

# Partition Participant Detector with Dynamic Paths in Mobile Networks

Luciana Arantes, Pierre Sens, Gaël Thomas

LIP6 - University of Paris 6 - INRIA

104, avenue du Président Kennedy

75016, Paris, France

Email:[Luciana.Arantes,Pierre.Sens,Gael.Thomas]@lip6.fr

Denis Conan, Léon Lim

Institut Télécom, Télécom Sud Paris - UMR CNRS Samovar

9 rue Charles Fourier

91011, Évry, France

Email:[Denis.Conan,Leon.Lim]@it-sudparis.eu

**Abstract**—Mobile ad-hoc networks, MANETs, are self-organized 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  $\diamond PD$  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.

## I. 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.

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 imply 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 the lack of links between nodes partition them into components. A MANET is thus a *partitionable system* [9], i.e., a system in which nodes that do not crash or leave the system might not be able of communicating between themselves. An example of a partitionable MANET is a mobile network where nodes are sparsely distributed (sparse MANET) [7].

**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, can eventually exist. Furthermore, in such a context it is essential 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 belong 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 are members of its partition. Similarly to failure detectors [6], the eventually perfect partition participant detector is thus characterized by both the *strong partition participant completeness* and *eventual strong partition participant accuracy* properties.

**Contributions of the paper:** Its contributions are twofold: (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*; (2) an *eventually perfect partition participant detector* whose algorithm considers our proposal model.

## II. 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  $\Pi$  of processes. Contrarily to a static environment, in a dynamic anonymous system, processes do not know  $\Pi$ .

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.

### A. 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. A process knows its identifier 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 and is considered as a new process. They can also be mobile and 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.

Processes execute by taking steps. Each process has a local clock that counts the number of steps 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.

## B. Communication Links

Considering a radio propagation model, a node in a MANET communicates directly with all the other nodes that are within its transmission range. Hence, we assume that a node in a MANET never sends a point-to-point message but broadcasts a message which will be received by those nodes that are in its transmission range. If a process  $q$  is within the communication range of a process  $p$  we say that there is a link between  $p$  and  $q$ . However, links between nodes are unidirectional. For instance, it might happen that a node can receive a message from another node but has insufficient remaining energy to broadcast it a message back.

We assume that our system does not modify the messages it carries, neither generate spontaneous messages nor duplicate them. Messages can be lost. 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 only if  $p$  previously sent  $m$  to  $q$ . Messages can be delivered out of order. We define  $\mathcal{M}$  as the set of all possible messages.

Furthermore, due to node movements, lack of energy, failures, arrivals or departures, links come up and down over time. Therefore, connectivity between two nodes in MANETs is built dynamically over time as discussed in the following.

## C. Dynamic Paths

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

We consider Lamport's happened-before relation between events [10]:  $a \rightarrow b$  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  $p$ .

Firstly, we define the notion of reachability: a "process  $q$  is 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  $m$ . 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  $\exists (p_1, p_2, \dots, p_n) \in \Pi^n$  with  $p_1 = p$  and  $p_n = q$ , and  $\exists (m_1, m_2, \dots, m_{n-1}) \in \mathcal{M}^{n-1}$  with  $m = m_1$  such that:

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

We denote  $S_{p,q,t,m,f}$  the set of sequences of processes  $(p_1, p_2, \dots, p_n)$  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 the existence of an end-to-end path between  $p$  and  $q$  at time  $t$ . The path is indeed built over time.

We can now define the concept of dynamic path which will model connectivity between two processes in MANETs.

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

It is worth pointing out that the reachability concept depends on the algorithm  $f$  and expresses that a process can communicate with another process. On the other hand, the dynamic path concept does not depend on any algorithm. It ensures that if a process  $p$  sends any message  $m$ , then  $q$  will receive a message that causally depends on  $m$ . Both concepts are instantaneous, i.e., at time  $t$ .

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 (denoted  $p \rightsquigarrow_t^* q$ ): there exists unknown  $\delta_{pq}$  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  $trec(P, t, m, f) - t < \delta_{pq}$ .

Finally, 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 (denoted  $p \rightsquigarrow_t^{**} q$ ):  $p \rightsquigarrow_t^{**} q$  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  $p \rightsquigarrow_t^{**} q \Rightarrow p \rightsquigarrow_t q \Leftrightarrow q$  is reachable from  $p$  at time  $t$  for all messages  $m$  and all

algorithms  $f$ .

#### D. 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  $\diamond PART_p$ , is composed of the same set of correct processes that can always communicate to each other through simple timely dynamic paths. Thus, processes within  $\diamond PART_p$  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.

If a node  $q$  is reachable from  $p$ , and vice-versa, and if  $p$  is stable, then  $q$  must be in the partition of  $p$ . We define therefore the set of nodes that can be mutually reachable through a process  $p$  at a time  $t$ . These nodes form cycles which include  $p$ . 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)  $\forall t' \geq t : Cycle_p(t') = Cycle_p(t)$
- (2)  $\forall t' \geq t : q, r \in Cycle_p(t') \Rightarrow q \overset{*}{\rightsquigarrow}_{t'} r$
- (3)  $\exists N : \forall t_0 : |Cycle_p(t_0)| \leq N$

**Definition 7.** A node  $p$  is  $\diamond$ stable if  $\forall q \in Cycle_p(t)$ ,  $q$  has the *stability property*.

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

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

**Definition 8.**  $\diamond PART_p \stackrel{def}{=} Cycle_p(ST_p)$

Axiom 1 of Definition 6 states that the set of nodes of cycles of  $p$  does not change. This set is therefore the partition. Note that processes that belong to the partition are fixed but paths between processes can evolve during time. Axiom 2 imposes the existence of timely links between all nodes of a partition. It ensures therefore that communication time between nodes of the partition is bounded and that if a process  $p$  sends a message  $m$ , then each process of the partition will receive a message which is causally dependent on  $m$ . Since we make no assumption on the total number of nodes of the system, Axiom 3 fixes a bound on cycles size. Indeed, without this bound, it is impossible to build an algorithm that detects the partition in a finite time because communication time would

thus be unbounded too<sup>1</sup>. Finally, Definition 7 states that a node is  $\diamond$ stable if all nodes of its cycles have the stability property. It ensures that a process  $q$  of  $p$ 's partition is also  $\diamond$ stable and that  $p$ 's and  $q$ 's partitions are equals (see Lemma ??).

We should remark 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. On the other hand, Axiom 1 of Definition 6 ensures that if  $p$  can send messages to  $q$  through timely paths, and vice-versa, then  $q$  is in  $p$ 's partition.

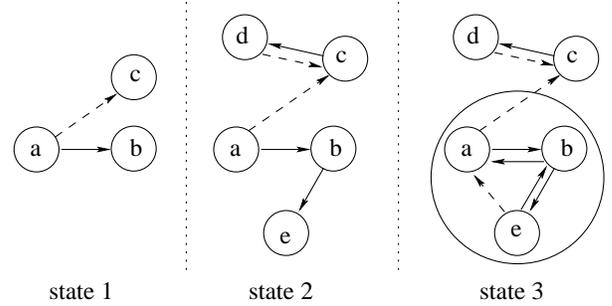


Fig. 1. Illustration of a  $\diamond$ stable process

Figure 1 illustrates the definition of  $\diamond$ stable nodes. All nodes in 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 as there does not exist any stable partition, i.e., there is no set 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.,  $\diamond PART_a = \diamond PART_b = \diamond PART_e$ .

### III. 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  $\diamond PD$ . 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  $\diamond PD$ . When invoked,  $\diamond PD$  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,  $\diamond PD$  will return the nodes that belong to the *stable partition*  $\diamond PART_p$  and only these nodes.

<sup>1</sup>We can prove it by contradiction: we suppose the existence of a partition detection bound time  $t$  and we build a long simple dynamic path such that a message takes more than  $t$  to travel through the path.

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 being included in a set of 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.

---

**Algorithm 1** Implementation of Eventually Perfect Partition Participant Detector

---

```

1  Init:
2  Begin
3  | { Processes supposed to be in  $\diamond PART_p$  }
4  |  $inPart \leftarrow \{p\}; output \leftarrow \{p\};$ 
5  |  $Timeout \leftarrow \alpha;$ 
6  | set timer to  $Timeout;$ 
7  |  $broadcast_{nbg}(\langle ALIVE, p \rangle);$ 
8  End

9

10 Task T1: upon reception of  $(\langle ALIVE, path \rangle)$ 
11 Begin
12 | If first node in  $path = p$  then
13 | | For all  $q: q$  appears after  $p$  in  $path$  do
14 | | |  $inPart \leftarrow inPart \cup \{q\};$ 
15 | | Else
16 | | | If  $p$  appears at most once in  $path$  then
17 | | | |  $broadcast_{nbg}(\langle ALIVE, path \cdot p \rangle);$ 
18 | End

19

20 Task T2: upon expiration of Timeout
21 Begin
22 | If  $output \neq inPart$  then
23 | |  $Timeout \leftarrow Timeout + 1;$ 
24 | |  $output \leftarrow inPart;$ 
25 | | set timer to  $Timeout;$ 
26 | |  $inPart \leftarrow \{p\};$ 
27 | |  $broadcast_{nbg}(\langle ALIVE, p \rangle);$ 
28 | End

29

30 Task T3: when membership() is invoked by the upper layer
31 Begin
32 | return(output);
33 End

```

---

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.

*A. 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 of the set of processes that belong to its partition by consulting the variable `output` (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  $\langle p, \dots \rangle$ ,  $p$  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.

*B. 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  $\diamond\mathcal{PD}$ .

The key of the proof is to show that if there exists  $t_1$  such that  $\forall t \geq t_1, q \in Cycle_p(t)$  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*). The full proof is available in [2].

IV. RELATED WORK

Modeling dynamic systems is an open issue and new models aiming at capturing different aspects of such dynamics have been defined. Some articles, [3], [11], [12], [5] propose a

model for dynamic systems such as MANET or peer-to-peer systems.

Like in our work, in [11], 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 of time 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.

Piergiovanni et al. [12] 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. The paper shows that there is no protocol that can ensure overlay network connectivity during perturbed periods since network partitions can happen. This work is thus particularly interested in the eventual connectivity of the overlay, i.e., a stable period where there is no partition.

In [3], 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.

Aiming at characterizing the dynamics of a system caused by the arrival and departures of participating processes of a system, Baldoni et al. present in [4] a generic model of churn which is based on deterministic joining and departure distributions of nodes where the number of processes in the systems remains, at any time, in a given range. However, even if these works consider the dynamics of the system due to arrival/departure of nodes, they do neither provide a means of modeling partitionable networks nor dynamic paths.

In [5], 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. Like in our model, evolving graphs capture the notion of *path over time*. Nevertheless, evolving graphs are based on time-step schedulers, nodes have a global view of the connectivity graph, 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 but a per process heartbeat value array.

In a previous work [8], we have proposed an eventual partition failure detector for MANETs that uses information provided both by the above mentioned Aguilera et al.'s  $\mathcal{HB}$  failure detector [1] 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.

## V. 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 time as well as the concept of stable partitions, where a finite set of nodes are connected through 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 partitionable 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] Luciana Arantes, Pierre Sens, Gaël Thomas, Denis Conan, and Léon Lim. Partition Participant Detector with Dynamic Paths in MANETs. Research Report RR-7002, INRIA, 2009.
- [3] R. Baldoni, M. Bertier, M. Raynal, and S. Tucci Piergiovanni. Looking for a definition of dynamic distributed systems. In *PaCT*, pages 1–14, 2007.
- [4] R. Baldoni, S. Bonomi, and M. Raynal. Regular register: an implementation in a churn prone environment. In *16th International Colloquium on Structural Information (SIROCCO 2009)*, 2009.
- [5] 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.
- [6] T.D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267, 1996.
- [7] Y. Chen, Y. Yang, W. 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.
- [8] D. Conan, P. Sens, L. Arantes, and M. Bouillaguet. Failure, disconnection and partition detection in mobile environment. In *NCA*, pages 119–127, 2008.
- [9] 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.
- [10] L. Lamport. Time, clocks and the ordering of events in a distributed system. *Communications of the ACM*, 21(7), July 1978.
- [11] 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.
- [12] S. Tucci Piergiovanni and R. Baldoni. Connectivity in eventually quiescent dynamic distributed systems. In *LADC*, pages 38–56, 2007.