Deterministic Computations in Time-Varying Graphs: Broadcasting under Unstructured Mobility

Arnaud Casteigts, Paola Flocchini, Bernard Mans, Nicola Santoro

Deterministic Computations in Time-Varying Graphs: Broadcasting under Unstructured Mobility

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

[1] University of Ottawa, Ottawa, Canada,
{casteig,flocchin}@site.uottawa.ca
[2] Macquarie University, Sydney, Australia,
bernard.mans@mq.edu.au
[3] Carleton University, Ottawa, Canada,
santoro@scs.carleton.ca

Abstract. Most highly dynamic infrastructure-less networks have in common that the assumption of *connectivity* does not necessarily hold at a given instant. Still, communication routes can be available between any pair of nodes over time and space. These networks (variously called delay-tolerant, disruptive-tolerant, challenged) are naturally modeled as *time-varying graphs* (or *evolving graphs*), where the existence of an edge is a function of time. The existing theoretical research on these networks and graphs has focused on probabilistic assumptions and analysis. The few deterministic results have been restricted to the well structured mobility patterns described by the class of periodically-varying graphs. In this paper we study *deterministic* computations under *unstructured* mobility, that is when the edges of the graph appear infinitely often but without any (known) pattern. In particular, we focus on the problem of *broadcasting with termination detection*. We explore the problem with respect to three possible metrics: the date of message arrival (*foremost*), the time spent doing the broadcast (*fastest*), and the number of hops used by the broadcast (*shortest*). We prove that the solvability and complexity of this problem vary with the metric considered, as well as with the type of knowledge a priori available to the entities. These results draw a complete computability map for this problem when mobility is unstructured.

Keywords: time-varying graphs, evolving graphs, dynamic networks, distributed algorithm, broadcast, termination detection, recurrent edges.

1 Introduction

1.1 The Framework

The past few years have seen increasing research efforts devoted to the study of infrastructure-less highly dynamic networks, whose topologies change as a function of time. Most of these networks, variously called *delay-tolerant*, *disruptive-tolerant*, *challenged*, *opportunistic*, have in common that the assumption of *connectivity* does not necessarily hold at a given instant. The network may even be disconnected at every time instant. Still, communication routes can be available over time and space, and make broadcast and routing feasible. Indeed an extensive amount of research has been devoted, mostly by the engineering community, to the problems of broadcast and routing in such highly dynamical environment (*e.g.* [3,4,13,14,15,19,21,22,23,24]).

The highly dynamic features of these networks can be described by means of *time-varying* graphs (also called *evolving* graphs), where links exist only at some times, a priori unknown to the algorithm designer (see [2,7,9,12]). Thus, in these graphs, the set of edges existing at a given time might not form a connected graph. Due to the complexity of these systems, it is not surprising that very few analytical results exist, all obtained under a set of restrictive assumptions that make the investigated problems amenable to analysis. An example of basic assumption is that the existence of these graphs is continuous over time; that is, the network does not suddenly cease forever to exist.

Almost all the work in this area considers these computations in time-varying graphs from a *probabilistic* standpoint [6,7,8,16], assuming e.g. that the edge schedule obeys a Markovian process. The design and analysis of *deterministic* solutions has been carried out under very strong assumptions. For example, knowing the complete edge schedule ahead of time in a central entity allows to compute optimum solutions to the broadcast and routing problems [2]. Intermediate assumptions have been investigated, such as the fact that the network is always connected [20]. A hierarchy of basic assumptions for distributed algorithms in dynamic networks is discussed in [5].

Clearly any a-priori knowledge about the edge schedule can be employed in the design and analysis of (possibly deterministic) solutions. This is also true from a practical point of view, and indeed an intensive investigation exists on *mobility patterns* [1,18,17,10]. Some classes of infrastructure-less networks have indeed specific mobility patterns. For example, in networks such as public transports with fixed timetables, low earth orbiting (LEO) satellite systems, security guards' tours, etc. the edge-schedule is *periodic*, and deterministic protocols for routing and exploration of such networks have been devised (e.g., [12,11,19]). Periodicity is interesting not only because it models several classes of dynamic systems, but also because the infinite mobility pattern defining it is highly *structured*. The existing results show that the existence of such a structure allows the development of deterministic solutions to fundamental problems.

The question immediately arises of what happens when the mobility is *unstructured*. More precisely, what happens if encounters between mobile entities occur infinitely often but without any (known) pattern? what happens if there is no known pattern but there is a time bound on the re-appearance of edges? What can be done *deterministically* in such cases?

In this paper we address these questions and provide some answers on the computability and complexity aspects with regards to the basic problem of *broadcasting with termination detection*.

1.2 Problems and Contributions

Consider the class $\mathcal{R}$ of *recurrent* time-varying graphs whose edges appear infinitely often; that is if an edge (x, y) between nodes x and y exists at time t (i.e., entities x and y are able to communicate at time t), then there exists a time $t' > t$ when (x, y) also exists (let us assume the set of apparition of a given edge as *enumerable*). Let $\mathcal{B} \subset \mathcal{R}$ be the class of time-bounded recurrent time-varying graphs, where two successive appearance of a same edge is bounded by some duration. We consider the basic problem of *broadcasting with termination detection* in $\mathcal{R}$ and in $\mathcal{B}$: there is a node (the source, also called *emitter*) that has a message that must be distributed to all other nodes; the source must be notified when the entire process has been completed. This problem is more difficult than simple broadcast, and is required in more complex operations, e.g. *sequence transmission*, where the i-th sequence item must only be transmitted after the $(i-1)^{th}$ item has been received by all nodes.

We explore the problem with respect to the three possible metrics discussed in [2]: the date of message arrival (*foremost*); the number of hops used (*shortest*); and the time spent doing the effective broadcast (*fastest*). Interestingly, the solvability and complexity of the problem vary with the type of metric considered, as well as with the knowledge available to the nodes. Note that broadcasting with termination detection involves two processes: the actual dissemination of information achieved by exchange of *information messages*, and termination detection achieved by exchange of (typically smaller) *control messages*. In the paper we make a distinction between these two types of messages and we analyze them separately. Also notice that a byproduct of a broadcast algorithm might be the construction of a (delay-tolerant) spanning tree of the underlying graph, which could possibly be reused for subsequent broadcasts, sometimes for the dissemination process (thus reducing the information messages), sometimes for termination detection (impacting the number of control messages), or for both. In each setting we discuss also the consequences on subsequent broadcasts in order to highlight the variation of benefits in reusability.

We first provide some impossibility results showing that broadcasting with termination detection cannot be solved in $\mathcal{R}$ without any knowledge of the underlying graph, nor in $\mathcal{B}$ without either the same knowledge or a bound on the recurrence time. We then analyze solvability and complexity of the problem in the various settings providing algorithms when it can be solved. The solvability results are summarized in Table 1 and the complexity results in Table 2, where n is the number of nodes, and Δ a bound on the recurrence time.

Metric	Class	Knowledge	Feasibility
Foremost	$\mathcal{R}$	$\emptyset$	no
		n	yes
	$\mathcal{B}$	$\emptyset$	no
		n	yes
		Δ	yes

Metric	Class	Knowledge	Feasibility
Shortest	$\mathcal{R}$	$\emptyset$	no
		n	no
	$\mathcal{B}$	$\emptyset$	no
		n	no
		Δ	yes
Fastest	$\mathcal{R}$ or $\mathcal{B}$	n or Δ	no

Table 1. Summary of contributions - Solvability.

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 $\left\{\right.$		$n\&\Delta$	$O(n\Delta)$	$O(m)$	$O(n) : n-1$	$O(n)$	0
		$n\&\Delta$	$O(n\Delta)$	$O(m)$	0	$O(m)$	0

Table 2. Summary of contributions - Complexity (for solvable cases)

2 Model and Basic Properties

2.1 Definitions and Terminology

Consider a system composed of a set of entities V that interact with each other over a (possibly infinite) time interval $\mathbb{T}$, called *lifetime* of the system (a subset of either $\mathbb{Z}$ (*discrete* time) or $\mathbb{R}$ (*continuous* time); our results hold in either case). The set of the times when the entities are in contact defines a *time-varying graph* (*TVG*, for short) $\mathcal{G} = (V, E, \rho)$, with $E \subseteq V \times V$ being the set of *intermittently available* edges such that $(u, v) \in E \Leftrightarrow u$ and v have at least one contact overlapping with $\mathbb{T}$, and $\rho : E \times \mathbb{T} \to \{0, 1\}$ indicates whether a given edge is *present* at a given time. In the following the terms entity and node will be used interchangeably.

This model is equivalent in substance to that of *evolving graphs* [9], where $\mathcal{G}$ is represented by the sequence of graphs $G_1, G_2, ..., G_i, ...$ each providing a *snapshot* of the system whenever a change (edge appearance/disappearance) takes place. In comparison, the definition used in this paper offers an *interaction-centric* view of the network evolution (the evolution of each edge can be considered irrespective of the global time sequence), which proves more convenient to express several properties.

An edge $e \in E$ is said to be *recurrent* if it appears infinitely often; that is, for any date t, $\rho(e, t) = 0 \implies \exists t' > t \mid \rho(e, t') = 1$. When all the edges of a TVG $\mathcal{G}$ are recurrent, we say that $\mathcal{G}$ is *recurrent*. Let $\mathcal{R}$ denote the class of recurrent TVGs. The recurrence of an edge e is said to be *time-bounded* (or simply *bounded*), if there exists a constant $\Delta(e)$ such that the time between any two successive appearances of e is at most $\Delta(e)$. When the recurrence of all the edges of a graph $\mathcal{G}$ is time-bounded, we say that $\mathcal{G}$ is *time-bounded recurrent*, call $\Delta(\mathcal{G}) = \max\{\Delta(e) : e \in E\}$, and denote by $\mathcal{B} \subset \mathcal{R}$ the class of time-bounded recurrent TVGs.

Given a TVG $\mathcal{G} = (V, E, \rho)$, the underlying graph $G = (V, E)$ is assumed simple (no self-loop nor multiple edges) and connected[4]. Each node v has a local function λ_v associating labels (or *port numbers*), to its incident edges (or *ports*). For each edge e there are two labels: $\lambda_u(e)$ local to u and $\lambda_v(e)$, local to v. These labels are locally unique and do not change from one appearance to another. 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). Finally, we note $\mathcal{G}_{[t_a,t_b)}$ the temporal subgraph of a TVG $\mathcal{G}$ with restricted lifetime $[t_a, t_b)$.

When an edge $e = (x, y)$ appears, the entities x and y can communicate. The time ζ necessary to transmit a message on any edge is called *crossing delay*, and is known by the nodes. The TVGs in the rest of this paper are assumed to have recurrent edges with a minimal duration of $2 \times \zeta$ for every edge presence (long enough for a back and forth exchange of message). This last assumption implies that

Property 1
1. If a message is sent just after an edge has appeared, the message and a potential answer are guaranteeed to be successfully transmitted.

[4] Broadcast, as well as any other global computation, would be impossible otherwise.

2. If the recurrence of an edge is bounded by some Δ, then this edge cannot disappear for more than $\Delta - 2 \times \zeta$.

The appearances and disappearances of edges are instantaneously 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, the message 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 has arrived or not. 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 on the 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 absent.

A sequence of couple $\mathcal{J} = \{(e_a, t_a), (e_b, t_b), ...\}$, with $e_i \in E$ and $t_i \in \mathbb{T}$ for all i, is called a *journey* in $\mathcal{G}$ iff $\{e_a, e_b, ...\}$ is a walk in G and for all t_i, $\rho(e_i)_{[t_i, t_i+\zeta)} = 1$ and $t_{i+1} \geq t_i + \zeta$, where ζ is the time required to transmit a message on an edge, called *crossing delay*. 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 possible journeys in a graph $\mathcal{G}$. We will say that $\mathcal{G}$ *admits* a journey from a node u to a node v, and note $\exists \mathcal{J}_{(u,v)} \in \mathcal{J}^*_{\mathcal{G}}$, if there exists at least one possible journey from u to v in $\mathcal{G}$. Note that the notion of journey is asymmetrical ($\exists \mathcal{J}_{(u,v)} \in \mathcal{J}^*_{\mathcal{G}} \not\Rightarrow \exists \mathcal{J}_{(v,u)} \in \mathcal{J}^*_{\mathcal{G}}$), regardless of whether edges are directed or undirected.

Because no end-to-end connectivity is assumed, the very notion of distance must incorporate the time factor. In fact, at least three notions of *length* can be defined for journeys (adapted from [2]): the *hop-count*, the *arrival date*, and the *duration* of a journey. Given a journey $\mathcal{J} = \{(e_1, t_1), (e_2, t_2) \ldots,$
$(e_k, t_k)\}$, its *hop-count* $|\mathcal{J}|_h$, is the number of couples in $\mathcal{J}$ (that is, k). The *arrival date* of $\mathcal{J}$, noted $|\mathcal{J}|_a$, is $t_k + \zeta$. Finally, the *duration* of $\mathcal{J}$, noted $|\mathcal{J}|_t$, is $|\mathcal{J}|_a - t_1$. Each of these metrics gives rise to a distinct definition of *distance* in $\mathcal{G}$.

- The *topological distance* between a node u and a node v, noted $d_h(u, v)$, is defined as $min\{|\mathcal{J}_{(u,v)}|_h : \mathcal{J}_{(u,v)} \in \mathcal{J}^*_{\mathcal{G}}\}$. A journey $\mathcal{J}_{(u,v)}$ whose length is $d_h(u, v)$ is qualified as *shortest* ;
- The *earliest arrival date* between u and v, noted $d_a(u, v)$ is defined as $min\{|\mathcal{J}_{(u,v)}|_a : \mathcal{J}_{(u,v)} \in \mathcal{J}^*_{\mathcal{G}}\}$. A journey $\mathcal{J}_{(u,v)}$ whose arrival date is $d_a(u, v)$ is qualified as *foremost* ;
- Finally, the *smallest delay* between u and v, noted $d_t(u, v)$ is $min\{|\mathcal{J}_{(u,v)}|_t : \mathcal{J}_{(u,v)} \in \mathcal{J}^*_{\mathcal{G}}\}$, and a journey $\mathcal{J}_{(u,v)}$ whose duration is $d_t(u, v)$ is qualified as *fastest*.

The *eccentricity* of a node u is defined as $max\{d_x(u, v) : v \in V\}$, where x is either h, a, or t, depending on the type of distance considered, and noted $\mathcal{E}_h(u)$, $\mathcal{E}_a(u)$, and $\mathcal{E}_t(u)$, respectively. Similarly, three notions of *diameter* of a graph $\mathcal{G} = (V, E, \rho)$ can be defined as $max(d_x(u, v) : u, v \in V)$, and noted $D_h(\mathcal{G})$, $D_a(\mathcal{G})$, or $D_t(\mathcal{G})$. Notice that D_h is closer to the usual notion of diameter (in hop-count) than D_a or D_t, which are both in the temporal domain. Observe also that all these notions are time-dependent in the sense that they may vary according to the time when they are considered.

2.2 Problems and Basic Limitations

The problem of *broadcast with termination detection*, `TDBroadcast`, 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 `TDBroadcast` in $\mathcal{G} \in \mathcal{R}$ if it solves it for any source $x \in V$ and time t. We say that it solves `TDBroadcast` in $\mathcal{R}$ if it solves `TDBroadcast` for any $\mathcal{G} \in \mathcal{R}$.

We are interested in three variations of the `TDBroadcast` problem, following the notions of distance defined above: `TDBroadcast`[*foremost*], where *each* node must receive the information at the *earliest* possible date following its *creation* at the emitter; `TDBroadcast`[*shortest*], where each node must receive the information within a minimal number of hops from the emitter, and `TDBroadcast`[*fastest*], where each node must receive the information at the *earliest* possible date following the beginning of its *emission*. For each of these problems, we require that the emitter detects termination, but this detection is not subjected to the same foremost, shortest, or fastest constraint.

Some *knowledge* of G, the underlying graph, is necessary even for simple broadcast in recurrent TVGs. In fact we have:

Theorem 2. *Without any knowledge of the underlying graph,* `TDBroadcast` *in $\mathcal{R}$ cannot be solved.*

Proof. By contradiction, let $\mathcal{A}$ be a algorithm that solves `TDBroadcast` in $\mathcal{R}$. Consider an arbitrary $\mathcal{G} = (V, E, \rho) \in \mathcal{R}$ 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 $\mathcal{G}' = (V', E', \rho') \in \mathcal{R}$ such that $V' = V \cup \{u\}$, $E' = E \cup \{(u, v) : v \in V\}$, $\rho'(e, t) = \rho(e, t)$ for all $e \in E, t \in \mathbb{T}$, $\rho'((u, v), t) = 0$ for all $t_0 \leq t < t_f$, and $\rho'((u, v), t) = 1$ for $t > t_f$. 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 $\mathcal{G}'$ will be 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 $\mathcal{A}$.

Indeed, as we will discuss later, some *metric* knowledge such as knowing the number of nodes $n = |V|$ or, in the case of bounded TVGs (class $\mathcal{B}$), knowing an upper bound Δ on the recurrence time $\Delta(\mathcal{G})$, can play an important role.

Theorem 3. *Without any knowledge of the underlying graph nor of Δ,* `TDBroadcast` *in $\mathcal{B}$ cannot be solved.*

Proof sketch. It follow the same lines as the proof of Theorem 2. The only difference is that both $\mathcal{G} = (V, E, \rho)$ and $\mathcal{G}' = (V', E', \rho')$ are in $\mathcal{B}$, and $\mathcal{G}'$ is such that $\Delta(\mathcal{G}') > t_f - t_0$.

Finally, let us conclude with a general impossibility result for fastest broadcast with termination, which cannot be solved even if both n and Δ are known.

Theorem 4. `TDBroadcast`[fastest] *is not solvable in $\mathcal{R}$, nor in $\mathcal{B}$, regardless of the fact that n or Δ are known.*

Proof sketch. The argument relates to the very existence of fastest journeys in an unstructured infinite setting. Indeed, consider any journey which at some point is the fastest journey that existed heretofore, clearly none of the above knowledge allows to decide that an even faster journey cannot exist at a future date. Consequently, no broadcast algorithm can recognize a fastest journey, and terminate.

Because of the impossibility of fastest broadcast, the rest of the paper focuses on `TDBroadcast`[*foremost*] and `TDBroadcast`[*shortest*] only, and on the impact on solvability and complexity of being in $\mathcal{R}$ or $\mathcal{B}$, and knowing n or Δ (if in $\mathcal{B}$).

3 `TDBroadcast`[*foremost*]

The objective is to have all the nodes receive the information at the *earliest* possible date following its creation at the emitter (*foremost* broadcast), then have the emitter detect termination. Clearly, achieving a foremost broadcast requires to use a flooding-based mechanism. Indeed, the very fact of probing a neighbor to determine whether it already has the information compromises the possibility of sending it in a foremost fashion (in addition to risking the disappearance of the edge between the probe and the real sending). The problem thus comes to minimize the number of messages and detect when all the nodes are informed. As we have seen in Theorem 2, the problem cannot be solved without any metric knowledge. We show that it becomes possible in the general class $\mathcal{R}$ if the number of nodes $n = |V|$ is known. Knowing n is however not required in the more specific case of $\mathcal{B}$, where the knowledge of an upper bound Δ on the recurrence time $\Delta(\mathcal{G})$ can also be used to solve the problem. If both n and Δ are known in $\mathcal{B}$, the termination detection can even become implicit, thereby saving a number of control messages.

3.1 `TDBroadcast`[*foremost*] in $\mathcal{R}$

In this section we discuss only knowledge of n since Δ cannot be known being the recurrent time unbounded by definition.

3.1.1 Knowledge of n

The problem is solvable when n is known, by using the procedures detailed in Algorithm 1, informally described as follows. Every time a new edge appears locally to an informed node, the node sends the information on this edge and remembers it. The first time a node receives the information, it chooses 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. 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. Each notification is then individually forwarded in the converge-cast tree using the $send\_retry$ primitive, and eventually collected by the emitter. When the emitter has received $n-1$ notifications, it knows all the nodes are informed (and the next broadcast can start, for example).

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

```
1:  $Edge\ parent \leftarrow nil$                          // edge the information was received from (for non-emitter nodes).
2:  $Integer\ nbNotifications \leftarrow 0$                 // number of notifications received (for the emitter).
3:  $Set<Edge>\ informedNeighbors \leftarrow \emptyset$     // neighbors known to have the information.
4:  $Status\ myStatus \leftarrow \neg$informed              // status of the node (informed or non-informed).
5:  initialization:
6:    if $isEmitter()$ then
7:      $myStatus \leftarrow$ informed
8:      $send(information)\ on\ I_{now()}$                  // sends the information on all its present edges.
9:  onAppearance of an edge $e$:
10:   if $myStatus ==$ informed and $e \notin informedNeighbors$ then
11:     $send(information)\ on\ e$
12:     $informedNeighbors \leftarrow informedNeighbors \cup \{e\}$   // (see Prop. 1).
13: onReception of a message $msg$ from an edge $e$:
14:   if $msg.type == Information$ then
15:     $informedNeighbors \leftarrow informedNeighbors \cup \{e\}$
16:     if $myStatus == \neg$informed then
17:       $myStatus \leftarrow$ informed
18:       $parent \leftarrow e$
19:       $send(information)\ on\ I_{now()} \setminus informedNeighbors$   // propagates.
20:       $send\_retry(notification)\ on\ e$              // notifies that a new node got 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:       $nbNotifications \leftarrow nbNotifications + 1$
24:       if $nbNotifications == n - 1$ then
25:         $terminate$                                  // at this stage, the emitter knows that all nodes are informed.
26:     else
27:       $send\_retry(notification)\ on\ parent$
```

Theorem 5. *When n is known,* `TDBroadcast`[foremost] *can be solved in $\mathcal{R}$ exchanging $O(m)$ information messages and $O(n^2)$ control messages, in unbounded time. (We call m the number of edges $|E|$).*

Proof sketch. 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 receive the information. 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 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. Note that the dissemination of information itself is performed in optimal time: $\mathcal{E}_a(emitter)$ in $\mathcal{G}_{[t,+\infty)}$, because the information is either directly relayed on edges that are present, or sent as soon as a new edge appears. However, since the recurrence of the edges is unbounded, this time, as well as the time required for termination detection, is necessarily unbounded.

Reusability for the subsequent broadcasts. By nature, a foremost tree is transient and cannot be re-used as such in subsequent broadcasts. However, it can be re-used by subsequent broadcasts as a converge-cast tree for the notification process where, instead of sending a notification as soon as a node is informed, each node notifies its parent in the converge-cast tree if and only if it is itself informed and has received a notification from each of its children. This would allow to reduce the number of control messages from $O(n^2)$ to $O(n)$, having only one notification per edge of the converge-cast tree.

3.2 `TDBroadcast`[*foremost*] in $\mathcal{B}$

If the recurrence is bounded, then either the knowledge of n or an upper bound $\varDelta$ on the recurrence time $\varDelta(\mathcal{G})$ can be used to detect termination.

3.2.1 Knowledge of n. Using the same algorithm as for class $\mathcal{R}$ (Algorithm 1) we can obviously solve the problem in $\mathcal{B}$ with the same message complexity, but bounded time. Moreover, the same observations regarding reusability for the subsequent broadcasts apply.

Theorem 6. *When n is known,* `TDBroadcast`[foremost] *can be solved in $\mathcal{B}$ exchanging $O(m)$ information messages and $O(n^2)$ control messages, in $O(n\varDelta)$ time.*

Proof sketch. The arrival-date-based eccentricity of the emitter ($\mathcal{E}_a(emitter)$ in $\mathcal{G}_{[t,+\infty)}$), which is itself bounded by the arrival-date-based diameter of the graph ($D_a(\mathcal{G}_{[t,+\infty)})$), is now clearly bounded by $\varDelta(n-1)$ (the worst case is when the foremost tree is a line). The detection of termination by the emitter may require an additional $\varDelta(n-1)$ for the propagation of the last notification. The overall time required for the emitter to detect termination is thus at most $\mathcal{E}_a(emitter)$ in $\mathcal{G}_{[t,+\infty)} + \varDelta(n-1)$, bounded by $\varDelta(2n-2)$.

3.2.2 Knowledge of $\varDelta$. The information dissemination is performed as in Algorithm 1, termination detection is however achieved differently and is based on knowledge of $\varDelta$. The code of the algorithm is reported in Algorithm 2 and we describe it here informally. Due to the time-bounded recurrence, no node can discover a new neighbor after a duration of $\varDelta$. Knowing $\varDelta$ can thus be used by the nodes to determine whether they are a leaf in the broadcast tree (if they have not informed any other node after the date they were informed at, plus $\varDelta$). This allows the leaves to terminate spontaneously while notifying their parent, which recursively terminate as they receive the notifications from all their children. Everytime a new edge appears locally to an informed node, this node sends the information on this edge, and remembers it. The first time a node receives the information, it chooses the sender as parent, memorizes the current time (say, 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 upward in the tree, but only intended to inform the direct parent about the existence of a new child (so that it knows it will have to wait for a notification by this node during the hierarchical notification). If an informed node has not received any affiliation message after a duration of $\varDelta + \zeta$ (see Figure 1), it sends a *notification* message to its parent using the $send\_retry$ primitive. If a node has one or several children, it waits until having received a notification message from each of them, then notifies its parent in the converge-cast tree in turn (using $send\_retry$ again). When the emitter has received a notification from each of its children, it knows that all nodes have received the information.

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

```
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 the 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 its 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:               $send\_retry(notification)\ on\ parent$   // notifies the parent in turn.
32:           $terminate$            // whether emitter or not, the node has terminated at this stage.
33:  when $now() == firstRD + \varDelta + \zeta$:          // tests whether the underlying node is a leaf.
34:    if $nbChildren == 0$ then
35:       $send\_retry(notification)\ on\ parent$
36:       $terminate$
```

Theorem 7. *When Δ is known,* `TDBroadcast`[foremost] *can be solved in $\mathcal{B}$ exchanging $O(m)$ information messages and $O(n)$ control message, in $O(n\Delta)$ time.*

Proof sketch. Correctness follows the same lines of the proof of Theorem 5, where however the correct construction of a converge-cast spanning tree is guaranteed by knowledge of Δ (the leaves of the tree recognize to be so because no new edges appear within Δ time) and where 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. However, the number of notification and affiliation messages decrease 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 version where n is known, that is, optimal with $\mathcal{E}_a(emitter)$ in $\mathcal{G}_{[t,+\infty)}$. The time required for the emitter to subsequently detect termination is an additional $\Delta + \zeta + \Delta(n-1)$ (the value $\Delta + \zeta$ corresponds to the time needed by the last informed node to detect that it is a leaf, and $\Delta(n-1)$ corresponds to the worst case of the notification process, chained from that node to the emitter).

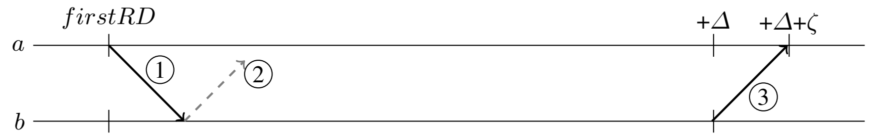

Fig. 1. Reasons why a node waits $\Delta + \zeta$ for receiving potential *affiliation* messages. First of all, note that *information* messages cannot be lost when they are sent on an appearing edge, neither can their potential *affiliation* answer (Prop. 1). The loss of such messages can only occur when the information is directly relayed by a node who received it (say, node a, relaying at time $firstRD$ the information to node b). If the information message is lost, then this is not a problem (it simply means that this edge at that time did not have to be used). If the *affiliation* message is lost, however, it must be sent again. In the worst case, the common edge disappears just before the affiliation message is delivered, and reappears only $\Delta - 2 \times \zeta$ later (Prop. 1). Affiliation messages can thus be received until $firstRD + \Delta + \zeta$.

Reusability for the subsequent broadcasts. Clearly, the number of nodes n, which is not *apriori* 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, which is discussed next and allows solving the problem with $O(m)$ information messages and no control messages.

3.2.3 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 $n\Delta$ (the worst case is when the foremost tree is a line). The termination detection can thus become implicit after this amount of time, which allows us to do without any control message (whether of affiliation or notification kinds). Note that subsequent broadcasts will have the same complexity.

Theorem 8. *When Δ and n are known,* `TDBroadcast`[foremost] *can be solved in $\mathcal{B}$ exchanging $O(m)$ info. messages and no control messages, in $O(n\Delta)$ time.*

4 `TDBroadcast`[*shortest*]

The objective is to have all nodes receive the information within a minimal *number of hops* from the emitter (*shortest* broadcast), then have the emitter detect termination. We show below that contrarily to the foremost case, knowing n is not enough to perform a shortest broadcast (even in $\mathcal{B}$). Considering only the two kind of knowledge we considered in this paper, it requires Δ to be known (and thus also to be in $\mathcal{B}$). In the following we then consider only the case of $\mathcal{B}$. Note that, contrarily to the foremost case, if a given tree is shortest for some particular emission date, then it is also shortest for any other emission dates (thanks to the recurrence of edges). Put it differently, the shortest quality of a tree is not time-dependent in recurrent TVGs. This allows shortest trees to be reused as is in subsequent broadcasts.

4.1 `TDBroadcast`[*shortest*] in $\mathcal{B}$

We first show that knowledge of n is not sufficient to perform shortest broadcast with termination detection in $\mathcal{B}$ and we then describe how to solve the problem when Δ is known or when both n and Δ are.

4.1.1 Knowledge of n

Theorem 9. *If n is the only knowledge available* `TDBroadcast`*[shortest] cannot be solved in $\mathcal{B}$.*

Proof. By contradiction, let $\mathcal{A}$ be a algorithm that solves `TDBroadcast`[*shortest*] in $\mathcal{B}$ with knowledge of n only. Consider an arbitrary $\mathcal{G} = (V, E, \rho) \in \mathcal{R}$ and $x \in V$. Execute $\mathcal{A}$ in $\mathcal{G}$ starting at time t_0 with x as the source obtaining a shortest tree T. Let t_f be the time when the algorithm terminates and all nodes have entered the terminal state. Let $\mathcal{G}' = (V', E', \rho') \in \mathcal{R}$ such that $V' = V$, $E' = E \cup \{(x, v) : v \in V, (x, v) \notin E\}$, $\rho'(e, t) = \rho(e, t)$ for all $e \in E, t \in \mathbb{T}$, $\rho'((u, v), t) = 0$ for all $t_0 \leq t < t_f$, and $\rho'((u, v), t) = 1$ for $t > t_f$. 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 $\mathcal{G}'$ 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 correctness of $\mathcal{A}$.

4.1.2 Knowledge of Δ

The idea is to propagate the message along the edges of a breadth-first spanning tree of the underlying graph. We report in Algorithm 3 the actual code, with the following informal description.

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 the underlying 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:         $roundStart \leftarrow now()$    // causes the procedure "when $now() == roundStart$:" (below) to execute.
10: onAppearance of an edge $e$:
11:     if $myStatus ==$ informed then
12:         if $e \notin informedNeighbors$ then
13:             $send(roundStart + \Delta - now() - \zeta)\ on\ e$    // time until the end of the 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 message and
        have no parent yet will take this node as parent and wait $\Delta$-$\zeta$ before informing new nodes.
33: when $now() == roundStart + \Delta + \zeta$:    // tests whether the underlying node is a leaf.
34:     if $|children| == 0$ then
35:         $send\_retry(notification)\ on\ parent$
```

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 plus Δ minus the current time minus the crossing delay, see Figure 2 for an intuitive explanation), so that the receiving node knows when to start informing new nodes in turn (if it had not the information yet). If a node has not informed any other node during its round, it notifies its parent. When a node has been notified by all its children, it notifies its parent. Note that 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 (with the same complication as already discussed in Figure 1). Finally, when the emitter has been notified by all its children, it knows that the broadcast is terminated.

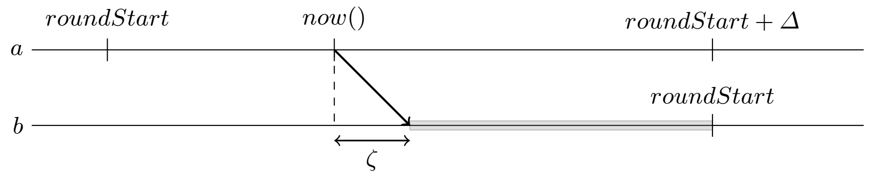

Fig. 2. Reasons why nodes transmit their own starting date plus Δ minus the current date minus ζ, when they attempt to become parent of a node (here, when a attempts to become b's parent). This duration corresponds to the exact amount of time the child would have to wait, if the relation is established, before integrating other nodes in turn.

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

Proof sketch. The fact that the algorithm constructs a (time-varying) breadth-first (and thus shortest) spanning tree follows from connectivity of the underlying graph and from knowledge of Δ. The bound on recurrence is in fact used to guarantee synchronization of rounds and correct identification of levels in the breadth-first spanning tree. 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: $((\mathcal{E}_h(emitter)) - 1)\Delta$ to reach the last node, plus $\Delta + \zeta$ at this node, plus $((\mathcal{E}_h(emitter)) - 1)\Delta$ to forward its notification. (The additional periods of ζ caused by the waiting of affiliation messages matter only for the last round, since the construction continues in parallel otherwise.) The total is $((2\times\mathcal{E}_h(emitter)) - 1)\Delta + \zeta$, which is upper bounded by $(2n - 1)\Delta + \zeta$.

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 tree built during the first broadcast, which reduces the number of information message from $O(m)$ to $O(n)$ in these subsequent broadcasts (assuming the nodes memorized the set of their children during the first broadcast). Moreover, if the depth d of the tree is reported through the first notification process, then all subsequent broadcasts can have an implicit termination detection which is optimal in time (after $d\Delta$ time), and no control message is needed.

4.1.3 Knowledge of n and Δ

When both n and Δ are known a possibility is to apply the same dissemination procedure as in Algorithm 3 and to use an implicit termination detection. In fact, as already discussed in Section 3.2.3 $n\Delta$ is an upper bound on the termination time. This allows the nodes to exchange no control messages at all.

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

Reusability for subsequent broadcasts. Note that the solution discussed above offers no gain on the number of information messages in the subsequent broadcasts. 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, however, 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 the expected number of broadcasts planned.

References

1. C. Bettstetter, G. Resta, and P. Santi. The node distribution of the random waypoint mobility model for wireless ad hoc networks. *IEEE Trans. on Mobile Comp.*, 2(3):257–269, 2003.
2. B. Bui-Xuan, A. Ferreira, and A. Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *Intl. J. of Foundations of Comp. Science*, 14(2):267–285, April 2003.
3. J. Burgess, B. Gallagher, D. Jensen, and B.N. Levine. Maxprop: Routing for vehicle-based disruption-tolerant networks. In *Proc. of the 25th Conference on Computer Communications (INFOCOM'06)*, pages 1–11, 2006.
4. I. Cardei, C. Liu, and J. Wu. Routing in Wireless Networks with Intermittent Connectivity. *Encyclopedia of Wireless and Mobile Communications, CRC Press, Taylor & Francis*, 2007.
5. A. Casteigts, S. Chaumette, and A. Ferreira. Characterizing topological assumptions of distributed algorithms in dynamic networks. In *Proc. 16th Intl. Conf. on Structural Information and Communication Complexity (SIROCCO'09)*, volume 5869 of *LNCS*, pages 126–140, 2009.
6. A. Clementi, C. Macci, A. Monti, F. Pasquale, and R. Silvestri. Flooding time in edge-markovian dynamic graphs. In *Proc. of the 27th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 213–222. ACM, 2008.
7. A. Clementi, A. Monti, F. Pasquale, and R. Silvestri. Information spreading in stationary markovian evolving graphs. In *Proc. of the 23rd IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1–12. IEEE Computer Society, 2009.
8. T. Dimitriou, S. Nikoletseas, and P. Spirakis. The infection time of graphs. *Discrete Applied Mathematics*, 154(18):25772589, 2006.
9. A. Ferreira. Building a reference combinatorial model for MANETs. *IEEE Network*, 18(5):24–29, 2004.
10. M. Fiore and J. Härri. The networking shape of vehicular mobility. In *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, pages 261–272. ACM, 2008.
11. P. Flocchini, M. Kellett, P. Mason, and N. Santoro. Mapping an unfriendly subway system. In *Proc. 5th International Conference on Fun with Algorithms*, 2010. To appear.
12. P. Flocchini, B. Mans, and N. Santoro. Exploration of periodically varying graphs. In *Proc. 20th Intl. Symposium on Algorithms and Computation (ISAAC'10)*, 2009.
13. S. Guo and S. Keshav. Fair and efficient scheduling in data ferrying networks. In *Proceedings of the 2007 ACM CoNEXT conference*. ACM New York, NY, USA, 2007.
14. P. Jacquet, B. Mans, and G. Rodolakis. Information propagation speed in mobile and delay tolerant networks. In *Proc. of the 28th Conference on Computer Communications (INFOCOM'09)*, Rio de Janeiro, Brazil, 2009.

15. S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. In *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 145–158. ACM New York, NY, USA, 2004.
16. D. Kempe and J. Kleinberg. Protocols and impossibility results for gossip-based communication mechanisms. In *43rd Symposium on Foundations of Computer Science (FOCS)*, pages 471–480, 2002.
17. M. Kim, D. Kotz, and S. Kim. Extracting a mobility model from real user traces. In *Proceedings of IEEE Infocom*, volume 6, pages 1–13. Citeseer, 2006.
18. J. Leguay, T. Friedman, and V. Conan. Evaluating mobility pattern space routing for DTNs. In *Proc. of the 25th IEEE International Conference on Computer Communications (INFOCOM'06)*, page 18, 2006.
19. C. Liu and J. Wu. Scalable routing in cyclic mobile networks. *IEEE Trans. Parallel Distrib. Syst.*, 20(9):1325–1338, 2009.
20. R. O'Dell and R. Wattenhofer. Information dissemination in highly dynamic graphs. In *DIALM-POMC '05: Proceedings of the 2005 joint workshop on Foundations of mobile computing*, pages 104–110, New York, NY, USA, 2005. ACM.
21. T. Spyropoulos, K. Psounis, and C.S. Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, page 259. ACM, 2005.
22. X. Zhang, J. Kurose, B.N. Levine, D. Towsley, and H. Zhang. Study of a bus-based disruption-tolerant network: mobility modeling and impact on routing. In *Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, pages 195–206. ACM, 2007.
23. 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.
24. W. Zhao, M. Ammar, and E. Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *MobiHoc '04: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, pages 187–198, New York, NY, USA, 2004. ACM.