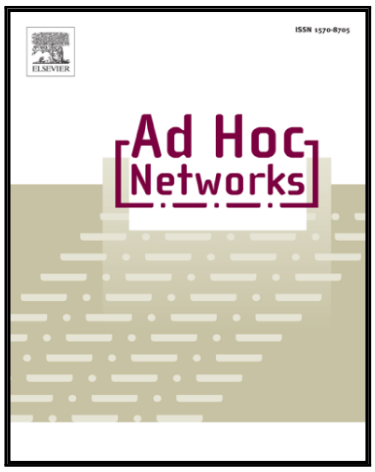

Dynamic p-graphs for capturing the dynamics of distributed systems

B. Ducourthial, A.M. Wade

PII: S1570-8705(16)30126-3
DOI: 10.1016/j.adhoc.2016.05.004
Reference: ADHOC 1390

To appear in: *Ad Hoc Networks*

Received date: 2 August 2015
Revised date: 16 April 2016
Accepted date: 13 May 2016

Please cite this article as: B. Ducourthial, A.M. Wade, Dynamic p-graphs for capturing the dynamics of distributed systems, *Ad Hoc Networks* (2016), doi: 10.1016/j.adhoc.2016.05.004

Dynamic p-graphs for capturing the dynamics of distributed systems

B. Ducourthial and A.M. Wade
Corresponding author: Bertrand.Ducourthial@utc.fr

Sorbonne universités, Université de Technologie de Compiègne
Heudiasyc UMR 7253, CS 60 319, 60 203 Compiègne cedex.

Abstract

The dynamics of a mobile dynamic distributed system depends on both the node mobility and the capacity of the underlying communication protocol. To qualify the dynamics of a distributed system, the *family of dynamic p-graphs* is introduced in this paper, constituting a finite set of dynamic graphs, each of them being a sequence of observed graphs, with the particularity that their edges allow transferring p messages. The family of dynamic p-graphs allows the characterization and the comparison of dynamic distributed systems of very different nature. It is also used to evaluate the ability of algorithms themselves to support a given dynamic distributed system.

Keywords: distributed algorithm, dynamic networks, routing, VANET

1. Introduction

Problem. A mobile dynamic distributed system is characterized by i) the moving nodes and ii) their communication means (generally implemented in a communicating device and composed of a low level protocol along with the underlying technology). Both are important to define the dynamics of the distributed dynamic systems from an algorithmic point of view.

Indeed when nodes move very fast, one may consider that the distributed system is highly dynamic. However, from an algorithmic point of view, in

[☆]Partially supported by the project Celtic Plus CoMoSeF (Cooperative Mobility Services of the Future) and the research project Toredy funded by the *Région Picardie* and the *Fond Européen de Développement Régional* (European Regional Development).

case the communication is very efficient, this is not always true because nodes could exchange a large amount of data before each neighborhood changes. Inversely, a distributed system with nodes moving very slowly but having a very poor communication mean for exchanging data could be considered as highly dynamic. Indeed, a distributed algorithm could have difficulties to achieve its aim (e.g. to stabilize on a result or to continually ensure a behavior) because very few data are exchanged between each neighborhood changes. This could be summarized as follows: running a distributed algorithm on a network of snails communicating using their antenna could be more difficult than running the same algorithm in a high speed vehicular network using IEEE 802.11p.

Related work. Dynamic networks have been studied for their structural properties [14, 3]. In [5, 12], the notion of *evolving graphs* is proposed. It introduces a time domain in the graph theory to capture the evolving nature of dynamic networks. Among other concepts, the *journey* is a path in the successive topologies ("spatiotemporal path"). Evolving graphs are used in [17] for designing routing algorithms. In [23], *temporal reachability graphs* are introduced, where an edge exists if there is a journey between the extremities. By comparison, our modeling relies on a family of dynamic graphs, each of them allowing to send a given number of messages per edge.

Several works deal with the relationship between the dynamic network and the feasibility of distributed algorithms. In [19], some algorithmic constraints are studied in order a node succeed in flooding the network or routing a message to a known destination. In [6], distributed local computations by means of graph relabelings and evolving graphs are used to analyze and compare distributed algorithms. In [7], a synthesis is proposed through a unified framework called *Time-Varying Graph* (TVG). Several classes of TVG are identified and the impact of their properties on distributed algorithms is studied. In [22], TVG are modeled using a quadruplet $G_d = (V, E, T, \Phi)$ by adding the set of dates T and by considering the starting and ending dates of each edge. By comparison, our modeling abstracts the time by considering several dynamic p-graphs.

In [15], the relation between the presence of a stable connected spanning subgraph during successive rounds and the feasibility and the complexity of several distributed computing problems is studied. A faster information dissemination algorithm is proposed in [13] using network coding. In [2], the impact of the dynamics of the network on geocast routing is studied.

In [1], a new model is introduced for studying information dissemination in mobile networks. It relies on two parameters α and β such that within α time slots, some nodes having the information are connected to nodes that have not for at least β time slots. By comparison, we propose to model the dynamics of *every* dynamic distributed system in the aim of expressing conditions ensuring the desired behavior of distributed algorithms.

Contribution. Considering a distributed system for wireless mobile ad hoc networks, we propose to model the evolving topology by a *family of dynamic p-graphs* $\mathcal{F} = (\mathcal{G}^1, \mathcal{G}^2, \mathcal{G}^3, \ldots)$, where each dynamic p-graph $\mathcal{G}^p = (G_i^p)_{i \in \mathbb{N}^*}$ for $p \in \mathbb{N}^*$ is a sequence of graphs $G_1^p, G_2^p, \ldots$ successively observed during the execution, such that the duration of each edge in these graphs is long enough to send p messages[1]. We prove that, for any observation, the family of dynamic p-graphs is finite and we provide an algorithm to build them.

We show how the *dynamic p-graphs* encompass both, the nodes movement and the underlying communication means (technology and link protocols), permitting to perform comparisons between systems of different technologies regarding their ability to run a given algorithm. Moreover, considering a family of dynamic p-graphs for different values of p leads to a complete representation of a dynamic network. We show that these families allow comparing different dynamic distributed systems regarding to their ability to support a given distributed algorithm, which is not possible or very complex with existing models. Also, they permit to give conditions on algorithms; we give an example with the topology-based routing in dynamic networks.

Finally we illustrate the interest of our modeling for studying distributed algorithms in dynamic distributed systems by giving conditions of success of several examples: communication with acknowledgment, cyclic diffusion from a node and propagation of information with feedback.

Hence, our approach reveals to be very interesting for studying dynamic distributed systems and distributed algorithms.

Road map. In Section 2, a distributed system for wireless mobile ad hoc networks is presented. Section 3 introduces the observation and the *timed p-graphs* provided by the observation. They are used for defining *the families of dynamic p-graphs* in Section 4. Section 5 presents properties of our modeling for comparing different dynamic distributed systems and for inferring

[1] $\mathbb{N}^* = \mathbb{N} \setminus \{0\}$

conditions on algorithmic problems. We give an example with the limitation of topology-based routing. In Section 6, we illustrate the interest of our approach by studying three distributed algorithms commonly used in dynamic distributed systems. Concluding remarks end the paper.

2. Distributed System

In this section, we define the dynamic distributed system used in this paper.

2.1. System

We consider dynamic distributed system $\mathcal{S}(\mathcal{N}, \mathcal{G})$ defined by a *network* $\mathcal{N}$ and its *dynamics* $\mathcal{G}$.

Network. The *network* $\mathcal{N}$ is composed of communicating computing nodes. Each node owns a local memory and a sequential computing unit so that it is able to run a local algorithm. Nodes are not synchronized. The local memory of a node v is composed by its *private memory*, an *input memory* and an *output memory*.

Nodes are equipped with a communicating device. Communication are done through a simple action called `push`: when a *sender* node u executes `push(m)`, the value m stored in its output memory is copied into the input memories of some *receiver* nodes v_1, v_2, ..., v_k. In order for a node v to receive the data pushed by a node u, several *communication conditions* have to be fulfilled. These conditions are related to the underlying communication technology and protocol and may vary. One may cite: bounded distance, available medium, no collision with other close communication, etc.

The receivers v_i of a `push` action by a node u are not necessarily known from the sender u and do not know u itself before the reception. They are determined by the communication conditions and could be different from those of a previous `push` by the same node u. When a data m pushed by u is received by v, this indicates that there is a *communication link* (u, v) between u and v. A link (u, v) may exist while the link (v, u) does not exist. The capacity of a link is a single message.

Dynamics. This communication scheme can be implemented (among other examples) on a mobile wireless network relying on the WiFi. A `push` is implemented using a local broadcast. Nodes move and frame collisions add/delete links according to the communication range.

The *dynamics* $\mathcal{G}$ of the system is defined by the appearance and disappearance of its communication links and the number of messages it is possible to send in each link. This depends on the physical characteristic of $\mathcal{S}$ but also on the nodes' moves (scenario): for a given system, each scenario may give a different dynamics. Describing such a dynamics in a convenient way for the study of distributed algorithms is the aim of this paper.

2.2. Algorithm

A *configuration* c of a distributed system $\mathcal{S}$ represents the state of the whole system $\mathcal{S}$, including :

- the state of processors (memories),
- the state of the communication links (messages in transit on the links),
- the underlying topology (set of communication links).

We denote by $c_{\restriction proc}$ the configuration c reduced to the information related to the processors. Similarly, we denote $c_{\restriction mesg}$ (respectively $c_{\restriction topo}$) the configuration reduced to the information related to the link state (respectively to the topology). We have $c = c_{\restriction topo} \cup c_{\restriction mesg} \cup c_{\restriction proc}$ though in a non-dynamic distributed system, c is only defined by $c_{\restriction mesg} \cup c_{\restriction proc}$ because the topology is a parameter of $\mathcal{S}$ and does not change.

A *distributed algorithm* $\mathcal{A}$ is a collection of local algorithms running on every node of $\mathcal{S}$. Processors actions change the global system configuration. An *execution* e is a sequence of configurations $c_1, c_2, \ldots$, where c_1 is the *initial configuration* of the execution e. We denote by $\mathcal{E}$ the set of executions of the system $\mathcal{S}$. For any given execution $e \in \mathcal{E}$, we denote by $e_{\restriction topo}$ the sequence $c_{1\restriction topo}, c_{2\restriction topo}, \ldots$. It gives the evolution of the topology of the system $\mathcal{S}$. In other words the dynamics $\mathcal{G}$ of the system $\mathcal{S}(\mathcal{N}, \mathcal{G})$ is defined by $e_{\restriction topo}$ and it may change for every execution of the algorithm (different scenarios).

The *specifications* of a distributed algorithm $\mathcal{A}$ are given by means of a predicate $\mathcal{P}_{\mathcal{A}}$ defined on the executions $\mathcal{E}$. For $e \in \mathcal{E}$, if $\mathcal{P}_{\mathcal{A}}(e)$ is true, we say that the algorithm $\mathcal{A}$ satisfies its specifications on e. When it satisfies its specifications on all the executions of $\mathcal{E}$, we say that $\mathcal{A}$ satisfies its specifications on $\mathcal{S}$.

The graph $U(\mathcal{G})$ having for vertices the computing nodes of the system and for edges all communication links appearing in $c_{1\restriction topo}, c_{2\restriction topo}, \ldots$ is called

underlying graph of the system $\mathcal{S}(\mathcal{N}, \mathcal{G})$. To the contrary of $\mathcal{G}$, which models the dynamics of $\mathcal{S}$, the underlying graph $U(\mathcal{G})$ describes a fix topology.

If an algorithm $\mathcal{A}$ satisfies its specifications on the non-dynamic system $\mathcal{S}(\mathcal{N}, U(\mathcal{G}))$, we say that the system $\mathcal{S}(\mathcal{N}, \mathcal{G})$ satisfies the *minimum requirements* of the Algorithm $\mathcal{A}$. In other words, the system has the adequate dimension (in terms of memory, computing power, etc.) to run the algorithm; if $\mathcal{A}$ does not satisfies its specifications on the dynamic system $\mathcal{S}(\mathcal{N}, \mathcal{G})$, it can be concluded that this is due to the dynamics $\mathcal{G}$ of the system. A more formal definition is done in Section 5 after introducing our model. Such a concept is used to study the relationship between the algorithms and the dynamics of the distributed systems.

3. Observing the dynamics of a distributed system

In this section, we discuss about the observation of a dynamic distributed system. We then define the timed graphs relying on the clock of an external observer to introduce the dynamic p-graphs in the next section.

3.1. Transfer duration function

For modeling the dynamics $\mathcal{G}$ of a dynamic distributed system $\mathcal{S}(\mathcal{N}, \mathcal{G})$, we consider graphs. The communicating computing nodes defined in Section 2.1 are represented by the vertices. As we consider finite systems, there is a finite set of vertices in the graphs.

The communication links are observed when a `push` action succeeded in forwarding a message from a node to another one. However others ought to be observed. Hence we define edges as follows: an edge is a communication link that ought to be observed if a `push` action would had been done and if it did not interfere with other ones. Similarly, we will consider edges able to send several successive messages before disappearing.

We introduce the following definition, to capture the fact that, in a dynamic distributed system, two close nodes may not have enough time to communicate before moving.

Definition 1. *A* transfer duration function $\delta : \mathbb{N}^* \to \mathbb{R}$ *gives the time* $\delta(p)$ *required to send* p *successive messages between two nodes.*

Obviously, δ is a linear function and depends on the underlying communication technology. The following holds.

Remark 1. *Let $\delta : \mathbb{N}^* \to \mathbb{R}$ be a transfer duration function. Then $\delta(p) \leq p \times \delta(1)$ and $p < q \Rightarrow \delta(p) \leq \delta(q)$.*

To be convinced of that, suppose that $p < q$ and $\delta(p) > \delta(q)$. Then, for sending p messages, we can send q messages including $q - p$ empty. As an illustration, in case a reservation is required for sending a message, it is advantageous to send several consecutive messages when possible. Using the transfer duration function, we introduce the *timed p-edge* to model the fact that it was possible to transfer p messages between two nodes at a given date.

3.2. Timed p-graphs

For the purpose of the following definition, we introduce an external imaginary observer which is able to note all the events in the dynamic distributed system[2]. The observer uses its own clock to date the events (nodes do not have any global clock and remain unsynchronized). By recording the appearance and disappearance of edges at the date t_i ($i \in \mathbb{N}, t_i \in \mathbb{R}^+$), the observer produces an observation O constituted by a sequence of observed graphs $O = (G_{t_0}, G_{t_1}, G_{t_2}, \ldots)$.

Figure 1 displays the beginning of such an observation. To fix the ideas, we consider here a convoy of 4 vehicles. The communication range is represented for the Vehicle 2 at date t_0. When the inter-vehicles distance increases (resp. decreases), some edges disappear (resp. appear).

Definition 2. *Consider a dynamic distributed system and its transfer duration function δ. A* timed p-edge *denoted $(u, v)^{t,p}$ is observed between nodes u and v at date $t \geq \delta(p)$ iff the edge (u, v) is present continuously during the time interval $[t - \delta(p), t]$. We denote by $E^{t,p}$ the set of timed p-edges observed at date t.*

In other words, a timed p-edge is long enough for allowing the transfer of p consecutive messages. Note that a timed p-edge $(u, v)^{t,p}$ may exist while u never really sent messages on this link. The observer only noticed that nodes

[2]In a real dynamic distributed system, an external imaginary observer can be implemented either by a supervising mean or by a post-computation of the log of positions of all nodes. For instance, for studying swarm of UAVs, we use an optitrack® system. For studying fleets of vehicles, we analyze their GPS traces after the road experiments. The accuracy of an observation can be increased by using several such observers that merge their observations. In the following, for sake of simplicity we consider a single observer.

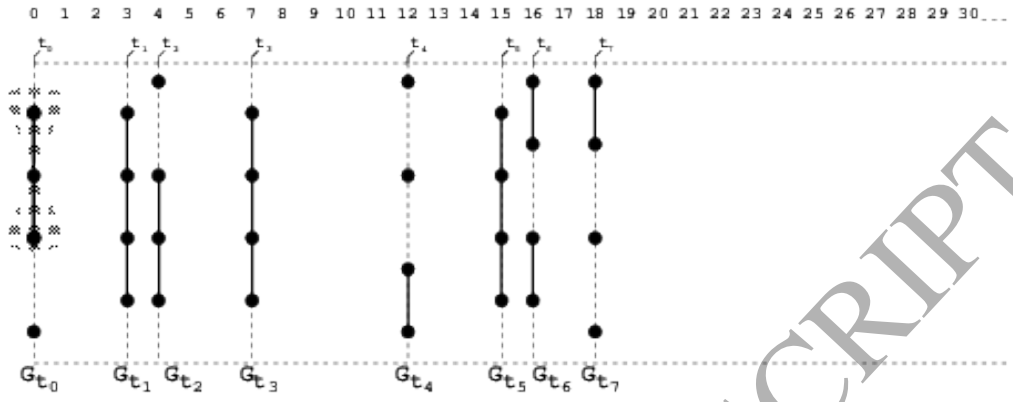

Figure 1: Observation $O = (G_{t_0}, G_{t_1}, G_{t_2} \ldots)$ of a dynamic distributed system, here a convoy of 4 vehicles. The time is provided here by an external imaginary observer (nodes are not synchronized).

u and v remain close each others long enough to send p consecutive messages. Note also that even if a timed p-edge $(u, v)^{t,p}$ exists, a communication may fail between u and v because some other communication conditions are not fulfilled (the duration is only one of such conditions). In particular, with usual wireless technologies, simultaneous receptions by a given node are not possible.

In a given system, if p is too large, no timed p-edge could be observed, meaning that no communication link exists during the time required to send p consecutive messages. In the same way, if $\delta(1)$ is very large (poorly efficient data transfer protocol), then no timed 1-edge could be observed, meaning that any communication attempt would fail in the dynamic distributed system. Hence, giving a communication mean, one may compute the maximal relative speed of the nodes allowing to communicate. Reciprocally, giving the nodes relative speed, one may compute the maximal transfer duration function, and then infer the kind of possible communication mean.

For instance, suppose that a vehicle passes away other vehicles in the convoy. Then, it is not sure it will be able to communicate with others because this depends on its speed and on the underlying communication protocol. This is solved from the observation knowing the transfer duration function δ. Starting from the timed p-edges, we introduce the *timed p-graphs*.

Definition 3. *Let $p \in \mathbb{N}^*$ and $t \in \mathbb{R}^+$. The* timed p-graph *at date t denoted*

by $G^{t,p}$ is defined by the pair $(V, E^{t,p})$, where V is the set of vertices and $E^{t,p}$ the set of timed p-edges observed at date $t \geq \delta(p)$.

4. Modeling the dynamics of a distributed system

Using the previously defined timed p-graphs, we propose a modeling for the dynamics of distributed system. For this purpose, we introduce the dynamic p-graphs. Their definition does not rely on the time, which is more convenient for specifying algorithmic properties.

4.1. Dynamic p-graphs

To define a dynamic p-graph, we introduce a binary relation on the timed p-graphs. Let define $\mathcal{R}^p$ as follows:

$$G^{\alpha,p}\ \mathcal{R}^p\ G^{\beta,p} \quad \overset{\text{def}}{\iff} \quad \left(\forall t \in [\min\{\alpha,\beta\}, \max\{\alpha,\beta\}] \quad E^{\alpha,p} = E^{t,p}\right)$$

In other words, two timed p-graphs are in relation by $\mathcal{R}^p$ if all other timed p-graphs between them (including themselves) have the same edges. This relation is reflexive: $G^{\alpha,p}\ \mathcal{R}^p\ G^{\alpha,p}$. It is symmetric because $G^{\alpha,p}\ \mathcal{R}\ G^{\beta,p} \Rightarrow G^{\beta,p}\ \mathcal{R}^p\ G^{\alpha,p}$. It is transitive because $\left(G^{\alpha,p}\ \mathcal{R}^p\ G^{\beta,p}\right) \wedge \left(G^{\beta,p}\ \mathcal{R}^p\ G^{\gamma,p}\right) \Rightarrow G^{\alpha,p}\ \mathcal{R}^p\ G^{\gamma,p}$. For this last property, consider $t = \beta$ belonging both in $[\min\{\alpha,\beta\}, \max\{\alpha,\beta\}]$ and $[\min\{\beta,\gamma\}, \max\{\beta,\gamma\}]$. Hence $\mathcal{R}^p$ is an equivalence relation on the timed p-graphs and we will consider classes of equivalence.

Since $\mathcal{R}^p$ relies on the increasing time of the external observer, we obtain by construction an ordered sequence of classes of equivalence, that we denote $G_1^p, G_2^p, G_3^p \ldots$ For an integer $i > 0$, the i^{th} class of equivalence G_i^p is defined by:

$$G^{\beta,p} \in G_i^p \quad \overset{\text{def}}{\iff} \quad \exists \alpha > 0 \text{ s.t. } G_i^p = G^{\alpha,p} \text{ and } G^{\alpha,p}\ \mathcal{R}^p\ G^{\beta,p}$$

Definition 4. *Let $p \in \mathbb{N}^*$. A* dynamic p-graph $\mathcal{G}^p = (G_1^p, G_2^p, G_3^p, \ldots)$ *of an observation O is the ordered sequence of the classes of equivalence induced by $\mathcal{R}^p$ on the timed p-graphs issued from O.*

Hence, instead of considering every timed p-graphs issued from an observation, we only retain the successive different ones. There is no more time in this definition. We now give an algorithm to determine the dynamic p-graphs starting from an observation.

4.2. Algorithm

For any given observation $O = (G_{t_0}, G_{t_1}, \ldots)$, the dynamic p-graphs of the observation are obtained as follows. We denote by T^0 the set of dates $t_0, t_1, \ldots$ provided by the observation.

1. Let $T^+ = \{t_0^+, t_1^+, \ldots\}$ the set of dates built from T^0 such that $t_i^+ = t_i + \delta(p)$ for any integer $i \geq 0$.
   Let $T^- = \{t_1^-, t_2^-, \ldots\}$ the set of dates built from T^0 such that $t_i^- = t_i - \delta(p)$ for any integer $i \geq 1$ such that $t_i \geq \delta(p)$.
   Let define $T = T^0 \cup T^+$ and $T' = T^0 \cup T^+ \cup T^-$.
2. Let $O' = (G_t)_{t \in T'}$ be the sequence of graphs defined as follows:
   - for any date $t \in T' \cap T^0$, $G_t = G_{t_i}$ where $t_i = t$;
   - for any date $t \in T' \setminus T^0$, $G_t = G_{t_i}$ where t_i is the largest date of T^0 which is smaller than t.
3. For $p \in \mathbb{N}^*$ and $t \in T$ such that $t > \delta(p)$, the timed p-graph at date t is obtained from the intersection of all graphs of O' present in the interval $[t - \delta(p), t]$:

$$G^{t,p} = \bigcap_{\substack{t' \in [t-\delta(p),\, t] \\ G_{t'} \in O'}} G_{t'}$$

4. To determine the dynamic p-graph of the observation O, we scan the ordered set of dates T, and for each date $t \in T$, we compute the timed p-graphs at date t. If this gives a graph different from the previously computed p-graph, then it gives the next p-graph (the next class of equivalence). The sequence of all p-graphs obtained from this computation defines the dynamic p-graph of the observation O.

This is summarized in Algorithm 1. Figure 2 displays the dynamic 1-graph built from the example of observation in Figure 1. The shaded rectangles display the windows of width $\delta(1)$ considered for computing the intersections.

4.3. Families of dynamic p-graphs

Any observation of a dynamic distributed system $\mathcal{S}$ admits a unique dynamic p-graph $\mathcal{G}^p$, for a given integer $p > 0$. We now introduce the family of dynamic p-graphs by considering all such integers p.

Algorithm 1 Dynamic p-graph construction

Require: the observation O, sets of dates T and T'
initialization: $G_0^p = \emptyset$; $\mathcal{G}^p = \emptyset$; $j = 1$; $O' = \emptyset$
for any date $t \in T'$ **do**
  $G_t = G_{t_i}$ where $t_i \leq t < t_{i+1}$
  $O' = O' \cup G_t$
  if $t \in T$ and $t > \delta(p)$ **then**
    $G_j^p = \bigcap_{t' \in [t-\delta(p),t]} G_{t'}$
    if $G_j^p \neq G_{j-1}^p$ **then**
      $\mathcal{G}^p = \mathcal{G}^p \cup G_j^p$
      $j{+}{+}$
    end if
  end if
end for

Definition 5. *The* family of dynamic p-graphs $\mathcal{F}$ *of a dynamic distributed system* $\mathcal{S}$ *is the sequence of dynamic* p*-graphs* $\mathcal{G}^p$ *for* $p \in \mathbb{N}^*$*:* $\mathcal{F} = (\mathcal{G}^1, \mathcal{G}^2, \ldots)$.

Figure 3 displays the family of dynamic p-graphs built from the example of observation in Figure 1.

Theorem 1. *For any dynamic distributed system* $\mathcal{S}$*, the family of dynamic* p*-graphs* $\mathcal{F} = (\mathcal{G}^1, \mathcal{G}^2, \mathcal{G}^3, \ldots)$ *of the dynamic distributed system is finite.*

Proof. Consider an edge of $\mathcal{S}$ observed from date t to date $t + \Delta$.

Either there is no $p \in \mathbb{N}^*$ such that $\delta(p) > \Delta$, i.e., $\delta()$ is bounded by Δ. In this case, any timed p-graph $G^{t,p}$ will include this edge. Then all the dynamic p-graphs will become identical regarding this edge.

Or there exists $q \in \mathbb{N}$ such that $\delta(q) > \Delta$. In this case the edge disappears from the timed p-graphs $G^{t,p}$ starting from $p \geq q$ (cf. Figure 3). Hence, all the dynamic p-graphs with $p \geq q$ are identical regarding this edge.

There is a finite number of nodes and then a finite number of possible edges. So there exists Δ and $q \in \mathbb{N}^*$ such that for any $p \in \mathbb{N}^*$, if $p > q$ then $\mathcal{G}^p$ is empty or $\mathcal{G}^p = \mathcal{G}^q$. This implies that the family is finite and that $\mathcal{G}^q$ is the last element of the family. □

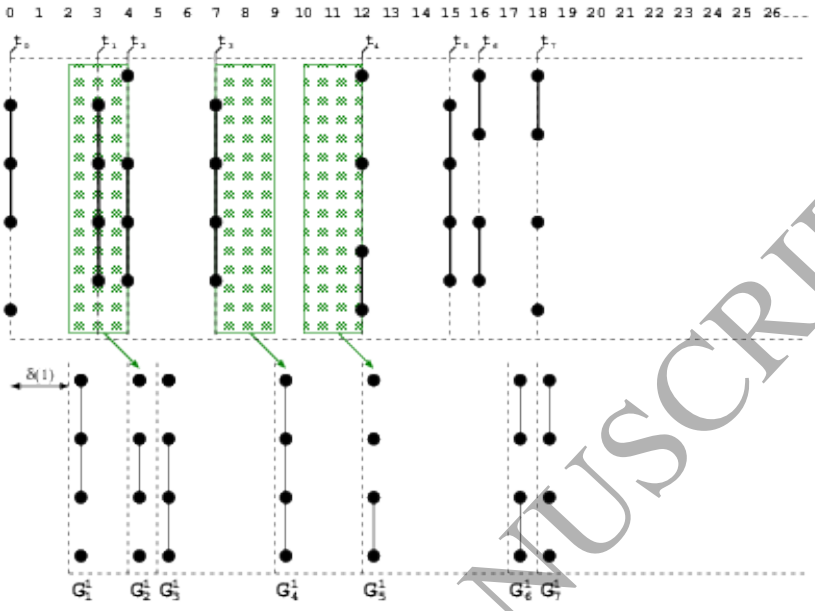

Figure 2: Dynamic 1-graph built from the observation of Figure 1 by considering intersections of graphs of the observation. The shaded rectangles display the windows of width $\delta(1)$ considered for computing the intersections.

In the rest of the paper, we will denote the family of dynamic p-graphs by $\mathcal{F} = (\mathcal{G}^1, \mathcal{G}^2, \mathcal{G}^3, \ldots, \mathcal{G}^q)$, with $\mathcal{G}^q$ the last element of the family.

We claim that the families of dynamic p-graphs fully characterize dynamic distributed systems as defined in Section 2. By comparison, the existing models (e.g evolving graphs, TVG, ...) for dynamic networks give no indication on the number of messages it is possible to send on edges. Two very different dynamic systems could lead to the same evolving graph while a given algorithm could work only in one of them. Moreover, a single dynamic p-graph gives no indication on the possibility to send $p+1$ messages on a given edge. By considering all the family of such graphs, we obtain a complete description of the dynamics abstracting both the structural properties and the communication technologies. It then allows some comparisons of dynamic distributed systems whatever the nodes speed or the communication mean are (underlying technology and communication protocol). Moreover it does not rely on the time (to the contrary of the TVG). It is then more adapted to the analysis of distributed algorithms, as we illustrate in the next section.

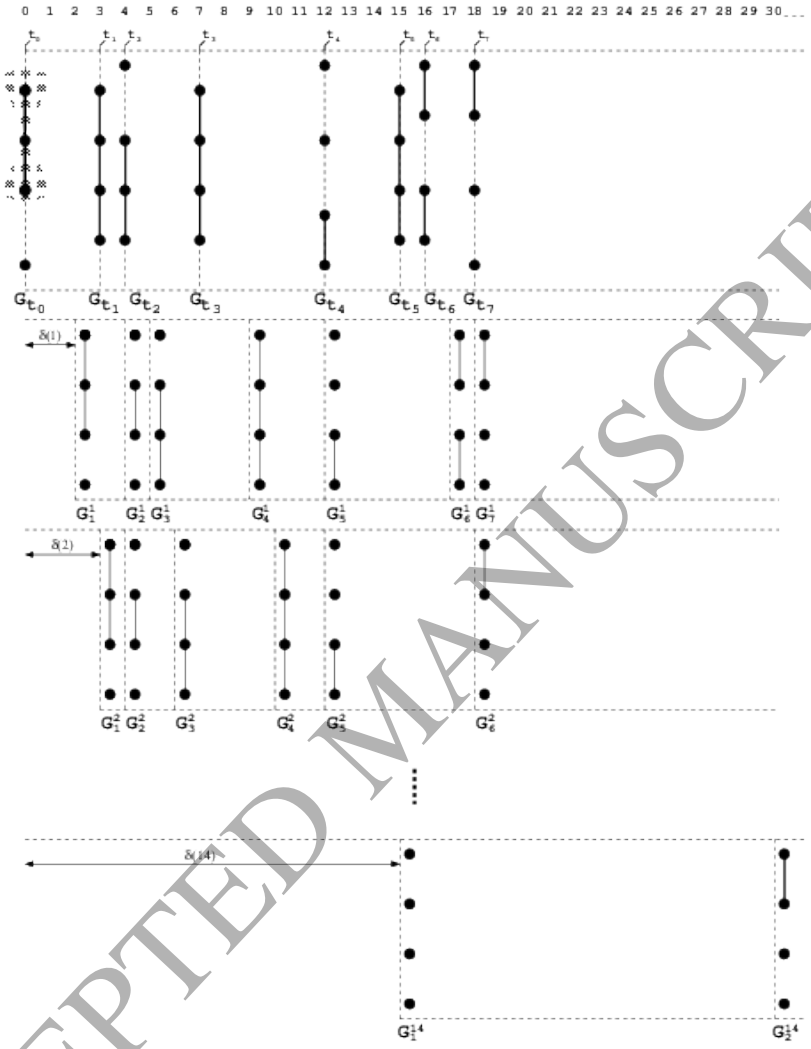

Figure 3: Family of dynamic p-graphs built from the observation of Figure 1.

5. Properties

In the previous section, we introduced the families of dynamic p-graphs. In this section, we give some of their properties useful for studying both distributed systems and distributed algorithms. First we show how the dynamic p-graphs allow comparing different dynamic distributed systems regarding to

their ability to support a given distributed algorithm. Then we explain how using the families of dynamic p-graphs for inferring properties on distributed algorithmic problems. As an illustration, we explicit some intrinsic limitations of topology-based routing in dynamic distributed systems.

5.1. Comparing dynamic distributed system

The families of dynamic p-graphs we introduced are more convenient than the observations for comparing dynamic distributed systems. We begin by showing that the relationship between observations and families is not straightforward.

Theorem 2. *Let $\mathcal{S}_a$ and $\mathcal{S}_b$ two different dynamic systems, O_a and O_b their respective observations, $\delta_a(p)$ and $\delta_b(p)$ their transfer duration functions and $\mathcal{F}_a$ and $\mathcal{F}_b$ their family of dynamic p-graphs. Then the following holds:*

1. $O_a = O_b \nRightarrow \mathcal{F}_a = \mathcal{F}_b$
2. $\mathcal{F}_a = \mathcal{F}_b \nRightarrow O_a = O_b$
3. $(\delta_a = \delta_b$ *and* $O_a = O_b) \Rightarrow \mathcal{F}_a = \mathcal{F}_b$
4. $(\delta_a = \delta_b$ *and* $\mathcal{F}_a = \mathcal{F}_b) \nRightarrow O_a = O_b$

Proof. Using Theorem 1, we note that two families of dynamic p-graphs $\mathcal{F}_a = (\mathcal{G}_a^1, \mathcal{G}_a^2, \ldots, \mathcal{G}_a^q)$ and $\mathcal{F}_b = (\mathcal{G}_b^1, \mathcal{G}_b^2, \ldots, \mathcal{G}_b^{q'})$ are equal if and only if $q = q'$ and $\mathcal{G}_a^p = \mathcal{G}_b^p$ for all $p \in \{1, \ldots, q\}$.

1. For the first case, we use a counter example. Let $\mathcal{S}_a$ a dynamic distributed system composed with two nodes u and v. Let O_a an observation of $\mathcal{S}_a$ such that the edge (u, v) appears at date $t = 2k$ and disappears at date $t = 2k + 1$ for $k \in \mathbb{N}$. Let $\mathcal{S}_b$ another distributed system composed with nodes u' and v' such that $\mathcal{S}_a \neq \mathcal{S}_b$ and let O_b an observation of $\mathcal{S}_b$ such that $O_a = O_b$. Let $\delta_a(p) = 2p$ and $\delta_b(p) = p$, $\forall p \in \mathbb{N}^*$. Then, for $p = 1$, we have $\mathcal{G}_a^1 = \emptyset$ and $\mathcal{G}_b^1 = O_a$. This proves that $O_a = O_b \nRightarrow \mathcal{F}_a = \mathcal{F}_b$.

2. We use again a counter example. Let $\mathcal{S}_a$, O_a, δ_a and $\mathcal{S}_b$ as described in the first part above. Let O_b an observation of $\mathcal{S}_b$ such that at date $t = 4k$ the edge (u', v') appears and at date $t = 4k + 1$ it disappears. Let $\delta_b(p) = 2p$. We then have $\mathcal{F}_a = \mathcal{F}_b = \emptyset$ while $O_a \neq O_b$. This proves that $\mathcal{F}_a = \mathcal{F}_b \nRightarrow O_a = O_b$.

3. By construction of the families (see Alg. 1), we have $(\delta_a = \delta_b \wedge O_a = O_b) \Rightarrow \mathcal{F}_a = \mathcal{F}_b$.

4. We use again a counter example. Let $\mathcal{S}_a$, O_a, δ_a and O_b as described in the second part above. Let O_b an observation of $\mathcal{S}_b$ such that no edge is observed. Let $\delta_b(p) = 2p$. Then we have $\delta_a = \delta_b$ and $\mathcal{F}_a = \mathcal{F}_b = \emptyset$ while $O_a \neq O_b$. This proves that $(\delta_a = \delta_b \wedge \mathcal{F}_a = \mathcal{F}_b) \not\Rightarrow O_a = O_b$. □

This theorem shows that two different dynamic distributed systems with different observations can give the same family of dynamic p-graphs. Likewise, two different dynamic distributed systems with the same observation can give two different families of dynamic p-graphs. Hence an observation does not fully characterize a dynamic distributed system. As a complement, the following theorem shows that the families of dynamic p-graphs allow comparing two different systems (that can be of different nature, see the example of snail network versus vehicular network introduced in Section 1), regarding their ability to run a given distributed algorithm.

Theorem 3. *Let $\mathcal{S}_a$ and $\mathcal{S}_b$ two different dynamic distributed systems and $\mathcal{F}_a$ and $\mathcal{F}_b$ their families of dynamic p-graphs.*

For any distributed algorithm $\mathcal{A}$ such that $\mathcal{S}_a$ and $\mathcal{S}_b$ satisfy both its minimum requirement, if $\mathcal{F}_a = \mathcal{F}_b$ then $\mathcal{A}$ has the same behavior on both systems.

Proof. Suppose that there is a distributed algorithm $\mathcal{A}$ which satisfies its specifications in $\mathcal{S}_a$ and does not satisfy them on in $\mathcal{S}_b$ while $\mathcal{S}_a$ and $\mathcal{S}_b$ satisfied the minimum requirements of $\mathcal{A}$. This means that there is at least one communication completed in $\mathcal{S}_a$ and that could not be done in $\mathcal{S}_b$. So there is at least one edge which has allowed sending a number of messages q in $\mathcal{S}_a$ larger than the number of messages q' it has allowed sending in $\mathcal{S}_b$. So $\mathcal{G}_a^q \neq \mathcal{G}_b^q$ implying that $\mathcal{F}_a \neq \mathcal{F}_b$, which contradicts the initial assumption. □

This theorem shows that if two dynamic distributed systems – whatever their nature – have the same family of dynamic p-graphs, then any distributed algorithm will give the same results on both dynamic distributed systems (providing both systems are able to run the algorithm, in term of capacity).

5.2. Studying distributed algorithms

In addition to the comparison of different dynamic distributed systems, the families of dynamic p-graphs allow inferring properties on the algorithms themselves. We begin by introducing the *p-graph at/from a configuration*

for dealing with communication which are possible just before or just after a given configuration. Then we give some properties about topological information in dynamic distributed systems.

Definition 6. *For a given observation O, let t_c be the date when the distributed system reaches a given configuration c. The* p-graph at configuration *c denoted by $G^p_{c|}$ is the timed p-graph $G^p_{t_c}$ observed at the date t_c. The* p-graph from configuration *c denoted by $G^p_{|c}$ is the timed p-graph $G^p_{t_c+\delta(p)}$ observed at the date $t_c + \delta(p)$.*

The graph $G^p_{|c}$ is composed of all edges allowing to send p messages from configuration c. In an execution $e = c_1, c_2, \ldots$, the graph $G^1_{|c_1}$ is the initial topology. It is important to note that the initial topology of a dynamic distributed system cannot be observed before a delay of $\delta(1)$ as there is no guarantee that two close nodes at $t = 0$ will not move before $\delta(1)$, avoiding any communication possibility (see the previous example with the vehicle passing the convoy in Section 3.2).

A dynamic distributed system is *stable* during the execution e if, for any integer $p > 0$, $G^1_{|c_1} = G^p_{|c_1}$, where c_1 is the initial configuration of e. This does not imply that no communication link appears nor disappears in the dynamic system but these changes are not significant from the point of view of the messages. The dynamic distributed system contains some *stable edges* during the execution e if $G^p_{|c_1} \neq \emptyset$ for every integer $p > 0$. It contains some *stable edges starting from configuration c* if $G^p_{|c} \neq \emptyset$ for every integer $p > 0$.

Obviously if an edge allows sending p messages, it also allows to send $q < p$ messages. This gives the following proposition (notation $G(V, E) \subseteq G'(V, E')$ means that any edge of E belongs to E'):

Proposition 1. *For any configuration c in an execution and for any integer p and q, if $p > q$, then $G^p_{|c} \subseteq G^q_{|c}$ and $G^p_{c|} \subseteq G^q_{c|}$.*

Proof. Assume that there exists a configuration c and two integers p and q with $p > q$ such that $G^p_{|c} \not\subseteq G^q_{|c}$. Then there is at least one edge e such that $e \in G^p_{|c}$ and $e \notin G^q_{|c}$. From Definition 6, $e \in G^p_{|c}$ and $e \notin G^q_{|c}$ means that e exists during the time interval $[t_c, t_c + \delta(p)]$ and does not exist during the whole time interval $[t_c, t_c + \delta(q)]$. However, since $p > q$, we have $\delta(p) \geq \delta(q)$ (Remark 1) and we obtain a contradiction. Thus $G^p_{|c} \subseteq G^q_{|c}$.

By the same way, we can prove that $G^p_{c\downarrow} \subseteq G^q_{c\downarrow}$. □

In other words, when p increases, there is less and less edges in a timed p-graph. A timed p-graph is built by withdrawing edges with too short duration. By comparison a *Temporal Reachability Graph* [23] is built by adding some edges. In the first case, there is an edge when several consecutive messages can be sent while in the second there is an edge when several hops (a journey) allow joining the extremities.

Now we defined the p-graph at/from a configuration, we illustrate the interest of the families of dynamic p-graphs for inferring properties on distributed computations. The next theorem gives a condition to obtain a valid local information related to the topology at several hops. By *valid*, we intend that, each time a local topology change is propagated, it is still up-to-date when it is received. For instance, suppose that u informs a remote node v about one of its new neighbor u'. Then v will receive this information later due to the distance between u and v. If u' is still a neighbor of u when v receives the information, the information is said *valid*.

Theorem 4. *Consider a dynamic distributed system and its family of dynamic p-graphs* $\mathcal{F} = ((\mathcal{G}^p)_{p\in\mathbb{N}^*})$. *If* $\mathcal{G}^{q-1} \neq \mathcal{G}^q$ *then no distributed algorithm can give a valid local description of the topology at q hops.*

Proof. If $\mathcal{G}^{q-1} \neq \mathcal{G}^q$ then, for any algorithm running in the dynamic distributed system, there exists at least one configuration c in the execution such that $G^{q-1}_{\downarrow c} \neq G^q_{\downarrow c}$. Let (u, u') be one of the edge existing in $G^{q-1}_{\downarrow c}$ and not in $G^q_{\downarrow c}$. The duration of this edge is then less than $\delta(q)$.

Sending a message from u up to q hops requires a delay of at least $q \times \delta(1)$. As $\delta(q) \leq q \times \delta(1)$ (Remark 1), a delay of at least $\delta(q)$ is required. Then, when the message sent by u is received by a node v such that $\text{dist}(u, v) = q$ in $G^q_{\downarrow c}$, edge (u, u') does not exist anymore. Thus, no valid local description of the topology can be sent at q hops if $\mathcal{G}^{q-1} \neq \mathcal{G}^q$. □

Theorem 4 gives an interesting indication regarding the use of local topology description for routing. Indeed, by studying the family of dynamic p-graphs of a given dynamic distributed system, it is possible to deduce the maximal distance from which it is no more interesting to forward local topological information.

6. Applications

In order to illustrate the practical usefulness of our modeling, we present in this section some examples.

6.1. Cumulative acknowledgment

Consider a bi-directional communication between two neighbor nodes u and v where u is the sender willing to send a bunch of messages to the receiver v. To enforce the robustness of the communication, v acknowledges the received messages. However, to better use the network resources, v may acknowledge several messages with a single acknowledgment. We define the *n-ack* algorithm as follows: the receiver acknowledges the n previously received messages using a single acknowledgment returned to the sender.

Obviously the nodes ability to exchange data using such an algorithm depends on the dynamic of the network. However the dynamic p-graphs permit to easily determine them.

Theorem 5. *Consider a dynamic distributed system $\mathcal{S}$ and its family of dynamic p-graphs $\mathcal{F} = ((\mathcal{G}^p)_{p\in\mathbb{N}^*})$.*

The set of pairs sender-receiver that have the ability to communicate using the n-ack algorithm in $\mathcal{S}$ from the configuration c is equal to the set of edges of $G^{n+1}_{|c}$.

Proof. If the edge (u, v) belongs to $G^{n+1}_{|c}$, it permits to send n messages and an acknowledgment (graphs are not oriented), satisfying the n-ack algorithm.

Reciprocally, suppose that starting from configuration c, Node u sent n messages to v that returned an acknowledgment received by u. Then the edge (u, v) permitted to send at least $n+1$ messages. Thus it belongs to $G^{n+1}_{|c}$. □

6.2. Cyclic diffusion of fragments

Consider a node that has to diffuse a large piece of data in n fragments. For this purpose, it sends to its neighbors n messages (each of them containing a single fragment) before restarting the process indefinitely. A neighbor will then receive the data after n messages, whatever is the first fragment received.

Such a use case is encountered in vehicular networks where a Road-Side-Unit (RSU) has to broadcast data to arriving vehicles. Depending on the

size of the data and the MTU[3], several fragments are required to receive the whole data. If the RSU broadcasts cyclically the n fragments (ie. it sends one fragment after the other in its vicinity, restarting from the first one when the last one has been sent), an approaching vehicle obtains the whole data as soon as it has received n messages from the RSU. Taking into account the packets losses rate leads to a similar result (with more messages) but to simplify, we consider no loss here.

The problem consists in determining the vehicles able to receive the complete data, for a given number of fragments n. This depends on the dynamic of the distributed system. By modeling the dynamic using the families of dynamic p-graphs, we obtain easily the answer.

Recall that the underlying graph $U(\mathcal{G}^p)$ is composed with the nodes of the system and all edges of the successive graphs $G_1^p, G_2^p, \ldots$ of the dynamic graph $\mathcal{G}^p$.

Theorem 6. *Consider a dynamic distributed system $\mathcal{S}$ and its family of dynamic p-graphs $\mathcal{F} = ((\mathcal{G}^p)_{p\in\mathbb{N}^*})$. Let u a node sending a large data using a cyclic diffusion of n fragments.*

Then all nodes v such that the edge (u, v) belongs to $U(\mathcal{G}^n)$ receives the n fragments composing the data.

Proof. Let denote by $G_1^p, G_2^p, G_3^p, \ldots$ the p-graphs composing $\mathcal{G}^p$. If (u, v) belongs to $U(\mathcal{G}^n)$, there exists an integer $k \in \mathbb{N}$ such that (u, v) belongs to the graph G_k^n. Hence the edge (u, v) permitted to send at least n successive fragments. Thus Node v received the whole data. □

6.3. Propagation information with feedback

Our last example focuses on the PIF algorithm. We propose an adaptation for dynamic distributed systems named DPIF and we prove a condition ensuring that DPIF satisfies its specifications. We begin by explaining the original PIF algorithm.

[3]MTU: Maximum Transfer Unit.

6.3.1. PIF definition

The propagation information with feedback (PIF) has been introduced in [20]. It builds a spanning tree to gather information in a fix network. Like most of general *wave algorithms* [21], the PIF algorithm works in two steps – both of them evoking a wave.

Consider a fixed network modeled by a graph $G(V, E)$ (where V is the set of vertices and $E \subseteq V \times V$ is the set of edges). The first step is a *flooding phase* started by the initiator node $u \in V$. During this step, the initiator sends a broadcast message to all its neighbors. When a node $w \in V$ receives a broadcast message for the first time, it considers the sender $v \in V$ as its parent by setting its local variable parent_w to v ($\text{parent}_w = v$). Then it forwards the broadcast message to all its neighbors except its parent. Behaving like this, a spanning tree is built in the network.

The second step is a *feedback phase* started by the leaves of the spanning tree. More precisely, each time a node v receives a broadcast message from a node w which is not its parent ($\text{parent}_v \neq w$), it sends back to w an acknowledgment message. When a node receives an acknowledgment from all its neighbors except its parent, it sends back to its parent an acknowledgment. When the initiator itself has received an acknowledgment from all its neighbors, the algorithm ends. Information of each node has been gathered by the initiator thanks to the acknowledgments sent by each node (except the initiator) to its parent.

Let denote by PIF(u) this algorithm starting from Node u. Let $V_u \subset V$ be the set of vertices of G belonging to the connected component of u and E_u be the set of corresponding edges: $E_u = (V_u \times V_u) \cap E$. We denote by $K_u(G)$ the subgraph of G composed with the set of vertices V_u and the set of edges E_u. PIF(u) gathers information from all nodes in V_u after building a spanning tree of $K_u(G)$. The height of this tree can reach $|V_u| - 1$ in some cases.

A variant of the PIF algorithm consists in stopping the flooding phase after a deepness of d; we denote it by $\text{PIF}_d(u)$. Here, the height of the tree built by $\text{PIF}_d(u)$ is at most d. Let denote by $V_{u,d} \subseteq V_u$ the vertices of V_u at distance less than or equal to d from u and by $E_{u,d} \subseteq E_u$ the set of corresponding edges: $E_{u,d} = (V_{u,d} \times V_{u,d}) \cap E$. We denote by $K_{u,d}(G)$ the subgraph of G composed with the set of vertices $V_{u,d}$ and the set of edges $E_{u,d}$. In a synchronous system, $\text{PIF}_d(u)$ gathers information from all nodes in $V_{u,d}$ after building a spanning tree of $K_{u,d}(G)$.

6.3.2. DPIF: PIF in dynamic network

The PIF algorithm is mainly applied on wired fixed networks. Obviously it cannot be used in dynamic networks efficiently because it assumes a stable spanning tree over the whole system. We then propose an adaptation of PIF_d for dynamic networks called DPIF_d where nodes communicate without knowing their neighbors by sending periodically some beacons in their vicinity using the `push` primitive. Note that periodic beacons are commonly used in dynamic networks to sense the environment (eg. vehicular networks, networks of robots...). In this context, we use piggybacking to implement the PIF_d on the top of the beacons, leading to a synchronous distributed system.

The node u that initiates the $\text{DPIF}_d(u)$ adds (d, h) to the next beacon it will push, where d is the deepness of the willing spanning tree and h is the height of the sender ($h = 0$ here because the sender is the root). Each neighbor receiving such a message for the first time learns about the running DPIF. They compute their own height ($h \leftarrow h + 1$) and adds (d, h) to the next beacon they will send (as the root did, except the height has been incremented). Moreover they plan to answer by adding all received data collected from now in the next $(2(d-h)+1)^{\text{th}}$ beacon. Acting like this, more and more nodes are involved in the DPIF until reaching the leaves where $h = d$. Such nodes add their own data to be collected into the next beacon to push. The parents of the leaves in the tree will receive such data and add them as well as their own data into the next beacon. From nodes to nodes, the collected data are gathered towards the root u.

For instance, if $d = 3$, a neighbor v of the root u contributes to $\text{DPIF}_d(u)$ when it receives the beacon from u. It will send the collected data to u $2(3-1)+1 = 5$ beacons latter, including all received data in this beacon. A neighbor w of v at distance 2 of the root will warn the leaves with its own beacon and will wait for $2(3-2)+1 = 3$ beacons to answer. When receiving such a beacon, the leaves answer with their next beacons because $2(3-3)+1 = 1$. Node w has then enough time to receive the answers from its sons in the tree before including them into its beacon in such a way Node v receives them. In turns, v does the same, waiting enough to receive the data from its sons before including them into its beacon. Finally the root u receives all the data before sending its $2(3-0)+1 = 7^{th}$ beacon since the beginning of the algorithm $\text{DPIF}_d(u)$.

6.3.3. Result

When starting $\text{DPIF}_d(u)$ on dynamic distributed system, it is expected that, as $\text{PIF}_d(u)$ would do on a fixed network, it gathers on u all data from nodes at distance less than or equal to d from u. Obviously, the dynamic of the system may prevent $\text{DPIF}_d(u)$ to satisfy such specifications.

Let formalize the problem. Consider a dynamic distributed system $\mathcal{S}$ and its family of dynamic p-graphs $\mathcal{F} = ((\mathcal{G}^p)_{p\in\mathbb{N}^*})$. The problem consists in determining, for a given deepness $d \in \mathbb{N}^*$ and a given configuration c_i, the nodes u of $\mathcal{S}$ such that $\text{DPIF}_d(u)$ starting at configuration c_i succeeds in gathering all data of nodes of $K_{u,d}(G^1_{|c_i})$. Alternatively, for a given node u, the problem consists in determining the maximal deepness d so that $\text{DPIF}_d(u)$ succeeds. Such questions are essential in vehicular networks where vehicles take benefit of data produced by sensors and calculators of close vehicles.

Obviously, the $\text{DPIF}_d(u)$ algorithm satisfies its specifications on a distributed system which is stable (see Section 5.2), or partially stable on $K_{u,d}(U(\mathcal{G}))$ where $U(\mathcal{G})$ is the underlying graph of the system (see Section 2.2). Nevertheless, thanks to the families of dynamic p-graphs, we propose a simple and less restrictive answer to the problem, relying only on graphs considerations, avoiding reasoning with the timing information of edges.

Theorem 7. *Consider a dynamic distributed system $\mathcal{S}$ and its family of dynamic p-graphs $\mathcal{F} = ((\mathcal{G}^p)_{p\in\mathbb{N}^*})$. Let d a positive integer.*

If there exists an integer $p > 2d$ such that $K_{u,d}(G^1_{|c_i}) = K_{u,d}(G^p_{|c_i})$, then $\text{DPIF}_d(u)$ starting from the configuration c_i will satisfy its specifications.

Proof. The $\text{DPIF}_d(u)$ algorithm starts with the first beacon of u containing $(d, 0)$. Before u sends its second beacon, the algorithm involves neighbors of u. Before u sends its $(d+1)^{th}$ beacon, the leaves are reached. Before it sends its $(2d+1)^{th}$ beacon, it has received all data of reached nodes. Since $K_{u,d}(G^1_{|c_i}) = K_{u,d}(G^p_{|c_i})$, every edge of $K_{u,d}(G^1_{|c_i})$ allows sending p beacons, meaning that it still exists when the root receives the collected data. □

Note that such a condition can easily be checked in practice after computation of the family of dynamic p-graphs from GPS traces using Algorithm 1.

7. Conclusion

In this paper, we introduced a new model of dynamic distributed systems: the dynamic p-graphs. This modeling takes into account the ability of the dynamic network to send consecutive messages on a link, encompassing both the nodes move and the capacity of the underlying communication technology. We claim that the dynamics of dynamic distributed systems can be fully described with the finite family of dynamic p-graphs for different values of $p \in \mathbb{N}^*$. The importance of such modeling for algorithmic issues is shown in this paper.

First, we have shown that dynamic distributed systems of very different nature can be compared according to their ability to support a given algorithm. Then we proved the intrinsic limitation of topology-based routing protocols for a given dynamic distributed system. Finally we illustrated the interest of our modeling by exhibiting conditions on the dynamics that ensure the desired behavior of three distributed algorithms commonly used in dynamic distributed systems.

Our future work will focus on the comparison of real dynamic distributed systems and on the proof of complex algorithms and protocols in the context of vehicular networks and swarm of UAVs.

Acknowledgment

The authors would like to thank the reviewers for their valuable comments that permitted to significantly improve this paper.

This work has been partially supported by the project Celtic Plus CoMoSeF (Cooperative Mobility Services of the Future) and the research project Toredy funded by the *Région Picardie* and the *Fond Européen de Développement Régional* (European Regional Development).

References

[1] A. F. Anta, A. Milani, M. A. Mosteiro, and S. Zaks. Opportunistic information dissemination in mobile ad-hoc networks: the profit of global synchrony. *Distributed Computing*, 25(4):279–296, 2012.

[2] R. Baldoni, A. Fernández Anta, K. Ioannidou, and A. Milani. The impact of mobility on the geocasting problem in mobile ad-hoc networks:

Solvability and cost. *Theor. Comput. Sci.*, 412(12-14):1066–1080, March 2011.

[3] P. Borgnat, E. Fleury, J.-L. Guillaume, and C. Robardet. Characteristics of the dynamic of mobile networks. In *Proc. BIONETICS 2009*, vol. 6811, pages 130–139.

[4] A. Boukerche, editor. *Algorithms and Protocols for Wireless, Mobile Ad Hoc Networks*. John Wiley and Sons, 2008.

[5] B.-M. Bui-Xuan, A. Ferreira, and A. Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *In International Journal of Foundations of Computer Science*, 14(2):267–285, 2003.

[6] A. Casteigts, S. Chaumette, and A. Ferreira. On the Assumptions about Network Dynamics in Distributed Computing *CoRR*, abs/1102.5529, 2011.

[7] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-varying graphs and dynamic networks. In *Proc. ADHOC-NOW 2011*, pp. 346–359.

[8] B. Ducourthial, Y. Khaled, and M Shawky. Conditional transmissions: performances study of a new communication strategy in VANET. *IEEE Transactions on Vehicular Technology*, 56(6):3348–3357, November 2007.

[9] B. Ducourthial, S. Khalfallah, and F. Petit. Best-effort group service in dynamic networks. In *Proceedings of the 22nd ACM SPAA*, Greece, June 2010.

[10] F. El Ali and B. Ducourthial. On-line videos and screenshot movies of the pth algorithm. https://www.hds.utc.fr/airplug/doku.php?id=en:doc:movies:start.

[11] F. El Ali and B. Ducourthial. A distributed algorithm for path maintaining in dynamic networks. In *International Workshop on Dynamicity (DYNAM'11), collocated with (OPODIS'11)*, Toulouse, France, December 2011.

[12] A. Ferreira. Building a reference combinatorial model for manets. *IEEE Network*, 18(5):24–29, 2004.

[13] B. Haeupler and D.R. Karger. Faster information dissemination in dynamic networks via network coding. Proceedings of the 30th Annual *ACM* Symposium on Principles of Distributed Computing, *PODC* 2011, San Jose, CA, USA, pages 381–390, 2011

[14] F. Harary and G. Gupta. Dynamic graph models. *Mathematical and Computer Modelling*, 25(7):79 – 87, 1997.

[15] F. Kuhn, N. Lynch, and R. Oshman. Distributed computation in dynamic networks. In *Proceedings of the 42nd ACM symposium on Theory of computing*, STOC '10, pages 513–522, New York, NY, USA, 2010. ACM.

[16] N. Meghanathan. Survey and taxonomy of unicast routing protocols for mobile ad hoc networks. *The International Journal on Applications of Graph Theory in Wireless Ad hoc Networks and Sensor Networks*, 1(1), 2009.

[17] J. Monteiro, A. Goldman, and A. Ferreira. Performance evaluation of dynamic networks using an evolving graph combinatorial model. *WiMob*, 2006.

[18] S. Olariu and M. C. Weigle, editors. *Vehicular Networks: From Theory to Practice*. CRC Press, 2010.

[19] R. ODell and R. Wattenhofer. Information dissemination in highly dynamic graphs. In *Proceedings of DIALM-POMC*, pages 104–110, 2005.

[20] A. Segall. Distributed network protocols. *IEEE Transactions on Information Theory*, 29(1):23 – 34, 1983.

[21] G. Tel. *Introduction to Distributed Algorithms*. Cambridge University Press, 1994

[22] K. Wehmuth, E. Fleury, and A. Ziviani. A New Model for Time-Varying Graphs. In *Temporal and Dynamic Networks: From Data to Models*, , March 2013.

[23] J. Whitbeck, M. Dias de Amorim, V. Conan, and J.-L. Guillaume. Temporal reachability graphs. In *Proceedings of ACM Mobicom*, Istambul, 2012.

Bertrand DUCOURTHIAL received the Ph.D. degree in Computer Science from Paris Sud University in 1999. He joined the Université de Technologie de Compiègne and the Heudiasyc lab. He became full Professor in 2010. His research deals with dynamic ad-hoc networks (e.g., VANET, UAVs): modeling, networking, distributed algorithms, security, robustness, embedded software (Airplug Software Distribution)... His work is funded by industrial, regional, national and European projects.

Dr. Ahmed Wade is postdoc in Heudiasyc laboratory, Université de Technologie de Compiégne. He studies dynamic vehicular networks. A. Wade defended his PhD in January 2014 at LaBRI (Laboratoire Bordelais de Recherche en Informatique), the computer science laboratory of the Université de Bordeaux, France. The subject deals with several problems of distributed algorithms in dynamic networks. Research interests of A. Wade concern dynamic networks and algorithms for mobile entity (agents).