2W-FD: A Failure Detector Algorithm with QoS

Alejandro Z. Tomsic, Pierre Sens, Joao Coelho Garcia, Luciana Arantes, Julien Sopena

2W-FD: A Failure Detector Algorithm with QoS

Alejandro Z. Tomsic*, Pierre Sens*, João Garcia†, Luciana Arantes* and Julien Sopena*
*Sorbonne Universités, UPMC Univ Paris 06,
CNRS, Inria, LIP6
F-75005, Paris, France
Email: firstname.lastname@lip6.fr
†Instituto Superior Técnico, Universidade de Lisboa,
Email: joao.c.garcia@tecnico.ulisboa.pt

***Abstract*—Failure detection plays a central role in the engineering of distributed systems. Furthermore, many applications have timing constraints and require failure detectors that provide quality of service (QoS) with some quantitative timeliness guarantees. Therefore, they need failure detectors that are fast and accurate.**

We introduce the Two Windows Failure Detector (2W-FD), an algorithm that provides QoS and is able to react to sudden changes in network conditions, a property that currently existing algorithms do not satisfy.

We ran tests on real traces and compared the 2W-FD to state-of-the-art algorithms. Our results show that our algorithm presents the best performance in terms of speed and accuracy in unstable scenarios.

***Keywords*-Failure Detectors, Quality of Service, Fault Tolerance, Distributed Algorithms, Reliability, Quiescence.**

I. Introduction

Distributed systems should provide reliable and continuous services despite the failures of some of their components. As a consequence, failure detection plays a central role in the engineering of such systems. A failure detector (FD) provides suspicion information on which processes have crashed. FDs are used in a wide variety of settings, such as network communication and group membership protocols, computer cluster management and distributed storage systems.

Many applications have timing constraints. They require a FD that provides quality of service (QoS) with quantitative timeliness guarantees as the QoS of the FD greatly influences the QoS that upper layers provide. There is an inherent tradeoff between *conservative* failure detection, i.e., reducing the risk of wrongly suspecting a correct process, and *aggressive* failure detection, i.e., quickly detecting the occurrence of a real crash. There is a continuum range of valid choices between these two extremes and the correct choice depends on the particular needs of each application in terms of QoS.

Existing FDs [4], [1], [2], [6], [7], [11], [12], [10] keep a sliding window that contains information about received messages to make an estimate of the state (trusted or suspected of having failed) of a monitored process. These FDs assume that the network behaviour follows some stable or slowly changing probability distribution in terms of message delay and message loss, but are not designed to adapt their behaviour to sudden changes in network conditions.

In this work, we present the Two Windows Failure Detector (2W-FD), a FD that adapts to sudden changes in unstable network scenarios. This situation is likely to occur in WAN scenarios, where packets travel across routes with varying number of hops and are subject to latency jitter, as well as present in LAN scenarios, possibly due to contention in hardware switches or end systems, e.g. when a large amount of data is suddenly sent to a machine. Virtualisation may exacerbate the latter problem if applications with different workloads are co-located on a shared machine. The 2W-FD uses two sliding windows of past received messages; a small one that stores very recent history (information about the past few messages), and a bigger one that stores a larger recent history. The small window allows the 2W-FD to react rapidly to abrupt condition changes in network conditions. The long-term window allows the 2W-FD to make better estimations on stable periods or periods where conditions change gradually.

We evaluated and compared the QoS of our algorithm to the best-known existing ones [4], [1], [7], [12] in terms of mistake rate and query accuracy probability. These algorithms and concepts will be introduced in sections II and III, respectively. We ran experiments over real traces taken from an unstable WAN and a stable LAN scenario. Our results show that the 2W-FD outperforms the others in the unstable scenario and is similar to the best performing algorithms in the stable scenario.

The rest of this paper is organised as follows. Section II provides an introduction to the different technical areas related to our work, Section III discusses related work, Section IV introduces the 2W-FD algorithm. Section V evaluates it, and compares it to other FD algorithms. Finally, in Section VI we present conclusions on our work.

II. Background

We assume an asynchronous underlying system, where it is impossible to precisely determine whether a remote process has failed or has just been very slow [3]. Therefore, we consider *unreliable* failure detectors [3] that can only *suspect* a process failure. Unreliable FDs may be inaccurate, i.e., suspect a process that has not failed, and incomplete, i.e., overlook suspecting a failed process.

A. Quality of Service (QoS) Specification for Failure Detectors

In this section, we present our considered model and notions on QoS for failure detectors.

1) Model for QoS Specification: We consider a system of two processes, p and q. The failure detector at q monitors p. Real time is continuous and ranges from 0 to ∞. At any given time t, the output of the failure detector at q can be either S, suspect, or T, trust. Whenever the output of the failure detector in q changes, we say that a *transition* occurs: an *S-transition* occurs when the output of q changes from T to S; and a T-transition happens when the output of the failure detector at q changes from S to T. Only a finite number of transitions can take place during a finite period of time.

2) QoS Metrics for Failure Detectors: QoS metrics measure how *fast* and *accurate* a failure detector is. These metrics are applicable to all failure detectors, regardless of how they are implemented. The most relevant metrics, as introduced by Chen et al. [4], are described in this section.

- ***Detection Time*** ***(T_D)*** is the time that elapses from the moment that process p crashes until the failure detector at q detects the failure and starts suspecting p for ever (see Figure 1). More precisely, T_D measures the time that elapses from the moment that the crash of p occurs to the moment when the final S-transition occurs (at q) and there are no further transitions (see Figure 1).

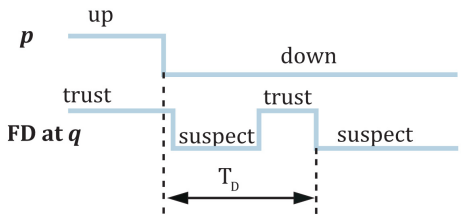


Figure 1: Detection Time T_D

- ***Average Mistake Rate*** ***(R_M)*** measures the rate at which a failure detector makes mistakes, i.e., it is the number of times q suspects a correct process p per unit of time (see figure 2). This metric is important for long-lived applications where a mistake results in a costly interrupt, such as group membership applications and cluster management protocols.
- ***Average Mistake Duration*** ***(T_M)*** measures the time a failure detector takes, on average, to correct a mistake (see figure 2). This metric is useful for applications that operate in a degraded mode when a process is incorrectly suspected.
- ***Query Accuracy Probability*** ***(P_A)*** is the probability that the failure detector's output is correct when queried at a random time. This metric is useful for applications that interact with the failure detector by querying it. It is easy to see that this metric can be derived from the previous two.

Note that the first metric is related to a failure detector's $speed$, while the remaining relate to its $accuracy$.

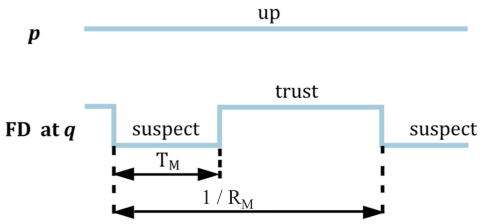


Figure 2: Mistake Duration T_M and Mistake Rate R_M

III. Related Work

In this section, we present the most relevant related failure detection algorithms.

A. Chen Failure Detector

Chen et al. developed a failure detector that provides QoS [4]. This algorithm estimates expected arrival times (EAs) of heartbeats, which are then used to compute *freshness points* (τs). A freshness point determines the moment until when the failure detector will wait for a message from the monitored process before starting to suspect that it has crashed. The next freshness point is given by the following equation:

$$\tau_{l+1} = EA_{l+1} + \alpha \tag{1}$$

where α is a constant safety margin chosen by the user based on her needs on detection time T_D and l is the largest sequence number of heartbeats received so far.

To compute *expected arrival times EA*s, Chen FD stores in a sliding window information regarding the n previous messages (for some n). Let s_1,s_2,...,s_n be the sequence number of those messages and A_1,A_2,...,A_n their receipt times at q. Then, EA_{l+1} is estimated by:

$$EA_{l+1} \approx \frac{1}{n}\left(\sum_{i=1}^{n} A_i - \eta.s_i\right) + (l+1)\eta \tag{2}$$

where η is the heartbeat sending interval, chosen by the user. This equation first normalises each A_i by shifting it

backwards ηs_i time units. Then, an average of the A'_is is computed and, finally, this computed average is shifted forward by $(l+1)\eta$.

B. Bertier Failure Detector

Bertier et al. introduced a failure detector principally intended for LAN environments [1]. Their algorithm uses the same mechanism as Chen for estimating expected arrival times *EA*s (see Equation 2), but a dynamic way of computing freshness points based on Jacobson's estimation [9], which is used in the TCP protocol to estimate the delay after which a transceiver retransmits a message. As in Chen FD, the arrival times of the *n* previous messages are kept in order to compute EAs. Jacobson's estimation supposes that the behaviour of the system is not constant, and it is used in this algorithm to adapt the safety margin each time a heartbeat is received. EA_{l+1} is calculated using Equation 2. With this two values computed, the next freshness point τ_{l+1} is computed exactly as in Equation 1 (by replacing α with the dynamic margin).

C. The φ Accrual Failure Detector

In the φ FD [7], the suspicion level is given by a value called φ, expressed on a scale that is dynamically adjusted to reflect current network conditions [5]. Let T_{last} denote the time when the most recent heartbeat was received, T_{now} the current time, and $P_{later}(t)$ the probability of a heartbeat arriving more than t time units after the previously received one. Then, the value of φ at current time is calculated as follows:

$$\varphi(T_{now}) = -log_{10}(P_{later}(T_{now} - T_{last})) \tag{3}$$

In this context, φ has the following meaning. Given a threshold Φ, if the failure detector suspects *p* when $\varphi \geq \Phi$, then the probability that the φ failure detector makes a mistake is about $\frac{1}{10^{\Phi}}$.

The estimation of φ is done as follows. When heartbeats arrive, their arrival times are stored in a sampling window (as in Chen and Bertier FD algorithms). These past samples are used to determine the distribution of inter-arrival times. Finally, the distribution is used to compute the current value of φ. The estimation of the distribution of inter-arrival times assumes that they follow a *normal distribution*. The parameters of the distribution are estimated by determining the mean μ and the variance σ^2 of the samples. Then, the probability $P_{later}(t)$ that a given heartbeat will arrive more than t time units later than the previous heartbeat is given by the following equation:

$$P_{later}(t) = \frac{1}{\sigma\sqrt{2\pi}} \int_t^{\infty} \mathrm{e}^{-\frac{(x-\mu)^2}{2\sigma^2}} \mathrm{d}x \tag{4}$$

$$= 1 - F(t) \tag{5}$$

where $F(t)$ is the cumulative distribution function of a normal distribution with mean μ and variance σ^2. Finally, the value of φ at time T_{now} is computed by applying Equation 3. This process is repeated by *q* for every new heartbeat received.

D. Exponential Distribution Failure Detector (ED FD)

This FD [12] is based on the same principle as the φ accrual failure detector. The difference lies in the fact that the distribution considered for message delays by the ED FD is the exponential one. In the ED FD, the suspicion level is given by a value called e_d, which is calculated as follows:

$$e_d = F(T_{now} - T_{last}) \tag{6}$$

$$F(t) = 1 - \mathrm{e}^{-\frac{1}{\mu}t} \tag{7}$$

where T_{now}, T_{last} and μ have the same meaning as in the φ accrual failure detector (section III-C). For this FD, the threshold is called E_d.

E. Bursty Traffic

In some scenarios, the probabilistic behaviour of the network (message delay and message loss) can vary. The algorithms presented in this section adapt their behaviour to gradually changing network conditions (Chen failure detector can be made adaptive by recomputing the heartbeat interval η and timeout α every certain period of time [7]).

There are times when network conditions change very rapidly due to bursty traffic. Such variations are common in WAN networks, where message delays and message losses are more likely to occur. The presented algorithms that compute expected arrival times adapt well when the following conditions hold [4]:

1) the *occurrences* of bursts are independent of each other and follow some slowly changing probabilistic distribution.
2) the *duration* of each burst is short (smaller than the heartbeat interval η).

In this case, heartbeats behave according to some new slowly changing probability distribution that takes into account the occurrence of bursts.

When 1) or 2) do not hold, some mechanism to estimate the current behaviour of the network and adapt to it is necessary.

Summary

In this section we have introduced related work in the area of failure detection for distributed systems.

The algorithms presented throughout this section are able to provide QoS guarantees to applications when the message delay and message loss of the network behave according to some probability distribution. Nevertheless, in the presence of bursts of lost messages, these algorithms do not provide a mechanism to quickly adapt to such changes.

The estimation of expected arrival times and freshness points that these probabilistic approaches use are dependent on the past history of observed arrival times of a large amount of previous messages. Keeping such a large history prevents these algorithms from quickly reacting to sudden changes. In the next section, we further explain this problem and introduce our algorithm, which addresses unstable network behaviour.

IV. 2W-FD

In this section we introduce the *Two Windows Failure Detector* (2W-FD) and the idea behind it.

A. Rationale

In order to adapt to bursty-traffic conditions, we propose the use of two components for the estimation of expected arrival times EAs and freshness points τ_i. Namely:

1) a ***short-term*** component that considers only the most recent messages, which is used to quickly react to sudden changes in network conditions, possibly due to bursty traffic. Furthermore, we expect the algorithm to benefit from the use of this component when the network changes from a stable to an unstable state, i.e., the delay of the last few messages has increased with respect to the delay of the previous ones.
2) a ***long-term*** component that considers a bigger amount of recently received messages that is not sensitive to momentary fluctuations. The estimation calculated by using the information stored by this component is useful for periods of stability, and periods when the network changes from an unstable to a stable state.

Whenever a message m_l is received by *q*, both components are used to estimate the freshness point τ_{l+1} for the next expected message m_{l+1}, as we will explain in the following section.

B. The Algorithm

Algorithm 1 presents the pseudo-code for the Two Windows Failure Detection (2W-FD) algorithm for stable periods of behaviour (after both windows have been filled). 2W-FD keeps two sliding windows, W_1 and W_2 (of sizes n_1 and n_2 respectively), of recently received heartbeat arrival times. Whenever a message m_l sent by *p* is received by *q*, *q* adds the arrival time of message m_l to W_1 and to W_2 (lines 15 and 16). The following step is to compute, using the values stored in W_1 and to W_2, the expected arrival times $EA^{n_1}_{l+1}$ and $EA^{n_2}_{l+1}$ (line 20). The key of the algorithm is that, from the EAs computed, it uses the maximum of these estimations for the computation of the next freshness point τ_{l+1}:

$$\tau_{l+1} = max\left(EA^{n_1}_{l+1}, EA^{n_2}_{l+1}\right) + \alpha \tag{8}$$

where α is a constant safety margin. Finally, if message m_{l+1} is not received before time $t = \tau_{l+1}$, *q* starts suspecting *p* (line 23).

The formula used by the 2W-FD for computing expected arrival times (EAs) for each value of n, is the following:

$$EA_{l+1} \approx \frac{1}{n}\left(\sum_{i=1}^{n} A_i - \epsilon.s_i\right) + (l+1)\epsilon \tag{9}$$

This equation presents a difference from equation 2; the parameter ϵ replaces η, the parameter used by Chen and Bertier FD. η is a fixed input parameter that indicates the inter-sending arrival times at which the machine *p* is sending heartbeats, whereas ϵ is the average of inter-arrival times, as observed by *q*, computed by using the information stored in the larger window. The reason behind proposing ϵ is that the former approach poses a problem. When using η, it is highly probable that process *p* might not be sending messages at exactly that rate due to the fact that it is not possible to predict that a machine is able to schedule the sending of messages on an exact base. Even when that situation is not present, a clock skew between the clocks at p and q would mean that in reality η would have a different value for each process. As time passes, the value used by Chen and Bertier FD introduces an error that increases as the total number of samples observed does. Such a behaviour explains why Chen FD performs better with small window sizes, fact that has been previously observed but not explained [11]. We will provide empirical evidence supporting this statement in section V.

C. Consequences of Using Two Windows

Intuitively, our algorithm is expected to work better than the rest mainly in the presence of bursty traffic and rapid changes in network conditions, because of the reasons explained in section IV-B. Given two window sizes n_1 and n_2, our algorithm should be able to make fewer mistakes than using a single window when using any of n_1 or n_2 as for each analysed sample the 2W-FD computes the maximum of the expected arrival times that would be computed for each window size. The computation of the maximum implies that the 2W-FD will only make the mistakes that a single-windowed FD would make if it used both window sizes n_1 and n_2. This means:

$$Mistakes(n_1, n_2) = Mistakes(n_1) \cap Mistakes(n_2) \tag{10}$$

It is easy to see that, by picking the maximum of the estimations of expected arrival times of messages, the failure detector at *q* becomes more tolerant, i.e., more *conservative*. This occurs because for each heartbeat, the 2W-FD will wait for the maximum of the times estimated by each window before starting to suspect a crash, a fact that directly reduces the probability of making mistakes. The drawback of this approach is that it increases the detection time T_D of the algorithm. At first sight, this would suggest that the algorithm would not be able to work in *aggressive* ranges

Algorithm 1 Two Windows Failure Detector Algorithm

Process p: ▷ Using p's local clock

1: **for all** $i \geq 1$ **do**
2: at time $i \cdot \eta$ send heartbeat m_i to q
3: **end for**

Process q: ▷ Using q's local clock

4: ***Initialization***:
5: $\tau_0 = 0$;
6: l = -1; ▷ keeps the largest sequence number of messages seen
7: $\boldsymbol{W_1} = \{\}$; ▷ **contains the last n_1 message arrival dates**
8: $\boldsymbol{W_2} = \{\}$; ▷ **contains the last n_2 message arrival dates**
9: $EA_0 = EA^{n_1}_{l+1} = EA^{n_2}_{l+1} = 0$;
10: ***upon*** τ_{l+1} = the current time:
11: $output \leftarrow S$; ▷ suspect p
12: ***upon*** receive message m_j at time t:
13: **if** $j>l$ **then** ▷ Received a message with a higher sequence number
14: $l \leftarrow j$;
15: $\boldsymbol{W_1} \leftarrow \boldsymbol{W_1} \cup \{\boldsymbol{t_j}\}$
16: $\boldsymbol{W_2} \leftarrow \boldsymbol{W_2} \cup \{\boldsymbol{t_j}\}$
17: $\boldsymbol{W_1} \leftarrow \boldsymbol{W_1} \setminus \{\boldsymbol{t_{j-n_1}}\}$
18: $\boldsymbol{W_2} \leftarrow \boldsymbol{W_2} \setminus \{\boldsymbol{t_{j-n_2}}\}$
19: **Compute $\boldsymbol{EA_1[n_1]_{l+1}}$ and $\boldsymbol{EA_2[n_2]_{l+1}}$** ▷ **using Equation 9**
20: $\boldsymbol{EA_{l+1} \leftarrow max(EA^{n_1}_{l+1}, EA^{n_2}_{l+1})}$
21: $\tau_{l+1} \leftarrow EA_{l+1} + \alpha$ ▷ set the next freshness point using Equation 1
22: **if** $t<\tau_{l+1}$ **then**
23: $output \leftarrow T$; ▷ trust p
24: **end if**
25: **end if**

of detection, where the required T_D is very small. In the next section we will show that the trade-off between the increase in T_D and the gain in accuracy (P_A and R_M) is positive; our algorithm outperforms the others to which we have compared it to in scenarios where network conditions vary, particularly in the aggressive range.

V. Evaluation

In this section, we present the results of the experiments we conducted to study the performance of our algorithm. First, we evaluate the performance of the 2W-FD when using different window sizes and conclude about optimal configurations for the parameters n_1 and n_2. Later, we compare the performances of 2W-FD to the algorithms presented in section III.

A. Heartbeat Traces

All tests were performed on real traces. To generate them, a simple software was executed on two computers for an arbitrarily long period of time:

- *Computer 1*, periodically sending heartbeat messages to the other one,
- *Computer 2*, receiving heartbeats and logging their arrival information.

Whenever a heartbeat arrives to Computer 2, the heartbeat monitor logs its sample number and arrival time. The full logs are used later to replay the execution on each FD algorithm. Therefore, all failure detectors were compared under the same experimental conditions. Heartbeat messages were sent using the UDP/IP protocol. During the recording of traces, the average CPU load of both computers was nearly constant and below the full capacity of each computer.

Tests Scenarios: We used two different traces for our experiments. One taken from a WAN scenario, and the other one, from a LAN scenario.

WAN Scenario.: This traces were taken by Hashibara et al. for their evaluation of the φ FD [7]. They were also used by the authors of the ED FD [12], and for evaluation of other work on failure detectors [11], [10]. The traces are publicly available at [8].

The heartbeat sending computer was located in Switzerland, at the Swiss Federal Institute of Technology in Lausanne (EPFL). The monitoring computer, in Japan, at the Japan Advanced Institute of Science and Technology (JAIST). They communicated through a normal interconti-

nental Internet connection. The experiment lasted for a full week. Neither machine failed during the experiment. Heartbeats were sent at a frequency of one heartbeat every 100 ms. The measured sending rate was actually one heartbeat every 103.5 ms (standard deviation: 0.19 ms; min.: 101.7 ms; max.: 234.3 ms). Almost 6 million heartbeats were sent, from which about 0.4% were lost.

During the experiment, the round-trip time (RTT) was also measured. The average measured RTT was 283.3 ms with a standard deviation of 27.3 ms, a minimum of 270.2 ms, and a maximum of 717.8 ms.

It was observed that message losses tended to occur in bursts. The distribution of burst lengths, as well as more detailed information, can be found in [8].

There was a period where more messages were lost. According to the authors, such event was likely caused by an outbreak of the *W32/Netsky.T@mm* Internet worm, as dates coincided.

LAN Scenario.: The experiment used two identical computers connected through a single unshared 100 Mbps Ethernet hub. The heartbeat interval was set to 10 ms. More than 7 million samples were collected. The average received interval at the monitoring computer was around 12 ms. Not a single heartbeat was lost. The largest interval between the reception of two heartbeats was about 1.5 seconds. Nevertheless, the variance was very small. The average transmission delay was around 100 μs.

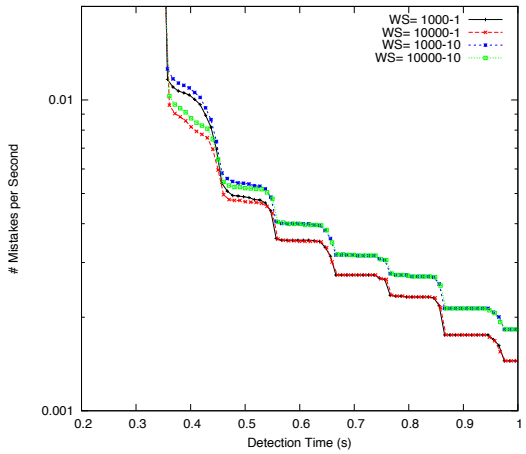
B. Experiments

All failure detectors considered in these experiments rely on window(s) of past samples to compute their estimations. As the behaviour of the failure detectors is stable only after the window is full, we do not include in our analysis data obtained before that moment.

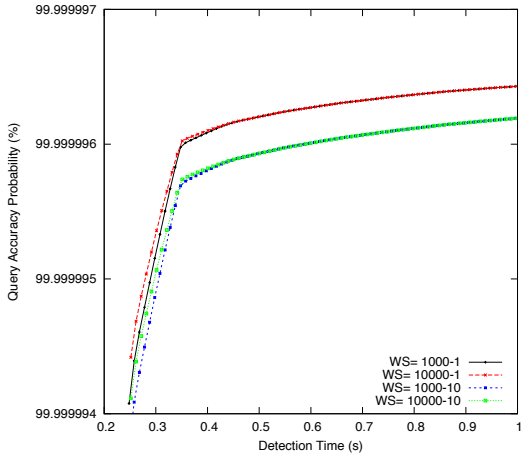
Detection Time T_D.: In all experiments, we have computed an *estimation* for the average detection time T_D as follows. Assuming that a crash would occur exactly after successfully sending a heartbeat (worst-case scenario), we measure the time elapsed until the failure detector would report a suspicion, for each analysed sample. In section V-B2, with the φ and ED failure detectors, we consider the algorithms' threshold values (Φ and E_d) and reverse the computation of φ and e_d to obtain the equivalent timeout Δ_{to}. We compute this equivalent timeout each time a new heartbeat is received and take the mean value Δ_{to}. We estimate the mean propagation time Δ_{tr} based on round-trip times. Then, for each sample, we compute the average (worst-case) detection time as follows.

$$D_T \approx \Delta_{tr} + \Delta_{to} \tag{11}$$

1) 2W-FD - Window Sizes: This experiment measures the effect of window sizes on the performance of our developed algorithm, 2W-FD. We varied the sizes of windows and



(a) R_M vs. T_D



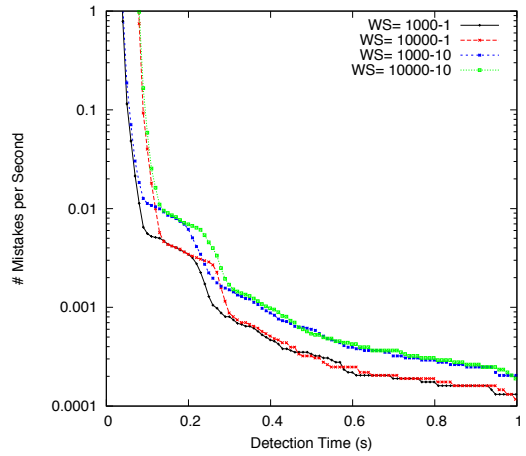
(b) P_A vs. T_D

Figure 3: 2WDF with different Window Sizes in a WAN scenario

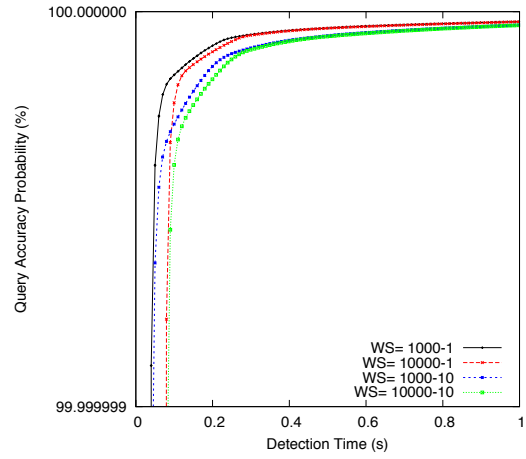
measured the accuracy obtained with our failure detector when run over the traces. In order not to overload the figures, we have only plotted the curves of the configurations that presented the best results. In the figures, the used notation is WS = n_1-n_2, where n_1 expresses the size of the long-term window and n_2, the size of the small one.

Results in a WAN.: Figure 3(a) shows the results on mistake rate R_M (number of mistakes per second) vs. detection time T_D in the unstable WAN scenario. R_M is represented on the vertical axes, expressed in logarithmic scale, and T_D in the horizontal axes. Figure 3(b) shows the results on query accuracy probability P_A vs. detection time T_D in the same scenario.

In terms of R_M, the 2W-FD presents the best performance when n_1=10,000-n_2=1. The curves with n_1=10,000 outperform the rest in the aggressive range, i.e., for T_D values smaller than 0.5 ms, when n_2=1, 10 . The configuration



(a) R_M vs. T_D



(b) P_A vs. T_D

Figure 4: 2WDF with different Window Sizes in a LAN scenario

n_1=1,000-n_2=1 presents a very good performance in the conservative range. Regarding P_A, the best results are obtained for the configurations n_1=10,000-n_2=1 and n_1=1,000-n_2=1.

Results in a LAN: Figures 4(a) and 4(b) show the results of the test over the traces taken from the stable LAN scenario. The results show that the configuration n_1=1,000-n_2=1 presents the best performance, both in terms of R_M and P_A, for all values of T_D.

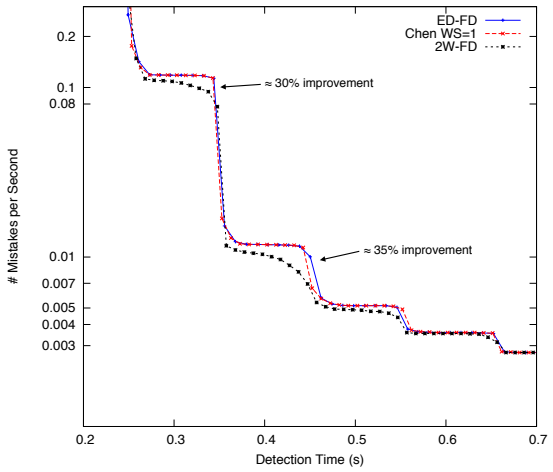
Experiment's Conclusions.: From the previous experiments we observe the following tendency. In terms of both R_M and P_A, our algorithm behaves better in unstable scenarios as the size of the big window increases and as the size of the small window decreases. It's also noticeable from the figures that curves for tests which share the same size for the short-term window tend to behave similarly. For the short-term window, the experiments suggest that the best size is one (1). Such observation ensures the principle of the short-term window, introduced in section IV-A, which is reactive to very recent behaviour.

2) Comparison to Other Algorithms: In this experiment, we compare the behaviour of the 2W-FD with four well known failure detectors. Namely, the ones introduced in section III. In this experiment we intend to show that the 2W-FD presents the best detection time to accuracy ratio in unstable network scenarios. Chen and 2W FD share a common tuning parameter, the safety margin α, which we use in our experiments to get the different values of detection time. The tuning parameter for the accrual failure detectors were the thresholds Φ and E_d. Unlike the other failure detectors, Bertier's has no tuning parameter. For this reason, its behaviour is plotted as a single point on the figures. The parameters of the algorithms were configured as follow: for Chen FD, the parameters are set the same as in [4], [7], [11], [12], [10]: $\alpha \in [0, 1000]$; For φ FD, the parameters are set the same as in [7], [11], [12], [10]: $\Phi \in [0.5, 16]$; $E_d \in [10^{-4}, 10]$ for ED FD, as in [12]. Window sizes were set to:

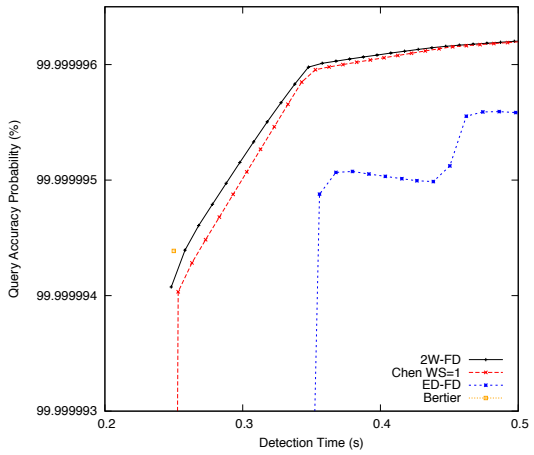
- 2W-FD: n_1=1,000 and n_2=1. These values were chosen as the algorithm presents the best tradeoff in terms of big window size and processing capacity required under such configuration.
- Chen FD: we plot two different configurations for this FD. First, n=1 because it was proven to work better as its window size decreases [11], as we also observed in experiments we performed but not present in this work, and n=1,000, as it is the commonly used value in related work experiments [7], [11], [12], [10]. This is the first work to include Chen FD with n=1 in its evaluation.
- φ and the ED failure detectors: n=1,000. These failure detectors benefit from using large window sizes [7], [12]. Furthermore, we have conducted experiments that show that for window sizes beyond 1,000 samples, the performance improvement of these algorithms is negligible (this fact was also observed by their authors [7], [12]) . Finally, this is the window size its authors used in the experiments described in their articles.
- Bertier FD: n=1,000, as that is the value their authors use in experiments presented in their article [1]. Furthermore, it is the common value used in related work [7], [11], [12], [10], and Bertier FD does not significantly vary its performance when varying its window size [11].

WAN results.: Figure 5(a) shows the results on mistake rate R_M vs. detection time T_D in the WAN scenario. R_M is represented on the vertical axes, expressed in logarithmic scale, and T_D in the horizontal axes. Figure 5(b) shows the results on query accuracy probability P_A. vs. detection time T_D in the same scenario.

The results indicate that all algorithms follow the same tendency. Our algorithm seems to outperform the others in

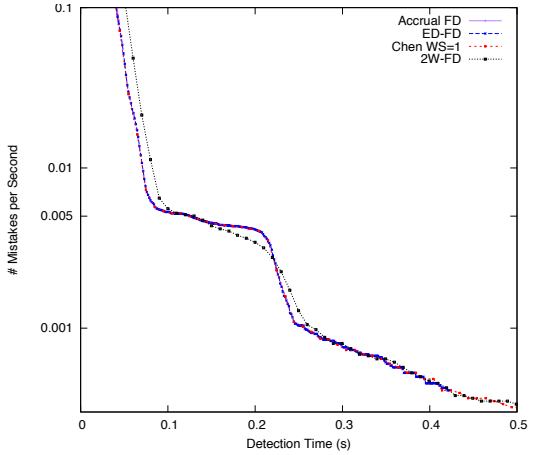


(a) R_M vs. T_D

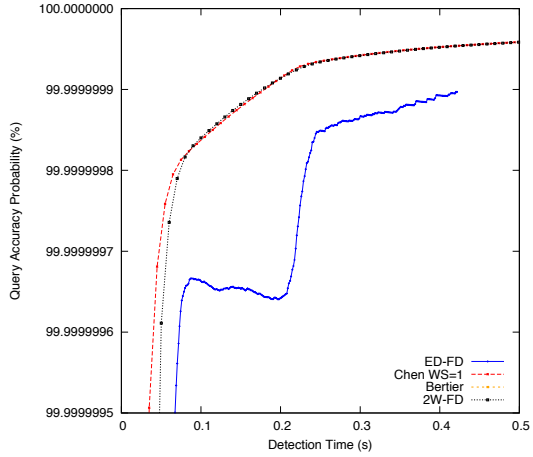


(b) P_A vs. T_D

Figure 5: Comparison of different algorithms in a WAN scenario



(a) R_M vs. T_D



(b) P_A vs. T_D

Figure 6: Comparison of different algorithms in a LAN scenario

the WAN scenario, mainly in the aggressive range ($T_D <$ 0.5s). It presents the lowest mistake rate (an improvement of up to 35%) and the best query accuracy probability for most measured detection times.

Note that in both graphs, the curve of the accrual failure detector with normal distribution (φ) does not appear as it is stopped early. This is due to the rounding error preventing the curves to the very conservative case. This fact that was also observed in related work [12]. Chen FD, with a window size of 1,000, is not present in the curves as its performance is worst (outside the bounds of the image).

LAN results.: Figures 6(a) and 6(b) show the results on mistake rate R_M and query accuracy probability P_A vs. detection time T_D in the LAN scenario. In this scenario, as the sample is very stable, the curve for the ED FD is also stopped early. In terms of R_M, the ED-FD and Chen with n=1 present the best performance. Nevertheless, the 2W-FD performs really close to them. The fact that we use for our computation of EAs, the maximum of the estimation of two windows, incurs in an increased T_D. In this scenario, where networks conditions are very stable, there is no benefit of using two windows. The results on P_A show that Chen with n=1 presents the best results in this scenario, and the 2W-FD behaves similarly.

Experiment's conclusions: From this experiment, we conclude that the 2W-FD presents the best performance in scenarios which present unstable network conditions (as the WAN trace), when compared to the most relevant existing algorithms for failure detection. Unfortunately, we were not able to generate traces in a LAN scenario that present unstable conditions. Remember, from section V-B1, that it is still possible to obtain significantly better results in the unstable case by using n_1=10,000 (remember the results shown in Figure 3(a) and the fact that accrual algorithms do not significantly increase their performance with bigger window

sizes [7], [12]). Since enlarging window sizes increases the processing and memory capacity that the algorithm requires, the decision of increasing the size of n_1 is left to the user depending on her particular needs and configuration.

In scenarios where network conditions are expected to be stable, we recommend using Chen with n=1. This algorithm requires less processing capacity than the others (the ϕ FD reduces performance as its window size decreases [7] and the ED FD performs similarly for small and big window sizes [12]), and presents the best results in stable scenarios. We believe it is really important to stress the fact that, to our knowledge, no previous work has evaluated Chen FD with n=1, even when it has been previously observed that Chen FD improves its performance as n decreases. We find this fact strange and to be a side contribution of our work, as it presents the best performance in stable scenarios and equal to the state of the art at a very low cost in terms of processing and memory capacity.

VI. Conclusions

Failure detection plays a very important role in dependable distributed systems. In this work, we introduced the Two Windows Failure Detector (2W-FD), an algorithm able to react to sudden network changes. By using two sliding windows of different sizes to store information about recent heartbeat history, the 2W-FD makes estimations on expected arrival times of future messages and decides on the failure suspicion of the monitored process.

The experiments we performed in both WAN and LAN scenarios indicate that our failure detector presents a better QoS in terms of false detections when comparing to existing FD algorithms in networks with unstable conditions. On stable networks, the Chen FD with the smallest window size slightly outperforms 2W-FD.

References

[1] Bertier, M., Marin, O., Sens, P.: Implementation and performance evaluation of an adaptable failure detector. In: Proceedings of the 2002 International Conference on Dependable Systems and Networks. pp. 354–363. DSN '02, IEEE Computer Society, Washington, DC, USA (2002), http://dl.acm.org/citation.cfm?id=647883.738261

[2] Bertier, M., Marin, O., Sens, P.: Performance analysis of a hierarchical failure detector. In: In Proceedings of the International Conference on Dependable Systems and Networks. pp. 635–644 (2003)

[3] Chandra, T.D., Toueg, S.: Unreliable failure detectors for reliable distributed systems. J. ACM 43(2), 225–267 (Mar 1996), http://doi.acm.org/10.1145/226643.226647

[4] Chen, W., Toueg, S., Aguilera, M.K.: On the quality of service of failure detectors. IEEE Trans. Comput. 51(5), 561–580 (May 2002), http://dx.doi.org/10.1109/TC.2002.1004595

[5] Défago, X., Urbán, P., Hayashibara, N., Katayama, T.: Definition and specification of accrual failure detectors. In: DSN. pp. 206–215. IEEE Computer Society (2005), http://dblp.uni-trier.de/db/conf/dsn/dsn2005.html#DefagoUHK05

[6] Deianov, B., Toueg, S.: Failure detector service for dependable computing. Proc, 2000 Int'l Conf. Dependable Systems and Neteworks pp. B14–B15 (June 2000)

[7] Hayashibara, N., Defago, X., Yared, R., Katayama, T.: The φ accrual failure detector. In: Reliable Distributed Systems, 2004. Proceedings of the 23rd IEEE International Symposium on. pp. 66–78 (2004)

[8] JAIST: http://ddsg.jaist.ac.jp/en/jst/data.html

[9] Paxson, V., Allman, M.: Computing tcp's retransmission timer (2000)

[10] Xiong, N., Vasilakos, A.V., Wu, J., Yang, Y.R., Rindos, A., Zhou, Y., Song, W.Z., Pan, Y.: A self-tuning failure detection scheme for cloud computing service. In: Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium. pp. 668–679. IPDPS '12, IEEE Computer Society, Washington, DC, USA (2012), http://dx.doi.org/10.1109/IPDPS.2012.126

[11] Xiong, N., Vasilakos, A.V., Yang, L.T., Song, L., Pan, Y., Kannan, R., Li, Y.: Comparative analysis of quality of service and memory usage for adaptive failure detectors in healthcare systems. IEEE J.Sel. A. Commun. 27(4), 495–509 (May 2009), http://dx.doi.org/10.1109/JSAC.2009.090512

[12] Xiong, N., Vasilakos, A.V., Yang, Y.R., Wei, S., Qiao, C., Wu, J.: General traffic-feature analysis for an effective failure detector in fault-tolerant wired and wireless networks. Tech. rep. (2011)