Adaptare-FD: A dependability-oriented adaptive failure detector

Mônica Dixit and António Casimiro
Department of Informatics
Faculty of Sciences, University of Lisboa
Lisboa, Portugal
mdixit,casim@di.fc.ul.pt

***Abstract*—Unreliable failure detectors are a fundamental building block in the design of reliable distributed systems. But unreliability must be bounded, despite the uncertainties affecting the timeliness of communication. This is why it is important to reason in terms of the quality of service (QoS) of failure detectors, both in their specification and evaluation.**

We propose a novel dependability-oriented approach for specifying the QoS of failure detectors, and introduce *Adaptare-FD*, an autonomous and adaptive failure detector that executes according to this new specification. The main distinguishing features of *Adaptare-FD* with respect to existing adaptive failure detection approaches are discussed and explained in detail.

A comparative evaluation of *Adaptare-FD* is presented. We highlight the practical differences between our approach and the well known Chen et al. approach for the specification of QoS requirements. We show that *Adaptare-FD* is easily configured, independently of the specific network environment. Furthermore, the results obtained using the PlanetLab platform indicate that *Adaptare-FD* outperforms other timeout-based solutions, combining versatility with improved QoS and dependability assurance.

***Keywords*-dependability; adaptation; failure detection;**

I. Introduction

Unreliable failure detectors (FDs) are a fundamental building block in the design of reliable distributed systems [1]. One important aspect is that they encapsulate the temporal uncertainties observed in asynchronous systems, freeing the system designer from the need to deal with them. However, as shown in [2], the actual implementation of the FD is fundamental to the overall system performance.

Two generic performance-related attributes of failure detectors are their speed (how fast they detect a failure) and their accuracy (how well they avoid making mistakes). With timeout-based failure detectors, as we consider in this paper, these attributes depend on the timeout values and on the period of heartbeats. Therefore, a fundamental problem in the implementation of failure detectors in asynchronous and evolving environments concerns the *configuration* of the operational parameters, which involves the need to handle (non-functional) user-level requirements and the ability to characterize the state of the operational environment.

Different facets of the problem of configuring and building adaptive failure detectors have been previously addressed by many authors. The work in [3] introduced a set of metrics for evaluating (and specifying) the QoS of failure detectors, abstracting from the specific implementation. Building on the basic *Push* and *Pull* styles of algorithm structuring, some works focus on algorithmic techniques to improve performance [4], others on the provision of adequate interfaces to support the varying requirements of applications [5], [6], and others on methods for detecting environment changes and adapting parameters accordingly [3], [7]–[10].

In this paper we propose *Adaptare-FD*, which advances on existing work by introducing a new dependability-oriented approach for the specification of the failure detector QoS and by integrating a recently developed framework for the dependable characterization of environments with varying stochastic behavior [11], [12]. We claim that our approach is well suited and can be easily integrated in the design of dependable systems. We discusse the relative merits of *Adaptare-FD*, based on a comparative analysis with other approaches. In particular, we focus on the comparison with Chen's approach [3], to reveal subtle differences that are not easily observed in a simple outlook.

An evaluation of *Adaptare-FD* is also provided, based on experiments performed in the PlanetLab platform [13]. Several results are presented, which allow to compare *Adaptare-FD* with other timeout-based and adaptive failure detectors [7]–[9]. The results show that *Adaptare-FD* is able to perform better, especially in more dynamic environments. A particularly interesting result is that the well known trade-off between the mistake recurrence time (T_{MR}) and the mistake duration (T_M) can be reduced with *Adaptare-FD*.

An additional advantage of our solution is that it is fully adaptive, even when the stochastic behavior of the environment changes. While *Adaptare-FD* is like a plug-and-play solution, the remaining approaches require some *a priori* configuration, either because they assume a given stochastic behavior, or because they define static parameters according to the expected conditions. They are thus unable to dependably adapt to significant environment changes.

The paper is organized as follows. In the next section we describe the assumed system model and the related adaptive failure detectors considered in our evaluation. In Section III we introduce *Adaptare-FD*, explaining its operation. In Section IV we discuss why our approach is more dependability-oriented. Section V presents the practical evaluation of *Adaptare-FD*. Finally, in Section VI we conclude the paper.

```
Process p:
  upon receive "are you alive?" message $mq_i$ from q:
    send "I'm alive" message $m_i$ to q
Process q:
  Initialization:
    output ← S
    $\eta$ ← compute_$\eta$   //Initialize interrogation period
    $\delta$ ← compute_$\delta$   //Initialize timeout
    lastReceived ← -1
    $\sigma_0$ ← currentTime
  for all i≥1, at time $\sigma_i = \sigma_{i-1} + \eta$
    send "are you alive?" message $mq_i$ to p
  for all i≥1, at time $\tau_i = \sigma_i + \delta$
    if lastReceived < i then
        output ← S
  upon receive "I'm alive" message $m_j$ at time t:
    if j > lastReceived then
        output  ← T
        $\eta$ ← compute_$\eta$ //Adapting interrogation period
        $\delta$ ← compute_$\delta$ //Adapting timeout
        lastReceived  ← j
```

Figure 1. Failure detection algorithm (process *q* monitoring process *p*)

II. Basic context

A. System model

We consider an asynchronous distributed system with a finite set of processes $\Pi = \{p, q, ..., s\}$. Processes are interconnected trough unreliable channels, which can loose messages or discard corrupted messages. Messages can also be arbitrarily delayed due to unbounded transmission and/or processing times. Regarding processes, we assume that they only fail by crashing, but otherwise behave correctly.

We consider a pull-style crash failure detection model where a process *q* (running in one node) monitors a process *p* (in another node), by periodically sending it *"are you alive?"* messages. Process *p* responds to every received *"are you alive?"* message with a corresponding *"I'm alive"* message. Depending on the reception times of *p*'s responses, the output of the failure detector may be T (*trust*), or S (*suspect*). Since we adopt a pull-style failure detector model, no assumptions about synchronized clocks are needed.

Figure 1 shows the basic algorithm that is typically used in pull-style failure detectors, which we also adopted. The distinguishing factor between the different adaptive failure detector approaches lies in the solutions used for computing the interrogation period, η, and the timeout, δ.

These two parameters are dynamically adapted during the execution, depending on the required QoS (initially and statically defined) and on the observed behavior of the communication environment (which may change). Therefore, the functions compute_η and compute_δ encapsulate the configuration procedures that take place during initialization, and periodically upon reception on new *"I'm alive"* responses. The different approaches for adaptation to which we compare our proposal are explained next.

B. Chen's failure detector

The first systematic study of the QoS of failure detectors was presented in [3]. In this work, Chen et. al. define a set of metrics to quantify the QoS of failure detectors, which are independent of their implementations. Then, they propose a failure detector that is configured according to some expected message behavior and to the required QoS. More specifically, the messages behavior is described by the message loss probability p_L, and by the expected value $(E(D))$ and the variance $(V(D))$ of message delays. The QoS of the failure detector is specified by three metrics: an upper bound on the detection time (T_D^U), a lower bound on the average mistake recurrence time (T_{MR}^L) and an upper bound on the average mistake duration (T_M^U).

Figure 2 (from [3]) presents a schematic view of Chen's FD, corresponding to the version that more closely matches our proposal. The authors mention that an adaptive version may be implemented by re-executing the configurator, leading to varying values for the period (η) and timeout (δ).

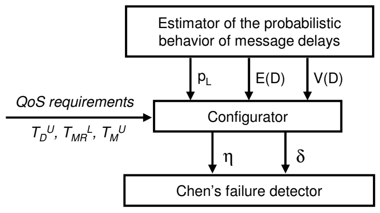
Estimator of the probabilistic behavior of message delays
p_L
E(D)
V(D)
QoS requirements
T_D^U, T_{MR}^L, T_M^U
Configurator
η
δ
Chen's failure detector

Figure 2. Schematic view of Chen's failure detector

C. Other timeout estimation methods

Most of the adaptive failure detectors described in the literature are based on fixed interrogation periods, and a common approach to estimate timeouts: the use of an estimator, plus a safety margin.

Estimators can vary from simple methods, like using the delay of the last received message [9] or the average delay of the last n received messages [7]–[9], to more elaborated approaches, e.g. applying an ARIMA (autoregressive integrated moving average) model to build a time series from the last delays, and using this model to predict the delay of the next message, also studied in [8], [9].

Safety margins may be static or dynamic. Static safety margins, as used in [3], although simpler, require some *a priori* analysis of the execution environment in order to be appropriately defined. The flexibility provided by dynamic safety margins [7] makes them more suitable to network environments susceptible to frequent changes.

The failure detectors used in our comparative evaluation that follow this approach combine two different estimators with two methods to compute dynamic safety margins. The estimators are defined as:

- WINMEAN(n): Computes the average of the delays of the last n received messages.

$$\hat{rtt}_{t+1} = \frac{\sum_{i=t-n}^{t} rtt_i}{t}$$

- LPF(β): A simplified ARIMA estimator, proposed in [9]. The parameter β represents the importance of the error in the last estimation.

$$\hat{rtt}_{t+1} = \hat{rtt}_t + \beta(rtt_t - \hat{rtt}_t)$$

Different approaches to compute dynamic safety margins were proposed in [8], and also used in [9]. They are:

- Cib(n,γ): Assumes that the predictor correctly models the communication delays, and the estimator is a linear function. In the equation, γ corresponds to the confidence level in the standard Normal distribution function, $\hat{\sigma}$ is the estimator of the standard deviation and $\bar{rtt}$ is the mean of the last n observed delays.

$$Cib_{t+1} = \gamma\hat{\sigma}\sqrt{1 + \frac{1}{n} + \frac{(rtt_t - \bar{rtt})^2}{\sum_{i=t-n}^{t}(rtt_i - \bar{rtt})^2}}$$

- Jac(ϕ,α): Based on the Jacobson estimation method [14], considers the error on the last estimation to adapt its value. α is a smoothing constant which defines the speed of the reaction to error variation [8], and ϕ is a multiplier factor that defines how conservative the margin will be.

$$Jac_{t+1} = \phi(Jac_t + \alpha(|rtt_t - \hat{rtt}_t|) - Jac_t$$

III. *Adaptare-FD*

In this work we propose *Adaptare-FD*, a dependability-oriented adaptive failure detector, built upon *Adaptare*, which is a framework for automatic and dependable adaptation, first introduced in [11] and fully described in [12].

The fundamental idea behind our approach is to achieve a failure detector (FD) that dynamically adapts to changing network conditions and is easily configured by specifying a lower bound on the average mistake recurrence time (T_{MR}^L, as with Chen's FD) and the minimum coverage (C^L) for the assumed timeouts (i.e., a lower bound on the probability that *"I'm alive"* replies will be received before the timeout expiration, avoiding mistakes). By specifying the required confidence on a fundamental assumption we achieve a *dependable approach for failure detection*. Figure 3 shows the architecture of *Adaptare-FD* and the two input parameters that determine its overall QoS.

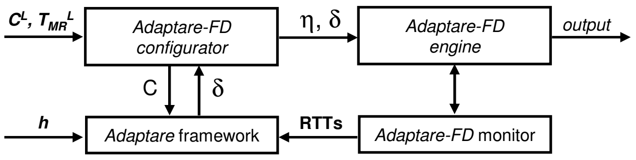

Figure 3. *Adaptare-FD* architecture

The interrogation period η is computed by the configurator module, and is fixed for a given pair (C^L, T_{MR}^L). The procedure is as follows. For failure-free runs, the coverage is specified by $C = 1 - (n\_mistakes)/(n\_observations)$. The number of observations is the total time of observation divided by the interrogation period η. Thus, $C = 1 - (\eta \cdot (n\_mistakes))/(t\_time)$. Since T_{MR} is given by the total time divided by the number of mistakes, we have $C = 1 - \eta/T_{MR}$. So, the value of η can be derived from the input parameters: $\eta = T_{MR}^L \cdot (1 - C^L)$.

The timeout δ is adapted in run-time, according to the environment conditions (network delays) and the required coverage. On the reception of each *"I'm alive"* message, the monitor module of *Adaptare-FD* measures the RTT and provides this information to the *Adaptare* framework, which computes a new dependable δ.

The *Adaptare* framework is a middleware library[1], designed to be generic and usable in different contexts, which is used in *Adaptare-FD* to determine appropriate timeout values. Although the detailed description and evaluation of the framework can be found in [12], here we present the fundamental aspects of its operation.

Essentially, the framework assumes that the system behaves stochastically, and the statistical processes that generate the data flow (e.g., RTTs) can change over time (alternating *stable phases* with *transient phases*). Using an appropriate number of samples (configurable through the input parameter *h*), the framework is able to determine the system state (stable or transient statistical processes), derive corresponding distributions whenever possible, and compute bounds that secure a given coverage.

The network characterization is composed by two phase detection mechanisms, based on the Kolmogorov-Smirnov (KS) and on the Anderson-Darling (AD) goodness-of-fit tests. Both AD and KS are distance tests that compare the cumulative distribution function (CDF) of the assumed distribution $\hat{D}$ to the empirical distribution function (EDF), which is a CDF built from the input samples. The current implementation of *Adaptare* tests five distributions: exponential, shifted exponential, Pareto, Weibull, and uniform. If the phase detection mechanisms do not identify a stable phase with any of these distributions, they assume that the environment is in a transient phase. The parameters of the postulated distributions are estimated using the maximum likelihood estimation and the linear regression methods.

The timeout is computed as follows. If a specific distribution is identified, the timeout is derived from this distribution CDF and the required coverage. Otherwise, a conservative timeout which holds for all distributions is computed based on the one-sided inequality of probability theory.

[1]Available in http://www.navigators.di.fc.ul.pt/software/Adaptare.jar

IV. Adapting for dependability

A. Adaptare-FD *vs. Chen's failure detector*

To implement a dependable failure detector, a fundamental issue is to limit its number of mistakes. This is achieved with both Chen's FD and *Adaptare-FD* by defining a lower bound on the mistake recurrence time (T_{MR}^{L}). However, this single (safety) condition is not sufficient to specify a useful failure detector – a trivial implementation would just produce an output every T_{MR}^{L} time units. At least one additional liveness condition is necessary.

Chen et al. propose to use upper bounds on the detection time (T_{D}^{U}) and on the average mistake duration (T_{M}^{U}). Instead, we propose to use the lower bound on the coverage of the assumed transmission delays (C^{L}). Either way, an upper bound for the period of heartbeats can be derived and thus liveness is achieved. But there are some differences, which we discuss next.

Adaptare-FD requires the specification of the coverage, which makes it a good approach when the designer knows *how dependable* it wants the FD to be. For instance, if a high coverage, close to one, is specified (e.g. $C^{L} = 0.9999$), this implies that the FD will use sufficiently high timeouts to avoid false positives with the specified probability. Additionally, the interrogation period can be (and will be) very small, improving detection speed and leading to a very accurate FD. On the other hand, a coverage close to zero will lead to small timeout values and hence more mistakes. In this case, to ensure a given T_{MR}^{L} the interrogation period will tend to approach T_{MR}^{L}, which means slow failure detection and poor overall accuracy. In functional terms, the timeout will be dynamically adjusted to always be the lowest possible, given the environment conditions and the required coverage. On the negative side, the designer must be aware that increasing the required coverage has a cost: more resources will be used as the interrogation period will decrease, which may render the approach infeasible.

In contrast, Chen's approach requires the designer to specify upper bounds on T_D and T_M, which is an alternative way of achieving a dependable FD, being particularly suitable for applications that rely on known bounds for the detection time. On the positive side, since Chen's FD configures $\delta = T_{D}^{U} - \eta$ [3], as T_{D}^{U} increases, the timeouts can be larger, reducing the possibility of mistakes. Moreover, this is achieved without significant impact on the interrogation period, which does not need to be reduced as with *Adaptare-FD*. On the negative side, the limitation is that requirements may become unfeasible during the execution, if the expected value of the transmission delay becomes larger than T_{D}^{U}. In addition, Chen's approach makes no effort to select optimal timeouts and hence optimize T_D.

B. Adaptare-FD *vs. other timeout-based failure detectors*

Except for Chen's failure detector discussed in the previous section, all timeout-based failure detectors considered in this paper combine some estimator with a safety margin in order to compute timeouts.

Comparing to them, a striking advantage of using *Adaptare-FD* is the possibility of specifying the desired coverage as a dependability requirement, which serves to automatically and dynamically configure the failure detector. In contrast, with other solutions the required coverage can only be achieved if some underlying parameters are carefully adjusted. This can only be done after verifying the results, following a typical trial and error approach that tries to optimize the failure detector parameters for that specific environment. Whenever the network or the environment changes, it will probably be necessary to perform another fine tuning process.

V. Evaluation

In this section we describe the performed experiments and discuss the evaluation results in terms of achieved QoS, comparing *Adaptare-FD* to other timeout-based adaptive failure detectors. However, Chen's FD could not be included in the evaluation since the collected traces were obtained using fixed interrogation periods, while this failure detector also adapts the interrogation period. To ensure a fair comparison, a different experimental setup would have to be considered, which is planned as future work to complement the analytical discussion presented in Section IV-A.

A. Data collection

We compared different adaptive FDs using the same set of collected RTT traces, ensuring an impartial analysis. These measurements were performed in the PlanetLab environment, which is a platform widely used by the research community as a test bed for distributed applications [13].

We selected four geographically distributed nodes from PlanetLab (Brazil, USA, Japan and Australia), in order to simulate a distributed failure detection service operating in a WAN. We are not considering LAN scenarios in this work, because the stability of LANs does not justify the use of adaptive failure detectors.

RTTs between each pair of nodes were measured during 24 consecutive hours in 3 different days, building up 36 trace files (each of the 4 nodes monitors the other 3 nodes). RTTs were measured using the ping application, and the interrogation period was set to one second, according to the specified QoS (see ahead), leading to traces with approximately 86000 samples each, representing failure-free runs.

Table I presents the RTT statistics. It is noticeable that the links connecting to Australia exhibited large instability, thus creating heterogeneous scenarios that were useful in our evaluation.

B. Failure detectors configuration

As presented in Section IV-B, most of the adaptive failure detectors proposed in the literature are based on timeouts

Table I
LINKS RTT STATISTICS (MS)

Link	AVG	SD	MIN	MAX
USA-BR	173.93	20.72	172.00	2153.00
USA-JP	203.82	19.39	198.67	2134.17
BR-JP	352.97	2.31	330.50	540.33
JP-AU	1332.51	1565.77	293.16	7979.83
USA-AU	1333.05	1576.47	252.67	7944.33
BR-AU	1408.97	1568.84	342.33	8095.50

computed using a simple estimator, plus some safety margin. By combining the WINMEAN ($n = 30$) and LPF ($\beta = 0.125$) predictors with three different levels of Cib and Jacobson's safety margins (high, med and low), we defined 12 different timeout-based failure detectors. Based on the configuration proposed in [9], we used the following safety margins: 3.31, 2.586, and 1.648 for Cib's γ parameter; and 4, 2, and 1 for Jac's ϕ parameter.

Adaptare-FD was configured with the following parameters and QoS requirements: history size $h = 30$, $T_{MR}^{L} = 1000\ sec$, and $C = 0.999$. These values imply an interrogation period $\eta = 1\ sec$. The history size defines how many past measurements are analyzed to produce a new timeout. In a previous evaluation presented in [12], we verified that a value around $h = 30$ provides the best results.

C. Evaluation results

This section analyzes the QoS results of the evaluated FDs in terms of average timeout ($timeout$), average mistake recurrence time (T_{MR}), average mistake duration (T_M), total mistake duration (TT_M), and average coverage (C).

Analyzing $timeout$ **results.** In timeout-based failure detectors, there is a trade-off involved in the timeout configuration: higher timeouts guarantee good accuracy, at the cost of increased detection time; lower timeouts improve the detection time, but increase the occurrence of mistakes. The challenge here is to reach an "ideal" value, which provides the best balance between accuracy and detection time. Figure 4 shows three timeout values for each implemented failure detector: the average timeout for the stable links (USA-BR, USA-JP, and BR-JP), the average timeout for all links, and the average timeout for unstable links (JP-AU, USA-AU, and BR-AU), in this order.

Observing these results, it is possible to classify the failure detectors in three groups, as shown in Figure 4. The first group comprises two failure detectors that had timeouts much higher (at least one order of magnitude) than the others; because their timeouts are overestimated, we refer to this group as the *conservative* one. The second group, referred as the *aggressive* group, includes the six failure detectors which presented the lowest timeouts, lower than 1.5 seconds in average. The last five failure detectors constitute the so-called *balanced* group, and presented average timeouts between 1.5 and 5 seconds.

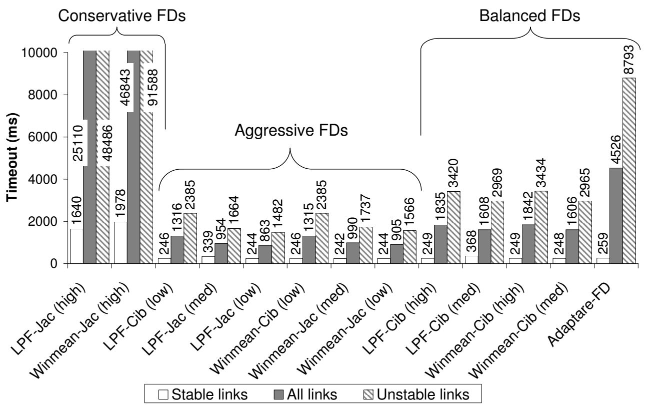

Figure 4. Average timeout

For the pull-style failure detector algorithm considered in this work, the detection time is bounded by the sum of the timeout δ and the interrogation period η. The average timeout values produced by the conservative group are much higher than the interrogation period, which becomes negligible in this case. Thus, the detection time is completely dictated by the timeout and is obviously worse than in the other groups. Note that the transmission of several heartbeats within one timeout interval creates some form of redundancy – any newer heartbeat can be used to reset the timeout. Therefore, if the timeout is larger than the period, the FD will do less mistakes. Since this is not explicitly considered in the configuration process of *Adaptare-FD*, the resulting behavior is more conservative than it could be.

Figures 5, 6, 7, and 8 provide comparative results for each metric, considering the best failure detector in each group (conservative, aggressive, balanced), plus *Adaptare-FD*.

Analyzing T_{MR}**,** T_M**, and** TT_M **results.** Reaching good values for both T_{MR} and T_M is a challenging task. The evaluation performed by Nunes and Jansch-Pôrto in [8] concluded that "no combination prediction/margin could ensure the best results on the accuracy evaluation because none of them ensures simultaneously the best values for T_M and T_{MR}". On a similar evaluation, Falai and Bondavalli [9] stated that "desirable higher values for T_{MR} are obtained only by paying higher T_M. Actually it is impossible to obtain at the same time the best values for both accuracy metrics".

Figures 5 and 6 show this trade-off in the failure detectors based on predictors and safety margins: the best T_{MR} values were obtained by the conservative FD, which presented the worst T_M. Its T_{MR} results are at least one order of magnitude higher than the others, due to its overestimated timeouts: mistakes only occur in the rare cases in which a change in the environment significantly increases the network delays. Consequently, this failure detector makes few mistakes, achieving excellent T_{MR} values.

Because the mistakes of the conservative group correspond to extreme situations of high network latencies, they

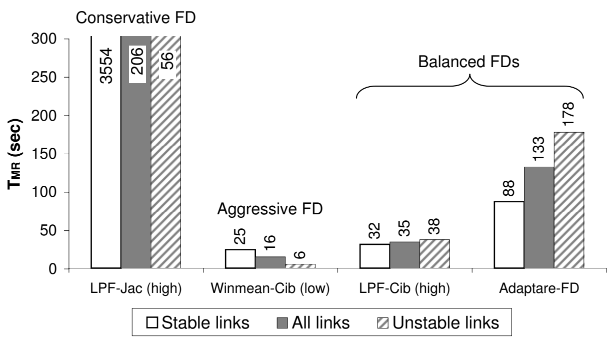

Figure 5. Average mistake recurrence time

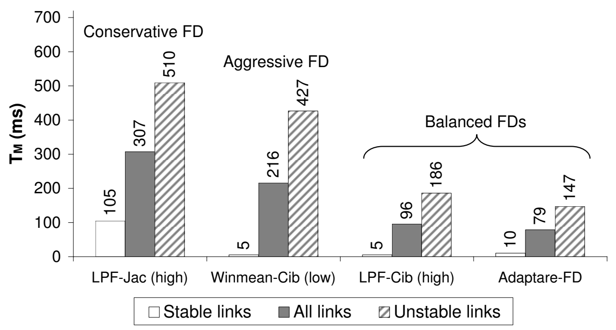

Figure 6. Average mistake duration

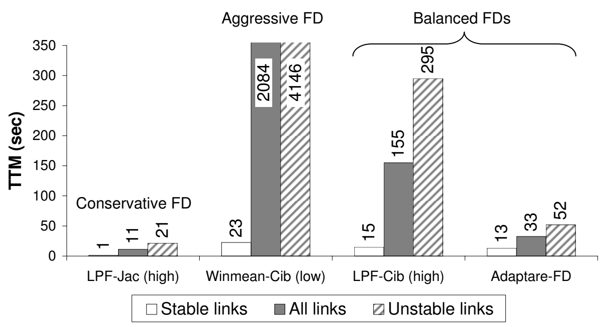

Figure 7. Total mistake duration

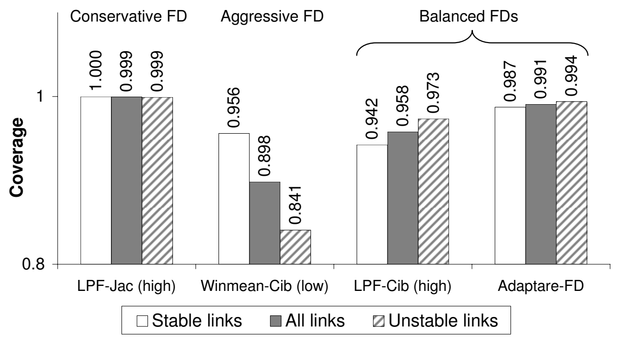

Figure 8. Average coverage

have longer durations, which explains the poor performance of this group on the T_M metric. The FD of the aggressive group also has higher T_M values than the balanced FD: this is because it tends to use too small timeouts, leading to premature mistakes that take longer to correct, in average.

Figure 7 shows the TT_M values, which are the sum of mistake durations. As expected, the lowest values were obtained by the conservative FD, due to its small T_{MR}.

Adaptare-FD exhibited the second best T_{MR} and TT_M values, being the best one if we disregard the conservative failure detector due to its high timeouts. It also presented the best T_M in average. The importance of these results is that they suggest that it is possible to achieve good values for both T_{MR} and T_M metrics, as obtained using *Adaptare-FD*.

Analyzing C results. Results for the coverage are presented in Figure 8. The best C values were obtained by the conservative failure detector. This was expected, because its overestimated timeouts decrease the possibility of receiving *"I'm alive"* messages after timeout expiration. Analogously, the worst results were obtained by the aggressive failure detector, due to too small timeouts.

Adaptare-FD achieved the best C of the balanced group, and the second best among all tested failure detectors. However, it still did not secure the required coverage, and consequently the specified T_{MR}^L. This can be explained by two factors. First, some mistakes occur due to environment dynamics, which may become higher than assumed by the probabilistic mechanisms implemented in the *Adaptare* framework. Arbitrary behaviors may occur occasionally, and it is impossible to predict them. The only possibility to avoid errors in these cases is by considering ad-hoc safety margins. Second, in our approach, omissions are equivalent to late messages, which means that for each lost message *Adaptare-FD* (and all the evaluated FDs) will make a mistake.

In cases of arbitrary behaviors (e.g. message bursts due to pointwise network perturbations), *Adaptare-FD* reaction is very fast: in our experiments we observed that it did not make two consecutive mistakes.

Interestingly, *Adaptare-FD* coverage in stable links is even lower than in unstable links, because in stable environments the probabilistic mechanisms become more susceptible: even small perturbations may have a negative impact.

Performance limits. As mentioned in Section IV-A, *Adaptare-FD* performance is limited, in particular, by the smallest implementable interrogation period. In practice, the limit is imposed by the *Adaptare* framework for the time it takes to recompute the timeout on every new input sample. Since the evaluation presented in [12] indicates that in a standard Pentium 4 PC the framework takes 1ms in average to perform this job (for a set of 30 samples), the approach seems practically realizable.

D. Securing coverage

As discussed in Section IV-B, a significant advantage of *Adaptare-FD* is the possibility of specifying the expected coverage as a dependability requirement, which is automatically secured as well as possible, without the need of any adjustment of algorithm parameters.

To emphasize the importance of this capability, we tried to configure the parameters of the *LPF-Cib (high)* failure detector (second best in our overall evaluation) to achieve $C = 0.999$ in all links. The adjustment was performed by incrementing the γ parameter of the *Cib* safety margin, trying to reach the expected coverage. We tried five different values (5, 10, 20, 30, and 50), and even though we only succeeded to achieve the requirement of $C = 0.999$ in the unstable links. Clearly, and in contrast with *Adaptare-FD*, this FD requires a fine tuning of γ for each environment.

VI. Conclusion

This paper proposed *Adaptare-FD*, a dependability-oriented adaptive failure detector. To the best of our knowledge this is the first FD that uses a coverage parameter in the configuration process.

We contrasted *Adaptare-FD* with the well-know approach of Chen et. al., emphasizing the expected impacts of the different configuration approaches on the QoS metrics. While Chen's approach is more suitable for applications that require a strict upper bound on the detection time, *Adaptare-FD* is more flexible and optimizes the detection speed with respect to environment conditions. Despite the clear theoretical advantages and impact of using coverage as a specification parameter, future work will be devoted to explore the practical extent of these advantages.

Our evaluation was devoted to compare *Adaptare-FD* with different adaptive timeout-based failure detectors based on the same pull-style FD algorithm. The results showed the advantages of *Adaptare-FD*: while aggressive approaches achieve better detection speed, and conservative ones achieve large mistake recurrence times and lower mistake rates, the best trade-off is obtained with the timeouts produced by our solution. In fact, when comparing with the remaining balanced FD approaches, a relevant result was that *Adaptare-FD* achieved the highest mistake recurrence time and coverage and, at the same time, one of the smallest mistake durations. The improvement of this trade-off is a clear advance over the state-of-the-art.

A significant feature of *Adaptare-FD* is adaptability to environment conditions. Other approaches can be optimized for given environments, but if the probabilistic behavior changes they may fail to deliver the required performance. Therefore, *Adaptare-FD* is comparably better for the implementation of distributed applications with dependability requirements executing in heterogeneous environments.

Acknowledgment

This work was partially supported by FCT through the Multiannual Funding and the CMU-Portugal Programs.

References

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

[2] N. Sergent, X. Défago, and A. Schiper, "Impact of a failure detection mechanism on the performance of consensus," in *In 2001 Pacific Rim International Symposium on Dependable Computing*, 2001, pp. 137–145.

[3] W. Chen, S. Toueg, and M. K. Aguilera, "On the quality of service of failure detectors," *IEEE Trans. on Computers*, vol. 51, no. 1, pp. 13–32, 2002.

[4] C. Fetzer, M. Raynal, and F. Tronel, "An adaptive failure detection protocol," in *In 2001 Pacific Rim International Symposium on Dependable Computing*, 2001, pp. 146–153.

[5] N. Hayashibara, X. Défago, R. Yared, and T. Katayama, "The F accrual failure detector," in *IEEE Symposium on Reliable Distributed Systems*, 2004, pp. 66–78.

[6] B. Satzger, A. Pietzowski, W. Trumler, and T. Ungerer, "A new adaptive accrual failure detector for dependable distributed systems," in *In 2007 ACM Symposium on Applied Computing*, Seoul, Korea, 2007, pp. 551–555.

[7] M. Bertier, O. Marin, and P. Sens, "Implementation and performance evaluation of an adaptable failure detector," in *In 2002 International Conference on Dependable Systems and Networks*, Washington, DC, USA, 2002, pp. 354–363.

[8] R. C. Nunes and I. Jansch-Porto, "Qos of timeout-based self-tuned failure detectors: The effects of the communication delay predictor and the safety margin," in *In 2004 International Conference on Dependable Systems and Networks*, Washington, DC, USA, 2004, p. 753.

[9] L. Falai and A. Bondavalli, "Experimental evaluation of the qos of failure detectors on wide area network," in *In 2005 International Conference on Dependable Systems and Networks*, Washington, DC, USA, 2005, pp. 624–633.

[10] I. Sotoma and E. R. M. Madeira, "Adaptation - algorithms to adaptive fault monitoring and their implementation on corba," in *DOA*, 2001, pp. 219–228.

[11] A. Casimiro, P. Lollini, M. Dixit, A. Bondavalli, and P. Verissimo, "A framework for dependable qos adaptation in probabilistic environments," in *23rd ACM Symposium on Applied Computing, Dependable and Adaptive Distributed Systems Track*, Ceara, Brazil, 2008, pp. 2192–2196.

[12] M. Dixit, A. Casimiro, P. Lollini, A. Bondavalli, and P. Verissimo, "A probabilistic framework for automatic and dependable adaptation in dynamic environments," Dep. of Informatics, Univ. of Lisboa, Tech Report TR-09-19, 2009.

[13] "PlanetLab: Version 3.0," PlanetLab Consortium, Tech. Rep. PDN–04–023, October 2004.

[14] V. Jacobson, "Congestion avoidance and control," *Innovations in Internetworking*, pp. 273–288, 1988.