs CPA on top of reliable authenticated channels, then if a message m is delivered by p_i, m was previously sent by the correct source p_s.*

Proof The proof trivially follows from CPA correctness in static networks with f-locally bounded adversary, considering that in the underlying graph $G = (V, E)$, we still have a f-locally bounded adversary. $\square_{Theorem\ 1}$

4.2 CPA Liveness in Dynamic Networks

The CPA liveness in static networks is based on the availability of a certain topology that supports the message propagation. Indeed every edge is always up so, once the communication network satisfies the topological constraints imposed by the protocol, the assumption that channels do not loose messages is sufficient to guarantee their propagation.

In dynamic networks, this is no longer true. Let us recall that each edge e in a TVG is up according to its presence function $\rho(e,t)$. At the same time, the message delivery latency are determined by the edge latency function $\zeta(e,t)$. As a consequence, in order to ensure a message m sent at time t from p_i to p_j is delivered, we need that (p_i,p_j) remains up until time $t+\zeta(e,t)$. Contrarily, there could exists a communication channel where every message sent has no guarantee to be delivered as the channel disappears while the message is still traveling.

Thus, in addition to topological constraints, moving to dynamic networks we need to set up other constraints on when edges appear and for how long they remain up.

Considering that processes have no information about the network evolution, they do not know if and when a given transmitted message will reach its receiver. Hence, without assuming extra knowledge, a correct process must re-send messages infinitely often.

As a consequence, CPA must be extended to the dynamic context incorporating the following additional steps:

- if process p_i delivers a message m, it forwards m to all of its neighbors infinitively often. In particular, p_i will retransmit at every time unit all the messages m received so far.

As a consequence, each time that the TVG changes the sets of neighbours of p_i, p_i tries to propagate the message to new nodes. Let us notice that infinite retransmission can be avoided/stopped only if a process get an acknowledge about the delivery of a sent message through a communication channel. This may require the channel to stay up more time or to appear again later on.
To ease of explanation, we will refer to this extended version of CPA as Dynamic CPA (DCPA).

We now characterize the conditions enabling a channel to deliver messages in order to argue about liveness. For this purpose, we define a boolean predicate whose value is true if and only if the TVG allows the reliable delivery of a message m sent from p_i to p_j at time t.

Definition 2 *Let $\mathcal{G}=(V,E,\rho,\zeta)$ be a TVG. We define the predicate* Reliable Channel Delivery at time t', $\mathsf{RCD}(p_i,p_j,t')$ *as follows:*

$$\mathsf{RCD}(p_i,p_j,t') = \begin{cases} \mathsf{true} & if\ \rho(<p_i,p_j>,\tau)=1,\ \forall \tau \in [t',t'+\zeta(e_{i,j},t')]. \\ \mathsf{false} & otherwise. \end{cases}$$

The communication channels don't usually have memory, thus we consider any message sent while the $\mathsf{RCD}()$ predicate is false as dropped.

Now that we are able to express constraints on each edge through the $\mathsf{RCD}()$ predicate, we need to define those that enable liveness of reliable broadcast. Let us define the *k-acceptance function*, that encapsulates temporal aspects for the three acceptance conditions of CPA.

Definition 3 *Let $p_s \in \Pi$ be a process that starts a reliable broadcast at time t_{br}. The* k-acceptance function $\mathcal{A}_k(p,t)$ *over the time $t \in \mathbb{N}$ is defined as follows:*

$$\mathcal{A}_k(p_j,t) = \begin{cases} 1 & \text{if } p_j = p_s \text{ with } t \geq t_{br} \quad \textit{(AK1)} \\ 1 & \text{if } \exists\, t' \geq t_{br} \ :\ \mathsf{RCD}(p_s,p_j,t') = \mathsf{true} \text{ with } t \geq t' + \zeta(e_{s,j},t') \quad \textit{(AK2)} \\ 1 & \text{if } \exists\, p_1,\ldots,p_k \ :\ \forall i \in [1,k],\ \mathcal{A}_k(p_i,t_i)=1 \text{ and } \exists\, t'_i \geq t_i \ :\ \mathsf{RCD}(p_j,p_i,t'_i) = \mathsf{true} \text{ with } t \geq t'_i + \zeta(e_{i,j},t'_i) \quad \textit{(AK3)} \\ 0 & otherwise \end{cases}$$

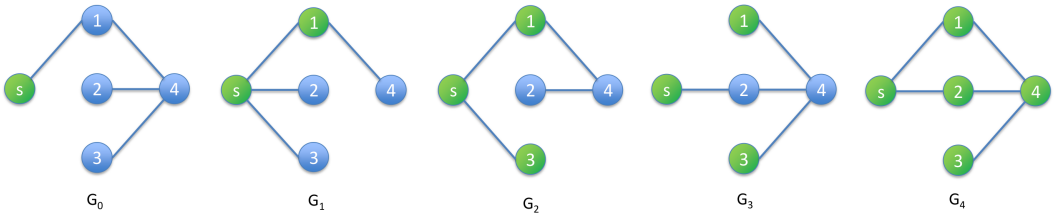


Figure 4: TVG example

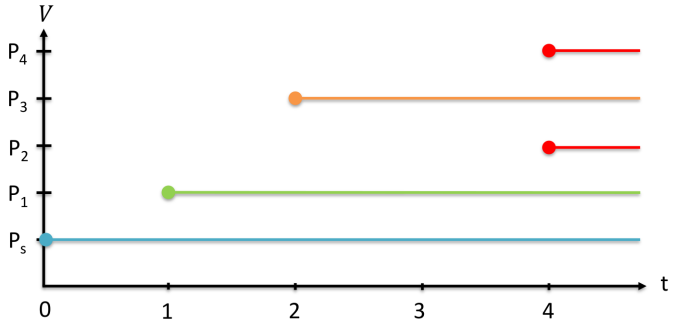


Figure 5: $\mathcal{A}_{k=2}(p_s, t)$ of Figure 4

Definition 4 *Let $\mathcal{G} = (V, E, \rho, \zeta)$ be a TVG, and let p_s be a node called* source. *A* temporal minimum k-level ordering *of $\mathcal{G}$ (TMKLO) is a partition the nodes in levels L_i defined as follows:*

$$p \in L_{t_i} \text{ iff } t_i = \min t \in \mathbb{N} \text{ such that } \mathcal{A}_k(p, t_i) = 1$$

Let us denote as P_k the partition identifying the temporal minimum k-level ordering.

As an example, let us consider the TVG presented in Figure 4: it evolves in five discrete time instants (*i.e.*, $t_0, t_1, \ldots, t_4$), its latency function $\zeta(e, t)$ is equal to 1 for every edge e at any time t. Now, let us consider process p_s as a source node that broadcasts m at time $t_{br} = 0$, and let us assume that $k = 2$. Such a TVG admits a temporal minimum 2-level ordering $P_2 = \{L_{t_0} = \{p_s\}, L_{t_1} = \{p_1\}, L_{t_2} = \{p_3\}, L_{t_4} = \{p_2, p_4\}\}$. Indeed:

- The 2-acceptance function $\mathcal{A}_2(p_s, t)$ is equal to 1 for $t \geq t_{br} = t_0$ according to *AK1*.
- The acceptance function evaluated on process p_1 is equal to 1 for $t \geq 1$ according to *AK2* (*i.e.*, $t' = 0$ and $RCD(p_s, p_1, 0) = true$ due to the presence function $\rho(< p_s, p_1 >, \tau) = 1,\ \forall \tau \in [0, 1]$).
- On processes p_3 and p_2, the acceptance function evaluates to 1 respectively for $t \geq 2$ and for $t \geq 4$, for the same reasons as p_1.
- The acceptance function on p_4 evaluates to 1 for $t \geq 4$ according to *AK3* (*i.e.*, $\mathsf{RCD}(p_i, p_4, t'_i) = \mathsf{true}$ for $p_i = p_1$, $t'_i = 1$, and for $p_i = p_3$, $t'_i = 3$).

A graphical representation of the acceptance function for $p_s, p_1, p_2.p_3, p_4$ is presented in Figure 5.

We now present a sufficient condition (Theorem 2) and a necessary condition (Theorem 3) for the liveness of reliable broadcast based on the TMKLO.

Theorem 2 (DCPA liveness sufficient condition) *Let $\mathcal{G} = (V, E, \rho, \zeta)$ be a TVG, let p_s be the* source *which broadcasts m at time t_{br}, and let us assume f-locally bounded Byzantine failures. If there exists a partition $P_k = \{L_{t_{br}}, L_{t_1} \dots L_{t_x}\}$ of nodes in V representing a TMKLO of $\mathcal{G}$ associated to m with $k > 2f$ then the message m spread using DCPA is eventually delivered by every correct process in $\mathcal{G}$.*

Proof We need to prove that if there exist a TMKLO with $k > 2f$ associated to message m, then any correct process eventually satisfies one of the CPA acceptance policies. A *TMKLO* with $k > 2f$ implies that there exist a time instant t such that the $2f + 1$-acceptance function $\mathcal{A}_k(p, t)$ is equal to 1 for every node of the network.

The process p_s belongs to any TMKLO due to AK1: as the source of the broadcast, p_s delivers the message according to AC1. Remind that the correct processes running DCPA spread the delivered messages over their neighborhood infinitively often. Then, the other nodes belong to the TMKLO due to the occurrence of AK2 or AK3.

If $AK2$ is satisfied by a node p_j from time t_j, then m: *(i)* can be delivered by the channel interconnecting p_s with p_j by definition of RCD, and *(ii)* it is transmitted by p_s, because t_j is greater than t_{br}. It follows that p_j delivers m according to $AC2$: indeed, p_j has received m directly from the source.

If $AK3$ is satisfied on a node p_j, it is possible to identify two scenarii:

- **Case 1**: $\mathsf{RCD}()$ is satisfied between p_j and $2f + 1$ nodes p_i where $AK2$ is already satisfied. We have shown that the processes satisfying $AK2$ accept m, and so they retransmit m. Assuming the f-locally bounded failure model, at most f nodes among the neighbors of p_i can be Byzantine and may not propagate m. Thus, p_j receives at least $f + 1$ copies of m from distinct neighbors. According to $AC3$ of DCPA p_j delivers m.
- **Case 2**: $\mathsf{RCD}()$ is satisfied between p_j and $2f + 1$ nodes p_i where $AK2$ or $AK3$ is already satisfied. Inductively, as the last considered nodes deliver m, it follows that the nodes p_j satisfying $AK3$ due to at least $2f + 1$ nodes p_i where $AK2$ or $AK3$ already holds also deliver m.

$\square_{Theorem\ 2}$

Theorem 3 (DCPA liveness necessary condition) *Let $\mathcal{G} = (V, E, \rho, \zeta)$ be a TVG, let p_s be the* source *that starts to broadcast m at time t_{br}, and let us assume f-locally bounded Byzantine failures. The message m can be delivered by every correct process in $\mathcal{G}$ only if a partition $P_k = \{L_{t_{br}}, L_{t_1} \dots L_{t_x}\}$ of nodes in V representing a TMKLO of $\mathcal{G}$ associated to m with $k > f$ exists.*

Proof Let us assume for the purpose of contradiction that: *(i)* every correct process in $\mathcal{G}$ delivers m, *(ii)* the Byzantine failures are f-locally bounded, and *(iii)* there does not exist a TMKLO associated to m with $k > f$. The latter implies that the TMKLO with $k = f + 1$ does not include all the nodes, *i.e.* $\exists p \in \Pi \mid \forall t \in \mathbb{N}, \mathcal{A}_{f+1}(p, t) = 0$.

The process p_s is always included in a *TMKLO* of any k. Thus, p_s is included in P_{f+1}. The nodes that deliver m according to $AC2$ have received m from p_s. Thus, the RCD predicate evaluated between p_s and p_i was true at least once after the delivery of m by p_s. It follows that the condition defined in $AK2$ is satisfied, and that nodes are included in P_{f+1}.

The remaining nodes that deliver according to $AC3$ have received the message from $f+1$ distinct neighbors. Let us initially assume that such neighbors have delivered the message by $AC2$. Again, the RCD predicate evaluated between the receiving node p_j and the distinct $f+1$ neighbors p_i has been true at least once after the respective deliveries of m. We already proved that such neighbors of p_i are included in P_{f+1}, therefore the condition defined in $AK2$ is satisfied by those p_j and they are included in P_{f+1}.

It naturally follows that the remaining nodes (the ones that have received the message from neighbors satisfying $AC2$ or $AC3$) are included in P_{f+1}. This is in contradiction with the assumptions because eventually every process satisfies one of the conditions AK1,AK2 or AK3, and the claim follows. $\square_{Theorem\ 3}$

5 On the Detection of DCPA Liveness

In Section 4, we proved that DCPA always ensure reliable broadcast safety and we provided the necessary and sufficient conditions about the dynamic network to ensure reliable broadcast liveness. In this section, we are investigating the ability of individual processes to detect whether the reliable broadcast liveness is actually achieved in the current network. In more details, we seek answers to the following questions:

- **(Conscious Termination):** Given a message m_s sent by a source p_s on TVG $\mathcal{G}$, is p_s able to detect if m_s is eventually delivered by every correct process?
- **(Bounded Broadcast Latency):** Given a message m sent by a source p_s on TVG $\mathcal{G}$, is p_s able to compute upper and lower bounds for reliable broadcast completion?

Obviously, if p_s has no knowledge about $\mathcal{G}$, nothing about termination can be detected. As a consequence, some knowledge about $\mathcal{G}$ is required to enable Conscious Termination and Bounded Broadcast Latency. We now formalize the notion of *Broadcast Latency*, and introduce oracles that abstract the knowledge a process may have about $\mathcal{G}$.

Definition 5 (Broadcast Latency (BL)) *Let $\mathcal{G} = (V, E, \rho, \zeta)$ be a TVG and let p_s be a node called* source *that broadcasts a message m at time t_{br}. We define as* Broadcast Latency BL *the period between t_{br} and the time of the last delivery of m by a correct process.*

We define the following knowledge oracles (from more powerful to least powerful):

- ***Full knowledge Oracle (FKO)***: FKO provides full knowledge about the TVG, *i.e.*, it provides $\mathcal{G} = (V, E, \rho, \zeta)$;
- ***Partial knowledge Oracle (PKO)***: given a TVG $\mathcal{G} = (V, E, \rho, \zeta)$, PKO provides the underlying static graph $G = (V, E)$ of $\mathcal{G}$;
- ***Size knowledge Oracle (SKO)***: given a TVG $\mathcal{G} = (V, E, \rho, \zeta)$, SKO provides the size of $\mathcal{G}$, that is $|V|$.

5.1 Detecting DCPA Liveness on Generic TVGs

In Section 4 we showed that the conditions guaranteeing the liveness property of reliable broadcast are strictly bounded to the network evolution. It follows that the knowledge provided by a FKO, in particular about the network evolution from the broadcast time t_{br}, is necessary to argue on liveness, unless further assumptions are taken into account. In the following, we clarify how a process can employs a FKO to detect Conscious Termination and Bounded Broadcast Latency.

Lemma 2 *Let $\mathcal{G} = (V, E, \rho, \zeta)$ be a TVG, let p_s be a node called* source *that broadcasts a message m at time t_{br} and let us assume f-locally bounded Byzantine failures. If p_s has access to a FKO then it is able to verify if there exists a TMKLO for the current broadcast on $\mathcal{G}$.*

Proof In order to prove the claim it is enough to show an algorithm that verifies if a TMKLO exists, given the full knowledge of the TVG provided by FKO.
Such algorithm works as follow: initially, the source p_s is placed in level t_{br} of the TMKLO. Then, the snapshots characterizing the TVG have to be analyzed, starting from $G_{t_{br}}$ and following their order. In particular, for each snapshot G_{t_i}, $t_i \geq t_{br}$, we need to verify that:

1. edges with only one endpoint already included in some level of the TMKLO are up enough to satisfy $\mathsf{RCD}()$ and
2. whenever $\mathsf{RCD}()$ is satisfied for a given edge $e_{i,j}$, we need to check if it allows p_j to be part of the TMKLO as it satisfies one condition among AK2 and AK3.

The algorithm ends when a TMKLO is found or when all the snapshots have been analyzed (and in the latter case we can infer that no TMKLO exists for the considered message on the given TVG). Assuming that $\mathcal{G}$ spans over T time instants, the complexity of this algorithm is:

$$O(|T||E) + O(|V| + |E|) = O(|V| + |T||E|)$$

A more detailed description of the algorithm is delegated to the Appendix A. $\square_{Lemma\ 2}$

Theorem 4 *Let $\mathcal{G} = (V, E, \rho, \zeta)$ be a TVG, let p_s be a node called* source *that broadcasts a message m at time t_{br} and let us assume f-locally bounded Byzantine failures. If p_s has access to a FKO then it is able to detect if eventually every correct process will deliver m.*

Proof The claim follows by considering that in order to assess the Conscious Termination of DCPA, the source process p_s (and in general any process in the system) needs to compute a TMKLO (i.e., it needs to check that eventually each correct process will be placed in a level) and due to Lemma 2 this can be done by accessing FKO. In particular, to detect Conscious Termination, a process p_i can first verify if the necessary condition holds and this can be done by computing a TMKLO with $k \geq f + 1$. If not, p_i can simply infer that m will not be delivered by every correct process. Contrarily, it can verify if the sufficient condition holds computing a TMKLO with $k \geq 2f + 1$. If it exists, p_i can infer that eventually every correct process will deliver the message otherwise, it needs to verify the necessary condition in every subgraph obtained by $\mathcal{G}$ removing all the possible disposition of Byzantine processes (remind that getting this answer corresponds to solve an NP-Complete problem even considering a static networks, thus same intractability follows also on dynamic networks) . If the necessary condition is always satisfied, it can infer Conscious Termination otherwise not. $\square_{Theorem\ 4}$

Let us note that if a process has the capability of computing the *TMKLO* for a message m sent at time t_{br}, then it can also establish a lower bound and an upper bound on the time needed by every correct process to deliver m simply evaluating the maximum level of the TMKLO that satisfy respectively the necessary and the sufficient condition for DCPA.

Theorem 5 *Let $\mathcal{G} = (V, E, \rho, \zeta)$ be a TVG and let p_s be a node called* source *that broadcasts a message m at time t_{br} and let us assume f-locally bounded Byzantine failures. Let $P_{f+1} = \{L_{t_0}, L_{t_1} \dots L_{t_x}\}$ be the*

TMKLO with $k = f + 1$ associated to m and let t_{max}^{f+1} be the time associated to the last level of P_{f+1}. Let assume the existence of the TMKLO with $k = 2f + 1$ associated to m, $P_{2f+1} = \{L_{t_0}, L_{t_1} \dots L_{t_x}\}$, and let t_{max}^{2f+1} be the time associated to the last level of P_{2f+1}.

The computed TMKLOs provide respectively a lower bound and an upper bound for BL such that:

$$t_{max}^{f+1} - t_{br} \leq BL \leq t_{max}^{2f+1} - t_{br}$$

Proof *Lower Bound:* Let us assume by contradiction that BL can be lower than $t_{max}^{f+1} - t_{br}$. It follows that the last process p_i delivering m, does it at a time $t_i < t_{max}^{f+1}$. Given the definition of TMKLO with $k = f + 1$, a level L_x is created each time that a process not yet inserted in the TMKLO delivers a message (due to AK2 or AK3). As a consequence, the last level of the TMKLO is created when the last process delivers the message. Thus, considering that p_i is the last process delivering the message, it follows that t_i is the time associated to the last level. Given P_{f+1}, it follows that $t_i = t_{max}^{f+1}$ and we have a contradiction.

Upper Bound: Let us assume by contradiction that BL can be greater than $t_{max}^{2f+1} - t_{br}$. It follows that the last process p_i delivering m, does it at a time $t_i > t_{max}^{2f+1}$. Given the definition of TMKLO with $k = 2f + 1$, a level L_x is created each time that a process not yet inserted in the TMKLO delivers a message (due to AK2 or AK3). As a consequence, the last level of the TMKLO is created when the last process delivers the message. Thus, considering that p_i is the last process delivering the message, it follows that t_i is the time associated to the last level. Given P_{2f+1}, it follows that $t_i = t_{max}^{2f+1}$ and we have a contradiction. $\square_{Theorem\ 5}$

Remind that, as the sufficient condition we provided is not strict, a TMKLO with $k = 2f + 1$ could not exist even if the reliable broadcast is possible. It is also possible to provide a stricter upper bound for BL as we explained in Theorem 4, but is not practical to compute. Finally, let us remark that the knowledge on the underlying topology is not enough on dynamic networks to argue on liveness.

Remark 1 *Let $\mathcal{G} = (V, E, \rho, \zeta)$ be a TVG and let p_s be a node called* source *that broadcasts a message m at time t_{br} and let us assume f-locally bounded Byzantine failures. If a process p_s has access only to a PKO (and not to an FKO) then it is not able to detect either Conscious Termination and Bounded Broadcast Latency. Indeed, as we highlighted in section 4.2, moving on dynamic network the knowledge on the underlying graph is not enough, because specific sequences of edge appearances are required in order to guarantee the message propagation (let us take again Figure 3 as clarifying example). Thus, PKO is not enough in arguing on liveness. The same can be said about Bounded Broadcast Latency as PKO provides no information about the time instants where the edge will appear.*

5.2 Detecting DCPA Liveness on Restricted TVGs

Casteigts *et al.* [7] defined a hierarchy of TVG classes based on the strength of the assumptions made on appearance of edges. So far, we considered the most general TVG[3]. In the following, we consider two more specific classes of the hierarchy where we show that liveness can be detected using oracles weaker than FKO. In particular, we consider the following classes that are suited to model recurring networks:

- ***Class recurrence of edges, ER*** : if an edge e appears once, it appears infinitively often[4].
- ***Class time bounded recurrences, TBER***: if an edge e appears once, it appears infinitively often and there exist an upper bound Δ between two consecutive appearances of e[5].

[3]Class 1 TVG according to Casteigts *et al.* [7]

[4]Class 6 TVG in Casteigts *et al.* [7].

[5]Class 7 TVG in Casteigts *et al.* [7].

Let us recall that assuming predicate $\mathsf{RCD}(e_{i,j}, t) = \mathsf{true}$ for every edge $e_{i,j}$ at some time t is necessary to guarantee liveness. While considering classes ER and TBER, such condition must be satisfied infinitely often, otherwise it is easy to show that the results presented in the previous section still apply. Let us also note that the conditions we defined in Section 4.2 are related to a single broadcast generated by a specific source p_s i.e., for a source p_s broadcasting a message at time t_{br} the conditions must hold from t_{br} on. Contrarily, exploiting the recurrence of edges it is possible to define different conditions that are valid for every broadcast from the same source p_s, independently from when it starts.

5.2.1 Detecting DCPA Liveness in ER TVG

In this section, we prove that considering TVG of class ER, we can get the following results: *(i)* PKO (an oracle weaker than FKO) is enough to enable Conscious Termination, *(ii)* despite the more specific TVG considered, FKO is still required to establish upper bounds for *BL*. Intuitively, this results follows from the fact that PKO allows to determine whether a MKLO exists on the static underlying graph, and this is enough to detect if eventually, every correct process is able to deliver the message. However, given the absence of information on when each edge is going to appear, it is impossible to compute an upper bound on the time required to deliver a message.

Lemma 3 *Let $\mathcal{G} = (V, E, \rho, \zeta)$ be a TVG of class ER, let p_s be a node called* source *that broadcasts a message m at time t_{br} and let us assume f-locally bounded Byzantine failures. Let $G = (V, E)$ be the underlying graph of $\mathcal{G}$. If p_s has access to a PKO then it can compute a MKLO on G.*

Proof The PKO provides knowledge on the topology of G. We reminded in Definition 1 that the MKLO is a partition of the nodes on the base of a topological conditions. It follows that it is possible to verify the MKLO on G with PKO through a modified breath-first search [17]. $\square_{Lemma\ 3}$

Lemma 4 *Let $\mathcal{G} = (V, E, \rho, \zeta)$ be a TVG of class* ER *that ensures* $\mathsf{RCD}()$ *infinitively often, let $G = (V, E)$ be the static underlying graph of $\mathcal{G}$, let p_s be a node called* source *and let us assume f-locally bounded Byzantine failures. If there exists the* MKLO *of $G = (V, E)$ associated to p_s then there always exists the* TMKLO *of $\mathcal{G}$ associated to a message m sent by p_s with the same k.*

Proof We prove the claim showing a mapping from MKLO to TMKLO. The source is placed inside the *TMKLO* at level t_{br}. Then, given the assumption on the channels and that every node in the MKLO has either (i) an edge connecting it with the source (ii) and/or k neighbors already included in MKLO, it follows that every node eventually satisfies at least one between AK2 and AK3. $\square_{Lemma\ 4}$

Theorem 6 *Let $\mathcal{G} = (V, E, \rho, \zeta)$ be a TVG of class ER that ensures* $\mathsf{RCD}()$ *infinitively often, and let p_s be a node called* source *that broadcasts m at time t_{br}, and let us assume f-locally bounded Byzantine failures. If p_s has access to a PKO, then it is able to detect if eventually every correct process delivers m.*

Proof It follows from Lemma 3 and Lemma 4 $\square_{Theorem\ 6}$

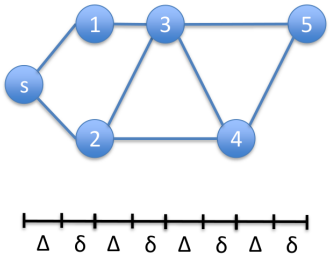


Figure 6: Worst case topology with respect BL.

5.2.2 Detecting DCPA Liveness in TBER TVG

The liveness condition enabling CPA to enforce reliable broadcast relays on network topology, therefore an oracle weaker that FKO cannot enable Conscious Termination unless further assumptions are made. On the other hand, the weaker oracle SKO allows a process to compute Bounded Broadcast Latency.

Lemma 5 *Let $\mathcal{G} = (V, E, \rho, \zeta)$ be a Class* TBER *TVG where each edge $e_{i,j}$ reappears in at most Δ time instants satisfying* $\mathsf{RCD}(e_{i,j}, t)$*. Let $\delta_{max} = max(\zeta(e, t))$. Let p_s be the source and let us assume f-locally bounded Byzantine failures. The Broadcast Latency BL is upper bounded by*

$$BL \leq |V|(\delta_{max} + \Delta)$$

Proof Given the assumptions on the TVG we know that every edge reappears in Δ and satisfies $\mathsf{RCD}()$. The worst case scenario, with respect the message propagation, is the one in which every node has to wait Δ to forward a message. The worst case scenario, with respect the network topology, is the one where every process has to wait the last one which has delivered to deliver (in other words, the partitions of the MKLO evaluated over the underlying graph $G(V, E)$, with the exception of the second level, have size equals to 1). We depicted such a worst case scenario in Figure 6. $\square_{Lemma\ 5}$

Lemma 6 *Let $\mathcal{G} = (V, E, \rho, \zeta)$ be a Class* TBER *TVG where each edge $e_{i,j}$ reappears in at most Δ time instants satisfying* $\mathsf{RCD}(e_{i,j}, t)$*. Let $\delta_{max} = max(\zeta(e, t))$. Let p_s be the source and let us assume f-locally bounded Byzantine failures. Let $P_{2f+1} = \{L_{t_0}, L_{t_1} \dots L_{t_x}\}$ be the MKLO with $k = 2f + 1$ computed on the underlying graph $G = (V, E)$ (if exists) and let S_{2f+1} be size of P_{2f+1}.*

An upper bound for BL can be computed from the MKLO with $k = 2f + 1$. In particular:

$$BL \leq S_{2f+1}(\delta_{max} + \Delta)$$

Proof Given the assumptions on the TVG $\mathcal{G}$ we know that every edge reappears in Δ and guarantees $\mathsf{RCD}()$. The worst case scenario with respect the message propagation is the one where every node as to wait Δ to forward a message.

The bound follows by Theorem 5 and Lemma 4, noting that every node in level L_i delivers in $(\delta_{max} + \Delta)i$ time instants. $\square_{Lemma\ 6}$

Theorem 7 *Let $\mathcal{G} = (V, E, \rho, \zeta)$ be a TVG of class TBER and let p_s be a node called* source *that broadcasts m at time t_{br}, and let us assume f-locally bounded Byzantine failures. If p_s uses SKO or PKO, then p_s is able to compute an upper bound for BL. Specifically:*

$$BL \leq |V|(\delta_{max} + \Delta)) \textit{using SKO}$$

$$BL \leq S_{2f+1}(\delta_{max} + \Delta) \textit{using PKO}$$

Proof The claim follows from Lemmas 6 and 5. $\square_{Theorem\ 7}$

6 Moving to an Asynchronous System

In this work we assumed a synchronous distributed systems. In this section, we briefly discuss consequences of asynchrony on the safety and liveness of DCPA.

In Section 4.1, we showed that a reliable and authenticated channel is necessary and sufficient to enforce safety trough CPA in an *f-locally bounded* failure model. Such channel properties are independent from the latency function as they just require that if a message m sent by a correct process is eventually received at its destination, it has not been compromised by the channel. As a consequence CPA (and DCPA as well) continues to enforce safety also on asynchronous dynamic networks.

In Section 4.2, we pointed out the need of having channels up long enough to allow the delivery of messages. This imposes constraints on the presence function that depend on the latency function. The asynchrony affects the latency function $\zeta(e, t)$ that basically is no more bounded. This makes impossible (in asynchronous system) to establish constraints for the liveness due to the fact it is no longer guaranteed the propagation of messages. It follows that we cannot argue on liveness of reliable broadcast on general TVG without making further assumptions.

In Section 5.2 we investigated about liveness in specialised classes of TVG. In particular, we showed in Theorem 6 that assuming recurrent *RCD* and having the knowledge on the underlying static graph it is possible to investigate about. It follows that, although RCDs are not identifiable over the time, if they are satisfied infinitively often, they enable the verification of liveness also in asynchronous systems.

7 Conclusion

We considered the reliable broadcast problem in dynamic networks represented by TVG. We analyzed the porting conditions enabling CPA to be correctly employed on dynamic network. The analysis of this simple algorithm is important as it works exploiting only local knowledge. This contrasts to the best result so far in the same setting [21], that demands an exponential costs to check when a message can be delivered. Moreover, we presented necessary and sufficient conditions to ensure safety and liveness DCPA. We analyzed how much knowledge of the TVG is needed to detect whether the liveness condition is satisfied, and its cost. Our work is a starting point to identify more general parameters of dynamic networks that guarantees the fulfilment of the conditions we provided, both in a deterministic and probabilistic way. Another interesting point to address in future work is the possibility to define a f-locally bounded failure model on a time dimension, considering the failures to be *f-locally bounded* in a temporal interval.

References

[1] Yehuda Afek, Baruch Awerbuch, and Eli Gafni. Applying static network protocols to dynamic networks. In *Foundations of Computer Science, 1987., 28th Annual Symposium on*, pages 358–370. IEEE, 1987.

[2] John Augustine, Gopal Pandurangan, and Peter Robinson. Fast byzantine agreement in dynamic networks. In Panagiota Fatourou and Gadi Taubenfeld, editors, *ACM Symposium on Principles of Distributed Computing, PODC '13, Montreal, QC, Canada, July 22-24, 2013*, pages 74–83. ACM, 2013.

[3] Baruch Awerbuch. On the effects of feedback in dynamic network protocols. *Journal of Algorithms*, 11(3):342–373, 1990.

[4] Baruch Awerbuch and Shimon Even. Efficient and reliable broadcast is achievable in an eventually connected network. In *Proceedings of the third annual ACM symposium on Principles of distributed computing*, pages 278–281. ACM, 1984.

[5] Baruch Awerbuch, Yishay Mansour, and Nir Shavit. Polynomial end-to-end communication. In *Foundations of Computer Science, 1989., 30th Annual Symposium on*, pages 358–363. IEEE, 1989.

[6] Baruch Awerbuch and Michael Sipser. Dynamic networks are as fast as static networks. In *Foundations of Computer Science, 1988., 29th Annual Symposium on*, pages 206–219. IEEE, 1988.

[7] Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012.

[8] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186, 1999.

[9] Danny Dolev. Unanimity in an unknown and unreliable environment. In *Foundations of Computer Science, 1981. SFCS'81. 22nd Annual Symposium on*, pages 159–168. IEEE, 1981.

[10] Danny Dolev. The byzantine generals strike again. *Journal of algorithms*, 3(1):14–30, 1982.

[11] John R Douceur. The sybil attack. In *International workshop on peer-to-peer systems*, pages 251–260. Springer, 2002.

[12] Vadim Drabkin, Roy Friedman, and Marc Segal. Efficient byzantine broadcast in wireless ad-hoc networks. In *Dependable Systems and Networks, 2005. DSN 2005. Proceedings. International Conference on*, pages 160–169. IEEE, 2005.

[13] Rachid Guerraoui, Florian Huc, and Anne-Marie Kermarrec. Highly dynamic distributed computing with byzantine failures. In Panagiota Fatourou and Gadi Taubenfeld, editors, *ACM Symposium on Principles of Distributed Computing, PODC '13, Montreal, QC, Canada, July 22-24, 2013*, pages 176–183. ACM, 2013.

[14] Akira Ichimura and Maiko Shigeno. A new parameter for a broadcast algorithm with locally bounded byzantine faults. *Information processing letters*, 110(12-13):514–517, 2010.

[15] 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. ACM, 2004.

[16] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

[17] Chris Litsas, Aris Pagourtzis, and Dimitris Sakavalas. A graph parameter that matches the resilience of the certified propagation algorithm. In *International Conference on Ad-Hoc Networks and Wireless*, pages 269–280. Springer, 2013.

[18] Alexandre Maurer and Sébastien Tixeuil. Byzantine broadcast with fixed disjoint paths. *J. Parallel Distrib. Comput.*, 74(11):3153–3160, 2014.

[19] Alexandre Maurer and Sébastien Tixeuil. Containing byzantine failures with control zones. *IEEE Trans. Parallel Distrib. Syst.*, 26(2):362–370, 2015.

[20] Alexandre Maurer and Sébastien Tixeuil. Tolerating random byzantine failures in an unbounded network. *Parallel Processing Letters*, 26(1), 2016.

[21] Alexandre Maurer, Sébastien Tixeuil, and Xavier Defago. Communicating reliably in multihop dynamic networks despite byzantine failures. In *Reliable Distributed Systems (SRDS), 2015 IEEE 34th Symposium on*, pages 238–245. IEEE, 2015.

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

[23] Kai Zeng, Kannan Govindan, and Prasant Mohapatra. Non-cryptographic authentication and identification in wireless networks [security and privacy in emerging wireless networks]. *IEEE Wireless Communications*, 17(5), 2010.

Appendix A An algorithm for verifying if a TMKLO exists

We present an algorithm enabling a process equipped with a FKO to compute TMKLO, if it exist.
The complete pseudo-code is provided in Figure 7. It is composed by:

- the main part, check_liveness_and_BL_bounds, where the liveness condition is verified and bounds for BL are computed,
- the code that computes a *TMKLO*, compute_temporal_minimum_level_ordering.

Going into details of the main part, it leverages on the TMKLO to verify the necessary condition (Theorem 3) on lines $2-3$ and the sufficient condition (Theorem 2) on lines $5-6$. It is clear that there exist cases where the necessary condition is verified while the sufficient one is not and, in order to argue on Conscious Termination and Bounded Broadcast Latency, we need to verify the necessary condition considering any possible Byzantine placement over the network (lines $12-18$) as in [17].

Moving on the part related to the *TMKLO* computation, the corresponding function analyzes the snapshots characterizing the TVG, starting from $G_{t_{br+1}}$ and following their order, verifying : (i) if the edges with only one endpoint already included in some level of the TMKLO are up enough to satisfy $\mathsf{RCD}()$ and (ii) whenever $\mathsf{RCD}()$ is satisfied for a given edge $e_{i,j}$, we need to check if it allows p_j to be part of the TMKLO as it satisfies one condition among AK2 and AK3. In the following we provide a summary of variables used in the algorithm:

- *t_level_ordering* is a map $\langle$process ID, acceptance time instant$\rangle$, it corresponds to the *TMKLO*;
- *transmitting_edges* is the set of edges which are under analysis for $\mathsf{RCD}()$;
- *onGoing_RCDs* is a map $\langle$edge, i$\rangle$ which keeps track of the duration of ongoing $\mathsf{RCD}()$;
- *latencies* is a map $\langle$edge, $\delta\rangle$ which keeps track of latencies of the channels;
- *succeded_RCDs* is the set of edges which have succeeded in $\mathsf{RCD}()$.

The algorithm takes as input:

- the TVG description;
- the ID of the source node;
- t_{br}, namely the the time instant when the broadcast starts;
- the value of k for *TMKLO*.

It computes the k-*TMKLO* or it returns an incomplete partition. It ends when all processes are included in the *TMKLO* (lines 30), or all network snapshots are analyzed and returns an incomplete *TMKLO* (line 32).
In particular, the k-acceptance function fixes the level of the source (AK1, line 21). Subsequently, the TVG is analyzed through its consecutive shapshots (line 22). Given a time instant t, the edges under analysis are the ones that interconnect processes who are already in the *TMKLO* with others not included (Lines $48-52$). If an edge does not remain up for enough consecutive time instants depending on the latency function $\zeta(e,t)$, it is not possible for such an edge to ensure $\mathsf{RCD}()$. This means that all ongoing transmissions (edges), registered inside the variable *onGoing_RCDs*, have to be compared with the ones included in *transmitting_edges*. In lines $45-47$ all ongoing $\mathsf{RCD}()$ over edges that disappear are dropped. Subsequently, the state of ongoing $\mathsf{RCD}()$ is updated considering the transmitting edges (line 26). When one among *AK2* or *AK3* is verified, the specific nodes are inserted inside the *TMKLO* (lines $39-44$).

```
check_liveness_and_BL_bounds(source, t_br, f):
(01) graph ← FKO;
(02) TMKLO_nec ← compute_temporal_minimum_level_ordering(graph, source, t_br, f + 1);
(03) if TMKLO_nec.is_complete(graph.size) then
(04)    BL_min ← TMKLO_nec.max_value() − t_br;
(05)    TMKLO_suf ← compute_temporal_minimum_level_ordering(graph, source, t_br, 2f + 1);
(06)    if TMKLO_suf.is_complete(graph.size) then
(07)       liveness ← 1;
(08)       BL_max ← TMKLO_suff.max_value() − t_br;
(09)    else
(10)       BL_max ← 0;
(11)       liveness ← 1;
(12)       for each (subgraph ∈ get_subgraphs_f_local(graph, source, f)) do
(13)             TMKLO_i ← compute_temporal_minimum_level_ordering(subgraph, source, t_br, f + 1);
(14)             if TMKLO_i.is_complete(subgraph.size) then
(15)                BL_max ← max(BL_max, TMKLO_i.max_value());
(16)             else
(17)                liveness ← 0;
(18)                break
(19) else
(20)    liveness ← 0;
```

```
compute_temporal_minimum_level_ordering(graph, source, t_br, k):
(21) t_level_ordering ← t_level_ordering ∪ {< source, t_br >};
(22) for (t ← t_br; t <= graph.max_T; t ← t + 1) do
(23)    transmitting_edges ← get_transmitting_edges(t);
(24)    check_ongoing_RCDs();
(25)    for each (e ∈ transmitting_edges) do
(26)          onGoing_RCDs ← onGoing_RCDs ∪ {< e, +1 >};
(27)          if e ∉ latencies then
(28)             latencies ← latencies ∪ {< e, graph.ζ(e, t) >};
(29)          check_RCD(e, t);
(30)    if t_level_ordering.keys = graph.nodes then
(31)       return t_level_ordering
(32) return t_level_ordering
```

```
check_RCD(e, t):
(33) edge_duration ← onGoing_RCDs.get(e)
(34) δ ← latencies.get(e)
(35) if edge_duration = δ then
(36)    out_node ← not_included(e)
(37)    succeeded_RCDs ← succeeded_RCDs ∪ {e}
(38)    onGoing_RCDs ← onGoing_RCDs \ {e}
(39)    if source ∈ e.endpoints then
(40)       t_level_ordering ← t_level_ordering ∪ {< out_node, t >}
(41)    else
(42)       k_acceptance ← k_acceptance{< out_node, +1 >}
(43)       if k_acceptance.getout_node) = k then
(44)          t_level_ordering ← t_level_ordering ∪ {< out_node, t >}
```

```
check_ongoing_RCDs():
(45) interrupted_RCDs ← onGoing_RCDs \ transmitting_edges
(46) onGoing_RCDs ← onGoing_RCDs \ interrupted_channels
(47) latencies ← latencies \ interrupted_channels
```

```
get_transmitting_edges(t):
(48) new_transmitting_edges ← ∅
(49)    for each (e ∈ graph.getSnapshot(t).edges) do
(50)          if |e.endpoints ∩ t_level_ordering.keys| = 1 AND e ∉ succeded_RCDs then
(51)             new_transmitting_edges ← new_transmitting_edges ∪ {e}
(52) return new_transmitting_edges
```

```
not_included(e):
(53) for each (p ∈ e.endpoints) do
(54)       if t_level_ordering.keys ∩ p ≠ ∅ then
(55)          return p
```

Figure 7: Process code with *FKO*

A.1 Algorithm complexity

We analyze the computation complexity of the presented algorithm, focusing on the single functions:

- The get_transmitting_edges(t) function analyses the edges present in a specific time instants, filtering out the ones that are not relevant for the *TMKLO*. The filtering process has constant complexity. It follows that the function has complexity $O(|E|)$.
- The check_ongoing_RCDs() performs the set difference of collection of size $O(|E|)$. Thus, this function has complexity $O(|E|)$.
- The check_RCD() function accesses an entry map and compares the returned value with δ. The if condition on line 35 is verified once for every edge and every process is inserted inside the *TMKLO* only once. It follows that the function has cost $O(|V| + |E|)$.

It follows the complexity of computing the *TMKLO* is:

$$O(|T||E) + O(|V| + |E|) = O(|V| + |T||E|)$$

We have to notice that even assuming reliable channels with instantaneous delivery (namely, $\delta = 0$ guaranteeing that every message sent over a present channel is delivered) the complexity of the algorithm does not change, because all the edges included inside the snapshot G_{t_i} have still to be analyzed.

We have shown that if the sufficient condition is verified, a process has to compute two TMKLO to get knowledge on conscious termination and bounded broadcast latency. Thus, the complexity of the entire algorithm is polynomial with respect our metrics (nodes, edges and time). In the worst case scenario, placed between the gap among the sufficient and necessary conditions, all possible Byzantine placements have to be verified. This, turn our algorithm in an exponential one as shown in [17].