A Failure Detector That Gives Information on the Degree of Confidence in the System

Anubis Graciela de Moraes Rossetto
Cláudio Geyer
Institute of Informatics
Federal University of Rio Grande do Sul (UFRGS)
Porto Alegre, Brazil
Email: {agmrossetto, geyer}@inf.ufrgs.br

Luciana Arantes
Pierre Sens
Sorbonne Universités, UPMC, CNRS, INRIA, LIP6
Paris, France
Email: {luciana.arantes, pierre.sens}@lip6.fr

***Abstract*—This work proposes a new and flexible unreliable failure detector, denoted Impact Failure Detector (FD), whose output gives the trust level of a set of processes. By expressing the relevance of each node by an impact factor value as well as an acceptable margin of failure in the system, the Impact FD enables the user to tune the failure detection configuration in accordance with the requirements of the application: in some scenarios, the failure of low impact or redundant nodes does not jeopardize the confidence in the system, while the crash resulting from a high impact factor may seriously affect it. Either a softer or stricter monitoring is thus possible. Performance evaluation results using real PlanetLab [1] traces confirm the degree of flexibility of our failure detector and that, due to the margin of failure, the number of false responses may be reduced when it is compared with traditional unreliable failure detectors.**

I. Introduction

An unreliable FD can be seen as an oracle that gives (not always correct) information, about process failures. Most of them are based on a binary model, in which monitored processes are either "trusted" or "suspected". Thus, the majority of existing FDs, such as those defined in [2] [3], output the set of processes that is currently suspected to have crashed. A non-binary approach is adopted in [4], the accrual failure detector, which outputs a suspicion level on a continuous scale.

This paper presents a new unreliable failure detector, called *Impact Failure Detector* (*Impact FD*), whose output is a trust level concerning a set S of monitoring processes. The output can be considered as the degree of confidence in S, i.e., the confidence in the system as a whole. Hence, an *impact factor* value is assigned to each process of the set. Furthermore, a *threshold* parameter defines a limit value above which the confidence degree of the set is not compromised. The *impact factor* indicates the relative importance of the process in the set S, while the *threshold* offers a degree of flexibility for failures and false suspicions, thus allowing a higher tolerance of instability in the system. For instance, in an unstable network, although there might be many false suspicions, depending on the value assigned to the *threshold*, the system might remain trustworthy.

If the output value of the impact FD, denoted *trust level*, is below the limit defined by the *threshold*, the user decides which measures should be taken and whether the latter are urgent or not, with regard to the value of the trust level. In the concept of the Impact FD, the monitored processes can also be grouped, based on some criterion such as process type (e.g. nodes, sensors) or their relevance. It is worth remarking that, together with the *threshold*, the group feature of the Impact FD can characterize processes redundancy.

The Impact FD can be applied to different distributed scenarios and is flexible enough to meet different requirements. It is suitable for environments where there is node redundancy. For instance, in Ubiquitous Wireless Sensor Networks (WSNs), often used to monitor physical conditions of geographical regions, sensors are usually of different types (humidity control, temperature control, etc.) and the number of sensors distributed in the region can vary. Sensors are prone to failure and, in this case, redundancy guarantees the coverage of the region and network connectivity [5]. Thus, the region can be regarded as a single set in which the sensors of the same type are grouped into subsets. Each sensor would have an *impact factor* value equal to 1, and the *threshold* would then be equal to the minimum number of sensors that each subset must have in order to ensure connectivity and application operations. Furthermore, if necessary, the degree of redundancy of the sensor subsets can be easily configured dynamically by simply changing the value of the *threshold*. A second example might be a system with a main server that offers a certain quality of service X (bandwidth, response time, etc.). If it fails, N backup servers can replace it, since each backup offers the same service but with a X/N quality of service. In this scenario, both the impact factor of the main server and the *threshold* would have the value of $N * Iback$ where $Iback$ is the impact value of the backup servers, i.e., the system becomes unreliable whenever the primary server and one or more of the N servers fail (or are suspected of being faulty).

We should emphasize that the output of the Impact FD does not depend on the identity of the processes. Thus, it is easily configurable to the needs of the problem, type of accuracy or system configuration. For instance, the Impact FD may be applied to anonymous[1] [6], non-anonymous, or homonymous [2] [7] systems.

This paper is structured as follows. Section II outlines some basic concepts of unreliable failure detectors. Section

[1]The anonymous processes cannot be distinguished one each other: they have no name and execute the same code

[2]Homonymous means that several processes may have the same identifier

978-1-4673-7194-0/15/$31.00 ©2015 IEEE

III describes the Impact Failure Detector. Section IV presents some preliminary evaluation results obtained from experiments conducted with real traces on PlanetLab [1]. Section V discusses some existing related studies. Finally, Section VI concludes the paper and outlines some of our planned future research work.

II. Unreliable Failure Detectors

An important abstraction for the development of fault tolerant distributed systems is the unreliable failure detector [2]. The aim of the latter is to encapsulate the uncertainty of the communication delay between two distributed entities.

In the following section of this article, we consider that there is one process by node (site) or sensor. Thus, the word process can mean either a node, a sensor, or a site. We define a *correct* process the one which never fails during the whole execution; otherwise it is *faulty*.

An unreliable FD can be seen as an oracle that gives (not always correct) information about process failures (either trusted or suspected). It usually provides a list of processes suspected of having crashed. According to [8], an unreliable FD module can make mistakes (1) by erroneously suspecting a correct process (false suspicion), or (2) by not suspecting a process that has actually crashed. If the FD detects its mistake later, it corrects it. For instance, a FD can stop suspecting at time t + 1, a process that it suspected at time t. Unreliable FDs are characterized by two properties, *completeness* and *accuracy*, as defined in [2]. Completeness characterizes the failure detector's capability of suspecting faulty processes, while accuracy characterizes the failure detector's capability of not suspecting correct processes, i.e., restricts the mistakes that the FD can make. FDs are then classified according to two completeness properties and four accuracy properties [2]:

- Strong (resp. weak) completeness: Eventually every process that crashes is permanently suspected by every (resp. some) correct process.
- Strong (resp. weak) accuracy: No (resp. some) process is suspected before it crashes.
- Eventual strong (resp. weak) accuracy: There is a time after which correct processes (resp. some correct process) are (resp. is) never suspected by any correct process.

In *synchronous* systems, there exist a known upper bound on the time required for a message transmission between two processes and on the relative speeds of processes. In *asynchronous* systems, such bounds do not exist while in *partial synchronous* systems, the bounds are known, but are only guaranteed to hold eventually, i.e., after a some unknown time t, denote Global Stabilization Time (GST) [2]. Therefore, the type of accuracy of a FD depends on the synchronism of the system. For instance, a strong accuracy requires synchronous systems while an eventual strong accuracy relies on partially synchronous systems.

III. Impact Failure Detector

We consider a distributed system which consists of a finite set [3] of processes $\Pi = \{q_1, \ldots, q_n\}$ with $|\Pi| = n$. Failures are only by crash. Other types of failures (e.g. misbehavior, transient, etc) are the object of a study that will be carried out in the near future. A crashed process never recovers. We assume the existence of some global time denoted T. A failure pattern is a function $F : T \rightarrow 2^{\Pi}$, where $F(t)$ is the set of processes that have failed before or at time t. The function $correct(F)$ denotes the set of correct processes, i.e., those that have never belonged to a failure pattern (F), while $faulty(F)$ denotes the set of faulty processes, i.e., the complement of $correct(F)$ with respect to Π.

The Impact FD can be defined as an unreliable failure detector that provides an output related to the trust level with regard to a set of processes. If the trust level provided by the detector, is equal to, or greater than, a given threshold value, defined by the user, the confidence in the set of processes is ensured. We can thus say that the system is trusted.

We denote FD (${I_p}^S$) as the Impact failure detector module of process p. Let S be a set of processes. Each process $q \in S$ has an *impact factor* value ($I_q | I_q > 0 : I_q \in \mathbb{R}$). Moreover, the set S can be partitioned into m disjoint subsets. Notice that the grouping feature of the Impact FD allows S to be partitioned into disjoint subsets, in accordance with a particular criterion. For instance, in a scenario where there are different types of sensors, those of the same type can be gathered in the same subset. Let then define $S^* = \{S_1^*, S_2^*, ...S_m^*\}$ as the set S partitioned into m disjoint subsets where each S_i^* is a set composed of the tuple $\langle id, impact \rangle$, where id is a process identifier and $impact$ is the value of the impact factor of the process in question.

When invoked in p, the Impact FD (${I_p}^S$) returns the ${trust\_level_p}^S$ value. The ${trust\_level_p}^S$ is a set that contains the trust level of each subset S^*, i.e., it expresses the confidence that p has in the set S.

An acceptable margin of failures, denoted as the $threshold^S$, characterizes the acceptable degree of failure flexibility in relation to set S^*. The $threshold^S$ is adjusted to the minimum trust level required for each subset, i.e., it is defined as a set which contains the respective threshold of each subset of S^*: $threshold^S = \{threshold_1, \ldots, threshold_m\}$.

We denote ${trusted_p}^S(t) = \{trusted_1, \ldots, trusted_m\}$, where each $trusted_i$ $(1 \leq i \leq m)$ contains the processes of subset S_i^* that are not considered faulty by p at time $t \in T$. Similarly to S_i^*, each $trusted_i$ is composed of the tuple $\langle id, impact \rangle$. The *trust level* at $t \in T$ of processes $p \notin F(t)$ of S is the function ${trust\_level_p}^S$ such that ${trust\_level_p}^S(t) = \{trust\_level_i | trust\_level_i = sum(trusted_i); 1 \leq i \leq m\}$. The function $sum(subset)$ returns the sum of the impact factor of all the elements of $subset$.

The $threshold^S$ is used by the application to check the confidence in the processes of S. If, for each subset of S^*, the $trust\_level_i(t) \geq threshold_i$, S is considered to be *trusted* at t by p, i.e., the confidence of p in S has not been compromised; otherwise S is considered *not trusted* by p at t.

Two points should be highlighted: (1) both the *impact factor* and $threshold^S$ render the estimation of the confidence in S flexible. For instance, it is possible that some processes in S might be faulty or suspected of being faulty but S can still be trusted; (2) the Impact FD can be easily configured to adapt

[3] In this work, 'set' and 'multiset' are used interchangeably. Unlike a set, an element of a multiset can appear more than once. This allows different processes to have the same identity.

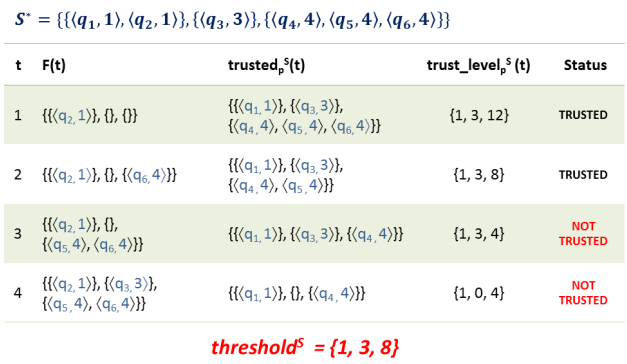

Fig. 1. Example of FD ($I_p{}^S$) output: S^* has three subsets

to the needs of the environment. The $threshold^S$ can be tuned to provide a more restricted or softer monitoring. This kind of adaptability is essential in dynamic environments (such as the ubiquitous one). Note that the Impact FD can also be applied when the application needs individual information about each process of S. In this case, each process must be defined as a subset of S^*.

In Figure 1, we consider a set S, where S^* is composed by three subsets: S_1, S_2, and S_3. The values of $threshold^S$ define that the subsets S_1 and S_2 (resp. S_3) must have at least one (resp. 2) correct process(es). The figure shows several possible outputs for FD ($I_p{}^S$): the set S is considered trusted whenever, for each subset S_i^*, $trust\_level_i(t) \geq threshold_i$.

IV. Performance Evaluation

In this section, we firstly describe the environment in which the experiments were conducted, the QoS metrics used for evaluating the results and how to estimate the heartbeats arrival time. Then, we discuss some of the results with different configurations of node sets with regard to both the impact factor and the threshold. We also compared the latter with some results related to Chen FD [9], whose output is a list of suspected processes.

A. Environment

We used realistic trace files collected from ten nodes of PlanetLab [1]. The experiment started on July 16, 2014 at 15:06 UTC, and ended exactly a week later. Each site sent heartbeat messages to other sites at a rate of one heartbeat every 100 ms (the sending interval). We should point out that these traces of PlanetLab contain a large amount of data concerning the sending and reception of heartbeats, including unstable periods of links and message loss which induce false suspicions. Thus, they can characterize any distributed system that uses FDs based on heartbeat. Furthermore, since the sending and arrival times of each heartbeat are recorded in the trace files, all the experiments were conducted with exactly the same scenarios and history of heartbeats.

Table I shows information about the heartbeat messages received by site number *1* (the monitor node). We observed that the mean inter-arrival times of received heartbeats was very close to 100 ms. However, in some sites, the standard deviation is very high, like in site *5* which the standard deviation was 310.958 ms with a minimum of 0.006 ms, and a maximum of

TABLE I. Sites and heartbeat sampling

Site	Messages	Mean (ms)	Min (ms)	Max (ms)	Stand. Dev.(ms)
0	5424326	100.058	0.025	26494.168	19.525
2	1759989	100.415	0.031	509.093	9.275
3	5426843	100.012	0.027	1227.349	1.709
4	5414122	100.247	0.003	1193.276	18.595
5	5413542	100.258	0.006	657900.226	310.958
6	5426700	100.015	0.003	3787.643	2.557
7	5424117	100.062	0.006	59603.188	31.229
8	5424560	100.054	0.027	11443.359	100.714
9	5422043	100.100	0.004	30600.076	18.798

657900.226 ms. Such a behavior shows that, for a certain time interval during execution, the site stopped sending heartbeats and started again afterwards. Note also that Site *2* stopped sending messages after approximately 48 hours and, therefore, there are just 1759990 received messages.

B. QoS Metrics

For evaluating the Impact FD, we used three of the QoS metrics proposed by [9]: detection time, average mistake rate, and query accuracy probability. Considering two processes, q and p, where p monitors q, the QoS of the FD at p can be evaluated from the transitions between the "trusted" and "not trusted" states with respect to q.

- Detection Time (T_D): the time that elapses from the moment that process q crashes until the FD at p starts suspecting q permanently.
- Average Mistake Rate (λ_R): represents the number of mistakes that FD makes in a unit time, i.e., the rate at each a FD makes mistakes.
- Query Accuracy Probability (P_A): the probability that the FD output is correct at a random time.

C. Implementation of the Imapct FD

The implementation of Impact FD for the evaluation was based on Algorithm 1 presented in [10]. This algorithm employs the timer-based approach which uses Chen's heartbeat estimation. Moreover, we consider an asynchronous system with message losses. In the *timer-based* strategy, FD implementations make use of timers to detect failures in processes. Every process q periodically sends a control message (*heartbeat*) to process p that is responsible for monitoring q. If p does not receive such a message from q after the expiration of a timer, it adds q to its list of suspected processes.

In order to calibrate the timer, we applied the estimation of heartbeat arrivals presented in [9], as described below.

D. Estimation of heartbeat arrivals

Aiming at reducing both the number of false suspicions and the time needed to detect a failure, Chen et al. [9] propose an approach to estimate the arrival of the next heartbeat which is based on the history of the arrival time of heartbeats and includes a safety margin (β). The timer is then set according to this estimation.

The estimation algorithm is the following: process p takes into account the n most recent heartbeat messages received from q, denoted by $m_1, m_2, \ldots, m_n$; $A_1, A_2, \ldots, A_n$ are their

TABLE II. SET CONFIGURATIONS

Config	Impact Factor of each site
Set 0	I_0=7; I_2=3; I_3=20; I_4=20; I_5=3; I_6=20; I_7=3; I_8=7; I_9=7;
Set 1	I_0=7; I_2=20; I_3=20; I_4=3; I_5=3; I_6=20; I_7=3; I_8=7; I_9=7;
Set 2	I_0=20; I_2=7; I_3=3; I_4=3; I_5=7; I_6=3; I_7=7; I_8=20; I_9=20;
Set 3	I_0=7; I_2=3; I_3=20; I_4=3; I_5=3; I_6=20; I_7=7; I_8=20; I_9=7;
Set 4	I_0=10; I_2=10; I_3=10; I_4=10; I_5=10; I_6=10; I_7=10; I_8=10; I_9=10;

actual reception times according to p's local clock. When at least n messages have been received, the theoretical arrival time $EA_{(k+1)}$ for a heartbeat from q is estimated by:

$$EA_{(k+1)} = \frac{1}{n} \sum_{i=k-n}^{k} (A_i - \Delta_i * i) + (k+1)\Delta_i$$

where Δ_i is the interval between the sending of two q's heartbeats. The next timeout value which will be set in p's timer and will expire at the next freshness point $\tau_{(k+1)}$, is then composed by $EA_{(k+1)}$ and the constant safety margin β:

$$\tau_{(k+1)} = \beta + EA_{(k+1)}$$

For the estimation of the arrival time, the authors suggest that the safety margin β should range from 0 to 2500 ms. For all experiments, we set the window size to 100 samples, which means that the FD only relies on the last 100 heartbeat message samples for computing the estimation of the next heartbeat arrival time.

E. Set Configuration

We defined a set composed of sites 0, 2, 3, 4, 5, 6, 7, 8 and 9 (S= {{*0, 2, 3, 4, 5, 6, 7, 8, 9*}}). Site *1* was the monitor node (p). Table II shows the five configurations with regard to impact factor values that have been considered for S in the experiments. The sum of the impact factor of the processes is 90 for all configurations.

In order to decide about the impact factor value to assign to each site, we have evaluated the stability of the sites. To this end, we monitored the sites individually using Chen's algorithm with β=400ms. Figure 2 shows the cumulative number of mistakes for each site during the whole trace period. We can observe that site or link periods of instability entail late arrivals or loss of heartbeats and, therefore, mistakes by the monitor node. For example, site *9* had a large number of cumulative mistakes at hour 48. After that, there is a stable period with regard to this site. It should also be noted that site *2* stopped sending messages after approximately 48 hours (it crashed) and, consequently, the monitor node made no more mistakes about it after this time. Finally, we can say that, considering the whole period, sites *3* and *6* (resp., *8* and *9*) are, in average, the most stable (resp., unstable) sites.

F. Experiments

1) Experiment 1 - Query Accuracy Probability: The aim of this experiment is to evaluate the Query Accuracy Probability (P_A) with different threshold values (64, 70, 74, 80, and 83) and different impact factor configurations (Table II). We considered the safety margin as being β=400ms.

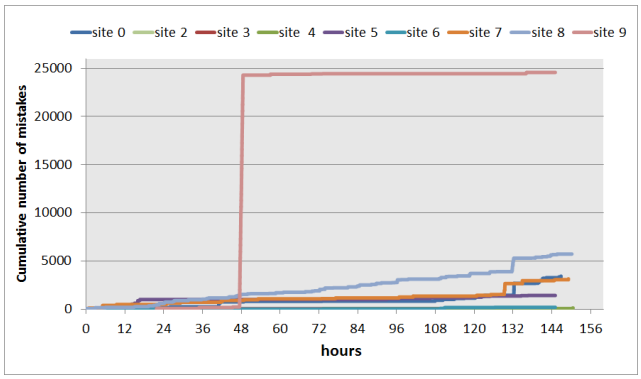

Fig. 2. Cumulative number of mistakes of each site

Figure 3 shows that the P_A decreases when the threshold increases. It should be remembered that the *threshold* is a limit value defined by the user and if the FD trust level output value is equal to, or greater than, the threshold, the confidence in the set of processes is ensured. Hence, the results confirm that when the threshold is more flexible, the Query Accuracy Probability is higher.

On the one hand, except for threshold 83, *"Set 0"* configuration has the highest P_A for most of the *thresholds* due to the assignment of high (resp., low) impact factors for the most stable (resp., unstable) sites. On the other hand, *"Set 2"* and *"Set 4"* have the lowest P_A since unstable sites have high impact factor values assignment. For instance, the hight impact factor value of sites *8* and *9* degrades the P_A of these sets.

"Set 4" shows a sharp decline when the *threshold* = 83. This behavior can be explained since, in this set configuration, all sites have the same impact factor (10) which implies that every false suspicion renders the *trust_level* smaller than the *threshold* (83), increasing the mistake duration. Therefore, the P_A decreases.

Notice that site *2* failed after approximately 48 hours. Thus, after its crash, the FD output, which indicates *trust_level* smaller than the *threshold*, is not a mistake, i.e. it is not a false suspicion. Hence, in *"Set 1"*, where the impact factor of site *2* is 20 (high), the *PA* is constant for a *threshold* greater than 70: after the crash of site *2*, the FD output is always smaller than the *threshold* and false suspicions related to other sites do not alter it. The average mistake duration in the experiment is thus smaller after the crash, which improves the P_A.

Finally, we have compared the P_A of the Impact FD and a FD approach that monitors processes individually by applying Chen's algorithm with *WS=100* and β=400ms. The mean P_A obtained was 0,979788. This result shows that, regardless of the set configuration, the Impact FD has a higher P_A than Chen FD since the former has enough flexibility to tolerate failures, i.e., the mistake duration only starts to be computed when the *trust_level* provided by Impact FD is smaller than the *threshold*, in contrast with individual monitoring, such as that by Chen FD, where every false suspicion increases the mistake duration.

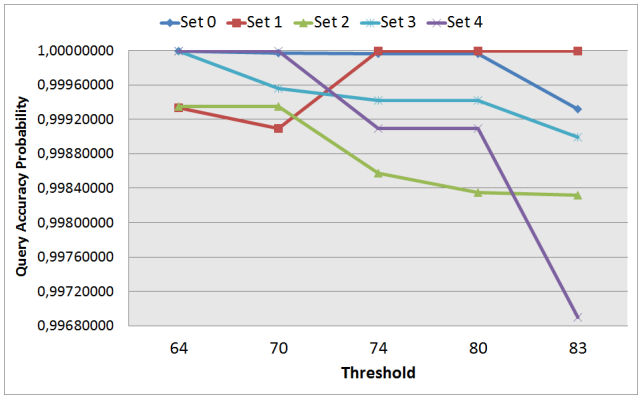

Fig. 3. P_A vs. threshold with different set configurations

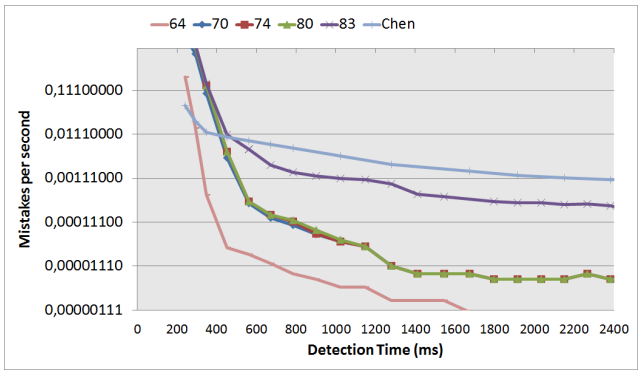

Fig. 4. λ_R vs. T_D with different thresholds

The results of this experiment highlight the fact that the assignment of heterogeneous impact factors to nodes can degrade the performance of the failure detector, especially when unstable sites have a high impact factor.

2) Experiment 2 - Average mistake rate: In the second experiment, we evaluated the average detection time (T_D) vs. the mistake rate (λ_R) (mistakes per second). In order to obtain different values for the detection time, we varied the safety margin (Chen's estimation) with intervals of 100 ms, starting at 100 ms. For this experiment, we chose the *"Set 0"* configuration since it presented the best P_A in Experiment 1. We also evaluated the (λ_R) and T_D for Chen's algorithm, which outputs the set of suspected nodes.

It can be observed in Figure 4 that for a high threshold and detection time close to 250 ms, the mistake rate of the Impact FD is higher, regardless of the threshold, because the safety margin (used to compute the expected arrival times) is, in this case, equal to 100 ms, which increases both the number of failure suspicion and mistake duration. However, when T_D is greater than 400 ms, the mistake rate of Impact FD is smaller than that of Chen. This behavior can be explained by the fact that the higher the safety margin, the smaller the number of false suspicions, and the shorter the mistake duration which confirms that, although failures are detected faster when the timeout is short, there is an increase in the likelihood of having false detections [11].

3) Experiment 3 - Cumulative number of mistakes: In this experiment, we evaluated the cumulative number of mistakes

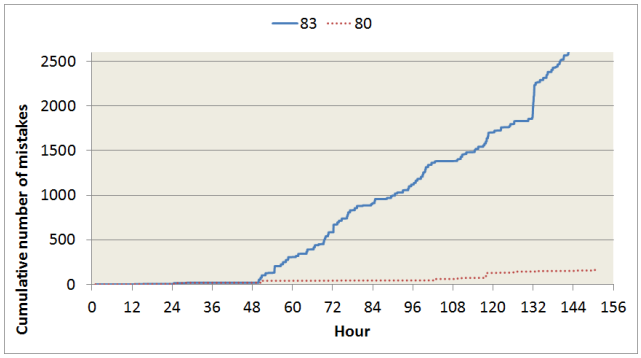

Fig. 5. Cumulative number of mistakes for *"Set 0"* configuration

for the *"Set 0"* configuration during the whole trace period, considering β=400ms and the threshold value of 80 and 83.

We can observe in Figure 5 that the cumulative number of mistakes is greater for the threshold value 83 (2754 mistakes) when compared to threshold value 80 (179 mistakes). The former made few mistakes until approximately the hour 48 (when the site *2* crashed). After that, the number of cumulative mistakes significantly increased because, as the threshold is high (83) and the failure of *2* was detected, false suspicions of any other site induce a *trust_level* value smaller than 83 in most cases. For instance, site *8* is highly unstable and has impact factor value of 7. Whenever there is a false suspicion about it, after the crash of site *2*, the *trust_level* value is 80. On the other hand, for the threshold 80, we can observe that there were fewer instability periods since the crash of site *2* does not have much impact on the confidence of the system. At hour 48, there was an increase in the cumulative number of mistakes due to the unstable period of site *9*, as shown in Figure 2. From hour 50 to 100, the FD made fewer mistakes. Such a behavior can be explained since, as observed in the same figure, all sites, with exception of site *8*, also had this same period of stability. After hour 108, there was a greater number of mistakes which is related to the instability of sites *0*, *7*, and *8* (see Figure 2).

4) Experiment 4 - Query Accuracy Probability vs. Time: In this experiment, we divided the execution trace time by fixed intervals and computed the average (P_A) for each of them. We chose the *"Set 0"* configuration, β=400ms, and the threshold values of 80 and 83. Similarly to the cumulative number of mistakes (Experiment 3), we can note in Figure 6 that instability periods have an impact on the P_A. For instance, for the threshold = 80, from hour 108, the cumulative number of mistakes increases very fast. Consequently, the PA decreases. The period of instability of site 9 is the responsible for the important reduction of the P_A at hour 60 (i.e., from hour 48 to 60) when threshold = 83. A new degradation of the P_A happens at hour 120 (i.e., from hour 108 to 120), due to unstable periods of the sites *0*, *7*, and *8*.

V. Related Work

Bertier et al. [3] introduced a failure detector principally intended for LAN environments. They propose a different estimation function, which combines Chen's estimation with another estimation, due to Jacobson [12] and developed in a different context. It adapts the safety margin each time it

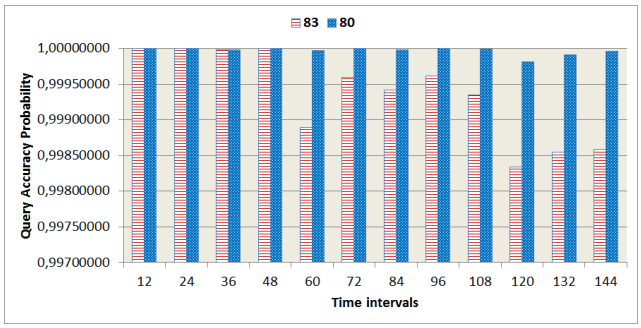

Fig. 6. P_A vs. Time

receives a message. The adaptation is based on the error in the last estimation. Its proposition provides a shorter detection time, but generate more wrong suspicions than when Chen's estimation.

Most of the unreliable fault detectors in the literature are based on a binary model and provide as output a set of process identifiers, which usually reveal the number of processes currently suspected of having failed ([2] [3]). However, in some detectors, such as class Σ (resp., Ω) [13], the output is the set of processes (resp., one process) which are (resp., is) not suspected of being faulty, i.e., $trusted$.

The ϕ Accrual failure detector [4] proposes an approach where the output is a suspicion level on a continuous scale, rather than providing information of a binary nature (trusted or suspected). It is based on an estimation of inter-arrival times which assume that the latter follow a normal distribution. The suspicion level captures the degree of confidence with which a given process is believed to have crashed. If the process actually crashes, the value is guaranteed to accrue over time and tends toward infinity. In [11], the authors extended the Accrual FD by exploiting histogram density estimation. Taking into account a sampled inter-arrival time and the time of the last received heartbeat, the algorithm estimates the probability that no further heartbeat messages arrive from a given process, i.e., it has failed. The aim of Accrual failure detectors is to decouple monitoring from interpretation.

Starting from the premise that applications should have information about failures to take specific and suitable recovery actions, the work in [14] proposes a service to report faults to applications. The latter also encapsulates uncertainty which allows applications to proceed safely in the presence of doubt. The service provides status reports related to fault detection with an abstraction that describes the degree of uncertainty.

Considering that each node has a probability of being byzantine, a voting node redundancy approach is presented in [15] in order to improve reliability of distributed systems. Based on such probability values, the authors estimate the minimum number of machines that the system should have in order to provide a degree of reliability which is equal to or greater than a threshold value.

VI. Conclusion and Future Work

In this paper, we have proposed a new unreliable failure detector, the Impact FD, which provides a single output value related to a set of processes and not to each one individually. Both its *impact factor* and *threshold* concepts offer a degree of flexibility since they enable the user to tune the Impact FD in accordance with the specific needs and acceptable margin of failures of the application. In some scenarios, they also might weaken the rate of false responses when compared to traditional unreliable failure detectors. The performance evaluation results show that the assignment of a high (resp. low) impact factor to more stable (resp. unstable) nodes increases the Query Accuracy Probability of the failure detector.

In a future study, we intend to extend the Impact FD so that it addresses the question of misbehavior failures. We are also working on the reduction and equivalence of Impact FD and other detectors, which will require some new assumptions and/or new definitions. A third aim is to conduct other experiments on different networks such as WiFi or LAN.

References

[1] PlanetLab, "Planetlab," http://www.planet-lab.org, 2014.

[2] T. D. Chandra and S. Toueg, "Unreliable failure detectors for reliable distributed systems," *Journal of the ACM (JACM)*, vol. 43, no. 2, pp. 225–267, 1996.

[3] M. Bertier, O. Marin, P. Sens *et al.*, "Performance analysis of a hierarchical failure detector." in *DSN*, vol. 3, 2003, pp. 635–644.

[4] N. Hayashibara, X. Defago, R. Yared, and T. Katayama, "The φ accrual failure detector," in *SRDS*. IEEE, 2004, pp. 66–78.

[5] D. Geeta, N. Nalini, and R. C. Biradar, "Fault tolerance in wireless sensor network using hand-off and dynamic power adjustment approach," *Journal of Network and Computer Applications*, vol. 36, no. 4, pp. 1174–1185, 2013.

[6] F. Bonnet and M. Raynal, "The price of anonymity: Optimal consensus despite asynchrony, crash, and anonymity," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 6, no. 4, p. 23, 2011.

[7] S. Arévalo, A. Fernández Anta, D. Imbs, E. Jiménez, and M. Raynal, "Failure detectors in homonymous distributed systems (with an application to consensus)," in *ICDCS*. IEEE, 2012, pp. 275–284.

[8] N. Hayashibara, X. Défago, and T. Katayama, "Two-ways adaptive failure detection with the ϕ-failure detector," in *WADiS*. Citeseer, 2003, pp. 22–27.

[9] W. Chen, S. Toueg, and M. K. Aguilera, "On the quality of service of failure detectors," *Computers, IEEE Transactions on*, vol. 51, no. 5, pp. 561–580, 2002.

[10] A. G. Rossetto, C. F. Geyer, L. Arantes, and P. Sens, "Impact: an unreliable failure detector based on processes relevance and the confidence degree in the system," Tech. Rep., 2015, iNRIA N hal-01136595. [Online]. Available: https://hal.inria.fr/hal-01136595

[11] B. Satzger, A. Pietzowski, W. Trumler, and T. Ungerer, "A new adaptive accrual failure detector for dependable distributed systems," in *ACM SAC*. ACM, 2007, pp. 551–555.

[12] V. Jacobson, "Congestion avoidance and control," in *ACM SIGCOMM Computer Communication Review*, vol. 18, no. 4. ACM, 1988, pp. 314–329.

[13] C. Delporte-Gallet, H. Fauconnier, R. Guerraoui, V. Hadzilacos, P. Kouznetsov, and S. Toueg, "The weakest failure detectors to solve certain fundamental problems in distributed computing," in *PODC*. ACM, 2004, pp. 338–346.

[14] J. B. Leners, T. Gupta, M. K. Aguilera, and M. Walfish, "Improving availability in distributed systems with failure informers," in *NSDI*, 2013.

[15] Y. Brun, G. Edwards, J. Y. Bang, and N. Medvidovic, "Smart redundancy for distributed computation," in *ICDCS*, 2011, pp. 665–676.