Shortest, Fastest, and Foremost Broadcast in Dynamic Networks *

Arnaud Casteigts[1], Paola Flocchini[2], Bernard Mans[3], and Nicola Santoro[4]

[1] University of Bordeaux, France
[2] University of Ottawa, Canada
[3] Macquarie University, Sydney, Australia
[4] Carleton University, Ottawa, Canada

Highly dynamic networks rarely offer end-to-end connectivity at a given time. Yet, connectivity in these networks can be established over time and space, based on temporal analogues of multi-hop paths (also called *journeys*). Attempting to optimize the selection of the journeys in these networks naturally leads to the study of three cases: shortest (minimum hop), fastest (minimum duration), and foremost (earliest arrival) journeys. Efficient centralized algorithms exists to compute all cases, when the full knowledge of the network evolution is given.

In this paper, we study the *distributed* counterparts of these problems, i.e. shortest, fastest, and foremost broadcast with termination detection (TDB), with minimal knowledge on the topology. We show that the feasibility of each of these problems requires distinct features on the evolution, through identifying three classes of dynamic graphs wherein the problems become gradually feasible: graphs in which the re-appearance of edges is *recurrent* (class $\mathcal{R}$), *bounded-recurrent* ($\mathcal{B}$), or *periodic* ($\mathcal{P}$), together with specific knowledge that are respectively n (the number of nodes), Δ (a bound on the recurrence time), and p (the period). In these classes it is not required that all pairs of nodes get in contact – only that the overall *footprint* of the graph is connected over time.

Our results, together with the strict inclusion between $\mathcal{P}$, $\mathcal{B}$, and $\mathcal{R}$, implies a feasibility order among the three variants of the problem, i.e. TDB$[foremost]$ requires weaker assumptions on the topology dynamics than TDB$[shortest]$, which itself requires less than TDB$[fastest]$. Reversely, these differences in feasibility imply that the computational powers of $\mathcal{R}_n$, $\mathcal{B}_\Delta$, and $\mathcal{P}_p$ also form a strict hierarchy.

Keywords: dynamic networks, distributed algorithm, time-varying graphs, delay-tolerant broadcast, recurrent edges.

1. Introduction

Dynamic networks are widely addressed in distributed computing. Contexts of interest are as varied as fault-tolerance, interaction scheduling, dynamic membership, planned mobility, or unpredictable mobility. The recent emergence of scenarios where entities are truly mobile and can communicate without infrastructure (e.g. vehicles, satellites, robots, or pedestrian smartphones) brought to the fore the most versatile of these environments. In these *highly* dynamic networks, changes are not anomalies but rather integral part of the nature of the system.

The need to categorize and understand highly dynamic networks led the engineering community to design a variety of *mobility models*, each of which captures a particular con-

*Preliminary results were presented at the 6th IFIP International Conference on Theoretical Computer Science. This work has been supported in part by the ARC (Australia), NSERC (Canada) and Dr. Flocchini's University Research Chair.

text by means of rules that determine how the nodes move and communicate (see e.g. [22]). A popular example includes the well-known *random waypoint* model [5]. The main interest of these models is to be able to reproduce experiments and compare different solutions on a relatively fair basis, thereby providing a common ground to solve practical challenges in highly dynamic networks, e.g. routing and broadcasting [8, 20, 24, 25, 27, 30, 31].

In the same way as mobility models enable to federate practical investigations in highly dynamic networks, *logical properties* on the graph dynamics, that is, *classes of dynamic graphs*, have the potential to guide a more formal exploration of their analytical aspects. A number of special classes were recently identified, for instance graphs in which the nodes interact infinitely often (*e.g.*, uniform random scheduler for *population protocols* [1, 2, 12]); graphs whose dynamics is unrestricted but remains connected at any instant [16, 28]; graphs in which there exists a stable connected spanning subgraph in any T-time window (*a.k.a. T-interval connectivity*) [21, 26]; graphs whose edges appear or disappear with given probabilities [4, 13, 14, 29]; graphs that have a stable root component [6]; graphs whose schedule is periodic [10, 18, 19, 23] or guarantees minimal reachability properties [9]. These classes (among others) were characterized within a common formal framework and organized into a hierarchy in [11].

In this paper we are interested in studying specific relationship between some of these classes, namely three subclasses of those networks called *delay-tolerant networks* (DTNs), in which instant connectivity is never guaranteed, but still connectivity can be achieved over time and space (see e.g. [3]). These classes are:

- Class $\mathcal{R}$ of all graphs whose edges cannot disappear forever (*recurrent edges*). That is, if an edge appears once and disappears, then it will eventually re-appear at some unknown (but finite) date. It is not required that all pairs of nodes share an edge, but only that the *footprint* of all edges forms a connected graph (otherwise, even temporal connectivity is not guaranteed). This class corresponds to Class 6 in [11].
- Class $\mathcal{B}$ (for *bounded-recurrent* edges) consisting of those graphs with recurrent edges in which the recurrence time cannot exceed a given duration Δ. And again, the footprint is connected. This class corresponds to Class 7 in [11].
- Class $\mathcal{P}$ (for *periodic* edges) consisting of those graphs in which all topological events (appearance or disappearance) repeat identically modulo some period p. And again, the footprint is connected. This class corresponds to Class 8 in [11].

As far as *inclusion* is concerned, it clearly holds that $\mathcal{P} \subset \mathcal{B} \subset \mathcal{R}$, but what about the *computational relationship* between these classes? Considering different types of knowledge, namely the number n of nodes in the network, a bound Δ on the recurrence time, and (any multiple of) the period p, we look at the relationship between $\mathscr{P}(\mathcal{R}_n)$, $\mathscr{P}(\mathcal{B}_\Delta)$, and $\mathscr{P}(\mathcal{P}_p)$, where $\mathscr{P}(\mathcal{C}_K)$ is the set of problems one can solve in class $\mathcal{C}$ with knowledge K.

The investigation is carried out by studying a fundamental problem in distributed computing: *broadcast* with termination detection at the emitter (or TDB); this problem is also known in the literature with different names (Echo, Propagation of Information with Feed-

back, etc.) or context (synchronizers, etc.). It can have at least three distinct definitions in highly dynamic networks: TDB$[foremost]$, in which the date of delivery is minimized at every node; TDB$[shortest]$, where the number of hops used by the broadcast is minimized relative to every node; and TDB$[fastest]$, where the overall duration of the broadcast is minimized (however late the departure be). These three metrics were considered in the seminal work by Bui-Xuan, Ferreira, and Jarry [7] where the authors solved the offline problem of computing all shortest, fastest, and foremost journeys from a given node, given a complete schedule of the network.

Main contributions

In this paper we examine the feasibility and reusability of the solution (and to some extent, the complexity) of TDB$[foremost]$, TDB$[shortest]$, and TDB$[fastest]$ in $\mathcal{R}$, $\mathcal{B}$, $\mathcal{P}$ with knowledge $\emptyset$, n, or Δ. We additionally draw some observations from existing results in $\mathcal{P}$ with knowledge p [10], that complete our general picture of feasibility and reusability of broadcast in the three classes. Here is a short summary of some of the contributionss.

Feasibility: We first show that none of these problems are solvable in any of the classes unless additional knowledge is considered. We then establish several results, both positive and negative, on the feasibility of TDB$[foremost]$, TDB$[shortest]$, and TDB$[fastest]$ in $\mathcal{R}$, $\mathcal{B}$, and $\mathcal{P}$. In particular, we prove that knowing n makes it possible to solve TDB$[foremost]$ in $\mathcal{R}$, but this is not sufficient to solve TDB$[shortest]$ nor TDB$[fastest]$, even in $\mathcal{B}$. TDB$[shortest]$ becomes in turn feasible in $\mathcal{B}$ if Δ is known, but this is not sufficient to solve TDB$[fastest]$; this later problem becomes solvable in $\mathcal{P}$ knowing p [10]. These results allow us to show that

$$\mathscr{P}(\mathcal{R}_n) \subsetneq \mathscr{P}(\mathcal{B}_\Delta) \subsetneq \mathscr{P}(\mathcal{P}_p) \tag{1}$$

where $\mathscr{P}(\mathcal{C}_K)$ denotes the set of problems solvable in every $\mathcal{G} \in \mathcal{C}$ with knowledge K. In other words, the computational relationships between these three contexts form a *strict* hierarchy.

In the universe $\mathcal{U} = \{\mathcal{R}_n, \mathcal{B}_n, \mathcal{B}_\Delta, \mathcal{B}_{\{n,\Delta\}}, \mathcal{P}_n, \mathcal{P}_\Delta, \mathcal{P}_{\{n,\Delta\}}, \mathcal{P}_p\}$ of the classes of dynamic networks with knowledge considered here, let $P_1 \preceq P_2$ denote the fact that P_1 is "no more difficult" than P_2, that is, if P_2 is solvable in $\mathcal{G}_K \in \mathcal{U}$ so is P_1; and let $P_1 \prec P_2$ denote the fact that P_1 is "less difficult" than P_2, that is, $P_1 \preceq P_2$ and there exists $\mathcal{G}_K \in \mathcal{U}$ in which P_1 is solvable but P_2 is not. Our results show the existence of a *strict* hierarchy between these problems with respect to *feasibility*:

$$\text{TDB}[foremost] \prec \text{TDB}[shortest] \prec \text{TDB}[fastest] \tag{2}$$

These results are summarized in Table 1.

Reusability: Regarding the possibility to reuse a solution, that is, a same broadcast tree, over several broadcasts, we establish several results; in particular we show the intriguing fact that reusability in TDB$[shortest]$ is easier than that of TDB$[foremost]$. Precisely,

<table>
<tr><th>Knowledge / Class</th><th>$\emptyset$</th><th>n</th><th>Δ</th><th>p</th></tr>
<tr><td>$\mathcal{R}$</td><td rowspan="3">$\neg$ Foremost
$\neg$ Shortest
$\neg$ Fastest</td><td rowspan="3">Foremost
$\neg$ Shortest
$\neg$ Fastest</td><td>n/a</td><td rowspan="2">n/a</td></tr>
<tr><td>$\mathcal{B}$</td><td rowspan="2">Foremost, Shortest
$\neg$ Fastest</td></tr>
<tr><td>$\mathcal{P}$</td><td>Foremost, Shortest, Fastest*</td></tr>
</table>

Table 1. Feasibility of broadcast with termination detection, depending on the class of dynamic networks ($\mathcal{R}, \mathcal{B}, \mathcal{P}$) and of knowledge ($\emptyset, n, \Delta, p$). Unfeasibility is denoted by "$\neg$". The feasibility of Fastest in $\mathcal{P}$ with knowledge p (* in table) is from [10].

when TDB$[shortest]$ becomes feasible in $\mathcal{B}$, it enables at once reusability of the broadcast trees, whereas TDB$[foremost]$, although it was already feasible in $\mathcal{R}$, does not enable reusability until in $\mathcal{P}$ [10].

For reusability, let relations $\leq$ and $<$ be the analogous of $\preceq$ and $\prec$, respectively; and let $P_1 \equiv P_2$ denote that both $P_1 \leq P_2$ and $P_2 \leq P_1$. Our results imply that:

$$\text{TDB}[shortest] < \text{TDB}[foremost] \equiv \text{TDB}[fastest] \tag{3}$$

Complexity: Although complexity is not the main focus here, we characterize the time complexity and message complexity of our algorithms and observe some interesting facts. For instance, the message complexity of our algorithm for TDB$[foremost]$ is lower knowing Δ than knowing n, and even lower if both are known. These results are summarized in Table 2. Note that TDB involves two processes: the actual dissemination of *information messages*, and the exchange of typically smaller *control messages* (e.g. for termination detection), both of which are separately analyzed. Regarding time complexity, we observe that for all algorithms but those which terminate implicitly, the termination detection phase takes the same order of time as the dissemination phase. Thus, Table 2 does not distinguish both phases.

2. Model and Basic Properties

2.1. *Definitions and Terminology*

Consider a system composed of a finite set of n entities V (or nodes) that interact with each other over a (possibly infinite) time span $\mathcal{T} \subseteq \mathbb{T}$ called *lifetime* of the system, where $\mathbb{T}$ is the temporal domain (typically, $\mathbb{N}$ or $\mathbb{R}^+$ for discrete and continuous-time systems, respectively). In this paper we consider a continuous-time setting with $\mathbb{T} = \mathbb{R}^+$.

Following [11], we describe the network as a *time-varying graph* (*TVG*, for short) $\mathcal{G} = (V, E, \mathcal{T}, \rho, \zeta)$, where $E \subseteq V \times V$ is a set of m (possibly *intermittent*) undirected edges such that $(u, v) \in E \Leftrightarrow u$ and v have at least one contact over $\mathcal{T}$; $\rho : E \times \mathcal{T} \to \{0, 1\}$ (*presence function*) indicates whether a given edge is *present* at a given time; and $\zeta : E \times \mathcal{T} \to \mathbb{T}$ (*latency function*), indicates the time it takes to cross a given edge (*i.e.*, send a message) if starting at a given time. In this paper we assume ζ to be constant over

Metric	Class	Knowl.	Time	Info. msgs (1^{st} run)	Control msgs (1^{st} run)	Info. msgs (next runs)	Control msgs (next runs)
Foremost	$\mathcal{R}$	n	unbounded	$O(m)$	$O(n^2)$	$O(m)$	$O(n)$
	$\mathcal{B}$	n	$O(n\Delta)$	$O(m)$	$O(n^2)$	$O(m)$	$O(n)$
		Δ	$O(n\Delta)$	$O(m)$	$O(n)$	$O(m)$	0
		$n\&\Delta$	$O(n\Delta)^*$	$O(m)$	0	$O(m)$	0
Shortest	$\mathcal{B}$	Δ	$O(n\Delta)$	$O(m)$	$O(n)$: $2n-2$	$O(n)$	0
either of {		$n\&\Delta$	$O(n\Delta)$	$O(m)$	$O(n)$: $n-1$	$O(n)$	0
		$n\&\Delta$	$O(n\Delta)^*$	$O(m)$	0	$O(m)$	0

Table 2. Complexity of TDB in different classes of dynamic networks with associated knowledge. *(The * indicates that the emitter terminates implicitely, even in the first run.)*

all edges and dates, and call ζ the *crossing delay*; thus we use the shorthand notations $\mathcal{G} = (V, E, \mathcal{T}, \rho)$. We also assume that, for every edge $e \in E$, the union of dates when $\rho(e) = 1$, is a set of disjoint closed time intervals of length at least ζ. Finally, the (static) graph formed by V and E, taken alone, is the *footprint* of $\mathcal{G}$ (also called *underlying graph* or *interaction graph*). In this work, we do not consider the footprint to be a complete graph in general (some nodes may never interact), but we consider it to be connected.

The TVG formalism essentially encompasses that of *evolving graphs* [17], where $\mathcal{G}$ is represented as a sequence of graphs $G_1, G_2, ..., G_i, ...$ each providing a *snapshot* of the system at different times (which correspond either to discrete steps or given dates). In comparison, TVGs offer an *interaction-centric* view of the network evolution, where the evolution of each edge can be considered irrespective of the global time sequence.

A graph G is said to be *recurrent* if none of the edges E can disappear forever; that is, for any date t and edge e, $\rho(e,t) = 0 \implies \exists t' > t : \rho(e,t') = 1$. Strictly speaking, we do not say that an edge must appear infinitely often because here an edge might also remain present continuously (and this would satisfy the property). Let $\mathcal{R}$ denote the class of recurrent TVGs whose footprint $G = (V, E)$ is *connected*.

A graph $\mathcal{G} \in \mathcal{R}$ is said to be *time-bounded recurrent* (or simply *bounded*), if there exists a constant Δ such that, for every edge $e \in E$, the time between two successive appearances of e is at most Δ. We denote by $\mathcal{B} \subset \mathcal{R}$ the class of time-bounded recurrent TVGs whose footprint is connected.

A graph is said to be *periodic* if there exists a constant p such that $\forall e \in E$, $\rho(e,t) = \rho(e, t+kp)$ for every positive integer k; the smallest such p is called the *period* of the graph. We denote by $\mathcal{P} \subset \mathcal{B}$ the class of periodic TVGs whose footprint is connected.

Given a TVG $\mathcal{G} = (V, E, \mathcal{T}, \rho)$, we consider that $G = (V, E)$ is always simple (no self-loop nor multiple edges) and that nodes possess unique identifiers.

The set of edges being incident to a node u at time t is noted $I_t(u)$ (or simply I_t, when

the node is implicit).

When an edge $e = (x, y)$ appears, the entities x and y can communicate. The time ζ necessary to transmit a message (*crossing delay*) is known to the nodes. The duration of edge presence is assumed to be at least ζ (*i.e.,* long enough to send a message). Algorithmically, this allows the following observations:

Property 1.
1. If a message is sent just after an edge has appeared, the message is guaranteed to be successfully transmitted.
2. If the recurrence of an edge is bounded by some Δ, then this edge cannot disappear for more than $\Delta - \zeta$.

The appearance and disappearance of edges are instantly detected by the two adjacent nodes (they are notified of such an event without delay). If a message is sent less than ζ before the disappearance of an edge, it is lost. However, since the disappearance of an edge is detected instantaneously, and the crossing delay ζ is known, the sending node can locally determine whether the message was successfully delivered. We thus authorize the special primitive $send\_retry$ as a facility to specify that if the message is lost, then it is automatically re-sent upon next appearance of the edge, and this sending is necessarily successful (Property 1). Note that nothing precludes this primitive to be called while the corresponding edge is even absent (this actually simplifies the expression of some algorithms).

A sequence of couples $\mathcal{J} = \{(e_1, t_1), (e_2, t_2), ..., (e_k, t_k)\}$, with $e_i \in E$ and $t_i \in \mathcal{T}$ for all i, is called a *journey* in $\mathcal{G}$ iff $\{e_1, e_2, ...\}$ is a walk in G and for all t_i, $t_{i+1} \geq t_i + \zeta$ and $\rho(e_i)_{t'} = 1$ for all $t_i \leq t' \leq t_i + \zeta$. We denote by $departure(\mathcal{J})$, and $arrival(\mathcal{J})$, the starting date t_1 and last date $t_k + \zeta$ of $\mathcal{J}$, respectively.

Journeys can be thought of as *paths over time* from a source node to a destination node (if the journey is finite). Let us denote by $\mathcal{J}^*_{\mathcal{G}}$ the set of all finite journeys in a graph $\mathcal{G}$. We will say that node u can reach node v in $\mathcal{G}$, and note $\exists \mathcal{J}_{(u,v)} \in \mathcal{J}^*_{\mathcal{G}}$ (or simply $u \rightsquigarrow v$, if $\mathcal{G}$ is clear from the context), if there exists at least one possible journey from u to v in $\mathcal{G}$. Note that the notion of journey is asymmetrical ($u \rightsquigarrow v \not\Leftrightarrow v \rightsquigarrow u$), regardless of whether edges are directed or undirected.

Because journeys take place *over time*, they have both a topological length and a temporal length. The *topological length* of $\mathcal{J}$ is the number $|\mathcal{J}|_h = k$ of couples in $\mathcal{J}$ (i.e., number of *hops*), and its *temporal length* is its duration $|\mathcal{J}|_t = arrival(\mathcal{J}) - departure(\mathcal{J}) = t_k - t_1 + \zeta$. This yields two distinct definitions of distance in a graph $\mathcal{G}$:

- The *topological distance* from a node u to a node v at time t, noted $d_{u,t}(v)$, is defined as $Min\{|\mathcal{J}|_h : \mathcal{J} \in \mathcal{J}^*_{(u,v)} \wedge departure(\mathcal{J}) \geq t\}$. For a given date t, a journey whose departure is $t' \geq t$ and topological length is equal to $d_{u,t}(v)$ is called *shortest*;
- The *temporal distance* from u to v at time t, denoted by $\hat{d}_{u,t}(v)$ is defined as $Min\{arrival(\mathcal{J}) : \mathcal{J} \in \mathcal{J}^*_{(u,v)} \wedge departure(\mathcal{J}) \geq t\} - t$. Given a date t, a journey whose departure is $t' \geq t$ and arrival is $t + \hat{d}_{u,t}(v)$ is called *foremost*; if the set $\hat{d}_{u,t'}(v)$ has a minimum, say d', any journey whose temporal distance is d' is called the *fastest*.

Informally, a *foremost* journey is one that minimizes the date of arrival at destination; a *shortest* journey is one that uses the least number of hops; and a *fastest* journey is one that minimizes the time spent between departure and arrival (however late the departure be) [7].

2.2. *Problems*

We consider the *distributed* problem of performing *broadcast with termination detection* at the emitter, or TDB, according to the shortest, fastest, or foremost metrics.

TDB in general requires all nodes to receive a message with some information initially held by a single node x, called *source* or *emitter*, and the source to enter a terminal state after all nodes have received the information, within finite time. A protocol solves TDB in a graph $\mathcal{G}$ if it solves it for any source $x \in V$ and time $t \in \mathcal{T}$. We say that it solves TDB in a class $\mathcal{C}$ if it solves TDB for any $\mathcal{G} \in \mathcal{C}$. We are interested in three variations of this problem, following the optimality metrics defined above:

- TDB$[foremost]$, where *each* node receives the information at the *earliest* possible date following its *creation* at the emitter;
- TDB$[shortest]$, where each node receives the information within a minimal number of hops from the emitter;
- TDB$[fastest]$, where the overall duration between first global emission and last global reception is minimized.

For each of these problems, we require that the emitter detects termination, however this detection is not subject to the same optimality constraint (it just has to be finite). TDB thus involves two processes: the actual dissemination of *information messages*, and the exchange of typically smaller *control messages* used for termination detection, both being considered separately in this paper.

Finally, we call *broadcast tree* the hierarchy of nodes and edges along which the broadcast takes place, without consideration to the dates when the edges are used (*i.e.* the footprint of the part of the TVG that is used). A broadcast tree is said to be *reusable* if the same hierarchy of nodes and edges can be purposedly followed to perform a subsequent optimal (*i.e.* foremost, shortest, or fastest) broadcast.

3. Basic Results and Limitations

Let us first state a general property on the computational relationship between the main three contexts of interest, namely knowing n in $\mathcal{R}$ (noted $\mathcal{R}_n$), knowing Δ in $\mathcal{B}$ (noted $\mathcal{B}_\Delta$), and knowing p in $\mathcal{P}$ (noted $\mathcal{P}_p$). Let $\mathscr{P}(\mathcal{C}_K)$ denote the set of problems solvable in every $\mathcal{G} \in \mathcal{C}$ with knowledge K.

Theorem 2. $\mathscr{P}(\mathcal{R}_n) \subseteq \mathscr{P}(\mathcal{B}_\Delta) \subseteq \mathscr{P}(\mathcal{P}_p)$

Proof. The right inclusion is straight from the fact that $\mathcal{B} \subset \mathcal{P}$ and p is a valid bound Δ on the recurrence time. The left inclusion follows from the facts that $\mathcal{R} \subset \mathcal{B}$ and n can be inferred in $\mathcal{B}$ if Δ is already known. This can be done by performing, from any

node (say u), a depth-first token circulation that will explore the underlying graph G over time. Having a bounded recurrence time indeed allows every node to learn the list of its neighbors in G within Δ time (all incident edges must appear within this duration). As the token is circulated to unvisited nodes, these nodes are marked as visited by u's token and the token is incremented. Upon returning to u, the token value is n. □

These inclusions will be shown strict later on.

We now establish a negative result that justifies the need for additional knowledge in order to solve TDB in any of the considered contexts. In fact we have:

Theorem 3. TDB *cannot be solved in $\mathcal{P}$ without additional knowledge.*

Proof. By contradiction, let $\mathcal{A}$ be an algorithm that solves TDB in $\mathcal{P}$. Consider an arbitrary $\mathcal{G} = (V, E, \mathcal{T}, \rho) \in \mathcal{P}$ and $x \in V$. Execute $\mathcal{A}$ in $\mathcal{G}$ starting at time t_0 with x as the source. Let t_f be the time when the source terminates (and thus all nodes have received the information). Let $\mathcal{G}' = (V', E', \mathcal{T}', \rho') \in \mathcal{P}$ such that $V' = V \cup \{v\}$, $E' = E \cup \{(u, v)$ for some $u \in V\}$, for all $t_0 \leq t < t_f$, $\rho'(e, t) = \rho(e, t)$ for all $e \in E$ and $\rho'((u, v), t) = 0$. Now, consider $\rho'((u, v), t) = 1$ for some $t > t_f$, and the period of $\mathcal{G}'$ is some $p' > t - t_0$. Consider the execution of $\mathcal{A}$ in $\mathcal{G}'$ starting at time t_0 with x as the source. Since (u, v) does not appear from t_0 to t_f, the execution of $\mathcal{A}$ at every node in $\mathcal{G}'$ is exactly as at the corresponding node in $\mathcal{G}$. In particular, node x will have entered a terminal state at time t_f with node v not having received the information, contradicting the correctness of $\mathcal{A}$. □

We thus have the following corollary, by inclusion of $\mathcal{P}$.

Corollary 4. TDB *cannot be solved in $\mathcal{B}$ nor $\mathcal{R}$ without any additional knowledge.*

Hence, additional knowledge of some kind is required to solve TDB in these classes. We consider three types of knowledge, namely, the number of nodes $n = |V|$, an upper bound Δ on the recurrence time (when in $\mathcal{B}$), or the period p (in $\mathcal{P}$).

To prove some impossibility results on these problems with a given knowledge (Theorems 12 and 16, later in the paper), we make use of a specific family of TVGs with the same footprint, and establish some basic limitations they expose.

Consider the graph $G = (V, E)$ with $V = \{u, v, x, y\}$ and $E = \{e_1 = (u, x), e_2 = (u, y), e_3 = (x, v), e_4 = (y, v)\}$. Consider the infinite family $\{\mathcal{G}_i\} = \{\mathcal{G}_0, \mathcal{G}_1, \ldots\}$ of periodic TVGs with footprint G where, for each $\mathcal{G}_i$, $\zeta = 1$ and ρ is as follows, where $[t]_j$ denotes t modulo j:

- e_1 and e_2 are present only during the intervals $[t, t + 1]$ with $t \in \mathbb{N}$ and $[t]_4 = 0$;
- e_3 is present only during the intervals $[t, t + 1]$ where $t \in \mathbb{N}$ and $[t]_4 = 2$;
- If $i = 0$, then e_4 is present only during the intervals $[t, t + 1]$ where $t \in \mathbb{N}$ and $[t]_4 = 3$.
- If $i > 0$, then e_4 is present only during the intervals $[t, t + 1]$ where $t \in \mathbb{N}$ and either $[t]_4 = 3$ or $[t]_{4(i+1)} = 4i + 1$.

Notice that $\mathcal{G}_i \in \mathcal{P}$ and its period is $4(i+1)$. Also notice that in all of these graphs, $n = 4$ and the minimum Δ is 4.

Let TEST($\{\mathcal{G}_i\}$) be the problem of observing the evolution of a graph chosen by an adversary from $\{\mathcal{G}_i\}$, and deciding in finite time whether or not it is $\mathcal{G}_0$.

Theorem 5. TEST*($\{\mathcal{G}_i\}$) is undecidable.*

Proof. At any time t, the evolution of $\mathcal{G}_0$ from time 0 to t is indistinguishable from that of any $\mathcal{G}_i$ with $i > \frac{t-1}{4}$. Since any solution algorithm must terminate in finite time, any decision taken at that time, say $\hat{t}$, can be made incorrect by the adversary by choosing the graph to be any $\mathcal{G}_i$ with $i > \frac{\hat{t}-1}{4}$ if the answer was $\mathcal{G}_0$, and by choosing it to be $\mathcal{G}_0$ otherwise. □

A knowledge K about $\{\mathcal{G}_i\}$ may reduce the choices of the adversary. Indeed, given enough knowledge, the TEST problem can become decidable. The following lemma identifies conditions on K for TEST to be undecidable.

Let $\{\mathcal{G}_i\} \setminus K$ denote the subset of $\{\mathcal{G}_i\}$ still available to the adversary in spite of knowledge K.

Theorem 6. *If $\mathcal{G}_0 \in \{\mathcal{G}_i\} \setminus K$ and $|\{\mathcal{G}_i\} \setminus K| = \infty$, then* TEST$(\{\mathcal{G}_i\} \setminus K)$ *is undecidable.*

Proof. Since $|\{\mathcal{G}_i\} \setminus K| = \infty$, for any t there is always a TVG with $i > \frac{t-1}{4}$ in the set $\{\mathcal{G}_i\} \setminus K$, and its evolution from time 0 to t is indistinguishable from that of $\mathcal{G}_0$. Since any solution algorithm must terminate in finite time, any decision taken at that time, say $\hat{t}$, can be made incorrect by the adversary by choosing the graph to be any $\mathcal{G}_j \in \{\mathcal{G}_i\} \setminus K$ with $j > \frac{\hat{t}-1}{4}$ if the answer was $\mathcal{G}_0$, and by choosing it to be $\mathcal{G}_0$ otherwise.

□

As an immediate consequence, we have that n and Δ are not sufficient:

Lemma 7. TEST$(\{\mathcal{G}_i\} \setminus \{n, \Delta\})$ *is undecidable.*

Proof. Let n and Δ be known. Since n and Δ are the same for every $\mathcal{G}_i$, then $\{\mathcal{G}_i\} \setminus n = \{\mathcal{G}_i\} \setminus \Delta = \{\mathcal{G}_i\} \setminus \{n, \Delta\} = \{\mathcal{G}_i\}$. Thus, by Theorem 5, the lemma follows. □

4. TDB[$foremost$]

Solving TDB[$foremost$] in $\mathcal{R}$ or $\mathcal{B}$ clearly requires some sort of flooding, because the very fact of probing a neighbor to determine if it already has the information compromises the possibility to send it in a foremost fashion (in addition to risking the disappearance of the edge in-between the probe and the real sending). As a consequence of Theorem 3, this problem cannot be solved without knowledge. In this section we first show that it becomes possible in $\mathcal{R}$ if the number of nodes $n = |V|$ is known. The proof is constructive by means of Algorithm 1, whose termination is however not bounded in time. Being in $\mathcal{B}$ with the

same knowledge allows its termination to be bounded. Knowing Δ instead of n in $\mathcal{B}$ then allows us to propose another solution (described in Algorithm 2) that has a lower message complexity. This complexity can be further improved if both Δ *and* n are known, as in this case we have the possibility to terminate *implicitly*.

4.1. TDB$[foremost]$ *in* $\mathcal{R}$

Since Δ and p are not defined for $\mathcal{R}$, we need to focus only on the knowledge of n. We show that the problem is solvable when n is known.

The algorithm proceeds as follows (see Algorithm 1 for details). Every time a *new* edge appears locally to an informed node, this node sends the information message onto this edge, and remembers that this edge now leads to an informed node. The first time a node receives the information, it records the sender as parent, transmits the information on its available edges, and sends back a notification message to the parent. Note that these notifications create a parent-relation and thus a converge-cast tree. Each notification is propagated along the converge-cast tree and eventually collected at the emitter. When the emitter has received $n-1$ notifications, it knows all nodes are informed. Observe that the notification messages are sent using the special primitive $send\_retry$ discussed in Section 2.1, to ensure that the parent eventually receives it even if the edge disappears during the first attempt. Information messages, on the other hand, are sent using the normal $send$ primitive. Indeed, if the propagation of such a message fails because the corresponding edge disappears, it simply means that this edge at that particular time did not have to be used (i.e., it did not belong to a valid journey).

Theorem 8. *When n is known,* TDB$[foremost]$ *can be solved in $\mathcal{R}$ exchanging $O(m)$ information messages and $O(n^2)$ control messages, in unbounded time.*

Proof. Since a node sends the information to each new appearing edge, it is easy to see, by connectivity of the *underlying* graph, that all nodes will eventually receive the information. The dissemination itself is necessarily foremost because the information is either directly relayed on edges that are present, or sent as soon as a new edge appears. As for termination detection: every node identifies a unique parent and a converge-cast spanning tree directed towards the source is implicitly constructed; since every node notifies the source (through the tree) and the source knows the total number of nodes, termination is guaranteed. Since information messages might traverse every edge in both directions, and an edge cannot be traversed twice in the same direction, we have that the number of *information* messages is in the worst case $2m$. Since every node but the emitter induces a notification that is forwarded up the converge-cast tree to the emitter, the number of *notification* messages is the sum of distances in the converge-cast tree between all nodes and the emitter, $\sum_{v \in V \setminus \{emitter\}} d_{h\_tree}(v, emitter)$. The worst case is when the graph is a line where we have $\frac{n^2-n}{2}$ control messages. Regarding time complexity, the termination of the algorithm is unbounded due to the fact that the recurrence of the edges is itself unbounded. □

Algorithm 1 Foremost broadcast in $\mathcal{R}$, knowing n.

```
1:  Edge parent ← nil                    // edge the information was received from (for non-emitter nodes).
2:  Integer nbNotifications ← 0          // number of notifications received (for the emitter).
3:  Set<Edge> informedNeighbors ← ∅      // neighbors known to have the information.
4:  Status myStatus ← ¬informed          // status of the node (informed or non-informed).
5:  initialization:
6:    if isEmitter() then
7:      myStatus ← informed
8:      send(information) on I_now()     // sends the information on all present edges.
9:  onAppearance of an edge e:
10:   if myStatus == informed and e ∉ informedNeighbors then
11:     send(information) on e
12:     informedNeighbors ← informedNeighbors ∪ {e}     // (see Prop. 1).
13: onReception of a message msg from an edge e:
14:   if msg.type == Information then
15:     informedNeighbors ← informedNeighbors ∪ {e}
16:     if myStatus == ¬informed then
17:       myStatus ← informed
18:       parent ← e
19:       send(information) on I_now() \ informedNeighbors     // propagates.
20:       send_retry(notification) on e     // notifies that this node has the info.
                (this message is to be resent upon the next appearance, in case of failure).
21:   else if msg.type == Notification then
22:     if isEmitter() then
23:       nbNotifications ← nbNotifications + 1
24:       if nbNotifications == n − 1 then
25:         terminate     // at this stage, the emitter knows that all nodes are informed.
26:     else
27:       send_retry(notification) to parent
```

Use in subsequent broadcasts. Foremost trees are *time-dependent* in the sense that they might be optimal for some emission dates and not be so for other dates. Still, they remain *valid* trees (though, possibly non-foremost ones) regardless of the considered date. As such, they can be memorized by the nodes in order to be used as *converge-cast* trees for termination detection in subsequent broadcasts. Indeed, while the broadcast is required to be foremost, the detection of termination does not have such constraint. Hence, instead of sending a notification each time a new node is informed (as done previously), nodes can notify their parents (in the converge-cast tree) if and only if they are themselves informed and have received a notification from each of their children (in the converge-cast tree). This reduces the number of control messages from $O(n^2)$ to $O(n)$, having only one notification per edge of the converge-cast tree.

4.2. TDB[$foremost$] *in* $\mathcal{B}$

If the recurrence time is bounded, then either the knowledge of n or an upper bound Δ on the recurrence time can be used to solve the problem (with various message complexities).

4.2.1. *Knowledge of* n.

Since $\mathcal{B} \subseteq \mathcal{R}$, one can obviously solve TDB[$foremost$] in $\mathcal{B}$ using Algorithm 1 (and the same observations apply regarding reusability of the converge-cast tree). Here, however, the termination time becomes bounded due to the fact that the recurrence of edges is itself bounded.

Theorem 9. *When n is known,* TDB[$foremost$] *can be solved in $\mathcal{B}$ exchanging $O(m)$ information messages and $O(n^2)$ control messages, in $O(n\Delta)$ time.*

Proof. Since all edges in E are recurrent within any Δ time window, the delivery of the information at the last node must occur within $(n-1)\Delta$ global time. The same property holds for the latest notification, bounding the overall process to a duration of $\Delta(2n-2)$. The rest follows from Theorem 8. □

4.2.2. *Knowledge of* Δ.

The information dissemination is performed as in Algorithm 1, but the termination detection is different. Thanks to the time-bound Δ on edge recurrence, a node can discover all of its neighbors within 2Δ time (back and forth messages, each on a different Δ period at the worst). This fact can be used by a node to determine whether it is a leaf in the broadcast tree (*i.e.*, if it has not heard back from a potential child within 2Δ time following its own reception time). This allows the leaves to terminate spontaneously and notify their parent, which recursively terminate after receiving the notifications from all their children and notifying their own parent. This combination of broadcast-convergecast, originally described in [15], is a standard process for distributed computing in *static* graphs. The temporal adaption for bounded-recurrent TVGs is as follows (see Algorithm 2 for details). First, everytime a *new* edge appears locally to an informed node, this node sends the information on the edge, and records the edge. The first time a node receives the information, it chooses the sender as parent, memorizes the current time (in a variable $firstRD$), transmits the information on its available edges, and returns an *affiliation* message to its parent using the $send\_retry$ primitive (starting to build the converge-cast tree). This affiliation message is not relayed further up the tree: it is only intended to inform the parent about the existence of a new child (so this parent knows it must wait for a future notification by this node). If an informed node has not received any affiliation message after a duration of 2Δ, then it sends a *notification* message to its own parent using the $send\_retry$ primitive.

As for Algorithm 2, if the information message is lost, then it simply means that this edge at that time did not have to be used. On the other hand, if the *affiliation* message is lost, it must be sent again ($send\_retry$). However, in the worst case, the common edge

disappears just before the affiliation message is delivered, and reappears less than $\Delta - \zeta$ later (Prop. 1). The overall back and forth exchange thus remains within 2Δ time.

If a node has one or more children, it waits until it receives a notification message from each of them, then notifies its parent in the converge-cast tree (using $send\_retry$ again). Once the emitter has received a notification from each of its children, it knows that all nodes are informed.

Algorithm 2 Foremost broadcast in $\mathcal{B}$, knowing a bound Δ on the recurrence time.

1: $Edge\ parent \leftarrow nil$ *// edge the information was received from (for non-emitter nodes).*
2: $Integer\ nbChildren \leftarrow 0$ *// number of children.*
3: $Integer\ nbNotifications \leftarrow 0$ *// number of children that have terminated.*
4: $Set<Edge>\ informedNeighbors \leftarrow \emptyset$ *// neighbors known to have the information.*
5: $Date\ firstRD \leftarrow nil$ *// date of first reception.*
6: $Status\ myStatus \leftarrow \neg$`informed` *// status of the node (informed or non-informed).*
7: ***initialization***:
8: **if** $isEmitter()$ **then**
9: $myStatus \leftarrow$ `informed`
10: $send(information)\ on\ I_{now()}$ *// sends the information on all present edges.*
11: ***onAppearance*** of an edge e:
12: **if** $myStatus ==$ `informed` **and** $e \notin informedNeighbors$ **then**
13: $send(information)\ on\ e$
14: $informedNeighbors \leftarrow informedNeighbors \cup \{e\}$ *// (see Prop. 1).*
15: ***onReception*** of a message msg from an edge e:
16: **if** $msg.type == Information$ **then**
17: $informedNeighbors \leftarrow informedNeighbors \cup \{e\}$
18: **if** $myStatus == \neg$`informed` **then**
19: $myStatus \leftarrow$ `informed`
20: $firstRD \leftarrow now()$ *// memorizes the date of first reception.*
21: $parent \leftarrow e$
22: $send(information)\ on\ I_{now()} \setminus informedNeighbors$ *// propagates.*
23: $send\_retry(affiliation)\ on\ e$ *// informs the parent that it has a new child.*
24: **else if** $msg.type == Affiliation$ **then**
25: $nbChildren \leftarrow nbChildren + 1$
26: $informedNeighbors \leftarrow informedNeighbors \cup \{e\}$
27: **else if** $msg.type == Notification$ **then**
28: $nbNotifications \leftarrow nbNotifications + 1$
29: **if** $nbNotifications == nbChildren$ **then**
30: **if** $\neg isEmitter()$ **then**
31: $send\_retry(notification)\ to\ parent$ *// notifies the parent in turn.*
32: $terminate$ *// whether emitter or not, the node has terminated at this stage.*
33: ***when*** $now() == firstRD + 2\Delta$: *// tests whether the underlying node is a leaf.*
34: **if** $nbChildren == 0$ **then**
35: $send\_retry(notification)\ on\ parent$
36: $terminate$

Theorem 10. *When Δ is known,* TDB$[foremost]$ *can be solved in $\mathcal{B}$ exchanging $O(m)$ information messages and $O(n)$ control message, in $O(n\Delta)$ time.*

Proof. Correctness follows the same lines of the proof of Theorem 8. However the correct construction of a converge-cast spanning tree is guaranteed by the knowledge of Δ (i.e., the nodes of the tree that are leaves detect their status because no new edges appear within Δ time) and the notification starts from the leaves and is aggregated before reaching the source. The number of information messages is $O(m)$ as the exchange of information messages is the same as in Algorithm 1, but the number of notification and affiliation messages decreases to $2(n-1)$. Each node but the emitter sends a single affiliation message; as for the notification messages, instead of sending a notification as soon as it is informed, each node notifies its parent in the converge-cast tree if and only if it has received a notification from each of its children resulting in one notification message per edge of the tree. The time complexity of the dissemination itself is the same as for the foremost broadcast when n is known ($(n-1)\Delta$). The time required for the emitter to subsequently detect termination is an additional $2\Delta + (n-1)\Delta$ in the worst case (*i.e.* self-detection by the leaves, followed by the longest notification chain up the emitter). This gives a total of $2n\Delta$. □

Clearly, the number of nodes n, which is not *a priori* known here, can be obtained through the notification process of the first broadcast (by having nodes reporting their number of descendants in the tree, while notifying hierarchically). All subsequent broadcasts can thus behave as if both n and Δ were known. Next we show this allows to solve the problem without any control messages.

4.2.3. *Knowledge of both n and Δ*

In this case, the emitter knows an upper bound on the broadcast termination date; in fact, the broadcast cannot last longer than $(n-1)\Delta$ (this worst case is when the foremost tree is a line). Termination detection can thus become implicit after this amount of time, which removes the need for any control message (whether of affiliation or of notification).

Theorem 11. *When Δ and n are known,* TDB$[foremost]$ *can be solved in $\mathcal{B}$ exchanging $O(m)$ information messages and no control messages, in $O(n\Delta)$ time.*

4.3. *Reusability*

As we have seen, TDB$[foremost]$ is feasible even in $\mathcal{R}$ with knowledge n. However our algorithms in $\mathcal{R}_n$ or even in $\mathcal{B}_{\{n,\Delta\}}$ do not provide reusability. This is not accidental; in fact, we will now show that achieving reusability in TDB$[foremost]$ is *impossible* even in $\mathcal{P}_{\{n,\Delta\}}$.

Theorem 12. *Foremost broadcast trees are not reusable in $\mathcal{P}_{\{n,\Delta\}}$.*

Proof. Consider the infinite family $\{\mathcal{G}_i\}$ of TVGs defined in Section 3. In $\mathcal{G}_0$ the foremost journey from u to v at time t is always along the edges e_1 and e_3. In $\mathcal{G}_i$, $i > 0$, the foremost

journey from u to v at time $t > 0$ with $[t]_{4i} = 0$ is also along the edges e_1 and e_3, *except* when $[t]_{4(i+1)} = 0$, in which case the foremost journey is along the edges e_2 and e_4. By Lemma 7, even knowing n and Δ, the evolution of $\mathcal{G}_0$ until time t with $[t]_{4i} = 0$ is undistinguishable from that of $\mathcal{G}_i$ with $i \geq \frac{t}{4}$. Then, by just observing the evolution until time t with $[t]_{4i} = 0$, it is impossible to decide at that time whether the graph is $\mathcal{G}_0$ or not; hence, it is impossible to decide whether the foremost path from u to v at that time is $\{e_1, e_3\}$ (foremost in $\mathcal{G}_0$) or $\{e_2, e_4\}$ (foremost in $\mathcal{G}_i$ with $i \geq \frac{t}{4}$). □

While n and Δ are not sufficient for reusability in $\mathcal{P}$, it has been shown in [10] that the knowledge of the period p is sufficient.

Theorem 13. [10] *Foremost broadcast trees can be reused for subsequent broadcasts in $\mathcal{P}$ with knowledge p.*

The basic argument is that a foremost broadcast tree for time t remains optimal for all times $t + jp$, where j is a positive integer.

5. TDB[$shortest$]

Recall that the objective of TDB[$shortest$] is to deliver the information to each node within a minimal *number of hops* from the emitter, and to have the emitter detect termination within finite time. We show below that contrary to the foremost case, knowing n is insufficient to perform a shortest broadcast in $\mathcal{R}$ or even in $\mathcal{B}$. This becomes however feasible in $\mathcal{B}$ when Δ is also known. Moreover any shortest tree built at some time t will remain optimal in $\mathcal{B}$ relative to any future emission date $t' > t$. This feature allows the solution to TDB[$shortest$] to be possibly reused in subsequent broadcasts.

5.1. TDB[$shortest$] *in* $\mathcal{B}$

As we will show later (see Theorem 16), knowing n is not sufficient to solve TDB[$shortest$] in $\mathcal{P}$, and thus also in $\mathcal{B}$ (or $\mathcal{R}$). Therefore, the only knowledge still of interest are Δ and the combination of Δ and n.

5.1.1. *Knowledge of* Δ

The idea is to propagate the message along the edges of a *breadth*-first spanning tree of the underlying graph. We present the pseudo-code in Algorithm 3, and provide the following informal description.

Assuming that the message is created at some date t, the mechanism consists of authorizing nodes at level i in the tree to inform new nodes only between time $t + i\Delta$ and $t + (i + 1)\Delta$ (doing it sooner would lead to a non-shortest tree, while doing it later is pointless because all the edges have necessarily appeared within one Δ). So the broadcast is confined into rounds of duration Δ as follows: whenever a node sends the information to another, it sends a time value that indicates the remaining duration of its round (that is, the starting date of its own round minus the current time, plus Δ, minus the crossing delay

ζ), so the receiving node, if it is a new child, knows when it should start informing new nodes in its turn. For instance in Figure 1 when the node a attempts to become b's parent, node a transmits its own starting date plus Δ minus the current date minus ζ. This duration corresponds to the exact amount of time the child would have to wait, if the relation is established, before integrating other nodes in turn. If a node does not detect any children before 2Δ following its own reception date (same as Algorithm 2), then it detects that it is a leaf and notifies its parent. Otherwise, it waits for the final notifications of all its children, then notifies its parent. As before, this requires parents to keep track of the number of children they have, and thus children need to send *affiliation* messages when they select a parent. Finally, when the emitter has been notified by all its children, it knows the broadcast is terminated.

Theorem 14. $\mathrm{TDB}[shortest]$ *can be solved in* $\mathcal{B}$ *knowing* Δ*, exchanging* $O(m)$ *info. messages and* $O(n)$ *control messages, in* $O(n\Delta)$ *time.*

Proof. The fact that the algorithm constructs a breadth-first (and thus shortest) delay-tolerant spanning tree follows from the connectivity over time of the underlying graph and from the knowledge of the duration Δ. The bound on recurrence is used to enable a rounded process whereby the correct distance of each node to the emitter is detected. The number of *information* messages is $2m$ as the dissemination process exchanges at most two messages per edge. The number of *affiliation* and *notification* messages are each of $n-1$ (one per edge of the tree). The time complexity for the construction of the tree is at most $(n-1)\Delta$ to reach the last node, plus 2Δ at this node, plus at most $(n-1)\Delta$ to aggregate this node's notification. (The additional ζ caused by waiting affiliation messages matters only for the last round, since the construction continues in parallel otherwise.) The total is thus at most $2n\Delta$. □

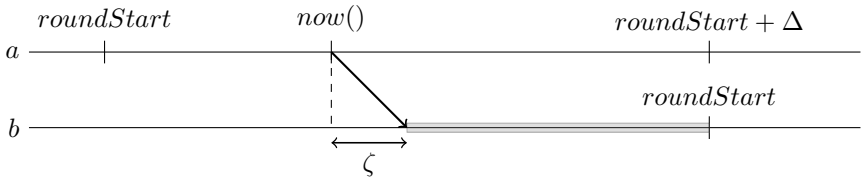

Figure 1. Propagation of the rounds of duration Δ.

Algorithm 3 Shortest broadcast in $\mathcal{B}$, knowing a bound Δ on the recurrence.

1: $Edge\ parent \leftarrow nil$ *// edge the information was received from (for non-emitter nodes).*

2: $Date\ roundStart \leftarrow +\infty$ *// date when this node starts informing new nodes.*
3: $Set <Edge>\ children \leftarrow \emptyset$ *// set of children from which a notification is expected.*
4: $Integer\ nbNotifications \leftarrow 0$ *// number of children that have sent their notification.*
5: $Set<Edge>\ informedNeighbors \leftarrow \emptyset$ *// set of neighbors known to have the info.*
6: $Status\ myStatus \leftarrow \neg$`informed` *// status of the node (informed or non-informed).*

7: ***initialization***:
8: **if** $isEmitter()$ **then**
9: $roundStart \leftarrow now()$

10: ***onAppearance*** of an edge e:
11: **if** $myStatus ==$ `informed` **then**
12: **if** $e \notin informedNeighbors$ **then**
13: $send(roundStart - now() + \Delta - \zeta)$ *on* e *// time until end of round.*
14: $informedNeighbors \leftarrow informedNeighbors \cup \{e\}$ *// (see Prop. 1).*

15: ***onReception*** of a message msg from an edge e:
16: **if** $msg.type == Duration$ **then**
17: $informedNeighbors \leftarrow informedNeighbors \cup \{e\}$
18: **if** $parent == nil$ **then**
19: $parent \leftarrow e$
20: $roundStart \leftarrow now() + msg$
21: $send\_retry(affiliation)$ *on* e
22: **else if** $msg.type == Affiliation$ **then**
23: $children \leftarrow children \cup \{e\}$
24: **else if** $msg.type == Notification$ **then**
25: $nbNotifications \leftarrow nbNotifications + 1$
26: **if** $nbNotifications == |children|$ **then**
27: **if** $\neg isEmitter()$ **then**
28: $send\_retry(notification)$ *on* $parent$
29: $terminate$

30: ***when*** $now() == roundStart$:
31: $myStatus \leftarrow$ `informed`
32: $send(\Delta\text{-}\zeta)$ *on* $I_{now()} \setminus informedNeighbors$ *// nodes that receive this and have no parent yet will take this node as parent and wait* Δ*-*ζ *before informing new nodes.*

33: ***when*** $now() == roundStart + 2\Delta$: *// tests whether the underlying node is a leaf.*
34: **if** $|children| == 0$ **then**
35: $send\_retry(notification)$ *on* $parent$

Reusability for subsequent broadcasts: Thanks to the fact that shortest trees remain shortest regardless of the emission date, all subsequent broadcasts can be performed within the same, already known tree, which reduces the number of information message from $O(m)$ to $O(n)$. Moreover, if the depth d of the tree is detected through the first notification process, then all subsequent broadcasts can enjoy an implicit termination detection that is itself optimal in time (after $d\Delta$ time). No control message is needed.

5.1.2. *Knowledge of n and Δ*

When both n and Δ are known, one can apply the same dissemination procedure as in Algorithm 3 combined with an implicit termination detection that avoids using control messages at all. Indeed, each node learns (and possibly informs) all of its neighbors within Δ time. Since the underlying graph is connected, the whole process must then complete within $n\Delta$ time. Hence, if the emitter knows both Δ and n, it can simply wait $n\Delta$ time, then terminate implicitly.

Theorem 15. *When n and Δ are known,* TDB$[shortest]$ *can be solved in $\mathcal{B}$ exchanging $O(m)$ info. messages and no control messages, in $O(n\Delta)$ time.*

However, such a strategy would prevent the emitter from learning the depth d of the shortest tree, and thus prevent lowering the termination bound to $d\Delta$ time. An alternative solution would be to achieve explicit termination for the first broadcast in order to build a reusable broadcast tree (and learn its depth d in the process). In this case, dissemination is achieved with $O(m)$ information messages, termination detection is achieved similarly to Algorithm 3 with $O(n)$ control messages (where however affiliation messages are not necessary, and the number of control messages would decrease to $n-1$). In this way we would have an increase in control messages, but the subsequent broadcasts could reuse the broadcast tree for dissemination with $O(n)$ information messages, and termination detection could be implicit with no exchange of control message at all after $d\Delta$ time. The choice of either solution may depend on the size of an information message and on the expected number of broadcasts planned.

5.2. TDB$[shortest]$ *in $\mathcal{P}$*

Feasibility in $\mathcal{P}_\Delta$ is implied by feasibility in $\mathcal{B}_\Delta$ and it clearly implies feasibility in $\mathcal{P}_p$. The only case left to study is feasibility with only knowledge of n.

Theorem 16. TDB$[shortest]$ *is not feasible in $\mathcal{P}$ knowing only n.*

Proof. By contradiction, let $\mathcal{A}$ be an algorithm that solves TDB$[shortest]$ in $\mathcal{P}$ with the knowledge of n only. Consider an arbitrary $\mathcal{G} = (V, E, \mathcal{T}, \rho) \in \mathcal{P}$ and $x \in V$. Execute $\mathcal{A}$ in $\mathcal{G}$ starting at time t_0 with x as the source. Let t_f be the time when the source terminates and T the shortest broadcast tree along which broadcast was performed. Let $\mathcal{G}' = (V', E', \mathcal{T}', \rho') \in \mathcal{P}$ such that $V' = V$, $E' = E \cup \{(x, v)$ for some $v \in V : (x, v) \notin E\}$, $\rho'(e, t) = \rho(e, t)$ for all $e \in E, 0 \leq t \leq t_f$, $\rho'((x, v), t) = 0$

for all $t_0 \leq t < t_f$, and $\rho'((x,v),t) = 1$ for some $t > t_f$ (we can take the period p as large as needed here). Consider the execution of $\mathcal{A}$ in $\mathcal{G}'$ starting at time t_0 with x as the source. Since (x,v) does not appear between t_0 and t_f, the execution of $\mathcal{A}$ at every node in $\mathcal{G}'$ will be exactly as at the corresponding node in $\mathcal{G}$ and terminate with v having received the information in more than one hop, contradicting the fact that T is a shortest tree, and thus the correctness of $\mathcal{A}$. □

6. TDB[$fastest$]

The requirement for TDB$[fastest]$ is to deliver the information to each node using the least amount of time, regardless of the stating date, and to have the emitter detect termination within finite time.

We first show that, contrary to the foremost and shortest cases, knowing n and Δ is insufficient to perform a fastest broadcast even in $\mathcal{P}$.

Theorem 17. TDB$[fastest]$ *is not feasible in $\mathcal{P}$ with only knowledge of n and Δ.*

Proof. Consider the infinite family $\{\mathcal{G}_i\}$ of TVGs defined in Section 3. Notice that the duration of the fastest journey from u to v is 3 in $\mathcal{G}_0$ while it is 2 in any other $\mathcal{G}_i$. By Lemma 7, TEST is undecidable even if both n and Δ are known. It follows that it is undecidable whether the fastest journey from u to v has length 3 (in the case of $\mathcal{G}_0$) or 2 (for all other $\mathcal{G}_i$). □

The next question is what knowledge allows the problem to become feasible in $\mathcal{P}$. Observe that $\{\mathcal{G}_i\} \setminus p$ is a finite set, hence Theorem 6 does not apply. Indeed, it has been shown in [10] that, if the period p is known, TDB$[fastest]$ becomes feasible in $\mathcal{P}$.

Theorem 18 (from [10]) TDB$[fastest]$ *is feasible in $\mathcal{P}$ with a known period p, and the corresponding broadcast tree can be reused as such in the subsequent periods.*

We can actually show that TDB$[fastest]$ is feasible in $\mathcal{P}$ with the weaker knowledge of an upper bound on the period.

Theorem 19. TDB$[fastest]$ *is feasible in $\mathcal{P}$ with a known upper bound on p, and the broadcast tree can be reused for subsequent broadcasts.*

Proof. The construction of fastest (broadcast) trees in [10] is based on the observation that achieving a fastest broadcast from some node comes to performing a *foremost* broadcast from that node at the time of its minimum *temporal eccentricity*, that is, when it takes the minimum time to reach all other nodes. The algorithm in [10] consists of learning this date over a complete period p, then build a foremost broadcast tree that starts at that date (modulo p). In case of multiplicity, any of the minimum values qualifies just as well. Having an upper bound p^+ on the period p allows for the exact same technique. Indeed, any interval of time of length p^+ must contain an interval of length p, thus the time of minimum eccentricity will be detected. □

Note that in both cases (knowing p or $p+$), the broadcast tree that is built remains necessarily optimal in the future, since in $\mathcal{P}$ the network schedule repeats forever. It can thus be *memorized* for subsequent broadcasts, *i.e.*, the solution is *reusable*.

7. Computational Relationship

On the basis of this paper results, we can prove the validity of Equation 1 by showing the existence of a strict computational hierarchy between $\mathscr{P}(\mathcal{R}_n)$, $\mathscr{P}(\mathcal{B}_\Delta)$, and $\mathscr{P}(\mathcal{P}_p)$.

Theorem 20. $\mathscr{P}(\mathcal{R}_n) \subsetneq \mathscr{P}(\mathcal{B}_\Delta) \subsetneq \mathscr{P}(\mathcal{P}_p)$

Proof. The fact that $\mathscr{P}(\mathcal{R}_n) \subseteq \mathscr{P}(\mathcal{B}_\Delta) \subseteq \mathscr{P}(\mathcal{P}_p)$ was observed in Theorem 2. To make the left inclusion strict, one has to exhibit a problem P such that $P \in \mathscr{P}(\mathcal{B}_\Delta)$ and $P \notin \mathscr{P}(\mathcal{R}_n)$. By Theorem 16 and Theorem 14, TDB$[shortest]$ is one such example. The right inclusion is similarly proven strict, based on the fact that TDB$[fastest]$ is in $\mathscr{P}(\mathcal{P}_p)$ (Theorem 18) but it is not in $\mathscr{P}(\mathcal{P}_{\Delta,n})$ and thus in $\mathscr{P}(\mathcal{B}_\Delta)$ (Theorem 17). □

With regards to the *feasibility* of the three broadcast problems investigated here, the results established in the previous sections indicate the existence of a strict hierarchy. Given a class $\mathcal{C}$ of TVGs, a knowledge K, and two problems P_1, P_2, let $P_1 \preceq_{\mathcal{C}_K} P_2$ denote the fact that for all $\mathcal{G} \in \mathcal{C}$, if P_2 is feasible in $\mathcal{G}$ with knowledge K, so is P_1.

Given a (possibly infinite) set $\mathcal{L} = \{\mathcal{C}^1_{K_1}, \mathcal{C}^2_{K_2} \ldots\}$ of classes of TVGs with given knowledge, let $P_1 \preceq_{\mathcal{L}} P_2$ denote the fact that $P_1 \preceq_{\mathcal{C}_K} P_2$ for all $\mathcal{C}_K \in \mathcal{L}$. Let $P_1 \prec_{\mathcal{L}} P_2$ denote the fact that $P_1 \preceq_{\mathcal{L}} P_2$ and there exists at least one $\mathcal{C}_K \in \mathcal{L}$ such that $P_1 \in \mathscr{P}(\mathcal{C}_K)$ but $P_2 \notin \mathscr{P}(\mathcal{C}_K)$.

Let $\mathcal{U} = \{\mathcal{R}_n, \mathcal{B}_n, \mathcal{B}_\Delta, \mathcal{B}_{\{n,\Delta\}}, \mathcal{P}_n, \mathcal{P}_\Delta, \mathcal{P}_{\{n,\Delta\}}, \mathcal{P}_p\}$ be the universe of all the classes and knowledge considered in this paper.

Theorem 21. TDB$[foremost] \prec_{\mathcal{U}}$ TDB$[shortest] \prec_{\mathcal{U}}$ TDB$[fastest]$

Proof. By Theorem 8, TDB$[foremost]$ is feasible in $\mathcal{R}_n$ (and *a fortiori* in $\mathcal{B}_n$ and $\mathcal{P}_n$, which are subsets of $\mathcal{R}_n$). By Theorem 2, it is also feasible in $\mathcal{B}_\Delta$ (and in all the remaining combinations of $\mathcal{U}$, which are subsets of $\mathcal{B}_\Delta$). Thus TDB$[foremost]$ is feasible in $\mathcal{U}$ regardless of the class or knowledge considered. Now, by Theorem 16, TDB$[shortest]$ is unfeasible in $\mathcal{P}_n$ (among others). Thus, we have TDB$[foremost] \prec_{\mathcal{U}}$ TDB$[shortest]$. Similarly, by Theorem 14, TDB$[shortest]$ is feasible in $\mathcal{B}_\Delta$, and thus, among others, in $\mathcal{P}_p$ (Theorem 2), whereas TDB$[fastest]$ is only feasible in $\mathcal{P}_p$ (Theorem 17). It thus also holds that TDB$[shortest] \prec_{\mathcal{U}}$ TDB$[fastest]$. □

With regards to *reusability*, the relationship between the three broadcast problems is drastically different. For reusability, let $\leq_{\mathcal{L}}$, $<_{\mathcal{L}}$ be the analogous of $\preceq_{\mathcal{L}}$, $\prec_{\mathcal{L}}$ defined for feasibility. Furthermore, let $P_1 \equiv_{\mathcal{L}} P_2$ if $P_1 \leq_{\mathcal{L}} P_2$ and $P_2 \leq_{\mathcal{L}} P_1$.

Theorem 22. TDB$[shortest] <_{\mathcal{U}}$ TDB$[foremost] \equiv_{\mathcal{U}}$ TDB$[fastest]$

Proof. TDB$[shortest]$ enables reusability in $\mathcal{B}_\Delta$ (see the end of Section 5.1.1) and thus in all the stronger contexts $\{\mathcal{B}_{\{n,\Delta\}}, \mathcal{P}_\Delta, \mathcal{P}_{\{n,\Delta\}}, \mathcal{P}_p\}$. On the other hand, TDB$[foremost]$, although feasible in all of $\mathcal{U}$'s contexts, enables reusability only in $\mathcal{P}_p$ (Theorem 12 and 13). As for TDB$[fastest]$, it is only feasible (and reusable) in $\mathcal{P}_p$ (Theorems 17 and 18). □

Theorems 21 and 22 suggest that the difficulty of these problems is multi-dimensional, in that it depends on the aspect that is looked at (feasibility *vs.* reusability). Indeed, while TDB$[shortest]$ is harder than TDB$[foremost]$ in terms of feasibility, it is easier in terms of reusability. On the other hand, TDB$[fastest]$ is (among) the hardest in both terms.

8. Concluding Remarks

In this paper we focused on three particular problems (shortest, fastest, and foremost broadcast) in three classes of dynamic graphs (recurrent, time-bounded recurrent, and periodic graphs) with different types of applicable knowledge (size of the network, bound on edge recurrence, period, upper bound on period). By comparing the feasibility of these problems within each class depending on the available knowledge, we have observed the impact that knowledge has on feasibility and we have understood the computational relationship between the classes in this context. This has in turn allowed us to observe the relative "difficulty" of the problems under investigation.

Among other things our results show, for example, the special importance of periodic dynamic graphs with known period, the only combination of class and knowledge (in the universe considered here) where Fastest broadcast is feasible. It also stresses the inherent difference between reusability of Foremost broadcast (which is the "easiest" problem to solve but is reusable only in periodic graphs with known period), and Fastest and Shortest on the other, which can be reused whenever they can be solved. Another interesting observation stemming from our results is the intrinsic limitation of knowing only the number of nodes, in which case, regardless of the class of graphs considered, only Foremost broadcast can be performed (without being able to reuse it).

This study is a first step toward an understanding of computability in dynamic graphs and it opens the door to more general investigations on the computability power of different classes and their relationship with knowledge available to the nodes.

Acknowledgments: The authors would like to thank the anonymous referees. Their questions and comments have led to a stronger and clearer paper.

Bibliography

[1] D. Angluin, J. Aspnes, Z. Diamadi, M. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18(4):235–253, 2006.

[2] D. Angluin, J. Aspnes, D. Eisenstat, and E. Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4):279–304, 2007.

[3] B. Awerbuch and S. Even. Efficient and reliable broadcast is achievable in an eventually connected network. In *Proceedings of the 3rd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 278–281, 1984.

[4] H. Baumann, P. Crescenzi, and P. Fraigniaud. Parsimonious flooding in dynamic graphs. In *Proceedings of the 28th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 260–269.
[5] C. Bettstetter, G. Resta, and P. Santi. The node distribution of the random waypoint mobility model for wireless ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(3):257–269, 2003.
[6] M. Biely, P. Robinson, and U. Schmid. Agreement in directed dynamic networks. In *Proceedings of the 19th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, 2012.
[7] B. Bui-Xuan, A. Ferreira, and A. Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(2):267–285, 2003.
[8] J. Burgess, B. Gallagher, D. Jensen, and B.N. Levine. Maxprop: Routing for vehicle-based disruption-tolerant networks. In *Proceedings of the 25th IEEE Conference on Computer Communications (INFOCOM)*, pages 1–11, 2006.
[9] A. Casteigts, S. Chaumette, and A. Ferreira. Characterizing topological assumptions of distributed algorithms in dynamic networks. In *Proceedings of 16th Int. Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 126–140, 2009.
[10] A. Casteigts, P. Flocchini, B. Mans, and N. Santoro. Measuring temporal lags in delay-tolerant networks. *IEEE Transactions on Computers*, 63(2):397–410, Feb 2014.
[11] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012.
[12] I. Chatzigiannakis, O. Michail, and P. Spirakis. Mediated population protocols. *Proceedings of 36th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 363–374, 2009.
[13] A. Clementi, C. Macci, A. Monti, F. Pasquale, and R. Silvestri. Flooding time in edge-markovian dynamic graphs. In *Proceedings of the 27th ACM Symposium on Principles of distributed computing (PODC)*, pages 213–222, 2008.
[14] A. Clementi, A. Monti, F. Pasquale, and R. Silvestri. Information spreading in stationary markovian evolving graphs. In *Proceedings of the 23rd EEE International Symposium on Parallel & Distributed Processing(IPDPS)*, pages 1–12, 2009.
[15] E.W. Dijkstra and C.S. Scholten. Termination detection for diffusing computations. *Information Processing Letters*, 11(1):1–4, 1980.
[16] C. Dutta, G. Pandurangan, R. Rajaraman, Z. Sun, and E. Viola. On the complexity of information spreading in dynamic networks. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 717–736, 2013.
[17] A. Ferreira. Building a reference combinatorial model for MANETs. *IEEE Network*, 18(5):24–29, 2004.
[18] P. Flocchini, M. Kellett, P. Mason, and N. Santoro. Searching for black holes in subways. *Theory of Computing Systems*, 50(1):158–184, 2012.
[19] P. Flocchini, B. Mans, and N. Santoro. On the exploration of time-varying networks. *Theoretical Computer Science*, 469:53–68, 2013.
[20] S. Guo and S. Keshav. Fair and efficient scheduling in data ferrying networks. In *Proceedings of ACM Conference on Emerging Network Experiment and Technology (CoNEXT)*, 2007.
[21] B. Haeupler and F. Kuhn. Lower bounds on information dissemination in dynamic networks. *arXiv preprint arXiv:1208.6051*, 2012.
[22] J. Harri, F. Filali, and C. Bonnet. Mobility models for vehicular ad hoc networks: a survey and taxonomy. *IEEE Communications Surveys & Tutorials*, 11(4):19–41, 2009.
[23] D. Ilcinkas and A. Wade. On the power of waiting when exploring public transportation sys-

tems. *Proceedings of the 15th International Conference on Principles of Distributed Systems (OPODIS)*, pages 451–464, 2011.

[24] P. Jacquet, B. Mans, and G. Rodolakis. Information propagation speed in mobile and delay tolerant networks. *IEEE Transactions on Information Theory*, 56(1):5001–5015, 2010.

[25] S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. In *Proceedings of Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 145–158, 2004.

[26] F. Kuhn, N. Lynch, and R. Oshman. Distributed computation in dynamic networks. In *Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC)*, pages 513–522, 2010.

[27] C. Liu and J. Wu. Scalable routing in cyclic mobile networks. *IEEE Transactions on Parallel and Distributed Systems*, 20(9):1325–1338, 2009.

[28] R. O'Dell and R. Wattenhofer. Information dissemination in highly dynamic graphs. In *Proceedings of the Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, pages 104–110, 2005.

[29] Y. Peres, A. Sinclair, P. Sousi, and A. Stauffer. Mobile geometric graphs: Detection, coverage and percolation. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 412–428. SIAM, 2011.

[30] Z. Zhang. Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: Overview and challenges. *IEEE Communications Surveys & Tutorials*, 8(1):24–37, 2006.

[31] W. Zhao, M. Ammar, and E. Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *Proceedings of the 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, pages 187–198, 2004.