What Model and What Conditions to Implement Unreliable Failure Detectors in Dynamic Networks ?

Fabíola Greve
DCC - Federal University of Bahia (UFBA)
Campus de Ondina, Salvador, Bahia, Brasil
fabiola@dcc.ufba.br

Luciana Arantes, Pierre Sens
LIP6, University of Paris 6, CNRS, INRIA
4 - Place Jussieu, 75005, Paris, France
Firstname.Lastname@lip6.fr

***Abstract*—Failure detectors are classical mechanisms which provide information about process failures and can help systems to cope with the high dynamics of self-organizing, unstructured and mobile wireless networks. Unreliable failure detectors of class $\Diamond S$ are of special interest because they meet the weakest assumptions able to solve fundamental problems on the design of dependable systems. Unfortunately, a negative result states that no failure detector of that class can be implemented in a network of an unknown membership; but full membership knowledge as well as fully communication connectivity are no longer appropriate assumptions to the new scenario of dynamic networks. In this paper, we provide a discussion about the conditions and model able to implement failure detectors in dynamic networks and define a new class, namely $\Diamond S^{\mathcal{M}}$, which adapts the properties of the $\Diamond S$ class to a dynamic network with an unknown membership.**

I. Introduction

Modern distributed systems deployed over ad-hoc networks, such as MANETs (wireless mobile ad-hoc networks) and WSNs (wireless sensor networks) [6] are inherently dynamic and the issue of designing reliable services which can cope with the high dynamics of these systems is a challenge.

Unreliable failure detectors, namely FD, provide an elegant approach to design fault-tolerant and modular systems under dynamic environments [10]. They can informally be seen as a per process oracle, which periodically provides a list of processes suspected of having crashed. They are formally characterized by two properties: *completeness* and *accuracy*. The combination of them yield some classes of FDs which can be used to solve fundamental problems in a crash-prone asynchronous system (e.g., consensus) [10]. The *Eventually Strong Failure Detector*, also known as $\Diamond S$ class, can make an arbitrary number of mistakes; yet, there is a time after which some correct process is never suspected (*eventual weak accuracy* property). Moreover, eventually, every process that crashes is permanently suspected by every correct process in the system (*strong completeness* property).

FDs exempt the overlying protocol (e.g., consensus) to deal with the failure treatment and synchrony requirements, so that it can just take care about its inherent task. The protocol is designed and proved correct based only on the formal properties provided by a FD class and it is exempted to deal with low-level aspects. The FD implementation and practical assumptions can be addressed independently. In this sense, the implementation can be better adapted to the particular characteristics of each environment; moreover, one FD implementation can serve many applications. An advantage of providing a FD for dynamic networks is that existing applications that already run on top of static networks using FDs could be more easily ported to the new context.

The work of F. Greve is supported by grants from CAPES-Brazil and Paris City Hall, France.

Nonetheless, the nature of MANETs and WSNs creates important challenges for the development of failure detection protocols. The inherent dynamism of these networks prevents processes from gathering a global knowledge of the system's properties. The network topology is constantly changing and the best that a process can have is a local perception of these changes. Global assumptions, such as the knowledge about the whole membership, the maximum number of crashes, complete or reliable communication, are no more realistic.

Unfortunately, an algorithm that implements any of the classes of FDs requires every process to know the whole system membership. According to [15], if there is some processes in the system such that the rest of the processes have no knowledge whatsoever of its identity, there is no algorithm that implements a FD with weak completeness, even if links are reliable and the system is synchronous. On the other hand, the Ω FD can be implemented without the membership knowledge. This is because Ω just needs information about alive processes while the other FDs need to know the identity of every faulty process in order to ensure completeness. The Ω class (known as the *leader detector*) satisfies the following *eventual leadership* property: there is a time after which every correct process always trusts the same correct process in the system (i.e., the same leader). FDs of class $\Diamond S$ and Ω have the same computational power and they encapsulate the weakest synchrony necessary to circumvent the impossibility result of the fundamental consensus problem [10].

The fact is that many important services (e.g., group membership, atomic commitment, atomic broadcast) and middleware (e.g., group communication, replicated and transaction servers) make direct use of $\Diamond S$ FDs. Thus, seeking for conditions to implement FDs of such a class in dynamic networks is of utmost interest. Considering this importance, this paper provides a discussion about the different possible assumptions found in the literature to implement unreliable failure detectors (Section II). Then, it suggests a model and conditions under which FDs of a new class, namely $\diamond S^{\mathcal{M}}$, can be implemented (Section III). This class adapts the properties of the $\diamond S$ class to a network with an unknown membership

II. Strategies for Failure Detection

In this section we provide a discussion about models and protocols to implement FDs and the interest of their approach for dynamic networks: MANETs and WSNs.

Traditional Approach. A number of FD algorithms has been proposed so far for a traditional system of static networks. Most of them are based on an all-to-all communication approach where each process periodically sends "I am alive" messages to all processes [11]. The majority of these works consider a fully connected static network with reliable links where nodes fail by crashing. The maximum number of crashes is usually bounded (by f) and both the initial number of processes (namely, n) and the system composition (namely, Π) are known by all processes. As told, global assumptions about Π, n and f are no longer suitable for dynamic environments.

Scalable Approaches. Some scalable FD implementations do not require a fully connected network [13], [26]. They base the detection on the use of an adaptive heartbeat or follow the gossiping style communication, choosing only a few members or neighbors to disseminate information. Practically, the randomization makes the definition of timeout values difficult. It is worth remarking that none of these works tolerate mobility of nodes and wireless communications.

The Heartbeat Approach. Some works focus on the heartbeat FD for sparsely connected networks with partial membership knowledge. The heartbeat FD is a special class of FD which is able to implement quiescent reliable communication [2]. Nonetheless, it does not directly implement $\diamond S$ FDs, instead of lists of suspects, it outputs a vector of unbounded counters; if a node crashes, its counter eventually stops increasing. Tucci *et al.* [25] implements a heartbeat FD for the infinite arrival model [1] and shows how to use it to implement a Ω FD. The solution considers fair-lossy channels, but for a synchronous environment. Hutle [14] proposes a FD of the class $\diamond P$, which provide *strong completeness* and *eventual strong accuracy*: after some point in time no correct node will be suspected by another correct node. It considers sparsely connected unknown networks, subject to partitions; nonetheless, it assumes some knowledge about the neighborhood (which has a bounded number of processes) and about the jitter of the communication between direct neighbors. We should noticed that none of these works tolerate node mobility.

Probabilistic Approach for MANETs. Friedman *et al.* [12] propose a simple gossiping protocol which exploits the natural broadcast range of wireless networks to delimit the local membership of a node in a mobile network. Nodes may be mobile, may crash and messages may be lost. Unfortunately, this work assumes a known number of nodes (n) and provides probabilistic guarantees for the FD properties.

Tai *et al.* [24] exploit a cluster-based communication architecture to propose a hierarchical gossiping FD for a network of non-mobile nodes. The FD is implemented both via intra-cluster heartbeat diffusion and failure report diffusion across clusters, i.e., if a failure is detected in a local cluster, it will be further forwarded across the clusters. Unfortunately, this work assumes a known system composition and it does not consider node mobility. Moreover, it is for a specific type of network, considering a cluster-based communication architecture. Finally, it implements a $\diamond P$ FD providing probabilistic guarantees for the accuracy and completeness properties.

The Local Failure Detection Approach. Actually, most of the existing FDs ensure a "global failure detection" in the sense that the completeness and accuracy properties they satisfy are valid for the whole system. But, in dynamic systems, due to resource's restrictions, some computations are localized. Sridhar [23] advocates the use of a "local failure detection" as an appropriate abstraction to deal with node mobility and resource lack in WSNs. It proposes a FD that restraints the scope of detection to the neighborhood of a node and not to the whole system and provides an implementation of an eventually perfect *local* FD of the class $\diamond P_l^m$. This class provides $\diamond P$ properties, but with regard to a node's neighborhood. Sridhar [23] proposes a FD protocol that implement $\diamond P_l^m$ in a network composed of a set of known and mobile nodes, communicating via unreliable channels. The FD has two independent layers: a local one that builds a suspected list of crashed neighbors and a second one that detects mobility of nodes across network and able to correct possible mistakes.

A. Timer-Based x Time-Free Failure Detection

None of the FD classes can be implemented in a purely asynchronous system [10]. Indeed, while completeness can be realized by using "I am alive" messages and timeouts, accuracy cannot be safely implemented for all system executions. Thus, some additional assumptions on the underlying system should be made in order to implement them. With this aim, two orthogonal approaches can be distinguished: the *timer-based* and the *time-free* failure detection [20]. The timer-based is the traditional approach and supposes that channels in the system are eventually timely; this means that, for every execution, there are bounds on process speeds and on message transmission delays that will permanently hold. However, these bounds are not known and they hold only after some unknown time [10]. All the works mentioned above follow a timer-based approach to implement the FD classes [11], [13], [26], [14], [25], [24], [12], [23].

An alternative approach suggested by [18] and developed so far by [7], [19] is time-free and considers that the system satisfies a message exchange pattern on the execution of a *query-based* communication primitive. While the timer-based approach imposes a constraint on the physical time (to satisfy message transfer delays), the time-free approach imposes a constraint on the logical time (to satisfy a message delivery order). Mostefaoui *et al.* proposes a $\diamond S$ FD which is time-free; however, it assumes a network of fully connected nodes with global knowledge about Π, n and f. In their approach, the responses from some process p to a query launched by other processes arrive among the first ones, precisely, among the $n-f$ first messages. To implement a FD of the class $\diamond S$, a probabilistic analysis for the case $f=1$ shows that such a behavioral property on the message exchange pattern is always satisfied in practice [18]. Most recently, Cao *et al.* [7] have proposed a time-free Ω FD. It considers a hybrid network of stable mobile hosts (namely, MH) and mobile support stations (namely, MSS). A MH is considered *stable* if, once it entered the system, it does not crash or gets disconnected. Both MSSs and MHs can crash and the maximum number of MSSs that may crash (f) is known. The FD provides an eventual accuracy property, ensuring that eventually at least one stable MH is continuously trusted by the MSSs. The completeness ensures that a MH that crashes or permanently leaves the system is eventually no longer trusted by an MSS.

Both approaches (timer-based and time-free) are orthogonal and cannot be compared, but, in some cases, they can be combined at the link level in order to implement hybrid protocols with combined assumptions [20]. The time-free model has some potential advantages for being used in a dynamic set, e.g., processes can independently control their message exchange pattern and impose some delay in order to satisfy in practice the assumptions made for the failure detection. Moreover, since in dynamic networks the communication delays may frequently vary due to crashes, mobility, arrivals and departures, the statement of the transmission bounds is a real challenge.

Synthesis. As can be verified, none of the works proposed so far are able to implement deterministic $\diamond S$ FDs for dynamic networks of generic topologies. The next section provides thus a model with abstractions able to implement such a FD.

III. On the Specification of a Model and a $\diamond S$ FD for Dynamic Networks

We are particularly interested in systems deployed over a wireless mobile network, such as WMNs, WSNs and MANETs. The system is a collection of nodes which communicate by sending and receiving messages via a packet radio network. There are no assumptions on the relative speed of processes or on message transfer delays, thus the system is *asynchronous*. To simplify the presentation, we take the range $\mathcal{T}$ of the clock's tick to be the set of natural numbers. There is no global clock and processes do not have access to $\mathcal{T}$: it is introduced for the convenience of the presentation, to state properties and make proofs.

Finite arrival model [1]. The network is a dynamic system composed of infinitely many processes; but each run consists of a finite set Π of $n > 3$ nodes, namely, $\Pi = \{p_1, \ldots, p_n\}$. This model properly expresses dynamic networks where nodes join and leave the system as they wish. It is suitable for long-lived or unmanaged applications, as for example, sensor networks deployed to support crises management or help on dealing with natural disasters.

The membership is unknown. Processes are not aware about Π or n, because, moreover, these values can vary from run to run [1]. There is one process per node; each process knows its own identity, but it does not necessarily knows the identities of the others. Nonetheless, they can make use of the broadcast facility of the wireless medium to know one another. Thus, we consider that a process knows a subset of Π, composed of nodes with whom it previously communicated. A process may fail by *crashing*, *i.e.*, by prematurely or by deliberately halting (switched off); a crashed process does not recover. Until it possible crashes, a process behaves according to its specification.

Communication graph is dynamic. Due to arbitrary joins, leaves and failures, the network is represented by a communication graph with a dynamic topology, thus the relations between nodes take place over a time span $\mathcal{T} \subseteq \mathbb{N}$. Following [9], we consider that the dynamics of the system is represented by a *time-varying graph*, namely TVG, $\mathcal{G} = (V, E, \mathcal{T}, \rho, \varsigma)$, where $V = \Pi$ represents the set of nodes and $E \subseteq V \times V$ represents the set of logical links between nodes, $\rho : E \times \mathcal{T} \to \{0, 1\}$ is a *presence* function, indicating whether a given edge is available at a given time and $\varsigma : E \times \mathcal{T} \to \mathbb{N}$ is a *latency* function, indicating the time taken to cross a given edge if starting at a given date. Since the system is asynchronous, there is no bound for this time; thus, we consider that ς exists but cannot be estimated.

Let R_i be the wireless transmission range of p_i in the network, then all the nodes that are at distance at most R_i from p_i in the network are considered 1-hop *neighbors*, belonging to the same *neighborhood*. We denote N_i^t to be the set of 1-hop *neighbors* from p_i at time $t \in \mathcal{T}$. The neighborhood relationship establishes the edge set, in such a way that $p_j \in N_i^t$ *iff* $(p_i, p_j) \in E_i^t$. In this case, $\rho((p_i, p_j), t) = 1$. The *degree* of p_i is defined to be $Deg_i^t = |E_i^t|$.

Given a TVG $\mathcal{G}$, the graph $G = (V, E)$ is called the *underlying graph* of $\mathcal{G}$. G should be considered as a sort of *footprint* of $\mathcal{G}$ which flattens the time dimension and indicates only the pair of nodes that have relations at some time in $\mathcal{T}$. We can identify classes of TVG based on the temporal properties established by the entities. The classes are important because they imply necessary conditions and impossibility results for distributed computations. Notably, Class 3 is important for our study [9].

Class 3 (Connectivity over time): $\forall p_i, p_j \in V, p_i \rightsquigarrow p_j$; that is p_i *reaches* p_j. This means that the TVG is connected over time. Formally, a sequence of couples $\mathcal{J} = \{(e_1, t_1), (e_2, t_2), \ldots, (e_k, t_k)\}$, such that $\{e_1, e_2, \ldots, e_k\}$ is a walk in G, is a *journey* in $\mathcal{G}$ *if and only if* $\rho(e_i, t_i) = 1$ and $t_{i+1} \geq t_i + \varsigma(e_i, t_i)$ for all $i < k$. If a journey exists from p_i to p_j, we say that p_i *reaches* p_j or more simply, $p_i \rightsquigarrow p_j$.

Communication is fair-lossy. Local broadcast between 1-hop neighbors is *fair-lossy*. This means that messages may be lost, but, if p_i broadcasts m to processes in its neighborhood an infinite number of times, then every p_j in the neighborhood receives m from p_i an infinite number of times, or p_j is faulty. That is, if p_i starts to send m at time t an infinite number of times, then, if $\rho((p_i, p_j), t') = 1, \forall t' \in (t, \infty)$, p_j receives m an infinity number of times if p_j is correct. This condition is attained if the MAC layer of the underlying wireless network provides a protocol that reliably delivers broadcast data, even in presence of unpredictable behaviors, such as fading, collisions, and interference; solutions in this sense have been proposed in [17], [16], [4].

Mobility model. Nodes in Π may be mobile and they can keep continuously *moving* and *pausing* in the system. When a node p_m moves, its neighborhood may change. We consider a *passive mobility* model, i.e., the node that is moving does not know that it is moving. Hence, the mobile node p_m cannot notify its neighbors about its moving. Then, for the viewpoint of a neighbor, it is not possible to distinguish between a moving, a leave or a crash of p_m. During the neighborhood changing, p_m keeps its state, that is, the values of its variables.

A. Stability Assumptions

One important aspect on the design of FDs for dynamic networks concerns the time period and conditions in which processes are connected to the system. During unstable periods, certain situations, as for example, connections for very short periods or numerous joins or leaves along the execution (characterizing a churn) could block the application and prevent any useful computation. Thus, to implement any global computation, the system should present some stability conditions that when satisfied for long enough time will be sufficient to satisfy the requirements of the application and terminate.

In order to implement FDs with an unknown membership, processes should interact with some others to be known. If there is some process such that the rest of processes have no knowledge whatsoever of its identity, there is no algorithm that implements a FD with weak completeness [15]. *Completeness* characterizes the FD capability of suspecting every faulty process permanently. In this sense, the characterization of the *actual membership* of the system, that is, the set of processes which might be considered for the computation is of utmost importance.

We consider then that a process p_i *joins* the network at some point $t \in \mathcal{T}$ in time. Subsequently, p_i must somehow communicate with the others in order to be known. In a wireless network, this can be done by simply broadcasting its identity to the neighbors. Due to this initial communication, every process p_j is able to gather an initial partial knowledge $\Pi_j \subseteq \Pi$ about the system's membership which increases over the time along p_j's execution. Let $\Pi_j(t)$ be the partial knowledge of p_j by time t. When p_i *leaves* the network at time $t' > t$, it can re-enter the system with a new identity, thus, it is considered as a new process. Processes may join and leave the system as they wish, but the number of re-entries is bounded, due to the finite arrival assumption.

Definition 1: Process status. Let $\mathit{IN}(t) \subseteq \Pi$ be the set of processes that are in the system at time $t \in \mathcal{T}$, that is, after have joined the system before t, they neither leave it nor crash before t. A process p_i may assume the following status in the system.

$stable^t(p_i) \Leftrightarrow \exists t \in \mathcal{T}, \forall t' \geq t,\ p_i \in \mathit{IN}(t')$

$faulty^t(p_i) \Leftrightarrow \exists s, t \in \mathcal{T}, s < t,\ p_i \in \mathit{IN}(s) \wedge p_i \notin \mathit{IN}(t)$

$known^t(p_i) \Leftrightarrow \exists p_j, \exists t \in \mathcal{T}, stable^t(p_j) \wedge p_i \in \Pi_j(t)$

A process is *known* if, after have joined the system, it has been identified by some stable process. A *stable* process is thus a mobile process that, after had entered the system for some point in time, never departs (due to a crash or a leave); otherwise, it is *faulty*.

The *failure pattern* of the system, namely $F(t)$, is the set of processes that have failed in the system by time t. That is, $F(t) = \{p_i : faulty^t(p_i)\}$. Similarly, $S(t)$, is the set of processes that are stable in the system by time t. That is, $S(t) = \{p_i : stable^t(p_i)\}$.

Definition 2: Membership. The membership of the system is the KNOWN set.

$\text{STABLE} \overset{def}{=} \bigcup_{t \in \mathcal{T}} S(t)$

$\text{FAULTY} \overset{def}{=} \bigcup_{t \in \mathcal{T}} F(t)$

$\text{KNOWN} \overset{def}{=} \{p_i : \exists t \in \mathcal{T}, p_i \in \text{STABLE} \cup \text{FAULTY} \wedge known^t(p_i)\}$

Recent works about radio communication advocate a "local" fault model, instead of a "global" fault model, as an adequate strategy to deal with the dynamism and unreliability of wireless channels in spite of failures [16], [21], [3], [5]. They define bounds on the maximum number of local failures in order to reliably delivery data. Precisely, [3], [5] show that it is possible to achieve reliable broadcast if less than $1/2$ of nodes in any neighborhood fail by crash and impossible otherwise. Locality of failures can be interpreted as an uniform distribution of failures across the network and represents more accurately the reality of wireless channels. Following these recent works, the local fault model is the approach adopted in our work. Thus, we consider that f_i is the maximum number of faulty processes in p_i's neighborhood.

Let $V_{KS} = \text{KNOWN} \cap \text{STABLE}$ and $E_{KS} \subseteq V_{KS} \times V_{KS}$. The graph $G_{KS} = (V_{KS}, E_{KS}) \subseteq G$ is the graph induced from the stable known nodes in Π, defining the TVG $\mathcal{G}_{KS} = (V_{KS}, E_{KS}, \mathcal{T}, \rho) \subseteq \mathcal{G}$.

Assumption 1: Network connectivity. In the system, represented by the TVG $\mathcal{G}_{KS}$, $\exists t \in \mathcal{T}, \forall t' \geq t$, $\forall p_i, p_j \in V_{KS}, p_i \rightsquigarrow p_j$. That is, after t, there is a *journey* $\mathcal{J}$ $\forall p_i, p_j \in \text{KNOWN} \cap \text{STABLE}$. For a communication purpose, we assume that each edge e_i of $\mathcal{J}$ remains available until a message is delivered, thus $\rho_{[t_i, t_i + \varsigma(e_i, t_i))}(e_i) = 1$.

The connectivity assumption states that, in spite of changes in the topology, from some point in time t, the TVG $\mathcal{G}_{KS}$ is connected over time. This is a common assumption, mandatory to ensure reliable dissemination of messages to all stable processes in a dynamic network [8], [9] and thus to ensure the global properties of the failure detector [10], [15], [19], [5]. Notice that the connectivity assumption establishes a bound to tolerate faults, such that the minimum degree of a node p_i in G_{KS} is $Deg_i^t = |E_i^t| > f_i, \forall t \in \mathcal{T}$. If one considers the reliable broadcast model in [3], [5], this value should be $Deg_i^t = |E_i^t| > 2f_i, \forall t \in \mathcal{T}$. In practice, whenever a higher number of faults occur, the network may be disconnected for a while and then the progress may be compromised. But, the most important is to ensure the safety of the protocol during these bad periods. As soon as the network starts to behave better, the connectivity assumption will be satisfied and the protocol progresses.

B. Failure Detectors of Class $\diamondsuit S^M$

Unreliable failure detectors provide information about the liveness of processes in the system [10]. Each process has access to a local failure detector which outputs a list of processes that it currently suspects of being faulty. The failure detector is *unreliable* in the sense that it may erroneously add to its list a process which is actually correct. But if the detector later believes that suspecting this process is a mistake, it then removes the process from its list. Failure detectors are formally characterized by two properties: (i) *Completeness* characterizes its capability of suspecting every faulty process permanently; (ii) *Accuracy* characterizes its capability of not suspecting correct processes. Our work is focused on the class of *Eventually Strong* detectors, also known as $\diamondsuit S$. Nonetheless, we adapt the properties of this class in order to implement a FD in a dynamic set. Then, we define the class of *Eventually Strong Failure Detectors* with *Unknown Membership*, namely $\diamondsuit S^M$. This class keeps the same properties of $\diamondsuit S$, except that they are now valid to known processes, that are stable and faulty.

Definition 3: **Eventually Strong FD with Unknown Membership** ($\diamondsuit S^M$)

Let p_i, p_j be mobile nodes. Let $susp_j$ be the list of processes that p_j currently suspects of being faulty. The $\diamondsuit S^M$ class contains all the failure detectors that satisfy:

Strong completeness $\overset{def}{=}$ $\{\exists t \in \mathcal{T},\ \forall t' \geq t,\ \forall p_i \in \textsc{Known} \cap \textsc{Faulty} \Rightarrow p_i \in susp_j,\ \forall p_j \in \textsc{Known} \cap \textsc{Stable}\}$;

Eventual weak accuracy $\overset{def}{=}$ $\{\exists t \in \mathcal{T},\ \forall t' \geq t,\ \exists p_i \in \textsc{Known} \cap \textsc{Stable} \Rightarrow p_i \notin susp_j,\ \forall p_j \in \textsc{Known} \cap \textsc{Stable}\}$.

C. An Implementation of a FD of Class $\diamondsuit S^M$

The model and conditions proposed in this paper have been used in [22] to implement a deterministic FD algorithm of the class $\diamondsuit S^M$. This FD has a number of innovative features in comparison with those proposed so far. It is time-free; the failure detection does not make use of global information (e.g., Π, n, f) and is based on a local communication pattern; it tolerates message losses and node mobility, beyond arbitrary joins and leaves.

References

[1] M. K. Aguilera. A pleasant stroll through the land of infinitely many creatures. *SIGACT News*, 35(2):36–59, 2004.

[2] M. K. Aguilera, W. Chen, and S. Toueg. Heartbeat: A timeout-free failure detector for quiescent reliable communication. In *Proc. of the 11th International Workshop on Distributed Algorithms*, pages 126–140, 1997.

[3] V. Bhandari and N. H. Vaidya. On reliable broadcast in a radio network. In *Proc. of the 24th annual ACM symposium on Principles of distributed computing*, pages 138–147. ACM, 2005.

[4] V. Bhandari and N. H. Vaidya. Reliable local broadcast in a wireless network prone to byzantine failures. In *The 4th ACM SIGACT/SIGMOBILE International Workshop on FOUNDATIONS OF MOBILE COMPUTING*, 2007.

[5] V. Bhandari and N. H. Vaidya. Reliable broadcast in radio networks with locally bounded failures. *IEEE Transactions on Parallel and Distributed Systems*, 21:801–811, 2010.

[6] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications & Mobile Computing: Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2:483–502, 2002.

[7] J. Cao, M. Raynal, C. Travers, and W. Wu. The eventual leadership in dynamic mobile networking environments. In *Proc. of the 13th Pacific Rim International Symposium on Dependable Computing*, pages 123–130, 2007.

[8] A. Casteigts, S. Chaumette, and A. Ferreira. Characterizing topological assumptions of distributed algorithms in dynamic networks. *Structural Information and Communication Complexity*, pages 126–140, 2010.

[9] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-varying graphs and dynamic networks. Technical report, University of Ottawa, 2011.

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

[11] B. Devianov and S. Toueg. Failure detector service for dependable computing. In *Proc. of the 1st Int. Conf. on Dependable Systems and Networks*, pages 14–15, 2000.

[12] R. Friedman and G. Tcharny. Evaluating failure detection in mobile ad-hoc networks. *Int. Journal of Wireless and Mobile Computing*, 1(8), 2005.

[13] I. Gupta, T. D. Chandra, and G. S. Goldszmidt. On scalable and efficient distributed failure detectors. In *Proc. of the 20th annual ACM symposium on Principles of distributed computing*, pages 170–179, 2001.

[14] M. Hutle. An efficient failure detector for sparsely connected networks. *Proc. of the IASTED International Conference on Parallel and Distributed Computing and Networks*, pages 369–374, 2004.

[15] E. Jiménez, S. Arévalo, and A. Fernández. Implementing unreliable failure detectors with unknown membership. *Inf. Process. Lett.*, 100(2):60–63, 2006.

[16] C-Y. Koo. Broadcast in radio networks tolerating byzantine adversarial behavior. In *Proc. of the 23th annual ACM symposium on Principles of distributed computing*, pages 275–282, 2004.

[17] S. Min-Te, H. Lifei, A. A. Arora, and L.Ten-Hwang. Reliable mac layer multicast in ieee 802.11 wireless networks. In *Proc. of the International Conference on Parallel Processing*, pages 527–536, August 2002.

[18] A. Mostefaoui, E. Mourgaya, and M. Raynal. Asynchronous implementation of failure detectors. In *Proc. of Int. Conf. on Dependable Systems and Networks*, 2003.

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

[20] Achour Mostefaoui, Michel Raynal, and Corentin Travers. Time-free and timer-based assumptions can be combined to obtain eventual leadership. *IEEE Trans. Parallel Distrib. Syst.*, 17(7):656–666, 2006.

[21] A. Pelc and D. Peleg. Broadcasting with locally bounded byzantine faults. *Inf. Process. Lett.*, 93(3):109–115, 2005.

[22] P. Sens, L. Arantes, M. Bouillaguet, V. Simon, and F. Greve. Asynchronous implementation of failure detectors with partial connectivity and unknown participants. Technical Report RR6088, INRIA - France, http://hal.inria.fr/inria-00122517/fr/.

[23] N. Sridhar. Decentralized local failure detection in dynamic distributed systems. *The 25th IEEE Symp. on Reliable Distributed Systems*, 0:143–154, 2006.

[24] A. Tai, K. Tso, and W. Sanders. Cluster-based failure detection service for large-scale ad hoc wireless network applications. In *Proc. of the Int. Conf. on Dependable Systems and Networks*, pages 805–814, 2004.

[25] S. Tucci-Piergiovanni and R. Baldoni. Eventual leader election in infinite arrival message-passing system model with bounded concurrency. In *Dependable Computing Conference (EDCC), 2010 European*, pages 127 –134, 2010.

[26] R. Van Renesse, Y. Minsky, and M. Hayden. A gossip-style failure detection service. Technical report, Ithaca, NY, USA, 1998.