The Time-Free Approach to Byzantine Failure Detection in Dynamic Networks

Murilo Santos de Lima, Fabíola Greve
DCC - Federal University of Bahia (UFBA)
Campus de Ondina, Salvador, Bahia, Brasil

Luciana Arantes, Pierre Sens
LIP6, University of Paris 6, CNRS, INRIA
4 - Place Jussieu, 75005, Paris, France

***Abstract*—Modern distributed systems deployed over wireless ad-hoc networks are inherently dynamic and the issue of designing dependable services which can cope with the high dynamics of these systems is a challenge. Byzantine failure detectors provide an elegant abstraction for solving security problems; however, very few work has been proposed for the new context of dynamic networks. This paper advocates the adoption of the time-free approach to detect Byzantine failures in such systems. This approach does not rely on timers to detect progress failures; it is suitable to deal with the unpredictability of the node's behavior and communication medium on these networks, favoring scalability and adaptability.**

***Index Terms*—Byzantine failures, Byzantine failure detectors, dynamic distributed systems, wireless ad-hoc networks**

I. Introduction

Modern distributed systems, deployed over ad-hoc networks, such as wireless mesh networks (WMN), wireless sensor networks (WSN) are inherently dynamic. They are composed by a dynamic population of nodes, which randomly join and leave the network, at any moment of the execution, so that only a partial knowledge about the system's properties can be retained. Global assumptions, such as the knowledge about the whole membership, the maximum number of failures and complete communication, are no more realistic. Therefore, classical distributed protocols are no longer appropriate for this new context, since they make the assumption that the whole system is static and its composition is previously known.

Security is a major challenge on dynamic distributed systems as many factors favor the action of malicious agents, e.g., the dynamic population, the wireless communication medium, the necessity to cooperate in order to achieve fundamental tasks (e.g., routing). Thus, Byzantine fault tolerance (BFT) [LSP82] plays an important role on the development of dependable dynamic systems. It deals with a number of security problems by tolerating the presence of corrupted processes, which may behave in an arbitrary manner, trying to hinder the system to work accordingly to its specification.

Unreliable failure detectors, namely FD, provide an elegant approach to design dependable and modular systems under dynamic environments [CT96]. They give hints on which processes in the system are faulty and exempts the overlying protocol to deal with the failure treatment and synchrony requirements. But, differently from FDs for crash or "benign" failures, in which the FD implementation and practical assumptions can be addressed independently, FDs for Byzantine failures must rely the detection on the application algorithm that uses it as an underlying oracle. This is because the detection is made according with the message contents and communication pattern followed by the specific algorithm. This inherently symbiosis between the FD and application is perhaps at the cause of the very little work done until now in the domain of Byzantine fault detection.

Malkhi *et al.* [MR97] extend the theory of [CT96] and define a FD able to identify processes that prevent the progress of the algorithm using it. Doudou *et al.* [DS98] introduce the concept of *muteness failure detectors* in which the oracle detects when a process is mute, that is, when it ceases sending messages required by the algorithm. While these two works are restricted to a small subset of failures, the work of Kihlstrom *et al.* [KMMS03] extend the classical model of FDs [CT96] to propose new classes able to consider more generic Byzantine failures and to solve consensus. Baldoni *et al.* [BHRT03] provide a framework to solve consensus which integrates muteness failure detectors (for mute crashes) and a byzantine behavior detector (for other Byzantine failures). All these works consider a classical distributed system in which the communication graph is complete and neither the number nor set of participants are known. An exception is [AHNRR02], but it solves only a subset of the Byzantine failure detection problem to the specific application of routing.

Recently, Haeberlen *et al.* [HKD07] presented the PeerReview system and propose a concrete solution to the Byzantine fault problem based on the use of accountability to detect and expose node faults. The solution is suitable for dynamic systems which span multiple administrative domains, e.g., P2P and overlay multicast systems, but does not consider the case of systems in which the membership is unknown since the beginning. In a subsequent work [HK09], the same authors provide a formal study of the generic fault detection problem. They give a formal definition of commission (or security) and omission (or progress) faults [KMMS03] and identify

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

some bounds on the costs of solving a weak definition of failure detection problem in asynchronous systems with authenticated channels.

All the Byzantine FDs proposed so far adopt the *timer-based* model to detect progress failures. This is a common design principle which supposes that eventually some bound on the transmission delays will permanently hold. However, these bounds are not known and they hold only after some unknown time [CT96]. An alternative approach suggested by [MMR03] is *time-free* and considers that the system satisfies a message exchange pattern on the execution of a communication primitive. It does not rely on timers to detect crash failures and assumes that the responses from some correct process to a query launched by other processes permanently arrive among the first ones. This idea has been exploited by [SAB$^+$08] to develop a FD for dynamic networks, but for the crash failure model. 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). Both approaches (timer-based and time-free) are orthogonal and cannot be compared.

In dynamic networks, since the communication delays may frequently vary due to failures, arrivals and departures of nodes, the statement of the transmission bounds required by the timer-based detection becomes a big challenge. In this sense, the time-free model appears as a suitable alternative for being used in a dynamic set [MRT$^+$05].

This paper advocates the use of the time-free approach to provide Byzantine failure detection. It presents the main design principles and subtleties of an unreliable Byzantine FD adequate for dynamic networks with unknown membership and partial communication. To the best of our knowledge, the adoption of a time-free detection in networks with unknown membership is novel and this paper provides a first insight towards the understanding and implementation of such an approach.

The rest of the paper provides the model (Section II), the FD design principle (Section III) and conclusion (Section IV).

II. Proposed Model for Failure Detection in Dynamic Networks

We are particularly interested in systems deployed over wireless ad-hoc networks, such as WSNs and WMNs. The system is a set of nodes communicating by broadcasting messages via a packet radio network.

Finite arrival model [Agu04]. 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 what does happen in dynamic networks since nodes join and leave the system as they wish.

The membership is unknown. Processes are not aware about Π or n, because, moreover, these values can vary from run to run [Agu04]. 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 with nodes with whom it previously communicated.

Processes are subject to *Byzantine failures* [LSP82], i.e., they can deviate arbitrarily from the algorithm they are specified to execute and work in collusion to corrupt the system behavior. A process that does not follow its algorithm specification is said to be *Byzantine*; otherwise, it is *correct*. In particular, a Byzantine process may send messages not previously defined by its algorithm or may omit to send messages it is supposed to. In this sense, a process that crashes can be regarded as Byzantine. Notice that Π is a partition of correct and Byzantine processes. Every process is uniquely identified and a Byzantine process cannot obtain more than one identifier. Thus, it is impossible to launch a *sybil attack* against the system [Dou02].

The system is asynchronous. There are no assumptions on the relative speed of processes or on message transfer delays. There is no global clock, but to simplify the presentation, we take the range $\mathcal{T}$ of the clock's tick to be the set of natural numbers.

Communication graph is dynamic. The network is represented by a communication graph $G = (V, E)$ in which $V = \Pi$ represents the set of nodes and E represents the set of logical links. The topology of G is dynamic due to arbitrary joins, leaves and failures. A link between nodes p_i and p_j is bidirectional, meaning that p_i is within the wireless transmission range of p_j and vice-versa. Let R_i be the transmission range of p_i, 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 to be the set of 1-hop *neighbors* from p_i and $|N_i|$ its cardinality; thus, $(p_i, p_j) \in E$ *iff* $(p_i, p_j) \in N_i$.

Communication is fair-lossy. Local wireless channels are *authenticated* and *fair-lossy*. Thus, every process p_i holds a private key $\mathcal{K}_i$ with which it can sign its messages; and every process in the system can obtain the public key of every other node in order to authenticate the sender of any signed message [Sch96][1]. Moreover, a message m sent by a correct process p_i an infinite number of times is received by every correct process p_j in its neighborhood an infinite number of times, or p_j is faulty. In addition, there is no message duplication, modification or creation; this means that a Byzantine node is not allowable to interfere on message transmissions by correct processes, and even if it sends multiple versions of a message, the message will be perceived by the others

[1]Without authenticated channels, it is not possible to tolerate process misbehavior in an asynchronous system since a single faulty process can play the roles of all other processes to some (victim) process.

as only one message with the same contents [Koo04], [BV07]. The Fair-lossy assumption seems to be unrealistic for the dynamic environment; above all, wireless channels are inherently unreliable and can in addition suffer from a number of attacks, e.g., a malicious node can raise a collision attack in messages sent by honest nodes, preventing reception. Notice however that some works about ensuring reliability under wireless channels have recently appeared. They 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 Byzantine failures [Koo04], [PP05], [BV05], [BV07], [BV10], defining bounds on the maximum number of local failures in order to reliably delivery data. Precisely, [BV05], [BV10] show that it is possible to achieve reliable broadcast if less than $1/4$ of nodes in any neighborhood are Byzantine and impossible otherwise. Knowing that f_i is the maximum number of faulty processes in p_i's neighborhood, $f_i < |N_i|/4$.

Locality of failures can be interpreted as an uniform distribution of failures across the network and represents more accurately the reality of wireless channels. This is the approach followed in our work.

A. Stability Requirements

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, the system should present some stability conditions that when satisfied for longtime enough will be sufficient for the computation to progress 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 [JAF06]. *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: **Membership**. Let $t, t' \in \mathcal{T}$. Let $UP(t) \subseteq \Pi$ be the set of processes that are in the system at time t, that is, after having joined the system before t, they neither leave it nor crash before t. The membership of the system is the KNOWN set.

BYZANTIN $\stackrel{def}{=} \{\forall p_i : p_i$ *is Byzantine*$\}$

STABLE $\stackrel{def}{=} \{p_i : \exists t,\ \forall t' \geq t,\ p_i \in UP(t') \wedge p_i \notin$ BYZANTIN$\}$

FAULTY $\stackrel{def}{=} \{p_i : \exists t, t',\ t < t',\ (p_i \in UP(t) \wedge p_i \notin UP(t')) \vee p_i \in$ BYZANTIN$\}$

KNOWN $\stackrel{def}{=} \{p_i : \exists p_j, \exists t,\ (p_i \in$ STABLE$\cup$FAULTY$) \wedge (p_i \in \Pi_j(t), p_j \in$ STABLE$)\}$

The actual membership of the system is in fact defined by the KNOWN set. A process is *known* if, after having joined the system, it has been identified by some stable process. A *stable* process is thus a correct process that, after had entered the system for some point in time, never departs; otherwise, it is *faulty*. Following recent works about Byzantine radio communication [Koo04], [BV10], we adopt a local fault model in which f_i is the maximum number of faulty processes in p_i's neighborhood.

Assumption 1: **Network with Byzantine coverage**. Let $G($KNOWN $\cap$ STABLE$) = G(S) \subseteq G$ be the graph obtained from the stable known processes. Then, a system has *Byzantine coverage* if and only if $\exists t \in \mathcal{T}$, s.t: (1) there is a node-disjoint path between every pair of stable processes $p_i, p_j \in G(S)$; (2) the minimum degree of a node p_i in G is $|N_i| > 2f_i$.

Connectivity assumption (1) states that, in spite of changes in the topology of G, from some point in time t, the set of known stables forms a *strongly connected component* in G. This is a frequent assumption, mandatory to ensure reliable dissemination of messages to all stable processes and thus to ensure the global properties of the failure detector [CT96], [JAF06], [MRT+05], [Koo04], [BV05], [BV10].

Connectivity assumption (2) establishes a bound to tolerate Byzantine faults. It is a guarantee that information from/to process p_i is going to be sent/received to/by a minimum of stable nodes in its neighborhood. Precisely, at least $f_i + 1$ stable nodes can communicate with p_i, ensuring that initially $p_i \in \Pi_j$ of at least $f_i + 1$ stable processes. Afterwards, if p_i is faulty, eventually at least $f_i + 1$ stable processes will suspect p_i and may spread the suspicion to the remaining of the system, so that the *completeness* property of the FD can be satisfied.

B. Byzantine Failures

Two requirements must be satisfied in a system prone to Byzantine failures: (i) correct processes must have a coherent view of the messages sent by every process;

(ii) correct processes must be able to verify if a message is consistent with the requirements of the algorithm in execution. Thus, Byzantine failure detection is defined as a function of some algorithm $\mathcal{A}$. The first requirement may be addressed by two distinct techniques: information redundancy or unforgeable digital signatures; the second requirement can be met by adding certificates to the messages, so that its content may be validated [Sch96].

Two superclasses of Byzantine failures can be distinguished [KMMS03]: *detectable*, when the external behavior of a process provides evidence of the failure and *non-detectable*, otherwise. Non-detectable failures are grouped in unobservable, when other processes cannot perceive the occurrence of a failure (e.g., when a faulty process informs a parameter different from the supplied by the user) and undiagnosable, when it is not possible to identify the perpetrator of failure (e.g., the processes receive an unsigned message).

This work deals with detectable failures. They are classified in *omission* (or *progress*) failures and *commission* (or *security*) failures. Omission failures hampers the termination of the computation, since a faulty process does not send the messages required by the specification or sends it only to part of the system. Commission failures violate invariant properties to which processes must obey, and can be defined as the noncompliance of one of the following restrictions: (i) a process must send the same messages to every other; (ii) the messages sent must conform the algorithm $\mathcal{A}$ under execution.

C. Byzantine Failure Detector Definition

Kihlstrom *et al.* [KMMS03] define Byzantine failure detector classes which differ from those described by Chandra and Toueg [CT96], since the latter deals only with crash failures. Let $\mathcal{A}$ be an algorithm that uses the failure detector as a underlying module. The class $\Diamond\mathcal{S}(Byz, \mathcal{A})$ is an adaptation of the $\Diamond\mathcal{S}$ class to Byzantine failures. It is the focus of our work, but we should adapt its properties in order to implement it in a dynamic network.

With this aim, we define the class of *Eventually Strong Byzantine Failure Detectors with Unknown Membership*, namely $\Diamond\mathcal{S}^M(Byz, \mathcal{A})$. It keeps the same properties of $\Diamond\mathcal{S}(Byz, \mathcal{A})$, except that they are now valid to known processes, that are either stable or faulty. Informally, these properties are:

- *Strong Byzantine completeness* (for $\mathcal{A}$): eventually, every stable known process suspects permanently every process that has detectably deviated from $\mathcal{A}$;
- *Eventual weak accuracy*: eventually, one stable known process is never suspected by any stable known process.

Definition 2: **Eventually Strong Byzantine FD with Unknown Membership** ($\Diamond\mathcal{S}^M(Byz, \mathcal{A})$). Let $t, t' \in \mathcal{T}$. Let p_i, p_j be nodes. Let $susp_j$ be the list of processes that p_j currently suspects of being faulty. The $\Diamond\mathcal{S}^M(Byz, \mathcal{A})$ class contains all the failure detectors that satisfy:

Strong Byzantine completeness (for $\mathcal{A}$) $\stackrel{def}{=}$ $\{\exists t,\ \forall t' \geq t,\ \forall p_i \in \text{KNOWN} \cap \text{FAULTY} \Rightarrow p_i \in susp_j,\ \forall p_j \in \text{KNOWN} \cap \text{STABLE}\}$;

Eventual weak accuracy $\stackrel{def}{=}$ $\{\exists t,\ \forall t' \geq t,\ \exists p_i \in \text{KNOWN} \cap \text{STABLE} \Rightarrow p_i \notin susp_j,\ \forall p_j \in \text{KNOWN} \cap \text{STABLE}\}$.

III. Towards a Time-Free $\Diamond\mathcal{S}^M(Byz, \mathcal{A})$ Byzantine Failure Detector

A. Local Message Exchange Pattern

Most of the protocols for crash failure detection are based on the exchange of heartbeat messages of the failure detector. Nevertheless, in a Byzantine environment, due to the occurrence of Byzantine processes, such a mechanism is no longer enough. A faulty process may correctly answer the failure detector messages, yet without guaranteeing progress and safety to the algorithm under execution. Therefore, failure detection should be based on the pattern of the messages sent during the execution of the algorithm $\mathcal{A}$ which uses the failure detector as a building block [KMMS03].

We advocate the use of the time-free approach to raise suspicions and propose a FD protocol whose detection does not use timers but is based on the exchange of messages required by algorithm $\mathcal{A}$. Thus, when algorithm $\mathcal{A}$ requires the processes to exchange a message m, every process p_i waits until the reception of m from at least α_i distinct senders; for the remaining processes ($\in \Pi_i$), it raises an omission failure suspicion. In this case, p_i will send a SUSPICION message to processes in its neighborhood, carrying out its local view about suspicions. The detection follows a local message exchange pattern, i.e., between the nodes in the neighborhood [SAB+08]; thus, α_i corresponds to the minimum amount of stable known nodes in the neighborhood of p_i, i.e., $\alpha_i = N_i - f_i$. Knowing that $|N_i| > 2f_i,\ \alpha_i \geq f_i + 1$ (from the Byzantine coverage). The actual value of α_i depends on the type of dynamic network considered (WSN, WMN) as well as the topology of the network during execution.

B. Suspicion Generation

Every SUSPICION message of an omission faulty raised over p_j is related to a message m required by $\mathcal{A}$. That is, p_j is suspected of not sending the messages of $\mathcal{A}$ it should. Thus, messages must have unique identifiers. Suspicions are propagated on the network and a stable process will adopt a suspicion not generated by itself if and only if it receives it properly signed from at least $f_i + 1$ different senders. This requirement denies a Byzantine process to impose suspicions on stable processes. Since the network has a Byzantine coverage, at least (f_i+1) neighbors of p_i are stable and shall spread a suspicion of its failure to their respective neighbors. Since moreover there is a path formed only by stable processes between any stable process, eventually a stable process receives at least $f_i + 1$ occurrences of this suspicion and may adopt it. This ensures the satisfaction

of the *strong Byzantine completeness* property of the $\Diamond\mathcal{S}^M(Byz, \mathcal{A})$ FD.

C. Mistake Generation

Let p_i be a process that has been suspected of not sending a message m. If eventually a stable process p_j receives m properly signed from p_i, p_j will declare a *mistake* on the suspicion and will spread m to the remaining nodes, so that they can do the same. In a network with Byzantine coverage, there will be at least one path formed only by stable processes between p_j and every stable process. Then, every other stable process will receive m and will be able to remove the related suspicion. This behavior allows a Byzantine process to provoke a suspicion and revoke it continuously, masking part of the omission failures and degrading the failure detector performance. Nevertheless, it is not possible to distinguish that situation from the slowness of a process or an instability on the channel.

D. Security Failure Detection

In order to enable the *commission* (or *security*) failure detection, a message format must be established. Every message must also include a certificate that enables other processes to verify its coherence with algorithm $\mathcal{A}$. If a stable process detects the non validity of a received message, either for not obeying to the format or for a non valid justification, it will permanently suspect the sender and will forward the message to the remaining processes, so that the suspicion is propagated.

Notice that it is also necessary to detect *mutant messages*. This anomaly happens when a process sends two or more different versions of the same message. In their protocol, Kihlstrom *et al.* [KMMS03] deals with this problem by requiring stable processes to forward every received message. Moreover, processes should maintain a history of messages received by every process. Their model suppose, though, a point-to-point communication. In our model, processes communicate only through local broadcast under fair-lossy channels. Based on the recent advances and model propositions to implement reliable wireless channels [Koo04], [PP05], [BV05], [BV07], [BV10], we can assume that a message broadcast will be received with equal content by every stable process, so that it is impossible to send mutant messages.

E. Characteristics of the Overlying Algorithm

It is important to notice that, since the detection follows an asynchronous pattern in which suspicions are based on the message exchange, the communication pattern followed by algorithm $\mathcal{A}$ must be distributed. That is, all nodes must exchange messages, following a $n \rightarrow n$ pattern. So, the protocol followed by $\mathcal{A}$ should be symmetrical. Since the proposed detector uses a local message exchange pattern, this symmetrical communication must occur at least between processes in the same communication range. Thus, we conjecture to be impossible to detect omission failures if such a pattern is of the form $1 \rightarrow n$. That is, if at any moment of the algorithm execution, only one process is required to send messages. Otherwise, one could not distinguish an omission failure from a delay on the delivering of the message from that process, since the underlying system is asynchronous [CT96]. Thus, we identify the following conjecture, and if it is correct, it derives the following corollary.

Conjecture 1: In an asynchronous system, it is not possible to detect Byzantine omission failures time-freely if algorithm $\mathcal{A}$ allows a message exchange pattern $1 \rightarrow n$; that is, if algorithm $\mathcal{A}$ requires only a single process to send messages to the remaining.

Corollary 1: The time-free mode of Byzantine failure detection may only be adopted by symmetrical protocols, on which all nodes execute the same role.

In practice, if the communication pattern followed by algorithm $\mathcal{A}$ is not distributed, one can simulate this pattern by requiring processes to relay the messages received to the other processes. Thus, when a process receives a message for the first time, before proceeding with the computation, it must send it to all processes.

F. Behavioral System Properties

In order to satisfy *eventual weak accuracy* property of the $\Diamond\mathcal{S}^M(Byz, \mathcal{A})$ FD class, there must exist a stable process p_i whose messages from some point on are always among the first messages received by its neighbors, at every request of $\mathcal{A}$. Thus, eventually p_i will no longer be suspected by any stable process, and any previous suspicion will be revoked through mistake messages. The *Byzantine responsiveness* property characterizes this desired behavior.

Property 1: $Byz\mathcal{RP}$ (Byzantine Responsiveness Property). Let $t'', t', t \in \mathcal{T}$. Let $rec\_from_j^{t'}(rec\_from_j^{t''})$ be the set of processes from which p_j received the message required by $\mathcal{A}$ at its last step in execution until $t(t'')$. p_i satisfies $Byz\mathcal{RP}$ at time t if:

$Byz\mathcal{RP}^t(p_i) \stackrel{def}{=} (p_i \in \text{KNOWN} \cap \text{STABLE}) \wedge (\forall t' \geq t, \forall t'' > t', p_i \in rec\_from_j^{t'} \Rightarrow p_i \in rec\_from_j^{t''} \vee p_j$ *is faulty*$)$

Thus, the following behavioral assumption should be satisfied in the network in order to implement $\Diamond\mathcal{S}^M(Byz, \mathcal{A})$: $\exists p_i \in \text{KNOWN} \cap \text{STABLE} : Byz\mathcal{RP}^t(p_i)$ eventually holds.

As a matter of comparison, in the timer-based model, the $Byz\mathcal{RP}^t(p_i)$ property would approximate the following: there is a time t after which the output channels from a stable process p_i to every other process p_j that knows p_i are eventually timely. That assumption coincides to the classical one used to implement $\Diamond S$ FDs for crash failures in traditional networks.

G. Practical Issues

WSNs and WMNs are a good examples of networks who would satisfy the assumptions of our model, specially the $Byz\mathcal{RP}$ property and network assumptions. In a WMN, the client nodes move around a fixed set of nodes (the backbone of the network) and each mobile node eventually connects to a fix node. A WSN is composed of stationary nodes and can be organized in clusters, so that communication overhead can be reduced; one node in each cluster is designated the cluster head (CH) and the other nodes, cluster members (CMs). Communication inter-clusters is always routed through the respective CHs which act as gateway nodes and are responsible for maintaining the connectivity among neighboring CHs. For all these platforms, special nodes (the fixed nodes for WMN, CHs for WSN) eventually form a strongly connected component of stable nodes; additionally, some of these nodes can be regarded as fast, so that they will always answer messages faster than the other nodes, considered as slow nodes. Thus, one of these fast nodes may satisfy the $Byz\mathcal{RP}$ property. The stability conditions and the $Byz\mathcal{RP}$ may seem strong, but in practice they should just hold during the time the application needs the completeness and accuracy properties of FDs of class $\Diamond\mathcal{S}^M(Byz, \mathcal{A})$, as for instance, the time to execute a consensus algorithm [KMMS03].

IV. Conclusion

The ideas presented in this paper have been used to implement a Byzantine failure detector of class $\Diamond\mathcal{S}^M(Byz, \mathcal{A})$ and a detailed description of this work, including system model, algorithms and proofs, may be found at [dLGAS10]. We believe that this FD has two innovative characteristics: (i) it is suitable for dynamic distributed systems in which the membership is unknown and (ii) it does not rely on timers to detect omission failures. These features favor the scalability and adaptability and lead to an intriguing conjecture about the overlying algorithm that uses the FD as a building block: it should be symmetrical with a distributed communication pattern.

References

[Agu04] Marcos K. Aguilera. A Pleasant Stroll through the Land of Infinitely Many Creatures. *ACM SIGACT News*, 35(2):36–59, June 2004.

[AHNRR02] Baruch Awerbuch, David Holmer, Cristina Nita-Rotaru, and Herbert Rubens. An on-demand secure routing protocol resilient to byzantine failures. In *Proc. of the 1st ACM Workshop on Wireless Security*, pages 21–30, New York, NY, USA, 2002. ACM.

[BHRT03] R. Baldoni, J. Hélary, M. Raynal, and L. Tanguy. Consensus in Byzantine asynchronous systems. *J. Discrete Algorithms 1(2): 185-210 (2003)*, 1 2003.

[BV05] 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.

[BV07] Vartika Bhandari and Nitin H. Vaidya. Reliable local broadcast in a wireless network prone to byzantine failures. In *DIALM-POMC*, 2007.

[BV10] Vartika Bhandari and Nitin H. Vaidya. Reliable broadcast in radio networks with locally bounded failures. *IEEE Transactions on Parallel and Distributed Systems*, 21:801–811, 2010.

[CT96] Tushar Deepak Chandra and Sam Toueg. Unreliable failure detectors for reliable distributed systems. *J. ACM*, 43(2):225–267, 1996.

[dLGAS10] Murilo Santos de Lima, Fabíola Gonçalves Pereira Greve, Luciana Arantes, and Pierre Sens. Byzantine failure detection for dynamic distributed systems. Technical report, INRIA, Paris, France, 2010.

[Dou02] John R. Douceur. The sybil attack. In *Revised Papers from the First Int. Workshop on Peer-to-Peer Systems*, pages 251–260, London, UK, 2002. Springer-Verlag.

[DS98] Assia Doudou and André Schiper. Muteness detectors for consensus with byzantine processes. In *PODC '98: Proceedings of the seventeenth annual ACM symposium on Principles of distributed computing*, page 315, New York, NY, USA, 1998. ACM.

[HK09] Andreas Haeberlen and Petr Kuznetsov. The Fault Detection Problem. In *Proceedings of the 13th International Conference on Principles of Distributed Systems (OPODIS'09)*, December 2009.

[HKD07] Andreas Haeberlen, Petr Kuznetsov, and Peter Druschel. PeerReview: Practical accountability for distributed systems. In *Proceedings of the 21st ACM Symposium on Operating Systems Principles (SOSP'07)*, Oct 2007.

[JAF06] Ernesto Jiménez, Sergio Arévalo, and Antonio Fernández. Implementing unreliable failure detectors with unknown membership. *Inf. Process. Lett.*, 100(2):60–63, 2006.

[KMMS03] Kim Potter Kihlstrom, Louise E. Moser, and P. M. Melliar-Smith. Byzantine Fault Detectors for Solving Consensus. *The Computer Journal*, 46(1):16–35, January 2003.

[Koo04] Chiu-Yuen Koo. Broadcast in radio networks tolerating byzantine adversarial behavior. In *PODC '04: Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, pages 275–282, New York, NY, USA, 2004. ACM.

[LSP82] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[MMR03] Achour Mostefaoui, Eric Mourgaya, and Michel Raynal. Asynchronous Implementation of Failure Detectors. In *Proc. of the 2003 Int. Conf. on Dependable Systems and Networks*, page 351, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

[MR97] D. Malkhi and M. Reiter. Unreliable intrusion detection in distributed computations. In *Proc. 10th Computer Security Foundations Workshop*, pages 116–124, Rockport, MA, 1997. IEEE Computer Society Press, Los Alamitos, CA.

[MRT+05] Anchour Mostefaoui, Michel Raynal, Corentin Travers, Stacy Patterson, Divyakant Agrawal, and Amr El Abbadi. From Static Distributed Systems to Dynamic Systems. In *Proc. of the 24th IEEE Symp. on Reliable Distributed Systems*, pages 109–118, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

[PP05] Andrzej Pelc and David Peleg. Broadcasting with locally bounded byzantine faults. *Inf. Process. Lett.*, 93(3):109–115, 2005.

[SAB+08] Pierre Sens, Luciana Arantes, Mathieu Bouillaguet, Véronique Simon, and Fabíola Greve. An unreliable failure detector for unknown and mobile networks. In *Proc. of the 12th Int. Conf. on Principles of Distributed Systems*, pages 555–559, Berlin, Heidelberg, 2008. Springer-Verlag.

[Sch96] Bruce Schneier. *Applied Cryptography (2nd ed.)*. John Wiley & Sons, Inc., New York, NY, USA, 1996.