Partition Participant Detector with Dynamic Paths in MANETs

Luciana Arantes, Pierre Sens, Gaël Thomas, Denis Conan, Léon Lim

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Partition Participant Detector with Dynamic Paths in MANETs

Luciana Arantes — Pierre Sens — Gael Thomas — Denis Conan — Léon Lim

N° 7002

Août 2009

ISSN 0249-6399 ISRN INRIA/RR--7002--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
PARIS - ROCQUENCOURT

Partition Participant Detector with Dynamic Paths in MANETs

Luciana Arantes , Pierre Sens , Gael Thomas * , Denis Conan , Léon Lim †

Thème : Réseaux, systèmes et services, calcul distribué
Équipe-Projet Regal

Rapport de recherche n° 7002 — Août 2009 — 18 pages

Abstract: Mobile ad-hoc networks, MANETs, are self-organizing and very dynamic systems where processes have no global knowledge of the system. Due to node failures, mobility, disconnection and new arrivals, the network is not fully connected and it is not always possible to statically establish end-to-end paths between nodes. In this paper, we propose a model that characterizes the dynamics of MANETs in the sense that it considers that paths between nodes are dynamically built and the system can have infinitely many processes but the network may present finite stable partitions. We also propose an algorithm that implements an eventually perfect partition participant detector $\Diamond\mathcal{PD}$ which eventually detects the participant nodes of stable partitions. It is characterized by both the strong partition participant completeness and eventual strong partition participant accuracy properties.

Key-words: Participant detector, MANET, Partitionable networks, Models for dynamic systems

* LIP6 - University of Paris 6 - INRIA
† Institut Télécom, Télécom & Management Sud Paris, UMR CNRS Samovar

Centre de recherche INRIA Paris – Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay Cedex
Téléphone : +33 1 39 63 55 11 — Télécopie : +33 1 39 63 53 30

Détecteur des participants de partitions dans les réseaux MANETs en utilisant des chemins dynamiques

Résumé : Les réseaux mobiles sans fils tels que les MANETs sont des systèmes très dynamiques et auto-organisants où les processus n'ont pas une connaissance globale du système. À cause des défaillances des nœuds, de leurs déconnexions et arrivées, le réseau ne peut être à tout moment complètement connecté et il n'est pas toujours possible d'établir des chemins statiques entre les nœuds. Dans ce rapport, nous proposons un modèle pour caractériser la dynamicité des MANETs. Notre modèle considère des chemins construits dynamiquement entre les nœuds avec un nombre infini de processus. Cependant, nous supposons qu'il existe des partitions stables et finies. Nous proposons aussi un algorithme qui implémente un détecteur des participants de partitions ultimement parfait $\Diamond\mathcal{PD}$ qui détecte à terme tous les participants des partitions stables. Ce détecteur est caractérisé par les deux propriétés de complétude forte et de justesse à terme forte sur les participants des partitions.

Mots-clés : Détecteur de participants, MANET, Réseaux partionnables, Modèles pour les systèmes dynamiques

1 Introduction

A mobile ad hoc network (MANET) is a self-organized dynamic system composed of mobile wireless nodes. It lacks a fixed infrastructure. Nodes do not have a global knowledge of the system and the number of participant nodes is unknown. The network is not fully connected and a node can only send messages to nodes that are within its transmission range. Hence, it may be that a message sent by a node should be routed through a set of intermediate nodes until reaching the destination node. Furthermore, links between nodes are considered unidirectional. For instance, it might happen that a node can receive a message from another node but has insufficient remaining energy to send it back a message.

Due to arbitrary failures, disconnections, arrivals, departures, or node movements, a MANET is characterized as an extremely dynamic system where links between nodes change over time. Thus, the temporal variations in the network topology implies that a MANET can not be viewed as a static connected graph over which paths between nodes are established before the sending of a message. A path between two nodes is in fact dynamically built, i.e., a link between two intermediate nodes of a path is not necessarily established beforehand but when one node sends a message to the following one in the path. Another impact of the dynamics of MANET is that lack of links between nodes partition them into components. A MANET is thus a *partitionable system* [13], i.e., a system in which nodes that do not crash or leave the system might not be able of communicating between themselves. Other examples of partitionable MANETs are those mobile networks where nodes are sparsely distributed (sparse MANETs) [9].

Collaborative applications [5], distributed monitoring [16], resource allocation management [2] are examples of applications that support partitioning and can thus go on running in multiple partitions (components). However, such partitions must present some *eventual stability* (or a *stability* whose duration is long enough) in order to ensure that those applications can progress and terminate. When some stability conditions eventually take place for some set of processes, the latter forms a partition whose members are stable in the sense that they do neither crash nor leave the partition, and new members are not accepted. Nevertheless, we consider that nodes can move inside the partition. Members of the partition are mutually reachable from each other through links that do not lose messages and that ensure that a message arrives within a bounded delay. In this paper, we denote such a partition a *stable partition*. Notice that it may happen that some nodes (e.g., a mobile node that keeps on moving) will never join a *stable partition*, i.e., the MANET may present some eventually "stable regions" or "connectivity islands" while the rest of the network has a dynamic behavior where a stable connectivity among nodes is not possible.

Motivations of the paper: The above discussion shows that there is a need for a model that takes into account the dynamics of MANETs, as well as their "stable regions". In other words, a MANET should be modeled as a dynamic system where several *stable partitions*, not completely isolated from others, can eventually exist. Furthermore, in such a context it would also be interesting to be able to detect the existence of such *stable partitions*, i.e., to provide an eventually perfect partition participant detector. Participant detectors are oracles

associated with each process. The invocation of the oracle by a process gives the set of processes that belongs to its partition. A participant detector can make mistakes, but if a process p belongs to a *stable partition* eventually and permanently, it will obtain the set of processes that belong to its partition. Similarly to failure detectors [8], the eventually perfect partition participant detector is thus characterized by both the *strong partition participant completeness* and *eventual strong partition participant accuracy* properties.

A second important motivation for our work was Chockler *et al.*'s [10] work where the authors specify group membership service for partitionable systems. Basically, a group membership service specifies the *view* a process has on the current component (i.e., partition) that it belongs to. They argue that fault tolerant applications on top of a partitionable system usually rely on such a service. Contrarily to our work, they consider an asynchronous static distributed system composed of N processes which is fully connected. A *stable component* is defined by the authors as "a set of processes that are eventually alive and connected to each other, and the link from any process in this set to any process outside the set is down". It is to some extent similar to *stable partition* except that processes of a *stable component* do not communicate to the other processes of the network. According to them, the liveness properties of membership service for partitionable system must hold only in *stable components*. Moreover, this property ensures that each process of a *stable component* installs a final view that corresponds to the members of such a component. This is called a *Precise Membership*. On the other hand, the authors state that the latter can only be guaranteed for a *stable component* if an *eventually perfect partition participant detector*[1] is provided.

Contributions of the paper: Its contributions are threefold: (1) A model that characterizes as much as possible the behavior, dynamics, and the mentioned "stability per region" of MANETs. It also defines the conditions that the system must satisfy for supporting *stable partitions*. Our model considers that infinitely many processes can exist in the system but stable partitions are finite. Nodes do not have a global knowledge of the system, the network is not fully connected, and path between nodes are dynamically built over time; (2) an *eventually perfect partition participant detector* whose algorithm considers our proposal model. It has been inspired by the algorithm proposed by Aguilera *et al.* in [1]; (3) a support for Chockler *et al.*'s requirements necessary for a possible implementation of a *precise membership* service on top of a partitionable MANET system. Notice that the construction of the membership service itself for partitionable systems is not the focus of this paper. Our aim is just to provide a support for those partition-aware applications that have been built on top of Chockler's specification for portability sake towards MANET.

The rest of this paper is organized as follows. Section 2 defines the dynamic model that characterizes the MANET and the existence of *stable partitions*. Our *eventually perfect partition participant detector* algorithm is presented in Section 3. Finally, some related works are described in Section 4 and Section 5 concludes the paper.

[1] denoted *eventually perfect failure detector* in the authors' paper.

2 System Model

We consider a dynamic distributed system S composed of infinitely many mobile nodes. Considering one process per node, the system consists thus of an infinite countable set Π of processes. Contrarily to a static environment, in a dynamic anonymous system, processes do not know Π.

Processes: There is one process per node and they communicate by message-passing through an underlying wireless network. The words node and process are therefore interchangeable. Processes have unique and totally ordered identifiers, i.e., $\forall p \in \Pi$, p is the process identifier (*pid*). A process knows its identity but does not necessarily know the identities of the other processes.

The topology of the network is dynamic due to node arrivals, departures, crashes, and mobility. Processes can fail by crashing. A *correct* process is a process that does not crash during a run; otherwise, it is *faulty*. A *faulty* node will eventually crash and does not recover.

Nodes can dynamically enter the system or leave it (voluntarily disconnect themselves from the system). A correct process that voluntarily disconnects leaves the system. A process that leaves the system re-enters it with a new identity (pid) and is considered as a new process.

Nodes can also be mobile and they can keep continuously moving and pausing. When a node moves, its neighborhood may change and, in consequence, the set of logical links. Mobility can lead to involuntary disconnections when a process is isolated from other processes.

Due to node movements, failures, arrivals or departures, links come up and down over the time. Thus, paths between two nodes are built dynamically as far as connectivity between intermediate nodes are established. Furthermore, since a process may not know the identity of the other processes, it cannot send a point-to-point message to them. It can just broadcast a message which will be received by those nodes that are within in its transmission range. Finally, links between nodes are considered unidirectional. For instance, it might happen that a node can receive a message from another node but has insufficient remaining energy to send it back a message.

Processes execute by taking steps. Each process has a local clock that count the number of step since a fixed date. Processes are considered synchronous in the sense that we assume that there are lower and upper bounds on the rate of execution (number of steps per time unit) of any non-faulty process. Thus, to simplify our model and without loss of generality, we assume that local processing takes no time. Only message transfers take time.

Dynamic Paths: To simplify the presentation of the model, we consider the existence of a discrete global clock which is not accessible to the processes. We take the range $\mathcal{T}$ of the clocks' tick to be the set of natural numbers.

One of the goals of our model is to define dynamic paths, i.e., a concept of end-to-end connectivity through transfer of messages along a sequence of processes.

We assume that our system does not modify the messages they carry, neither generate spontaneous messages nor duplicate them. Each message m has a unique identifier id_m. The following *integrity* property is satisfied: q receives a

message m from p at most once and only if p previously sent m to q. Messages can be delivered in out of order. We define $\mathcal{M}$ as the set of all possible messages.

We consider Lamport's happened-before relation between events [14]: $a \rightarrow b$ if event a causally precedes event b. Let $send_p(m)$ be the sending event of m on process p and $rec_p(m)$ be the reception event of message m on p.

We also define $\mathcal{F}$ a set of functions from $\Pi \times \mathcal{M}$ to $\mathcal{M}$ which takes a process p and a message m as input and outputs a message $m' = f(p,m) \stackrel{def}{=} f_p(m)$. Elements of $\mathcal{F}$ model algorithms executed by processes. Notice that the output of f_p can depend upon the state of p.

At first, we define the notion of reachability: a process q being reachable from p at time t means that if p sends a message m at time t then q receives a message that is causally dependent upon m. More formally:

Definition 1. Reachability: $\forall (p,q,t,m,f) \in \Pi \times \Pi \times \mathcal{T} \times \mathcal{M} \times \mathcal{F}$, *$q$ is reachable from p at time t for the message m with the algorithm f: if $q = p$ or if there exists $send_p(m)$ event at time t, then $\exists (p_1, p_2, .., p_n) \in \Pi^n$ with $p_1 = p$ and $p_n = q$, and $\exists (m_1, m_2, ..., m_{n-1}) \in \mathcal{M}^{n-1}$ with $m = m_1$ such that:*

$$\begin{array}{ll} (1) & \forall i \in [1, n-1],\ send_{p_i}(m_i) \rightarrow rec_{p_{i+1}}(m_i) \\ (2) & \forall i \in [2, n-1],\ m_i = f_{p_i}(m_{i-1}) \end{array}$$

We denote $S_{p,q,t,m,f}$ the set of sequences of processes $(p_1, p_2, ..., p_n)$ that satisfy the above definition. For all $P = (p_i)_{i \in [1,n]} \in S_{p,q,t,m,f}$, we define $trec(P,t,m,f)$ the time at which q *receives* m_{n-1} and we define $mrec(P,t,m,f) = m_{n-1}$.

It is important to notice that reachability does not require an end-to-end link between p and q at time t. The link is indeed built over the time.

We can now define the concept of dynamic path which models asynchronous end-to-end connectivity in a MANET.

Definition 2. Dynamic path (noted $p \rightsquigarrow_t q$): $\forall (p,q,t) \in \Pi \times \Pi \times \mathcal{T}$ *there exists a dynamic path between p and q at time t: if* $\forall (m,f) \in \mathcal{M} \times \mathcal{F}$, $S_{p,q,t,m,f} \neq \emptyset$.

We also define the concept of timely dynamic path where communication delay between processes of such a path is bounded.

Definition 3. Timely dynamic path (noted $p \stackrel{\delta_{pq}}{\rightsquigarrow}_t q$): *there exists δ_{pq} such that* $p \rightsquigarrow_t q \Rightarrow \forall (m,f) \in \mathcal{M} \times \mathcal{F}$, $\exists P \in S_{p,q,t,m,f}$ *such that* $trec(P,t,m,f) - t < \delta_{pq}$.

Moreover, we define a useful property that ensures that a node appears at most once in a timely dynamic path.

Definition 4. Simple timely dynamic path (noted $p \stackrel{\delta_{pq}}{\overset{*}{\rightsquigarrow}}_t q$): $p \stackrel{\delta_{pq}}{\rightsquigarrow}_t q$ *and* $\exists (p_i)_{i \in [1,n]} \in S_{p,q,t,m,f} : (i \neq j \Rightarrow p_i \neq p_j)$ *and* $trec((p_i)_{i \in [1,n]}, t, m, f) - t < \delta_{pq}$.

To summarize our definitions, we have $p \stackrel{\delta_{pq}}{\overset{*}{\rightsquigarrow}}_t q \Rightarrow p \stackrel{\delta_{pq}}{\rightsquigarrow}_t q \Leftrightarrow$ *q is reachable from p at time t for all messages m and all algorithms f.*

Eventual Group Stabilization: As previously explained, a membership service on top of a partitionable network can only be provided for those groups of processes which present an eventual stabilization. We denote each of these groups a *stable partition*. Basically, the *stable partition* of a process p, denoted

$\diamondsuit PART_p$, is composed of the same set of correct processes that can always communicate to each other through simple timely dynamic paths. Thus, processes within $\diamondsuit PART_p$ neither crash nor leave it, and new node arrivals in the partition do not take place. However, dynamic paths can evolve and processes can move inside the stable partition as long as they keep being connected by a simple timely dynamic path.

At first, we define the set of nodes that can be mutually reachable through a process p at a time t. These nodes form cycles which includes p. The nodes that compose the cycles of p, denoted by $Cycle_p(t)$, are then defined as follows:

Definition 5. $Cycle_p(t) \stackrel{def}{=} \{q \mid \exists (m, f, n) \in \mathcal{M} \times \mathcal{F} \times N : \exists (p_i)_{i \in [1,n]} \in S_{p,p,t,m,f} : \exists k \in [1, n] : p_k = q\}$.

Definition 6. *We define the stability property of a node p if there exists t such that:*

$$(1) \quad \forall t' \geq t : Cycle_p(t') = Cycle_p(t)$$
$$(2) \quad \forall t' \geq t : q, r \in Cycle_p(t') \Rightarrow q \overset{*}{\rightsquigarrow}_{t'} r$$
$$(3) \quad \exists N : \forall t_0 : |Cycle_p(t_0)| \leq N$$

A node p is $\diamondsuit$stable *if $\forall q \in Cycle_p(t)$, q has the stability property.*

Axiom 1 defines that the set of nodes of cycles of p does not change whereas Axiom 2 imposes the existence of timely links between all nodes of these cycles. Axiom 3 fixes a bound on cycles size since we make no assumption on the total number of nodes. Finally, a node is $\diamondsuit$*stable* if all nodes of its cycles are also $\diamondsuit$*stable*.

Now, we define the *Stabilization Time* of a $\diamondsuit$*stable* node as the minimal time ST_p that satisfies the above definition. ST_p is unknown.

A *stable partition*, denoted by $\diamondsuit PART_p$ of a $\diamondsuit$*stable* process p, is defined as follows:

Definition 7. $\diamondsuit PART_p \stackrel{def}{=} Cycle_p(ST_p)$

It is worth remarking that nodes of a *stable partition* are not necessarily isolated from other nodes of the network. Depending on the network connectivity, it might be the case that one or more nodes of a stable partition can send or receive messages to nodes which do not belong to their partition.

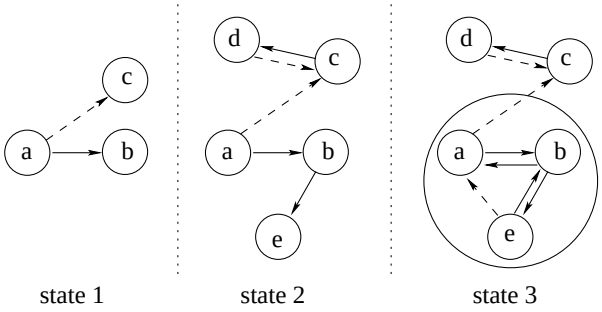

Figure 1: Illustration of a $\diamondsuit$*stable* process

Figure 1 illustrates the definition of $\Diamond stable$ nodes. All nodes on the figure are correct and the graph evolves from state 1 to state 3 and then remains in state 3. Solid arrows correspond to *timely dynamic paths*, otherwise the line is dashed. Before state 3, none of the nodes are $\Diamond stable$ and it does not exist any stable partition since there is no subset of processes that satisfies Definition 6. On the other hand, as there always exist timely paths between nodes a, b and e after state 3, these nodes are $\Diamond stable$ and form a partition, i.e., $\Diamond PART_a = \Diamond PART_b = \Diamond PART_e$.

3 Eventually Perfect Partition Participant Detector

Based on the system model defined in the previous section, we present in this section an algorithm for detecting the participants of a partition and then a sketch of proof which shows that this algorithm implements an eventually perfect partition detector $\Diamond\mathcal{PD}$. We also prove that our detector supplies the requirements for providing precise membership.

Each process p has locally an eventually perfect partition participant detector, denoted $\Diamond\mathcal{PD}$. When invoked, $\Diamond\mathcal{PD}$ returns to p the set of processes that are mutually reachable from p, i.e., those processes that it believes to belong to its partition. If p is a $\Diamond stable$ node, eventually, $\Diamond\mathcal{PD}$ will return the nodes that belong to the *stable partition* $\Diamond PART_p$.

Similarly to failure detectors, $\Diamond\mathcal{PD}$ is characterized by both the *completeness* and the *accuracy* properties. *Completeness* characterizes the capability of the $\Diamond stable$ node p of constructing an output set which contains the identification of the processes that belong to its partition while the *accuracy* characterizes the capability of that process of not including in such a set those processes which are not in its partition.

- *Strong partition participant completeness*: For each $\Diamond stable$ process p, if $q \in \Diamond PART_p$, then eventually p considers q as a member of its stable partition permanently.
- *Eventual strong partition participant accuracy*: For each $\Diamond stable$ process p, if $q \notin \Diamond PART_p$, then eventually p will no longer consider q as a member of its stable partition.

Note that if p is not a $\Diamond stable$ node the above properties not necessarily hold, i.e., the "eventually" and "permanently" characteristics of the properties can not be ensured.

Since processes do not know the identity of the other processes, they cannot send point-to-point messages to them. Thus, the only sending primitive provided to process p is the $broadcast_{nbg}$ primitive that allows p to send a message to all its current neighbors (nodes within its transmission range) without necessarily knowing their identity. Due to the dynamics of the system the set of neighbors of p can change during a run. A second remark is that a node q that received a broadcast message from p is not necessarily capable of broadcasting a message to p since links are unidirectional.

Algorithm 1 Implementation of Eventually Perfect Partition Participant Detector

```
1   Init:
2   Begin
3       { Processes supposed to be in ◇PART_p }
4       inPart ← {p}; output ← {p};
5       Timeout ← α;
6       set timer to Timeout;
7       broadcast_nbg(⟨ALIVE, p⟩);
8   End
9
10  Task T1: upon reception of (⟨ALIVE, path⟩)
11  Begin
12      If first node in path = p then
13          For all q: q appears after p in path do
14              inPart ← inPart ∪ {q};
15      Else
16          If p appears at most once in path then
17              broadcast_nbg(⟨ALIVE, path · p⟩);
18  End
19
20  Task T2: upon expiration of Timeout
21  Begin
22      If output ≠ inPart then
23          Timeout ← Timeout + 1;
24      output ← inPart;
25      set timer to Timeout;
26      inPart ← {p};
27      broadcast_nbg(⟨ALIVE, p⟩);
28  End
29
30  Task T3: when membership() is invoked by the upper layer
31  Begin
32      return(output);
33  End
```

3.1 Algorithm Description

Algorithm 1 implements an eventually perfect partition participant detector $\Diamond\mathcal{PD}$ for process p. By querying its local $\Diamond\mathcal{PD}$ (Line 30), process p obtains the current knowledge that the former has of the set of processes that belong to its partition (Line 32).

The local detector executes an initialization phase and then two concurrent tasks. At the *initialization* phase (Lines 4–7), it initializes its timer and sends to all its neighbors an $ALIVE$ message which includes just p.

Task $T1$ handles p's detector reception of an $\langle ALIVE, path\rangle$ message from those processes that have p as their neighbor. If $path$ is equal to $\langle p, \ldots\rangle$, p knows that its $\langle ALIVE, p\rangle$ message was forwarded through a cycle, i.e., all nodes that appear after p in $path$ are mutually reachable from it (Lines 13–14). Otherwise, if p does not appear in $path$ or appears just once, p's detector appends p to $path$ and forwards it to all its neighbors (Lines 16–17). Note that p's detector must forward the message even if p already appears once since it might be the case that there exists a cycle between q and r where p belongs both to the *simple path* from q to r and the *simple path* from r to q.

Task $T2$ is executed whenever the *timeout* expires. If the new set of nodes that p's detector believes to belong to p's partition ($inPart_p$) is different from the previous one, it increments the timeout value (Lines 22–23). This means that, if p is a $\Diamond$*stable* node, either the ST_p is not reached yet or it is reached but the timeout value is not enough for the message $\langle ALIVE, p\rangle$ sent from p to travel through the longest cycle from p. When both conditions happen, the set of processes in $inPart$, and thus in $output$, will always be the same. Finally, in Lines 24–27, p's detector initializes its timer and the variable $inPart$ and then broadcasts to all its neighbors an $ALIVE$ message that contains just p as reachable, as in the initialization phase.

3.2 Sketch of Proof

We present a sketch of proof of both the *strong partition participant completeness* and *eventual strong partition participant accuracy* properties of algorithm 1 that characterize the eventually perfect participant partition detector $\Diamond\mathcal{PD}$.

For simplicity's sake of the text, "p's detector" is just noted as p.

The key of the proof is to show that if eventually and permanently $q \in Cycle_p(t)$ then eventually and permanently $q \in output_p$ (*strong partition completeness*); otherwise, there exists a time after which $q \notin output_p$ permanently (*eventual strong partition accuracy*).

Definition 8. *Let p be a $\Diamond$stable process. We denote δ_p the maximum of $\{\delta_{qr} \mid (q,r) \in \Diamond PART_p \times \Diamond PART_p\}$. The value of δ_p exists because the number of nodes in $\Diamond PART_p$ is finite: $\Diamond PART_p = Cycle_p(ST_p)$ which is bound by Axiom 3 of Definition 6. Therefore, the number of possible process pairs (q,r) of $\Diamond PART_p \times \Diamond PART_p$ is finite. Moreover, the number of possible δ_{qr} is also bounded since paths of $\Diamond PART_p$ are timely (Axiom 2). However, δ_p is unknown.*

Definition 9. *We denote $inPart_p(t)$ the value of inPart of process p at time t and $output_p(t)$ the value of output of process p at time t. It is worth mentioning*

that after the initialization phase where p is added to inPart (Line 4), it is never more removed from inPart.

Lemma 1. *Let p be a $\Diamond$stable process and $q \in \Diamond PART_p$. $\exists \Delta : \forall t \geq ST_p : \exists t'' \geq t : q \in inPart_p(t'')$ and $t'' - t \leq \Delta$.*

Proof. Let $q \in \Diamond PART_p$ and $t \geq ST_p$. If $q = p$, then permanently $q \in output_p(t)$ thanks to the Lines 4, 24 and 25 of the algorithm. Otherwise, by definition of $\Diamond PART_p$, $q \in Cycle_p(t)$ and Axiom 2 of Definition 6 implies that $p \overset{*}{\rightsquigarrow}_t q$. We define the message $m = \langle ALIVE, p \rangle$ and the function $f_p(msg) = msg \cdot p$. Let $P = (p_i)_{i \in [1,k]} \in S_{p,q,t,m,f}$ be the sequence of processes that belong to a simple timely path between p and q at time t. Let $t' = trec(P, t, m, f)$. By definition, $\Diamond PART_p = Cycle_p(t')$ and Axiom 2 of Definition 6 implies also that $q \overset{*}{\rightsquigarrow}_{t'} p$. We define the message $m' = \langle ALIVE, p_1 \cdots p_k \rangle$. Let $P' = (p_i)_{i \in [k,n]} \in S_{q,p,t',m',f}$ be the sequence of processes that belong to a simple timely path between q and p at time t'. By the definition of simple paths, each p_i appears at most once in the sequences $(p_i)_{i \in [1,k]}$ and $(p_i)_{i \in [k,n]}$. We consider now the sequence $(p_i)_{i \in [1,n]}$. By construction, $p_1 = p_n = p$, $p_k = q$ and a process appears at most twice in the sequence.

Notice that for all $i \in [1, n]$, f_{p_i} is equivalent to algorithm 1. This is true since the condition at Line 16 holds for all p_i with $i \in [2, n-1]$ because process p_i appears at most once in $\langle ALIVE, p_1 \cdots p_{i-1} \rangle$ since it appears at most twice in $\langle ALIVE, p_1 \cdots p_{i-1} \cdot p_i \cdots p_n \rangle$.

To conclude, p will add $p_k = q$ in $inPart_p$ at $t'' = trec(P', t', m', f)$. Moreover, each of the link is timely and we have therefore $t'' - t' \leq (n-k)\delta_p$ and $t' - t \leq k\delta_p$, i.e., $t'' - t \leq n\delta_p$. Finally, a process appears at most twice in $(p_i)_{i \in [1,n]}$ and each of them is in $\Diamond PART_p$ which is bounded by N (Axiom 3 of Definition 6). We have therefore $n \leq 2N$ and $t'' - t \leq \Delta \overset{def}{=} 2N\delta_p$. □

Lemma 2. *Let p and q be two $\Diamond$stable processes such that $q \in \Diamond PART_p$. q is removed from $output_p$ a finite number of times.*

Proof. The proof of this Lemma is by contradiction. We suppose that q is removed an infinite number of times from $output_p$. Thus, q is also infinitely removed from $inPart_p$ since $output_p$ is only updated at Line 24 when $inPart_p$ is assigned to it. However, whenever q is removed from $inPart_p$, Lemma 1 ensures that q will be added later in $inPart$. Hence, at the next timeout expiration after the addition of q in $inPart$, the condition $inPart_p \neq output_p$ will hold. Due to our assumption, such a condition will hold an infinite number of times and thus, because of Lines 22–23 of the algorithm, the timeout value will grow indefinitely and will become higher than the Δ defined in Lemma 1 at a time T.

Since the timeout value is greater than Δ, the same Lemma 1 ensures that q will be in $inPart_p$ at a time t before the expiration of each timeout. Moreover, q will remain in $inPart_p$ from t to the timeout expiration since processes in $inPart$ can only be removed at Line 26 of the algorithm. At each timeout expiration, q will therefore be in $inPart$ and copied to $output_p$. Thus, q will always remain in $output_p$ what is contradictory with the initial assumption. □

Lemma 3. *Let p be a $\Diamond$stable process and $q \in \Diamond PART_p$. $\exists\ t \geq\ ST_p : \forall t' > t$: $q \in output_p(t')$.*

Proof. Thanks to Lemma 2, q is removed from $output_p$ a finite number of times. Let t be the last time of its removal. After t, q will remain in $output_p$. □

Lemma 4. Strong partition completeness*: Let p be a $\Diamond$stable process.* $\exists t : \forall t' > t : \Diamond PART_p \subset output_p(t')$.

Proof. Lemma 3 shows that for every q such that $q \in \Diamond PART_p$, there exists a time t_q, such that q remains definitely in $output_p$ after t_q. $\Diamond PART_p$ is finite (Axiom 3 of Definition 6) and there exists thus a time $t = \max\{t_q | q \in \Diamond PART_p\}$ such that all processes of $\Diamond PART_p$ remain definitely in $output_p$. □

Lemma 5. *Let p be a $\Diamond$stable process. There exists a time T such that if p receives an alive message of the form $\langle ALIVE, p \cdot p_2 \cdots p_{n-1} \rangle$ at a time $t' > T$, then $\forall i \in [2, n-1]$, $p_i \in \Diamond PART_p$.*

Proof. First, remark that if p receives a message of the form $\langle ALIVE, p \cdot p_2 \cdots p_{n-1} \rangle$ at a time t_r, then there exists t_e such that $\forall i \in [2, n-1]$, $p_i \in Cycle_p(t_e)$. Indeed, the reception of the message at t_r implies its emission at t_e and the content $p \cdot p_2 \cdots p_{n-1}$ of the message defines its path which is a cycle.

We consider now the set $ALIVES_p(t_e)$ the set of alive messages of the form $\langle ALIVE, p \cdot p_2 \cdots p_{n-1} \rangle$ received by p such that $\forall i \in [2, n-1]$, $p_i \in Cycle_p(t_e)$. This set is bounded because $Cycle_p(t_e)$ is bounded for all t_e (Axiom 3 of Definition 6) and because a process does not appear more than twice (Line 16 of the algorithm). For all t_e we can therefore consider the reception time $last(t_e)$ in p of the last message of $ALIVES_p(t_e)$.

We define $T = max\{last(t_e) \mid t_e < ST_p\}$. By construction of T, if p receives an alive message of the form $\langle ALIVE, p \cdot p_2 \cdots p_{n-1} \rangle$ at $t' > T$, it was clearly emitted by p at a time $t_s \geq ST_p$. Moreover, the initial remark of the proof shows that $\forall i \in [2, n-1]$, $p_i \in Cycle_p(t_s)$. Axiom 2 of Definition 6 implies that $Cycle_p(t_s) = Cycle_p(ST_p) = \Diamond PART_p$ and we have $\forall i \in [2, n-1]$, $p_i \in \Diamond PART_p$.

□

Lemma 6. *Let p be a $\Diamond$stable process, $\exists u : \forall t' \geq u : q \in inPart_p(t') \Rightarrow q \in \Diamond PART_p$.*

Proof. Let t be the bounded time of the previous lemma and u be the next timeout expiration after t. If $q \in inPart_p(t')$ with $t' \geq u$, q was added between u and t' upon the reception of a message $\langle ALIVE, p_1 \cdots p_{n-1} \rangle$ with $p_1 = p$ by Line 14 of the algorithm. The previous lemma ensures that $q \in \Diamond PART_p$, what concludes the demonstration. □

Lemma 7. Eventual strong partition accuracy*: Let p be a $\Diamond$stable process.* $\exists t : \forall\ t' \geq t : \forall q \notin PART_p \Rightarrow q \notin output_p(t')$.

Proof. Follows directly from Lemma 6. □

Theorem 1. *Let p be a $\Diamond$stable process.* $\exists u \in \mathcal{T} : \forall t > u, \forall t' > u : output_p(t') = output_p(t) = \Diamond PART_p$.

Proof. Lemma 4 shows that there exists a time u_1 such that $\forall t > u_1, \Diamond PART_p \subset output_p(t)$ while Lemma 6 shows that there exists a time v such that $\forall t > v : inPart_p(t) \subset \Diamond PART_p$. Moreover, $output_p$ is a copy of $inPart_p$ when

the timeout expires (Line 24). We define u_2 as the time of the first timeout expiration after the time v. $\forall t > u_2, output_p(t) \subset \Diamond PART_p$. Finally, we define $u = max\{u_1, u_2\}$, and we have $\forall t > u$, $\forall t' > u$, $output_p(t) = \Diamond PART_p = output_p(t')$. □

Theorem 2. *Algorithm 1 implements a partition participant detector $\Diamond\mathcal{PD}$ for $\Diamond$stable processes.*

Proof. Consider a $\Diamond$*stable* process p. To satisfy the *strong partition participant completeness* property, we must prove that eventually $\Diamond PART_p$ is permanently included in the *output* set of p. This claim follows directly from Lemma 4. To satisfy the *eventual strong partition participant accuracy* property, we must prove that there exists a time t after which $q \notin \Diamond PART_p$ is no longer included in the *ouptut* set of p. This claim follows directly from Lemma 7 and the theorem follows.

□

3.3 Precise Membership

In [10], Chockler *et al.* consider a static distributed partitionable network composed of N processes fully connected by unidirectional links. Nodes and links can crash. A *stable component* is defined to be a set of correct processes that are eventually connected to each other and for which links to them from all the other processes are down. Their definition of *stable component* corresponds to our *stable partition* definition except that *stable components* are isolated from the other nodes of the network.

The authors state that the liveness properties of membership service for partitionable system must hold only in *stable components* and if an *eventually perfect participant detector* is provided. In this case, the service offers a *precise membership*, i.e., it delivers the same last view to all members of a *stable component*: for every *stable component* C, there exists a view V with the members of C such that V is the last view of every process in C.

We can easily verify that our definition of *stable partition* and *eventually perfect participant detector* supply the requirements for providing *precise membership* in MANETs.

Lemma 8. *If p is a $\Diamond$stable process then $\forall q \in \Diamond PART_p$, $\Diamond PART_p \subset \Diamond PART_q$.*

Proof. We define $t = max\{ST_p, ST_q\}$. Let $(q, r) \in \Diamond PART_p \times \Diamond PART_p$. We will show that $r \in \Diamond PART_q$. First, $t \geq ST_p$ and we have therefore $(q, r) \in Cycle_p(t) \times Cycle_p(t)$. The axiom (2) of the definition 6 ensures that $q \overset{*}{\rightsquigarrow}_t r$. Let $(m, f) \in \mathcal{M} \times \mathcal{F}$. By definition of dynamic path, $\exists P = (p_i)_{i\in[1,k]} \in S_{q,r,t,m,f}$. Let $m' = mrec(P, t, m, f)$ and $t' = trec(P, t, m, f)$. We have $q \in \Diamond PART_p = Cycle_p(t')$ and $r \in \Diamond PART_p = Cycle_p(t')$ and the axiom (2) ensures also that $r \overset{*}{\rightsquigarrow}_{t'} q$: there exists $P' = (p_i)_{i\in[k,n]} \in S_{r,q,t',f_r(m'),f}$. We verify immediatly that $(p_i)_{i\in[1,n]} \in S_{q,q,t,m,f}$ and therefore that $r \in Cycle_q(t)$. Finally, $t \geq ST_q$ and $r \in \Diamond PART_q$. □

Lemma 9. *If p is a $\Diamond$stable process then $\forall q \in \Diamond PART_p$, $\Diamond PART_p = \Diamond PART_q$.*

Proof. $\Diamond PART_p \subset \Diamond PART_q$ thanks to the lemma 8. Moreover, $p \in \Diamond PART_p$, therefore q is a $\Diamond stable$ process and $p \in \Diamond PART_q$. The previous lemma also ensures that $\Diamond PART_q \subset \Diamond PART_p$. □

Theorem 3. Precise membership. *Eventually, $\Diamond \mathcal{PD}_p$ provides the same last participant view for all members of $\Diamond PART_p$*

Proof. p is a $\Diamond stable$ process. The strong partition completeness and the eventual strong partition accuracy of the algorithm ensure that there exists t_p such that $\forall t' \geq t_p$, $output_p(t') = \Diamond PART_p$. $\forall q \in \Diamond PART_p$, q is also a $\Diamond stable$ process and there exists t_q such that $\forall t' \geq t_q$, $output_q(t') = \Diamond PART_q$. The lemma 9 ensures that $\Diamond PART_p = \Diamond PART_q$ and therefore, after the time $max\{t_p, t_q\}$, we have $output_p = output_q = \Diamond PART_p = \Diamond PART_q$. □

4 Related Work

Similarly to our approach, some articles, [4], [15], [20], propose a model for dynamic systems such as MANET or peer-to-peer systems. However, none of them have considered dynamic construction of paths or the existence of several stable partitions.

Like in our work, in [15] the authors state that a dynamic system must present some stability period in order to guarantee progress and termination of the computation. However, in their work, there exists just a single *reliable core cluster* during a period of stability which consists of the minimal number of nodes that have to be simultaneously alive during a long enough period in order for the whole system to be able to progress. Hence, in their approach, it is not possible to have several *stable* groups simultaneously as in our approach. Furthermore, the number of processes in each run is bounded and links are considered to be bidirectional.

In [20], the authors also consider that a dynamic system can be characterized by perturbed periods followed by *quiescent* periods, i.e., periods where no more arrivals or departures take place. They then study the problem of overlay network connectivity in dynamic distributed systems. The paper shows that there is no protocol that can ensure such a connectivity during perturbed periods since network partitions can happen. Notice that even if the problem of network partition is considered during perturbed periods, this work is interested in the eventual connectivity of the overlay, i.e., a stable period where there is no partition.

In [4], the authors propose a model for dynamic systems where two parameters, the *number of nodes* (in a run or in all runs) and the *diameter* of the network, can be characterized (e.g., bounded/unbounded, known/ unknown) depending on the dynamics of the system. The first parameter allows to model continuous arrival and departure of nodes from the system while the second one allows to circumvent the impossibility of a node to have a global point-to-point connectivity view of the network. However, their model does not provide a means for characterizing partitionable networks neither dynamic paths.

In [6], the authors have introduced the notion of evolving graphs in order to model the temporal dependency of paths in dynamic systems such as MANET or DTN (disruption tolerant networks). Concisely, an evolving graph is a time-step indexed sequence of subgraphs, where the subgraph at a given time-step

corresponds to the network connectivity at the time interval indicated by the time-step value. To this end, each node or link has a "presence schedule" that indicates the moment during which the node takes part to the system. Like in our model, evolving graphs capture the notion of *path over time*. However, evolving graphs are based on time-step schedulers and path over time can not be characterized as timely. Furthermore, they do not support infinitely many nodes.

Aguilera *et al.* present in [1] a heartbeat failure detector, $\mathcal{HB}$, for partitionable network. The output of the failure detector at each process p is an array with one entry for each process of the system. The heartbeat sequence of every process not in the same partition of p is bounded. Our partition participant detector algorithm is inspired by this work. Contrarily to our approach, in the authors' work, the system is considered to be a fully-connected static one, the number of nodes of the system is known, nodes do not move or leave the system, and all links are fair lossy. Moreover, the output of $\mathcal{HB}$ at p is not the set of processes that belong to p's partition.

Chockler *et al.* [10] and Babagaolu *et al.* [3] have extended the definition of eventually perfect failure detectors to partitionable environments in order to provide a membership service. Basically, these detectors, as our *partition participant detector*, eventually detect mutual reachability among processes. Similarly to our approach, in [10], the failure detector behaves like an eventually perfect one provided a *stable component* exists. However, in both works, the considered partitionable systems are static and initially the network is fully connected.

In [7], the authors define a *participant detector* for self-organized networks (MANET). Like in our approach, both the identity and the number of nodes in the network are not initially known. However, the network is considered to be always connected through reliable bidirectional links and the *participant detectors* are defined by the authors for discussion about the minimal information that processes must have about the other participants in order to make the problem of consensus with unknown membership (CUP) solvable. Thus, a *participant detector* neither considers the physical topology of the network nor possible partitions but just outputs a view of the network.

Nesterenko and Schiper propose in [17] the *eventual reachability failure detector* $\diamondsuit\mathcal{R}$ which outputs a quorum to each process. They define the concept of a *reachabiliy graph* R that is a direct graph in which the nodes of R are the processes of the system and there is an edge from p' to p in R if the quorum outputted by p contains p'. The authors state that the concept of $\diamondsuit\mathcal{R}$ can be extended to partitionable networks if the *completness* and *intersection* properties of $\diamondsuit\mathcal{R}$ are reduced to processes of the same partition. Although the assumptions for the considered system are different from ours (the system is not dynamic, the number and identity of nodes are known, the links are reliable) and no implementation of $\diamondsuit\mathcal{R}$ is given, their approach is similar to ours since each process outputs a quorum which contains the membership view the process has of the system or the partition in the case of partitionable systems.

In a previous work [11], we have proposed an eventual partition failure detector for MANET that uses information provided both by Aguilera et al's $\mathcal{HB}$ failure detector and a disconnection detector. However, the number of nodes is known and the solution is neither based on periods of stability nor on dynamic paths.

Disruption or delay-tolerant networks (DTNs) [12], opportunistic wireless access networks [18], Vehicular ad hoc networks (VANETs) [19] are example of networks that also present some lack of continuous network connectivity and thus partitions. Their routing protocols adopt an "store and forward" or a collaborative opportunity of communication approach by exploiting the concept of dynamic *path over time* between source and destination nodes.

5 Conclusion

This paper proposes a model for dynamic networks, such as MANETs, which considers that the system is anonymous with an infinite set of processes. The model characterizes the concept of dynamic paths between processes built over the time as well as the concept of stable partitions, where a finite set of nodes are connected though timely dynamic paths. Based on this model, we propose an algorithm for an eventually perfect partition participant detector, $\Diamond\mathcal{PD}$, whose properties of strong completeness and eventual strong accuracy have been proved. We also show that $\Diamond\mathcal{PD}$ supplies the requirements for providing precise membership for partionable networks.

References

[1] M.K. Aguilera, W. Chen, and S. Toueg. Using the Heartbeat Failure Detector for Quiescent Reliable Communication and Consensus in Partitionable Networks. *Theoretical Computer Science*, 220(1):3–30, June 1999.

[2] T. Anker, D. Dolev, and I. Keidar. Fault tolerant video on demand services. In *In Proceedings of the 19th International Conference on Distributed Computing Systems*, pages 244–252, 1999.

[3] O. Babaoglu, R. Davoli, and A. Montresor. Group communication in partitionable systems: Specification and algorithms. *IEEE Trans. Softw. Eng.*, 27(4):308–336, 2001.

[4] R. Baldoni, M. Bertier, M Raynal, and S. Tucci Piergiovanni. Looking for a definition of dynamic distributed systems. In *PaCT*, pages 1–14, 2007.

[5] K. Birman, R. Friedman, M. Hayden, and I. Rhee. Middleware support for distributed multimedia and collaborative computing. *Softw. Pract. Exper.*, 29(14):1285–1312, 1999.

[6] B. Bui-Xuan, A. Ferreira, and A. Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *Int. J. Found. Comput. Sci.*, 14(2):267–285, 2003.

[7] D. Cavin, Y. Sasson, and A. Schiper. Consensus with unknown participants or fundamental self-organization. In *In Proceedings of the 3rd International Conference on ADHOC-NOW 2004*, pages 135–148, 2004.

[8] T.D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267, 1996.

[9] Yang Chen, Jeonghwa Yang, Wenrui Zhao, M. Ammar, and E. Zegura. Multicasting in sparse manets using message ferrying. In *Wireless Communications and Networking Conference, 2006. WCNC 2006. IEEE*, volume 2, pages 691–696, April 2006.

[10] G.V. Chockler, I. Keidar, and R. Vitenberg. Group communication specifications: a comprehensive study. *ACM Comput. Surv.*, 33(4), 2001.

[11] D. Conan, P. Sens, L Arantes, and M. Bouillaguet. Failure, disconnection and partition detection in mobile environment. In *NCA*, pages 119–127, 2008.

[12] K. Fall. A delay-tolerant network architecture for challenged internets. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–34, 2003.

[13] J. Hähner, D. Dudkowski, P. J. Marrón, and K. Rothermel. A quantitative analysis of partitioning in mobile ad hoc networks. *SIGMETRICS Perform. Eval. Rev.*, 32(1):400–401, 2004.

[14] L. Lamport. Time, clocks and the ordering of events in a distributed system. *Communications of the ACM*, 21(7), July 1978.

[15] A. Mostefaoui, M. Raynal, C. Travers, S. Patterson, D. Agrawal, and A. El Abbadi. From static distributed systems to dynamic systems. In *Proceedings of the 24th IEEE Symposium on Reliable Distributed Systems*, pages 109–118, 2005.

[16] P. Murray. A distributed state monitoring service for adaptive application management. In *Proceedings of the 2005 International Conference on Dependable Systems and Networks*, pages 200–205, 2005.

[17] M. Nesterenko and A. Schiper. On properties of the group membership problem. Technical Report TR-KSU-CS-2007-01, jun 2007.

[18] K. A. Phanse and J. Nykvist. Opportunistic wireless access networks. In *AcessNets '06: Proceedings of the 1st international conference on Access networks*, page 11, 2006.

[19] T. Spyropoulos, K. Psounis, and C.S. Raghavendra. Spray and focus: Efficient mobility-assisted routing for heterogeneous and correlated mobility. In *PERCOMW '07: Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 79–85, 2007.

[20] S. Tucci Piergiovanni and R. Baldoni. Connectivity in eventually quiescent dynamic distributed systems. In *LADC*, pages 38–56, 2007.

Contents

Centre de recherche INRIA Paris – Rocquencourt
Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399