A Connectivity Model for Agreement in Dynamic Systems ⋆

Carlos Gómez-Calzado[1], Arnaud Casteigts[2], Alberto Lafuente[1], and Mikel Larrea[1]

[1] University of the Basque Country UPV/EHU, Spain
{carlos.gomez, alberto.lafuente, mikel.larrea}@ehu.es
[2] LaBRI, University of Bordeaux, France
arnaud.casteigts@labri.fr

Abstract. The consensus problem is a fundamental paradigm in distributed systems, because it captures the difficulty to solve other agreement problems. Many current systems evolve with time, e.g., due to node mobility, and consensus has been little studied in these systems so far. Specifically, it is not well established how to define an appropriate set of assumptions for consensus in dynamic distributed systems. This paper studies a hierarchy of three classes of time-varying graphs, and provides a solution for each class to the problem of Terminating Reliable Broadcast (TRB). The classes introduce increasingly stronger assumptions on timeliness, so that the trade-off between weakness versus implementability and efficiency can be analysed. Being TRB equivalent to consensus in synchronous systems, the paper extends this equivalence to dynamic systems.

1 Introduction

The consensus problem is a central paradigm in distributed systems, as it represents many agreement problems, e.g., leader election, atomic commitment and total-order broadcast. Solving consensus has attracted a lot of attention in dependable computing and has generated fundamental results. In this regard, it is known that in crash-prone asynchronous distributed systems it is impossible to solve consensus deterministically due to the impossibility of distinguishing between "slow" processes and crashed ones, a result known as FLP impossibility [8]. Alternatively, consensus can be easily solved in synchronous systems, where perfect failure detection can be implemented [5].

Most of the research on consensus has considered a static distributed system with permanent connectivity among nodes. In many current distributed systems, however, these assumptions are not valid any more. Instead, these new systems exhibit a dynamic behavior, with nodes joining the system, leaving it or just moving, which implies uncertain connectivity conditions. Indeed, and unlike in classical static systems, these events are no longer considered incorrect or sporadic behaviors, but rather the natural dynamics of the system.

Clearly, even the synchrony assumptions of classical (static) models of distributed systems are not enough to solve agreement problems in dynamic systems. For example, having

⋆ Research supported by the Spanish Research Council, under grant TIN2013-41123-P, and the Basque Government, under grant IT395-10. Also, Carlos Gómez-Calzado is recipient of a doctoral fellowship from the Basque Government.

an upper bound on link latencies is pointless if the link is not available at the time of transmission of the message. Note however that the nodes could still communicate using an alternative path in the network. Thus, assumptions should consider the overall system connectivity, which encourages for a holistic approach to model dynamic distributed systems.

In recent years there was a rising interest in modeling dynamic distributed systems from the perspective of graph theory. In this regard, there exists several works that study the solvability of deterministic problems, including consensus, in highly-dynamic systems [2,3,7,10,11]. However, regarding consensus, none of them lowers the assumptions to the realm of *temporal* connectivity, i.e., not requiring that the graph be connected at every instant, but only that paths exist over time and space (temporal path, aka *journeys*). The time-varying graph formalism [4] (TVG, for short) provides a useful qualitative framework to model dynamic distributed systems. In this formalism, the dynamic network is represented as a graph, together with a presence function that tells whether a given edge is present at a given time and a latency function that tells how long it takes to cross a given edge at a given time. In [4], Casteigts et al. define a hierarchy of classes of dynamic networks, most of which are based on temporal connectivity concepts Among them, the *recurrent connectivity* class requires that a journey exists between any two nodes infinitely often (that is, recurrently). Nevertheless, this class lacks the necessary timeliness (i.e. time bounds in communication) to describe the specific assumptions that are required by synchronous agreement algorithms, such as TRB, to terminate. One of the goals of our paper is to extend some of the existing TVG classes by introducing timeliness constraints, together with practical considerations, and analyze the impact of these new constraints on solving consensus.

Our contribution

In this paper, we address timeliness in evolving systems (i.e., time-varying graphs, TVG) from a synchronous point of view, i.e., systems where the transmission delay of messages is bounded and the bound is known a priori by the processes. The resulting set of concepts and mechanisms make it possible to describe system dynamics at different levels of abstraction and with a gradual set of assumptions.

We first formulate a very abstract property on the temporal connectivity of the TVG, namely, that the temporal diameter (i.e. maximum duration of a foremost journey) of a component in the TVG is recurrently bounded by Δ. We refer to such a component as a Δ-*component*, and define the concept of correct process in terms of this component. We then specify a version of the Terminating Reliable Broadcast problem (TRB) for Δ-components, which we relate to the ability of solving agreement at component level.

Although Δ-components are proven to be a sufficient concept at the most abstract level, they rely on non implementable communication patterns in message-passing systems. Indeed, the solution to TRB proposed in this abstract model relies on an oracle that provides the algorithm with instantaneous knowledge of the appearance of an edge. Unfortunately, this oracle does not have a straightforward implementation in terms of real processes and communication links. Therefore, we introduce a first constraint to force the existence of journeys whose edges presence duration is lower-bounded by some duration β (which holds a relation to the maximal latency of a link), thereby enabling repetitive communication attempts to succeed eventually. These journeys are called β-journeys and their existence makes it possible to implement the TRB algorithm without oracle. We then look at a further constrained class of TVG, inspired by the work of Fernández-Anta et al [7], whereby the local appearance of the edge used by every next hop of (at least one of the possibly many) β-journeys also must be bounded by some duration α, yielding to the concept of (α, β)-journeys. The existence of recurrent (α, β)-journeys allows the nodes to stop sending a message α time after they receive it, which is much more efficient.

The rest of the paper is organized as follows. Section 2 introduces basic time-varying graph notations, used in Section 3 to define the abstract timely connectivity model based on Δ-components. In the same section we redefine the TRB problem with respect to Δ-components and give a solution to it. Then, in Section 4, we introduce β-journeys (and the corresponding β-components), which we show to be sufficient to implement an effective (i.e., oracle-free) version of the algorithm. We then define (α, β)-journeys and components, and discuss their advantages (and disadvantages) over β-journeys. In Section 5, we describe how consensus can be solved by using the TRB implementations introduced in Sections 3 and 4. We finally conclude in Section 6 with open questions and future work.

2 Time-Varying Graphs

A recent framework called *time-varying graphs*, proposed by Casteigts et al. [4], aims to provide a precise formalism for describing dynamic networks. As usual, the entities of the system and the communication links between them are represented as a graph. More specifically, a time-varying graph (TVG, for short) is defined as a tuple $\mathcal{G} = (V, E, \mathcal{T}, \rho, \zeta)$, where:

- V is the set of communicating entities (or nodes, or processes, interchangeably).
- E is the set of edges (or links, interchangeably) that interconnect the nodes in V. In this work, all edges are undirected.
- $\mathcal{T}$ is the *lifetime* of $\mathcal{G}$, *i.e.* the interval of time over which the graph is defined. It is a subset of the temporal domain $\mathbb{T}$, itself being $\mathbb{N}$ or $\mathbb{R}^+$ depending on whether time is discrete or continuous (in this work, it is continuous). For convenience, both endpoints of $\mathcal{T}$ are referred to as $\mathcal{T}^-$ and $\mathcal{T}^+$, the latter being possibly $+\infty$.
- $\rho : E \times \mathcal{T} \to \{true, false\}$, called the *presence* function, indicates whether a given edge is present at a given time (i.e., $\rho(e, t) = true$ if and only if edge e is present at time t)
- $\zeta : E \times \mathcal{T} \to \mathbb{T}$, called the *latency* function, indicates how long it takes to send a message across a given edge for a given emission time (assuming the edge is present at that time)

The kind of network we are addressing is possibly disconnected at every instant. Still, a form of communication can be achieved over time by means of *journeys* (*a.k.a. temporal path*). Formally, a journey $\mathcal{J} = \{((e_1, t_1), (e_2, t_2), \ldots, (e_k, t_k))\}$ is a sequence such that $(e_1, e_2, \ldots, e_k)$ is a valid path in the underlying graph (V, E), and (1) for every $i \in [1, k]$ edge e_i is present at time t_i long enough to send a message across (formally, $\rho(e_i, t_i + \delta) = true$ for all $\delta \in [0, \zeta(e_i, t_i)))$, and (2) the times when edges are crossed (we also say *activated*) and the corresponding latencies allow a sequential traversal (formally, $t_{i+1} \geq t_i + \zeta(e_i, t_i)$ for all $i \in [1, k)$). What makes this form of connectivity *temporal* is the fact that a journey can pause in between hops, e.g. if the next link is not yet available.

Given a journey $\mathcal{J}$, departure($\mathcal{J}$) and arrival($\mathcal{J}$) denote respectively its starting time t_1 and its ending time $t_k + \zeta(e_k, t_k)$. Journeys can be thought of as paths over time, having both a *topological length* k (i.e., the number of *hops*) and a *temporal length* (i.e., a duration) arrival($\mathcal{J}$)$-$departure($\mathcal{J}$) $= t_k + \zeta(e_k, t_k) - t_1$. Note that journeys describe *opportunities* of communication between an emitter and a receiver. $\mathcal{J}^*_{\mathcal{G}}$ is the set of all such opportunities over $\mathcal{G}$'s lifetime, while $\mathcal{J}^*_{(p,q)} \subseteq \mathcal{J}^*_{\mathcal{G}}$ are those journeys from p to q. A simplified way of denoting the existence of a journey between a process q and a process q, when the context of $\mathcal{G}$ is clear, is $p \rightsquigarrow q$. Finally, the graph is said to be *temporally connected* if for every $p, q \in V, p \rightsquigarrow q$.

An induced sub-TVG $\mathcal{G}' \subseteq \mathcal{G}$ is obtained by restricting either the set of vertices $V' \subseteq V$ or the lifetime $\mathcal{T}' \subseteq \mathcal{T}$, resulting in the tuple $(V', E', \mathcal{T}', \rho', \zeta')$ such that:

- (V', E') is the subgraph of (V, E) induced (in the usual sense) by V'
- $\rho' : E' \times \mathcal{T}' \rightarrow \{true, false\}$ where $\rho'(e, t) = \rho(e, t)$
- $\zeta' : E' \times \mathcal{T}' \rightarrow \{true, false\}$ where $\zeta'(e, t) = \zeta(e, t)$

If only the lifetime is rectricted, say to some interval $[t_a, t_b)$, then the resulting graph $\mathcal{G}'$ is called a *temporal* subgraph of $\mathcal{G}$ and denoted $\mathcal{G}_{[t_a,t_b)}$. The *temporal diameter* of a graph $\mathcal{G}$ *at time* t is the smallest duration d such that $\mathcal{G}_{[t,t+d)}$ is temporally connected.

Finally, following Bhadra and Ferreira in [1], we consider a temporal variant of connected components (hereafter, simply called *components*), which are maximal sets of nodes $V' \subseteq V$ such that $\forall p, q \in V', p \rightsquigarrow q$. Two variants are actually considered, whether the corresponding journeys can also use nodes that are in $V \setminus V'$ (*open* components) or not (*closed* components). Observe that a close component is equivalent to an induced sub-TVG being temporally connected.

3 A Timely Model for Dynamic Systems

This section focuses on the analysis of timeliness in dynamic systems at the most abstract point of view, i.e. considering only a general communication bound Δ for end-to-end communication. We first provide a set of definitions related to this bound, which leads to the formulation of a new class of TVGs that is a strict subset of Class 5 (*recurrent connectivity*) in [4]. We then specify a solution to the problem of Terminating Reliable Broadcast (TRB) in the corresponding context.

3.1 Definitions

We define the concept of bounded-time journey as follows:

Definition 1. *A journey $\mathcal{J}$ is said to be a* **Δ-journey** *if and only if $arrival(\mathcal{J}) - departure(\mathcal{J}) \leq \Delta$.*

Based on Δ-journeys we define the concept of bounded-time component. Unlike components, we require here that connectivity be also recurrent by definition.

Definition 2. *A* **Δ-component** *in $\mathcal{G} = (V, E, \mathcal{T}, \rho, \zeta)$ is a set $V' \subseteq V$ such that for every t in $[\mathcal{T}^-, \mathcal{T}^+ - \Delta]$, for every p, q in V', there exists a Δ-journey from p to q in $\mathcal{G}_{[t,t+\Delta)}$.*

Similarly to components, Δ-components can be open or closed, depending on whether the Δ-journeys use nodes in $V \setminus V'$. Observe that, a graph behaving in an open way provides flexibility in mobility, and therefore, a model allowing open Δ-components is weaker (in the sense that it requires less assumption) than a model strictly based on closed Δ-components. Henceforth we assume that in our system model Δ-components are open.

Informally, Δ-components allow us to think about subsets of nodes behaving timely with each other. Hence, nodes in a Δ-component are also said to be *timely connected*. We define the (parametrized) class of timely (and recurrently) connected TVGs $\mathcal{TC}(\Delta)$ as follows:

Definition 3. $\mathcal{G} \in \mathcal{TC}(\Delta) \iff V$ *is a* Δ*-component.*

3.2 Terminating Reliable Broadcast in $\mathcal{TC}(\Delta)$

According to [6], consensus is equivalent to Terminating Reliable Broadcast in static synchronous systems. We take this as a starting point and describe here a solution for TRB in the scope of a Δ-component.

First of all, we assume that processes know a global time. Processing times are negligible with respect to communication time. The system is composed by processes that can crash and recover, and leave and join the system. Processes that crash or leave the system, even if they recover or join again later, are by definition excluded from any Δ-component, however since we assume the existence of open Δ-components, they can punctually take part on various journeys.

Recall that a distributed system $\mathcal{G}$ may have several Δ-components. There may exist values of Δ for which a same processor belongs to different components, which are thus overlapping. However, since every component is *recurrently* connected, then overlapping components become naturally merged as the value for Δ increases, and transitively, there must exist a sufficiently large value of Δ such that all remaining components are disjoint. Henceforth, we consider Δ to be (an upper bound on) such a value.

We define now which processes are *correct* in terms of the classical terminology. In a classical partitioned system it can be considered that a process p behaves correctly in its partition, and incorrectly with respect to the other partitions in the system. Similarly, in our Δ-component based system a process p behaves correctly with respect to the Δ-component p belongs to, e.g. C. However p could still sporadically communicate timely with *some* processes in other Δ-component, C'. Obviously, we consider p incorrect with respect to C', but a message m from p received by some process in C' should either be delivered by all processes in C', or by none of them in order to hold the agreement property of reliable broadcast.

Thus, in $\mathcal{TC}(\Delta)$ a set of properties should be hold by a process *with respect to* a Δ-component in order to provide Δ-TRB:

- Δ-*Termination*: Every process in the same Δ-component eventually delivers some message.
- Δ-*Validity*: If a process in a Δ-component broadcasts a message m, then all processes in the same Δ-component eventually deliver m.
- Δ-*Agreement*: If a process in a Δ-component delivers a message m, then all processes in the same Δ-component eventually deliver m.

– *Δ-Integrity*: For any message m in a Δ-component, every process in the same Δ-component delivers at most one message, and if it delivers $m \neq SF$ then the sender(m) must have broadcast m.

As usual in TRB, the broadcast at time t_{init} of a message m should be considered in the scope of m.

To guarantee the Δ-Agreement property we should correctly understand when a message m broadcast by $p \notin C$ should be delivered by all processes in C. If p has been able to propagate m to some process $q \in C$, then we assume that there exists a Δ-journey from p to q. Observe that this assumption is consistent with the fact that our model allows the existence of open Δ-components.

```
To $\Delta$-TRBroadcast a message $m$ at time $t_{init}$:
if $p = p_B$ then
    for all $edge\ e = (p_B, -)$ s.t. $\rho(e, t_{init}) = true$ do
        $send(m)$ on $e$ at $t_{init}$

On appearance of $e = (p_B, -)$ at time $t \in [t_{init}, t_{init} + \Delta)$:
$send(m)$ on $e$ at $t$

On reception of a message $m$ for the first time at time $t_{rec} \in [t_{init}, t_{init} + 2\Delta)$:
if $p \neq p_B$ then
    for all $edge\ e = (p, -)$ s.t. $\rho(e, t_{rec}) = true$ do
        $send(m)$ on $e$ at $t_{rec}$

On appearance of $e = (p, -)$ at time $t \in [t_{init}, t_{init} + 2\Delta)$:
if a message $m$ has been previously received then
    $send(m)$ on $e$ at $t$

At time $t_{init} + 2\Delta$:
if a message $m$ has been previously received then
    $\Delta$-$TRDeliver(m)$
else
    $\Delta$-$TRDeliver(SF)$
```

Fig. 1. Terminating Reliable Broadcast for $\mathcal{TC}(\Delta)$.

A solution to the TRB problem is described in Figure 1. Informally, the distinguished process p_B Δ-TRBroadcasts a message m by sending m on all its active edges at time t_{init}. Whenever an edge in p_B's neighborhood appears[3], p_B also sends m on that edge. Every other process p, upon reception of m for the first time, forwards m on all its active edges, as well as upon the appearance of a new edge. Finally, at time $t_{init} + 2\Delta$ every process p Δ-TRBdelivers either m (if m has been received) or SF (*sender faulty* in the classical TRB terminology).

[3] We assume here the existence of an abstract *oracle* to capture events of edge appearance. In the next section we will board the implementation of such an oracle.

We explain next why a time of 2Δ is necessary and sufficient to deliver m.

Observe that, since we are assuming that p_B could be not in C, p_B could not be able to communicate to all nodes in C in Δ time, (otherwise $p_B \in C$), thus, after m is resent by q, every process in C will receive m into a second Δ time interval. Henceforth the bound for a process in C to $TRDeliver$ a message is 2Δ.

Correctness proof We proof that the specification in Figure 1 is a solution to Δ-TRBroadcast in a Δ-component.

Observation 1 *Observe that, by the system model assumptions, if a message m has been communicated between any two processes p and q then the communication time is bounded by Δ even if p and q are not in the same Δ-component.*

Lemma 1. *The specification in Figure 1 provides the Δ-Termination property.*

Proof. Observe that at time $t_{init} + 2\Delta$ a process p Δ-TRDelivers either m or a SF message.

Lemma 2. *The specification in Figure 1 provides the Δ-Validity property.*

Proof. Observe that p_B sends a message m at time t_{init} to all processes with an edge with p at t_{init} and p keeps sending m for every edge whenever it appears during 2Δ time. Since every process $p \in C$ resends m on the appearance of its edges during the same time interval, then m will be received by all processes in C and Δ-TRDelivered at time $t_{init} + 2\Delta$.

Lemma 3. *The specification in Figure 1 provides the Δ-Agreement property.*

Proof. By Lemma 3, every process q in a Δ-component C eventually receives and Δ-TRRdelivers m if $p_B \in C$. Else, if if $p_B \notin C$, we prove now that either (a) eventually every process $q \in C$ will Δ-TRDeliver m, or (b) no process in C will deliver m.

Assume first that a process $q \in C$ has received m. Since p_B has communicated with q, by Observation 1 m has been received by q not later than $t_{init} + \Delta$. Since q resends m whenever an edge appears during the next 2Δ time by definition of β-component C, and by the proof of Lemma 2 every process in C receives m before $t_{init} + 2\Delta$ and Δ-TRDelivers m at time $t_{init} + 2\Delta$.

Otherwise, if no process in C has received m before $t_{init} + \Delta$, again by Observation 1 m will not be received by any process in C, thus no process in C will deliver m.

Lemma 4. *The specification in Figure 1 provides the Δ-Integrity property.*

Proof. Observe that a process p Δ-TRDelivers a message just once. Observe also that m and SF are the only messages that can be delivered, been m the message that is Δ-TRBroadcast by process p_B.

Theorem 1. *The specification in Figure 1 satisfies the properties of Terminating Reliable Broadcast in a Δ-component.*

Proof. Straightforward from Lemmas 1, 2, 3 and 4.

4 Implementability of TRB

The specification of TRB provided in Figure 1 relies on an "oracle" available at every process p, which informs p instantaneously upon appearance of a new edge in its neighbourhood. Such an abstraction has been recently used by Raynal et al. [12] to implement a broadcast algorithm for recurrent dynamic systems. However, a strict implementation of this oracle in a real system is far from being trivial, as we discuss now.

Observe that the only temporal assumption on $\varDelta$-journeys is that they satisfy a given upper-bound $\varDelta$ in its temporal length, thus the duration of an edge may be as short as the latency of the message. In consequence, an implementation of this oracle should be able to allow the send of a message at the very same time that the edge get activated, which is unrealistic since the oracle should be able to predict the behaviour of the links in a real network. Alternatively, an algorithm could continuously send message m along the whole time interval in the hope that one of the sending attempts will success in the appearance of an edge. Observe, however, that this iteration would require a period of time zero between two consecutive sends. In other words, the algorithm should be able to send an infinite number of messages per unit of time, which is impossible.

Therefore, additional assumptions should be introduced in order to provide an implementation for the above specification of TRB. Specifically we first propose an extra assumption that allow to maintain active the edge not only for communicating the message but also to detect its appearance.

4.1 (Lower)-bounding the edge stability

We assume that the edge latency is bounded, i.e, there exist a bound on $\max\{\zeta(e,t) : t \in \mathcal{T}, e \in E\}$, that we call ζ_{MAX}. Additionally, we assume that edges are active at least β time. Let us call β-edge an edge that fulfils this bounded disposability. For this new model we define β-journeys as follows:

Definition 4. *A β-**journey** $\mathcal{J}$ at a time t is a $\varDelta$-journey such that:*

1. *$\zeta_{MAX} < \beta \leq \varDelta$.*
2. *$\forall i \in [0,k), e_i$ is a β-edge.*
3. *the times when edges are activated and their corresponding latencies allow a bounded sequential traversal (formally, $\forall i \in [0,k), t_{i+1} \geq t_i + \beta$).*

We now define β-components as a subset of $\varDelta$-components that uses β-journeys. Formally:

Definition 5. *A β-**component** is a $\varDelta$-component where a set $V' \subseteq V$ satisfies that $\forall t \in [\mathcal{T}^-, \mathcal{T}^+ - \varDelta], V'$ is a β-journey based temporal component in $\mathcal{G}_{[t,t+\varDelta)}$.*

We define the parametrized timely connectivity class $\mathcal{TC}'(\beta)$ as follows:

Definition 6. *$\mathcal{G} \in \mathcal{TC}'(\beta) \iff V$ is a β-component.*

```
W ← value ∈ (0, β − ζ_MAX]
To Δ-TRBroadcast a message m at time t_init:
    if p = p_B then
        while now() < t_init + Δ do
            send(m) to all
            wait(W)
On reception of a message m for the first time at time t_rec ∈ [t_init, t_init + 2Δ):
    Δ-TRDeliver(m)
    if p ≠ p_B then
        while now() < t_rec + Δ do
            send(m) to all
            wait(W)
At time t_init + 2Δ:
    if p has not Δ-TRDelivered any message then
        Δ-TRDeliver(SF)
```

Fig. 2. Terminating Reliable Broadcast for $\mathcal{TC}'(\beta)$.

TRB in $\mathcal{TC}'(\beta)$

We give now a TRB algorithm for the $\mathcal{TC}'(\beta)$ model, which is shown in Figure 2.

In the algorithm proposed in Figure 2 a process p_B sends at time t_{init} a message m by Δ-TRBroadcasting it, and p_B keeps sending m each W time in order to assure the correct send of m by every β-journey. Observe that, according to the definition of β-edge, for a β-edge $e = (p, q)$ in a β-journey, if process p sends a message m on e each $W \leq \beta - \zeta_{MAX}$ time during Δ, q will receive m at least once. When a process p receives the message m automatically Δ-TRDelivers m, and additionally, if $p \neq p_B$, p sends m each W time during Δ. Finally, if a process does not receive the message m, at time $t_{init} + 2\Delta$, it Δ-TRDelivers the special message SF.

Correctness proof We prove that Algorithm in Figure 2 solves Δ-TRBroadcast in a β-component. Thus, note that Δ-TRB properties hold on β-components.

Lemma 5. *Let $\beta > \zeta_{MAX}$, $0 < W \leq \beta - \zeta_{MAX}$ and let $e = (p, q)$ be a β-edge belonging to a β-journey $\mathcal{J}$. If a process p tries to send a message m on e each W from time $t \in [\mathcal{T}^-, \mathcal{T}^+ - \Delta]$ to time $t + \Delta$, q will receive m at least once.*

Proof. Since $\mathcal{J}$ is a β-journey, by definition $\mathcal{J}$ is also a Δ-journey and thus its temporal length is bounded by Δ. Also by definition of β-journey, e should appears at least once and be active for at least β time. Let t' be the time when the edge e appears, thus e is active in the interval $t' + \beta$. Observe that $t \leq t' \leq t + \Delta - \beta$.

We proof now that if p is sending m on e at times t, $t + W$, $t + 2W$, …, m will be received by q not later than $t + \Delta$.

Consider the worst-case situation, in which: (a) e becomes active only once in the interval (recall that $t' \leq t + \Delta - \beta$), (b) e has activated just after a sending attempt at time $t + kW$, and (c) the latency of the sending attempt at time $t + (k+1)W$ is the maximum latency we are assuming, ζ_{MAX}.

In this situation e is active in the interval $(t+kW, t+kW+\beta]$. Process p will try to send m at time $t + (k+1)W$, thus, in order to be a successful attempt, e should be active in the interval $[t + (k+1)W$ to $t + (k+1)W + \zeta_{MAX}]$. We should proof then that

$[t + (k+1)W, t + (k+1)W + \zeta_{MAX}] \subset (t + kW, t + kW + \beta]$

Observe first that at the time of the new sending of m by p, e continues to be active, since $W < \beta$. Observe now that the bound for m to be received by q is not higher than the time in which e disappears, since by definition $W \leq \beta - \zeta_{MAX}$. In effect,

$t + (k+1)W + \zeta_{MAX} \leq t + kW + \beta$

which results in

$\zeta_{MAX} \leq W + \beta$

Finally we show that m is received by q before $t + \Delta$.

In the limit, the only activation of e could happen at a time $t' \leq t + \Delta - \beta$. Thus, $t + kW < t + \Delta - \beta$. Since the last attempt of p sending q could be done as late as at time $t + (k+1)W$, m would be received by p before $t + \Delta - \beta + W + \zeta_{MAX}$. Again, since $W \leq \beta - \zeta_{MAX}$, the previous results in that m is received by q before $t + \Delta$.

Lemma 6. *Algorithm in Figure 2 provides the Δ-Termination property: Every process in the same β-component eventually delivers some message.*

Proof. Observe that by Lines 14-15 a process p executing the algorithm in Figure 2 Δ-TRDelivers a SF message at time $t_{init} + 2\Delta$ if p has not previously Δ-TRDeliver m by Line 8.

Lemma 7. *Algorithm in Figure 2 provides the Δ-Validity property: If a process in a β-component broadcasts a message m, then all processes in the same β-component eventually deliver m.*

Proof. Observe first that, since $W \in (0, \beta - \zeta_{MAX})$ by Line 1, Lemma 5 is applicable. By Lemma 5 and the definition of β-component, if a process p_B in a β-component C sends a message m to all processes at time t_{init} and p_B keeps sending m periodically with a period $W < \beta - \zeta_{MAX}$ (lines 4-6), then m will be received by Line 7 at least by one process in C, otherwise p_B is the only process in C. A process $q \in C$ receiving m by Line 7 will Δ-TRDeliver m, and will resend m by lines 10-12 of the Algorithm. Reasoning as previously by iteration on Lemma 5 and the definition of β-component, every process in C will Δ-TRDeliver m before $t_{init} + \Delta$.

Lemma 8. *Algorithm in Figure 2 provides the Δ-Agreement property: If a process in a β-component delivers a message m, then all processes in the same β-component eventually deliver m.*

Proof. By Lemma 7, every process q in a β-component C eventually receives and Δ-TRDelivers m if $p_B \in C$. Else, if $p_B \notin C$, we prove now that either (a) eventually every process $q \in C$ will Δ-TRDeliver m, or (b) no process in C will deliver m.

Assume first that a process $q \in C$ has received m (by Line 7). Before resending m by lines 10-12, q will Δ-TRDeliver m (by Line 8), thus we should prove now that every process in C will Δ-TRDeliver m. Since p_B has communicated with q, by Observation 1 m has been received by q not later than $t_{init} + \Delta$. Since q resends m by lines 10-12, by definition of β-component C, and by the proof of Lemma 7 every process in C eventually receives and Δ-TRDelivers m.

Otherwise, if no process in C has received m before $t_{init} + \Delta$, again by Observation 1 m will not be received by any process in C, thus no process in C will deliver m.

Lemma 9. *Algorithm in Figure 2 provides the Δ-Integrity property: For any message m present in a β-component, every process in the same β-component delivers at most one message, and if it delivers $m \neq SF$ then the sender(m) must have broadcast m.*

Proof. A process p executes the Δ-TRDeliver primitive (after receiving m, Line 8) just once since the Line 9 explicitly denotes "for the first time", or by SF by Line 15 at time $t_{init} + \Delta$. Observe that, by Line 14, Δ-TRDeliver(SF) is only executed if p has not previously delivered m by Line 8, thus either one message m or SF will be delivered.

Observe also by the Algorithm that m and SF are the only messages that can be delivered, been m the message that is Δ-TRBroadcast by process p_B.

Theorem 2. *The algorithm in Figure 2 satisfies the properties of Δ-TRB in a β-component.*

Proof. Straightforward from Lemmas 6, 7, 8 and 9.

4.2 (Upper)-bounding the edge appearance

Observe that, in the algorithm in Figure 2 messages are forwarded during the whole Δ interval. This is necessary because the ending edge of a β-journey could be activated at a time as late as $t_{init} + \Delta - \beta$. It is apparent that more efficient implementations of a TRB algorithm in terms of number of messages could be envisaged if stronger connectivity assumptions are introduced in the model. Specifically, in this section we introduce an additional timely assumption on the appearance of edges.

We adopt the assumption of [7], where, besides β, a bound α on the appearance of links is defined. We define a new type of journey, that we call (α, β)-journey. Formally:

Definition 7. *A (α, β)***-journey** *$\mathcal{J}$ at a time t is a β-journey such that:*

1. *The appearance of e_1 is bounded by $t_1 \leq t + \alpha$.*
2. *The appearance of the subsequent edges are also bounded by α. Formally, $t_{i+1} \leq t_i + \zeta(e_i, t_i) + \alpha$ for all $i \in [1, k)$).*

We define a (α, β)-component as follows:

Definition 8. *A (α, β)***-component** *is a β-component where a set $V' \subseteq V$ satisfies that $\forall t \in [\mathcal{T}^-, \mathcal{T}^+ - \Delta], V'$ is a (α, β)-journey based temporal component in $\mathcal{G}_{[t,t+\Delta)}$.*

We define the parametrized timely connectivity class $\mathcal{TC}''(\alpha, \beta)$ as follows:

Definition 9. *$\mathcal{G} \in \mathcal{TC}''(\alpha, \beta) \iff V$ is a (α, β)-component.*

```
$W \leftarrow value \in (0, \beta - \zeta_{MAX}]$
$\Gamma \leftarrow (\lceil \frac{\alpha}{W} \rceil + (|V| - 2)\lceil \frac{\zeta_{MAX}+\alpha}{W} \rceil)W + \zeta_{MAX}$
To $\Delta$-TRBroadcast a message $m$ at time $t_{init}$:
    if $p = p_B$ then
        send($m$) to all
        repeat
            wait($W$)
            send($m$) to all
        until now() $> t_{init} + \alpha$

On reception of a message $m$ for the first time at time $t_{rec}$:
    $\Delta$-TRDeliver($m$)
    if $p \neq p_B$ then
        send($m$) to all
        repeat
            wait($W$)
            send($m$) to all
        until now() $> t_{rec} + \alpha$

At time $t_{init} + \Gamma$:
    if $p$ has not $\Delta$-TRDelivered any message then
        $\Delta$-TRDeliver($SF$)
```

Fig. 3. Terminating Reliable Broadcast for $\mathcal{TC}''(\alpha, \beta)$.

TRB in $\mathcal{TC}''(\alpha, \beta)$

The algorithm in Figure 3, describes a TRB algorithm executable in a $\mathcal{TC}''(\alpha, \beta)$ dynamic system.

The new bound α, altogether with the latency bound β and ζ_{MAX}, allows to calculate global system bounds, namely the period W and a time to deliver Γ, strictly in terms of specific network parameters. In the algorithm proposed in Figure 3 a process p_B Δ-TRBroadcast a message m at time t_{init} by sending each W time m until the time is strictly higher than $t_{init} + \alpha$, in order to assure the correct send of m by every (α, β)-journey. When a process p receives the message m at time t_{rec} for the first time, automatically Δ-TRDelivers m and, additionally, if $p \neq p_B$, p sends m each W until the time is strictly higher than $t_{rec} + \alpha$. Finally, if any of the process in p does not receive the message m at time $t_{init} + \Gamma$, Δ-TRDelivers the special message SF denoting the sender failure.

It is important to note that in the TRB algorithm for $\mathcal{TC}''(\alpha, \beta)$, differently to the upper classes, processes need to known the network diameter, whic is bounded by $|V| - 1$. This is a consequence of the fact of considering strictly local bounds in $\mathcal{TC}''(\alpha, \beta)$. Instead, both $\mathcal{TC}(\Delta)$ and $\mathcal{TC}'(\beta)$ rely on a system-wide bound, Δ.

Bounding the time-to-deliver. We explain now how we calculate Γ, the bound used in the algorithm in Figure 3 for a process to Δ-TRBdeliver the message (see Figure 4 for a graphical illustration).

A process p_1 (the sender) will send a copy of m from t_{init} on, each W time units. In the worst case, the first edge of the journey will appear at $t_{init} + \alpha$, but p_1 will not success sending a copy of m until $t_{init} + \lceil \frac{\alpha}{W} \rceil W$, i.e., the $\lceil \frac{\alpha}{W} \rceil W$'s sent. Observe that, $t_{init} + \alpha < t_{init} + \lceil \frac{\alpha}{W} \rceil W \leq t_{init} + \alpha + W$.

For a journey including a single edge (p_1, p_2), the message m would be delivered by p_2 at time $t_{init} + \lceil \frac{\alpha}{W} \rceil W + \zeta_{MAX}$.

In general, for a journey including k nodes (and thus $k-1$ hops), excluding the first hop, the subsequent $k-2$ hops can be time-bounded as follows: a message m resent by a process p_i is received by p_{i+1} in ζ_{MAX} time units and p_{i+1} waits α time units until the appearance of edge e_{i+1} to resent m on this edge. Consequently, p_{i+1} will success in the sent made on e_{i+1} at a time not greater than $t_{init} + \lceil \frac{\alpha}{W} \rceil W + \lceil \frac{\zeta_{MAX}+\alpha}{W} \rceil W$. Summarizing, a message from p_1 to pk by a (α, β)-journey at time t_{init} will be delivered by p_k at time $t_{init} + \Gamma$, where $\Gamma = (\lceil \frac{\alpha}{W} \rceil + (k-2)\lceil \frac{\zeta_{MAX}+\alpha}{W} \rceil)W + \zeta_{MAX}$.

In the worst case, $k = |V| - 1$, thus the bound to deliver a message will be $t_{init} + \Gamma$, where $\Gamma = (\lceil \frac{\alpha}{W} \rceil + (|V|-2)\lceil \frac{\zeta_{MAX}+\alpha}{W} \rceil)W + \zeta_{MAX}$.

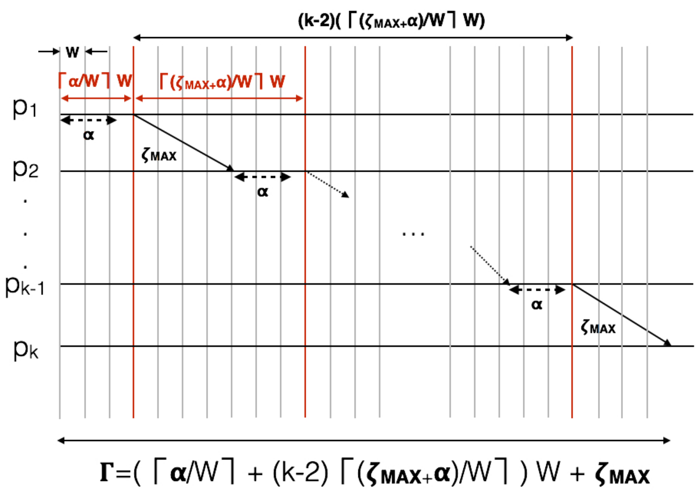

Fig. 4. A time-line explaining the Γ upper-bound for the worst case (α, β)-journey from a process p to another process q in the system.

Correctness proof

Lemma 10. *Algorithm in Figure 3 provides the Δ-Termination property: Every process in the same (α, β)-component eventually delivers some message.*

Proof. Observe that by Lines 19-20 a process p executing the algorithm in Figure 3 Δ-TRDelivers a SF message at time $t_{init} + \Gamma$ if p has not previously Δ-TRDeliver m by Line 11.

Lemma 11. *Algorithm in Figure 3 provides the Δ-Validity property: If a process in a $(alpha, \beta)$-component broadcasts a message m, then all processes in the same $(alpha, \beta)$-component eventually deliver m.*

Proof. By the definition of (α, β)-component, if a process p_B in a (α, β)-component C sends a message m to all processes at time t_{init} and p_B keeps sending m periodically with a period $W < \beta - \zeta_{MAX}$ (lines 5-9) during the maximum time for the appearance of the link, α, then m will be received by Line 10 at least by one process in C, otherwise p_B is the only process in C. Reasoning in the same way by iteration on lines 13-17 of the Algorithm, by the definitions of (α, β)-component, every process $q \in C$ will Δ-TRDeliver m by Line 11 before $t_{init} + \Gamma$. Observe by Line 2 that Γ has been set as a bound of the temporal length of an (α, β)-journey in a system with $|V|$ nodes.

Lemma 12. *Algorithm in Figure 3 provides the Δ-Agreement property: If a process in a (α, β)-component delivers a message m, then all processes in the same (α, β)-component eventually deliver m.*

Proof. By Lemma 11, every process q in a (α, β)-component C eventually receives and ΔTRDelivers m if $p_B \in C$. Else, if $p_B \notin C$, we prove now that either (a) eventually every process $q \in C$ will ΔTRDeliver m, or (b) no process in C will deliver m.

Assume first that a process $q \in C$ has received m (by Line 10). Since p_B has communicated with q, m has been received by Line 10 of q according the time bounds α and β following a journey topologically bounded by the maximum network diameter, $|V - 1|$, and q Δ-TRDelivers m by Line 11. By Line 2, Γ has been set as a bound on the temporal length of such an (α, β)-journey. By lines 13-17 m is resent by q during the maximum time for the appearance of the link, α, and, again by Lemma 11, every process in a (α, β)-component C eventually receives and ΔTRDelivers m. Again, Γ holds as the general bound, since it considerer the worst-case diameter, which includes all the processes in the system.

Otherwise, if no process in C has received m before $t_{init} + \Gamma$, every process in C will Δ-TRDeliver SF at time $t_{init} + \Gamma$.

Lemma 13. *Algorithm in Figure 3 provides the Δ-Integrity property: For any message m present in a (α, β)-component, every process in the same (α, β)-component delivers at most one message, and if it delivers $m \neq SF$ then the sender(m) must have broadcast m.*

Proof. A process p executes the Δ-TRDeliver primitive (after receiving m, Line 11) just once since the Line 10 explicitly denotes "for the first time", or by SF by Line 20 at time $t_{init}+\Gamma$. Observe that, by Line 20, Δ-TRDeliver(SF) is only executed if p has not previously delivered a message m by Line 11.

Observe also by the Algorithm that m and SF are the only messages that can be delivered, been m the message that is Δ-TRBroadcast by process p_B.

Theorem 3. *The algorithm in Figure 3 satisfies the properties of Terminating Reliable Broadcast in a Δ-component.*

Proof. Straightforward from Lemmas 10, 11, 12 and 13.

4.3 Relating timely classes

We have defined a hierarchy of classes with increasingly stronger timely assumptions. Being $\mathcal{TC}(\varDelta)$, $\mathcal{TC}'(\beta)$ and $\mathcal{TC}''(\alpha,\beta)$ the *parametrized* classes, we define now for each one the union of all its possible instances:

$$\mathcal{G} \in \mathcal{TC}^* \iff \exists \varDelta \neq \infty : \mathcal{G} \in \mathcal{TC}(\varDelta)$$

$$\mathcal{G} \in \mathcal{TC}'^* \iff \exists \beta \neq \infty : \mathcal{G} \in \mathcal{TC}'(\beta)$$

$$\mathcal{G} \in \mathcal{TC}''^* \iff \exists \alpha, \beta \neq \infty : \mathcal{G} \in \mathcal{TC}''(\alpha,\beta)$$

In spite of the different strength of the parametrized classes, we show that $\mathcal{TC}''^* \equiv \mathcal{TC}'^*$. Besides, $\mathcal{TC}'^* \subset \mathcal{TC}^*$ and $\mathcal{TC}''^* \subset \mathcal{TC}^*$.

Correctness proof

Theorem 4. $\mathcal{TC}''^* \equiv \mathcal{TC}'^*$

Proof. On the one hand, $\forall \mathcal{G} \in \mathcal{TC}''^*, \exists \beta : \mathcal{G} \in \mathcal{TC}'(\beta)$, since by definition, a (α,β)-component in $\mathcal{TC}'(\beta)$ is a β-component. More specifically, if $\mathcal{G} \in \mathcal{TC}''(\alpha,\beta)$ then $\mathcal{G} \in \mathcal{TC}'((|V|-1)(\alpha+\beta),\beta)$. On the other hand, $\forall \mathcal{G} \in \mathcal{TC}'^*, \exists \alpha,\beta : \mathcal{G} \in \mathcal{TC}'(\alpha,\beta)$. Observe that $\mathcal{G} \in \mathcal{TC}''(\alpha',\beta)$ such that α' is $\max(t-t') : t \in [\mathcal{T}^-, \mathcal{T}+], t' = t_0)$ where $\forall \mathcal{J} = \{(e_0,t_0),(e_1,t_1),\ldots\} \in \mathcal{J}^* \wedge arrival(\mathcal{J}) < t + \varDelta$ and $\mathcal{G} \in \mathcal{TC}'(\beta)$. Since, $\mathcal{TC}'^* \setminus \mathcal{TC}''^* = \emptyset$, then $\mathcal{TC}''^* \equiv \mathcal{TC}'^*$.

Theorem 5. $\mathcal{TC}'^* \subset \mathcal{TC}^*$

Proof. By the definition of $\mathcal{TC}'(\beta), \forall G \in \mathcal{TC}'^*, \exists \varDelta : G \in \mathcal{TC}(\varDelta)$. We proof now that there exists $\mathcal{G} \in \mathcal{TC}^*$ such that $G \notin \mathcal{TC}'^*$. Assume a graph $\mathcal{G}$ such that which any of its journey is composed by an edge e_i which is active during a β_i time where $\beta_i = \zeta_{MAX}$, and, by Definition 4, those journeys are not allowed in any parametrized $\mathcal{TC}'(\beta)$ class. Consequently, $\mathcal{TC}'^* \subset \mathcal{TC}^*$.

Theorem 6. $\mathcal{TC}''^* \subset \mathcal{TC}^*$

Proof. By Theorem 4 and Theorem 5, $\mathcal{TC}''^* \subset \mathcal{TC}^*$.

5 From $\varDelta$-TRB to $\varDelta$-Consensus in Dynamic Systems

In this section we analyse the equivalence between TRB and consensus, originally stated for synchronous static systems [6], in terms of a dynamic system as the one we have modelled.

In the former sections we have presented three $\varDelta$-TRB algorithms in the scope of respectively $\varDelta$-, β- and (α,β)-components. We show now how the consensus problem can be reduced[4] to a $\varDelta$-TRB problem. We will refer as $\varDelta$-Consensus to this kind of consensus in the scope of $\varDelta$-components.

By the properties of $\varDelta$-TRB, it is straightforward to define the $\varDelta$-Consensus properties as follows:

[4] We say that a problem A can be reduced to a problem B if A can be solved using B.

– Δ-*Termination:* Every process in the Δ-component eventually decides.
– Δ-*Agreement:* Every process in the Δ-component decides the same value.
– Δ-*Validity:* The decided value is a proposed one.

Without loosing generality we focus here on Δ-Consensus using the Δ-TRB specification of Figure 1 for the $\mathcal{TC}(\Delta)$ Class.

```
Vector $V_p(i) \leftarrow \bot : i \in [0, |V|)$
To $\Delta$-Propose $v$ at time $t_{init}$:
    $\Delta$-TRBroadcast($v$)
On $\Delta$-TRDeliver($m$) by $q$:
    $V_p(q) \leftarrow m$
At time $t_{init} + 2\Delta$:
    $\Delta$-decide($V_p(\min(i : V_p(i) \neq SF))$)
```

Fig. 5. Δ-TRB based Δ-Consensus algorithm for $\mathcal{TC}(\Delta)$.

The resulting Δ-Consensus algorithm is shown in Figure 5. Every process p holds a vector V_p initialized to $\bot$. At time t_{init}, $|V|$ instances of of Δ-TRB are started, one per process, being each process the sender in one instance. Every process p records in vector $V_p(q)$ the message m_q delivered from process q (or SF in case m_q has not been received on time). At time $t_{init} + 2\Delta$, p decides on the first non-SF value of V_p.

Note that solving consensus at system level would require a second Δ-TRB round to agree on the decision of the majority, provided that the temporal interval $[\mathcal{T}^-, \mathcal{T}^+]$ in which the Δ-component is defined covers both rounds. In other words, the stability of Δ-components must be temporally extended to solve consensus at system level.

6 Conclusions

In this paper we have studied how to introduce timeliness in evolving systems so that the resolution of agreement problems (specifically consensus) is possible. On the basis of previous works, we have adopted the concept of journey or temporal path and have introduced the necessary timeliness (i.e., time bounds) to describe the specific assumptions that are required by an agreement algorithm to terminate and satisfy the consensus properties.

We have first proposed a general class, $\mathcal{TC}(\Delta)$, with a very abstract property on the temporal connectivity of the TVG to provide the necessary stability conditions, namely, that the temporal diameter of a recurrent component in the TVG is bounded. We refer to such a component as a Δ-*component*. To approach the consensus problem we have defined a TRB specification in terms of Δ-*components,* Δ-*TRB*. However, Δ-TRB is not implementable in $\mathcal{TC}(\Delta)$ by message-passing without zero processing time assumptions. Henceforth, by introducing increasingly stronger connectivity assumptions, we have provided two *implementable* connectivity classes, namely $\mathcal{TC}'(\beta)$ and $\mathcal{TC}''(\alpha, \beta)$, as well as two respective implementations of Δ-TRB in these classes.

Finally, we have shown that consensus at Δ-component level is easily reduced to Δ-TRB.

An open issue is the search of the weakest connectivity class that allows to implement Δ-TRB (and henceforth consensus) in message-passing systems. Of additional interest is to extend the proposed classes to partially synchronous models. In this regard, in [9], leader election is implemented in a partially synchronous system with dynamic partitions that could be modelled as Δ-components.

References

1. Sandeep Bhadra and Afonso Ferreira. Complexity of connected components in evolving graphs and the computation of multicast trees in dynamic networks. In *Ad-Hoc, Mobile, and Wireless Networks*, pages 259–270. Springer, 2003.
2. Martin Biely, Peter Robinson, and Ulrich Schmid. Agreement in directed dynamic networks. In *Structural Information and Communication Complexity - 19th International Colloquium, SIROCCO 2012, Reykjavik, Iceland*, volume 7355 of *Lecture Notes in Computer Science*, pages 73–84. Springer, 2012.
3. Arnaud Casteigts, Paola Flocchini, Bernard Mans, and Nicola Santoro. Deterministic computations in time-varying graphs: Broadcasting under unstructured mobility. In *Theoretical Computer Science*, pages 111–124. Springer, 2010.
4. 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.
5. Tushar Deepak Chandra and Sam Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267, 1996.
6. Paul Feldman and Silvio Micali. Optimal algorithms for byzantine agreement. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 148–161. ACM, 1988.
7. Antonio Fernández-Anta, Alessia Milani, Miguel A Mosteiro, and Shmuel Zaks. Opportunistic information dissemination in mobile ad-hoc networks: The profit of global synchrony. *Distributed Computing*, 25(4):279–296, 2012.
8. Michael J. Fischer, Nancy A. Lynch, and Mike Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, 1985.
9. Carlos Gomez-Calzado, Alberto Lafuente, Mikel Larrea, and Michel Raynal. Fault-tolerant leader election in mobile dynamic distributed systems. In *Dependable Computing (PRDC), 2013 IEEE 19th Pacific Rim International Symposium on*, pages 78–87, Dec 2013.
10. Fabian Kuhn, Nancy A. Lynch, and Rotem Oshman. Distributed computation in dynamic networks. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 513–522. ACM, 2010.
11. Fabian Kuhn and Rotem Oshman. Dynamic networks: models and algorithms. *ACM SIGACT News*, 42(1):82–96, 2011.
12. M. Raynal, J. Stainer, Jiannong Cao, and Weigang Wu. A simple broadcast algorithm for recurrent dynamic systems. In *Advanced Information Networking and Applications (AINA), 2014 IEEE 28th International Conference on*, pages 933–939, May 2014.