On the Quality of Service of Failure Detectors*

Wei Chen† Sam Toueg‡ Marcos Kawazoe Aguilera§

Abstract

We study the *quality of service (QoS)* of failure detectors. By QoS, we mean a specification that quantifies (a) how fast the failure detector detects actual failures, and (b) how well it avoids false detections. We first propose a set of QoS metrics to specify failure detectors for systems with probabilistic behaviors, i.e., for systems where message delays and message losses follow some probability distributions. We then give a new failure detector algorithm and analyze its QoS in terms of the proposed metrics. We show that, among a large class of failure detectors, the new algorithm is optimal with respect to some of these QoS metrics. Given a set of failure detector QoS requirements, we show how to compute the parameters of our algorithm so that it satisfies these requirements, and we show how this can be done even if the probabilistic behavior of the system is not known. We then present some simulation results that show that the new failure detector algorithm provides a better QoS than an algorithm that is commonly used in practice. Finally, we briefly explain how to make our failure detector *adaptive*, so that it automatically reconfigures itself when there is a change in the probabilistic behavior of the network.

keywords: failure detectors, quality of service, fault tolerance, distributed algorithm, probabilistic analysis

1 Introduction

Fault-tolerant distributed systems are designed to provide reliable and continuous service despite the failures of some of their components. A basic building block of such systems is the *failure detector*. Failure detectors are used in a wide variety of settings, such as network communication protocols [10], computer cluster management [23], group membership protocols [5, 9, 7, 27, 22, 21], etc.

Roughly speaking, a failure detector provides some information on which processes have crashed. This information, typically given in the form of a list of *suspects*, is not always up-to-date or correct:

*Research partially supported by NSF grant CCR-9711403 and an Olin Fellowship.

†Oracle Corporation, One Oracle Drive, Nashua, NH 03062, USA. Email: Wei.Chen@oracle.com

‡DIX Departement d'Informatique, Ecole Polytechnique, 91128 Palaiseau Cedex, France. Email: sam@dix.polytechnique.fr

§Compaq Systems Research Center, 130 Lytton Avenue, Palo Alto, CA 94301-1044, USA. Email:aguilera@pa.dec.com

a failure detector may take a long time to start suspecting a process that has crashed, and it may erroneously suspect a process that has not crashed (in practice this can be due to message losses and delays).

Chandra and Toueg [12] provide the first formal specification of *unreliable failure detectors* and show that they can be used to solve some fundamental problems in distributed computing, namely, *consensus* and *atomic broadcast*. This approach was later used and generalized in other works, e.g., [20, 16, 17, 1, 3, 2].

In all of the above works, failure detectors are specified in terms of their *eventual* behavior (e.g., a process that crashes is eventually suspected). Such specifications are appropriate for asynchronous systems, in which there is no timing assumption whatsoever.[1] Many applications, however, have some timing constraints, and for such applications, failure detectors with eventual guarantees are not sufficient. For example, a failure detector that starts suspecting a process one hour after it crashed can be used to solve asynchronous consensus, but it is useless to an application that needs to solve many instances of consensus per minute. Applications that have timing constraints require failure detectors that provide a *quality of service (QoS)* with some quantitative timeliness guarantees.

In this paper, we study the QoS of failure detectors in systems where message delays and message losses follow some probability distributions. We first propose a set of metrics that can be used to specify the QoS of a failure detector; these QoS metrics quantify (a) how *fast* it detects actual failures, and (b) how *well* it avoids false detections. We then give a new failure detector algorithm and analyze its QoS in terms of the proposed metrics. We show that, among a large class of failure detectors, the new algorithm is optimal with respect to some of these QoS metrics. Given a set of failure detector QoS requirements, we show how to compute the parameters of our algorithm so that it satisfies these requirements, and we show how this can be done even if the probabilistic behavior of the system is not known. Finally, we give simulation results showing that the new failure detector algorithm provides a better QoS than an algorithm that is commonly used in practice. The QoS specification and the analysis of our failure detector algorithm is based on the theory of stochastic processes. To the best of our knowledge, this work is the first comprehensive and systematic study of the QoS of failure detectors using probability theory.

1.1 On the QoS Specification of Failure Detectors

We consider message-passing distributed systems in which processes may fail by crashing, and messages may be delayed or dropped by communication links.[2] A failure detector can be *slow*, i.e., it may take a long time to suspect a process that has crashed, and it can make *mistakes*, i.e., it may erroneously suspect some processes that are actually up (such a mistake is not necessarily permanent: the failure

[1] Even though the *fail-aware* failure detector of [17] is implemented in the "timed asynchronous" model, its specification is for the asynchronous model.

[2] We assume that process crashes are permanent, or, equivalently, that a process that recovers from a crash assumes a new identity.

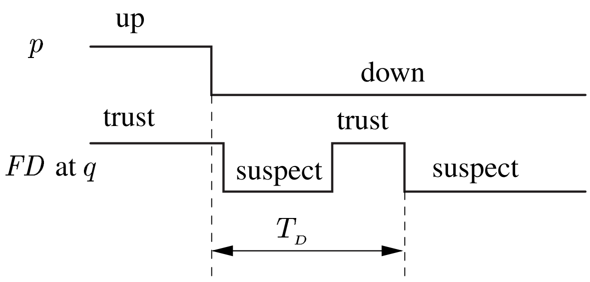

Figure 1: Detection time T_D

detector may later stop suspecting this process). To be useful, a failure detector has to be reasonably fast and accurate.

In this paper, we propose a set of metrics for the QoS specification of failure detectors. In general, these QoS metrics should be able to describe the failure detector's *speed* (how fast it detects crashes) and its *accuracy* (how well it avoids mistakes). Note that speed is with respect to processes that crash, while accuracy is with respect to processes that do not crash.

A failure detector's speed is easy to measure: this is simply the time that elapses from the moment when a process p crashes to the time when the failure detector starts suspecting p permanently. This QoS metric, called *detection time*, is illustrated in Fig. 1.

How do we measure a failure detector's accuracy? It turns out that determining a good set of accuracy metrics is a delicate task. To illustrate some of the subtleties involved, consider a system of two processes p and q connected by a lossy communication link, and suppose that the failure detector at q monitors process p. The output of the failure detector at q is either "I suspect that p has crashed" or "I trust that p is up", and it may alternate between these two outputs from time to time. For the purpose of measuring the accuracy of the failure detector at q, suppose that p does not crash.

Consider an application that queries q's failure detector at random times. For such an application, a natural measure of accuracy is the probability that, *when queried at a random time*, the failure detector at q indicates correctly that p is up. This QoS metric is the *query accuracy probability*. For example, in Fig. 2, the query accuracy probability of FD_1 at q is $12/(12+4) = .75$.

The query accuracy probability, however, is not sufficient to fully describe the accuracy of a failure detector. To see this, we show in Fig. 2 two failure detectors FD_1 and FD_2 such that (a) they have the same query accuracy probability, but (b) FD_2 makes mistakes more frequently than FD_1.[3] In some applications, every mistake causes a costly interrupt, and for such applications the *mistake rate* is an important accuracy metric.

Note, however, that the mistake rate alone is not sufficient to characterize accuracy: as shown in Fig. 3, two failure detectors can have the same mistake rate, but different query accuracy probabilities.

[3]The failure detector *makes a mistake* each time its output changes from "trust" to "suspect" while p is actually up.

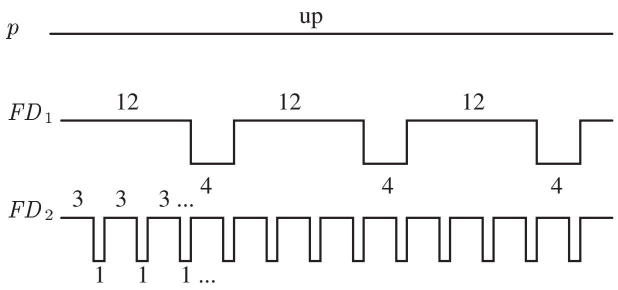

Figure 2: FD_1 and FD_2 have the same query accuracy probability of $.75$, but the mistake rate of FD_2 is four times that of FD_1

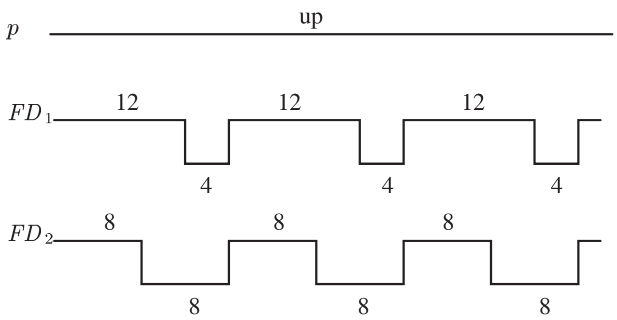

Figure 3: FD_1 and FD_2 have the same mistake rate $1/16$, but the query accuracy probabilities of FD_1 and FD_2 are $.75$ and $.50$, respectively

Even when used together, the above two accuracy metrics are still not sufficient. In fact, it is easy to find two failure detectors FD_1 and FD_2, such that (a) FD_1 is better than FD_2 in both measures (i.e., it has a higher query accuracy probability *and* a lower mistake rate), but (b) FD_2 is better than FD_1 in another respect: specifically, whenever FD_2 makes a mistake, it corrects this mistake faster than FD_1; in other words, the *mistake durations* in FD_2 are smaller than in FD_1. Having small mistake durations may be important to some applications.

As it can be seen from the above, there are several different aspects of accuracy that may be important to different applications, and each aspect has a corresponding accuracy metric.

In this paper, we identify six accuracy metrics (since the behavior of a failure detector is probabilistic, most of these metrics are random variables). We then use the theory of stochastic processes to quantify the relation between these metrics. This analysis allows us to select two accuracy metrics as the *primary* ones in the sense that: (a) they are not redundant (one cannot be derived from the other), and (b) together, they can be used to derive the other four accuracy metrics.

In summary, we show that the QoS specification of failure detectors can be given in terms of three basic metrics, namely, the detection time and the two primary accuracy metrics that we identified. Taken together, these metrics can be used to characterize and compare the QoS of failure detectors.

1.2 The Design and Analysis of a New Failure Detector Algorithm

In this paper, we consider a simple system of two processes p and q, connected through a communication link. Process p may fail by crashing, and the link between p and q may delay or drop messages. Message delays and message losses follow some probabilistic distributions. Process q has a failure detector that monitors p and outputs either "I suspect that p has crashed" or "I trust that p is up" ("suspect p" and "trust p" in short, respectively).

A Common Failure Detection Algorithm and its Drawbacks. A simple failure detection algorithm, commonly used in practice, works as follows: at regular time intervals, process p sends a heartbeat message to q; when q receives a heartbeat message, it trusts p and starts a timer with a fixed timeout value TO; if the timer expires before q receives a newer heartbeat message from p, then q starts suspecting p.

This algorithm has two undesirable characteristics; one regards its accuracy and the other its detection time, as we now explain. Consider the i-th heartbeat message m_i. Intuitively, the probability of a *premature timeout* on m_i should depend solely on m_i, and in particular on m_i's delay. With the simple algorithm, however, the probability of a premature timeout on m_i also depends on the heartbeat m_{i-1} that precedes m_i! In fact, the timer for m_i is started upon the receipt of m_{i-1}, and so if m_{i-1} is "fast", the timer for m_i starts early and this increases the probability of a premature timeout on m_i. This dependency on past heartbeats is undesirable.

To see the second problem, suppose p sends a heartbeat just before it crashes, and let d be the delay of this last heartbeat. In the simple algorithm, q would permanently suspect p only $d + TO$ time units after p crashes. Thus, the worst-case detection time for this algorithm is the *maximum* message delay plus TO. This is impractical because in many systems the maximum message delay is orders of magnitude larger than the average message delay.

The source of the above problems is that even though the heartbeats are sent at regular intervals, the timers to "catch" them expire at irregular times, namely the receipt times of the heartbeats plus a fixed TO. The algorithm that we propose eliminates this problem. As a result, the probability of a premature timeout on heartbeat m_i does *not* depend on the behavior of the heartbeats that precede m_i, and the detection time does *not* depend on the maximum message delay.

A New Algorithm and its QoS Analysis. In the new algorithm, process p sends heartbeat messages $m_1, m_2, \ldots$ to q periodically every η time units (just as in the simple algorithm). To determine whether to suspect p, q uses a sequence $\tau_1, \tau_2, \ldots$ of fixed time points, called *freshness points*, obtained by shifting the sending time of the heartbeat messages by a fixed parameter δ. More precisely, $\tau_i = \sigma_i + \delta$, where σ_i is the time when m_i is sent. For any time t, let i be so that $t \in [\tau_i, \tau_{i+1})$; then q trusts p at time t if and only if q has received heartbeat m_i or higher.

Given the probabilistic behavior of the system (i.e., the probability of message losses and the distribution of message delays), and the parameters η and δ of the algorithm, we determine the QoS of the new algorithm using the theory of stochastic processes. Simulation results given in Section 7 are consistent with our QoS analysis, and they show that the new algorithm performs better than the common one.

In contrast to the common algorithm, the new algorithm guarantees an upper bound on the detection time. Moreover, the new algorithm is optimal in the sense that it has the best possible query accuracy probability with respect to any given bound on the detection time. More precisely, we show that among all failure detectors that send heartbeats at the same rate (they use the same network bandwidth) and satisfy the same upper bound on the detection time, the new algorithm has the best query accuracy probability.

The first version of our algorithm (described above) assumes that p and q have synchronized clocks. This assumption is not unrealistic, even in large networks. For example, GPS and Cesium clocks are becoming accessible, and they can provide clocks that are very closely synchronized (see, e.g., [29]). When synchronized clocks are not available, we propose a modification to this algorithm that performs equally well in practice, as shown by our simulations. The basic idea is to use past heartbeat messages to obtain accurate estimates of the expected arrival times of future heartbeats, and then use these estimates to find the freshness points. This is explained in Section 6.

Configuring our Algorithm to Meet the Failure Detector Requirements of an Application. Given a set of failure detector QoS requirements (provided by an application), we show how to compute the parameters of our algorithm to achieve these requirements. We first do so assuming that one knows the probabilistic behavior of the system (i.e., the probability distributions of message delays and message losses). We then drop this assumption, and show how to configure the failure detector to meet the QoS requirements of an application even when the probabilistic behavior of the system is not known.

1.3 Related Work

In [19], Gouda and McGuire measure the performance of some failure detector protocols under the assumption that the protocol stops as soon as some process is suspected to have crashed (even if this suspicion is a mistake). This class of failure detectors is less general than the one that we studied here: in our work, a failure detector can alternate between suspicion and trust many times.

In [28], van Renesse *et. al.* propose a scalable gossip-style randomized failure detector protocol. They measure the accuracy of this protocol in terms of the *probability of premature timeouts*.[4] The probability of premature timeouts, however, is not an appropriate metric for the specification of failure detectors in general: it is implementation-specific and it cannot be used to compare failure detectors that use timeouts in different ways. This point is further explained at the end of Section 2.3.

In [24], Raynal and Tronel present an algorithm that detects member failures in a group: if some process detects a failure in the group (perhaps a false detection), then all processes report a group failure and the protocol terminates. The algorithm uses heartbeat-style protocol, and its timeout mechanism is the same as the simple algorithm that we described in Section 1.2.

[4] This is called "the probability of mistakes" in [28].

In [29], Veríssimo and Raynal study *QoS failure detectors* — these are detectors that indicate when a service does not meet its quality-of-service requirements. In contrast, this paper studies the QoS *of* failure detectors, i.e., how well a failure detector works.

Heartbeat-style failure detectors are commonly used in practice. To keep both good detection time and good accuracy, many implementations rely on special features of the operating system and communication system to try to ensure that heartbeat messages are received at regular intervals (see discussion in Section 12.9 of [23]). This is not easy even for closely-connected computer clusters, and it is very hard in wide-area networks.

The probabilistic network model used in this paper is similar to the ones used in [14, 6] for probabilistic clock synchronization. The method of estimating the expected arrival times of heartbeat messages is close to the method of remote clock reading of [6].

The rest of the paper is organized as follows. In Section 2, we propose a set of metrics to specify the QoS of failure detectors. In Section 3, we describe a new failure detector algorithm and analyze its QoS in terms of these metrics; we also present an optimality result. We then explain how to set the algorithm's parameters to meet some given QoS requirements — first in the case when we know the probabilistic behavior of messages (Section 4), and then in the case when this is not known (Section 5). In Section 6 we deal with unsynchronized clocks. We present the results of some simulations in Section 7, and we conclude the paper with some discussion in Section 8. Appendix A lists the main symbols used in the paper, and Appendices B to D give the proofs of the main theorems. More detailed proofs can be found in [13].

2 On the QoS Specification of Failure Detectors

We consider a system of two processes p and q. We assume that the failure detector at q monitors p, and that q does not crash. Henceforth, real time is continuous and ranges from 0 to ∞.

2.1 The Failure Detector Model

The output of the failure detector at q at time t is either S or T, which means that q suspects or trusts p at time t, respectively. A *transition* occurs when the output of the failure detector at q changes: An *S-transition* occurs when the output at q changes from T to S; a *T-transition* occurs when the output at q changes from S to T. We assume that there are only a finite number of transitions during any finite time interval.

Since the behavior of the system is probabilistic, the precise definition of our model and of our QoS metrics uses the theory of stochastic processes. To keep our presentation at an intuitive level, we omit the technical details related to this theory (they can be found in [13]).

We consider only failure detectors whose behavior eventually reaches *steady state*, as we now explain informally. When a failure detector starts running, and for a while after, its behavior depends on the

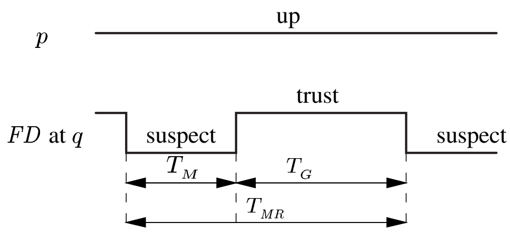

Figure 4: Mistake duration T_M, good period duration T_G, and mistake recurrence time T_{MR}

initial condition (such as whether initially q suspects p or not) and on how long it has been running. Typically, as time passes the effect of the initial condition gradually diminishes and its behavior no longer depends on how long it has been running — i.e., eventually the failure detector behavior reaches equilibrium, or steady state. In steady state, the probability law governing the behavior of the failure detector does not change over time. The QoS metrics that we propose refer to the behavior of a failure detector after it reaches steady state. Most of these metrics are random variables.

2.2 Primary Metrics

We propose three primary metrics for the QoS specification of failure detectors. The first one measures the speed of a failure detector. It is defined with respect to the runs in which p crashes.

Detection time (T_D): Informally, T_D is the time that elapses from p's crash to the time when q starts suspecting p permanently. More precisely, T_D is a random variable representing the time that elapses from the time that p crashes to the time when the final S-transition (of the failure detector at q) occurs and there are no transitions afterwards (Fig. 1). If there is no such final S-transition, then $T_D = \infty$; if such an S-transition occurs before p crashes, then $T_D = 0$.[5]

We next define some metrics that are used to specify the accuracy of a failure-detector. Throughout the paper, all accuracy metrics are defined with respect to *failure-free* runs, i.e., runs in which p does not crash.[6] There are two primary accuracy metrics:

Mistake recurrence time (T_{MR}): this measures the time between two consecutive mistakes. More precisely, T_{MR} is a random variable representing the time that elapses from an S-transition to the next one (Fig. 4).

Mistake duration (T_M): this measures the time it takes the failure detector to correct a mistake. More precisely, T_M is a random variable representing the time that elapses from an S-transition to the next T-transition (Fig. 4).

As we discussed in the introduction, there are many aspects of failure detector accuracy that may be important to applications. Thus, in addition to T_{MR} and T_M, we propose four other accuracy metrics in the next section. We selected T_{MR} and T_M as the primary metrics because given these two, one can

[5]We omit the boundary cases of other metrics since they can be similarly defined.
[6]As explained in [13], these metrics are also meaningful for runs in which p crashes.

compute the other four (this will be shown in Section 2.4).

2.3 Derived Metrics

We propose four additional accuracy metrics:

Average mistake rate (λ_M): this measures the rate at which a failure detector make mistakes, i.e., it is the average number of S-transitions per time unit. This metric is important to long-lived applications where each failure detector mistake (each S-transition) results in a costly interrupt. This is the case for applications such as group membership and cluster management.

Query accuracy probability (P_A): this is the probability that the failure detector's output is correct at a random time. This metric is important to applications that interact with the failure detector by querying it at random times.

Many applications can make progress only during *good periods* — periods in which the failure detector makes no mistakes. This observation leads to the following two metrics.

Good period duration (T_G): this measures the length of a good period. More precisely, T_G is a random variable representing the time that elapses from a T-transition to the next S-transition (Fig. 4).

For short-lived applications, however, a closely related metric may be more relevant. Suppose that an application is started at a random time in a good period. If the *remaining part* of the good period is long enough, the short-lived application will be able to complete its task. The metric that measures the remaining part of the good period is:

Forward good period duration (T_{FG}): this is a random variable representing the time that elapses from a random time at which q trusts p, to the time of the next S-transition.

At first sight, it may seem that, on the average, T_{FG} is just half of T_G (the length of a good period). But this is incorrect, and in Section 2.4 we give the actual relation between T_{FG} and T_G.

An important remark is now in order. For timeout-based failure detectors, the probability of premature timeouts has sometimes been used as the accuracy measure: this is the probability that when the timer is set, it will prematurely timeout on a process that is actually up. The measure, however, is not appropriate because: (a) it is implementation-specific, and (b) it is not useful to applications unless it is given together with other implementation-specific measures, e.g., how often timers are started, whether the timers are started at regular or variable intervals, whether the timeout periods are fixed or variable, etc. (many such variations exist in practice [10, 19, 28]). Thus, the probability of premature timeouts is not a good metric for the specification of failure detectors, e.g., it cannot be used to compare the QoS of failure detectors that use timeouts in different ways. The six accuracy metrics that we identified in this paper do not refer to implementation-specific features, in particular, they do not refer to timeouts at all.

2.4 How the Accuracy Metrics are Related

Theorem 1 below explains how our six accuracy metrics are related. We then use this theorem to justify our choice of the primary accuracy metrics. Henceforth, $Pr(A)$ denotes the probability of event A; $E(X)$, $E(X^k)$, and $V(X)$ denote the expected value (or mean), the k-th moment, and the variance of random variable X, respectively.

Parts (2) and (3) of Theorem 1 assume that in failure-free runs, the probabilistic distribution of failure detector histories is *ergodic*. Roughly speaking, this means that in failure-free runs, the failure detector slowly "forgets" its past history: from any given time on, its future behavior may depend only on its recent behavior. We call failure detectors satisfying this ergodicity condition *ergodic failure detectors*. Ergodicity is a basic concept in the theory of stochastic processes [26], but the technical details are substantial and outside the scope of this paper.

We have also determined the relations between our accuracy metrics in the case that ergodicity does *not* hold. The resulting expressions are more complex (they are generalized versions of those given below) and can be found in [13].

Theorem 1 *For any ergodic failure detector, the following results hold: (1)* $T_G = T_{MR} - T_M$. *(2) If* $0 < E(T_{MR}) < \infty$*, then* $\lambda_M = 1/E(T_{MR})$*, and* $P_A = E(T_G)/E(T_{MR})$*. (3) If* $0 < E(T_{MR}) < \infty$ *and* $E(T_G) = 0$*, then* T_{FG} *is always 0. If* $0 < E(T_{MR}) < \infty$ *and* $E(T_G) \neq 0$*, then (3a) for all* $x \in [0, \infty)$, $Pr(T_{FG} \leq x) = \int_0^x Pr(T_G > y)dy/E(T_G)$*, (3b)* $E(T_{FG}^k) = E(T_G^{k+1})/[(k+1)E(T_G)]$*. In particular, (3c)* $E(T_{FG}) = [1 + V(T_G)/E(T_G)^2]E(T_G)/2$.

The fact that $T_G = T_{MR} - T_M$ holds is immediate by definition. The proofs of parts (2) and (3) use the theory of stochastic processes. Part (2) is intuitive, while part (3), which relates T_G and T_{FG}, is more complex. In particular, part (3c) is counter-intuitive: one may think that $E(T_{FG}) = E(T_G)/2$, but part (3c) says that $E(T_{FG})$ is in general larger than $E(T_G)/2$ (this is a version of the "waiting time paradox" in the theory of stochastic processes [4]).

We now explain how Theorem 1 guided our selection of the primary accuracy metrics. Parts (2) and (3) show that λ_M, P_A and T_{FG} can be derived from T_{MR}, T_M and T_G. This suggests that the primary metrics should be selected among T_{MR}, T_M and T_G. Moreover, since $T_G = T_{MR} - T_M$, it is clear that given the joint distribution of any two of them, one can derive the remaining one. Thus, two of T_{MR}, T_M and T_G should be selected as the primary metrics, but which two? By choosing T_{MR} and T_M as our primary metrics, we get the following convenient property that helps to compare failure detectors: if FD_1 is better than FD_2 in terms of both $E(T_{MR})$ and $E(T_M)$ (the expected values of the primary metrics) then we can be sure that FD_1 is also better than FD_2 in terms of $E(T_G)$ (the expected values of the other metric). We would not get this useful property if T_G were selected as one of the primary metrics.[7]

[7]For example, FD_1 may be better than FD_2 in terms of both $E(T_G)$ and $E(T_M)$, but worse than FD_2 in terms of $E(T_{MR})$.

3 The Design and QoS Analysis of a New Failure Detector Algorithm

3.1 The Probabilistic Network Model

We assume that processes p and q are connected by a link that does not create or duplicate messages,[8] but may delay or drop messages. Note that the link here represents an end-to-end connection and does not necessarily correspond to a physical link.

We assume that the message loss and message delay behavior of any message sent through the link is probabilistic, and is characterized by the following two parameters: (a) *message loss probability* p_L, which is the probability that a message is dropped by the link; and (b) *message delay* D, which is a random variable with range $(0, \infty)$ representing the delay from the time a message is sent to the time it is received, under the condition that the message is not dropped by the link. We assume that the expected value $E(D)$ and the variance $V(D)$ of D are finite. Note that our model does not assume that the message delay time D follows any particular distribution, and thus it is applicable to many practical systems.

Processes p and q have access to their own local clocks. For simplicity, we assume that there is no clock drift, i.e., local clocks run at the same speed as real time (our results can be easily generalized to the case where local clocks have bounded drifts). In Sections 3, 4 and 5, we further assume that clocks are synchronized. We explain how to remove this assumption in Section 6.

For simplicity we assume that the probabilistic behavior of the network does not change over time. In Section 8, we explain how to modify the algorithm so that it dynamically adapts to changes in the probabilistic behavior of the system.

We assume that crashes cannot be predicted, i.e., the state of the system at any given time has no information whatsoever on the occurrence of future crashes (this excludes a system with program-controlled crashes [11]). Moreover, the delay and loss behaviors of the messages that a process sends are independent of whether (and when) the process crashes.

3.2 The Algorithm

The new algorithm works as follows. The monitored process p periodically sends heartbeat messages $m_1, m_2, m_3, \ldots$ to q every η time units, where η is a parameter of the algorithm. Every heartbeat message m_i is tagged with its sequence number i. Henceforth, σ_i denotes the sending time of message m_i. The monitoring process q shifts the σ_i's forward by δ — the other parameter of the algorithm — to obtain the sequence of times $\tau_1 < \tau_2 < \tau_3 < \ldots$, where $\tau_i = \sigma_i + \delta$. Process q uses the τ_i's and the times it receives heartbeat messages, to determine whether to trust or suspect p, as follows. Consider

[8] Message duplication can be easily taken care of: whenever we refer to a message being received, we change it to the *first copy* of the message being received. With this modification, all definitions and analyses in the paper go through, and in particular, our results remain correct without any change.

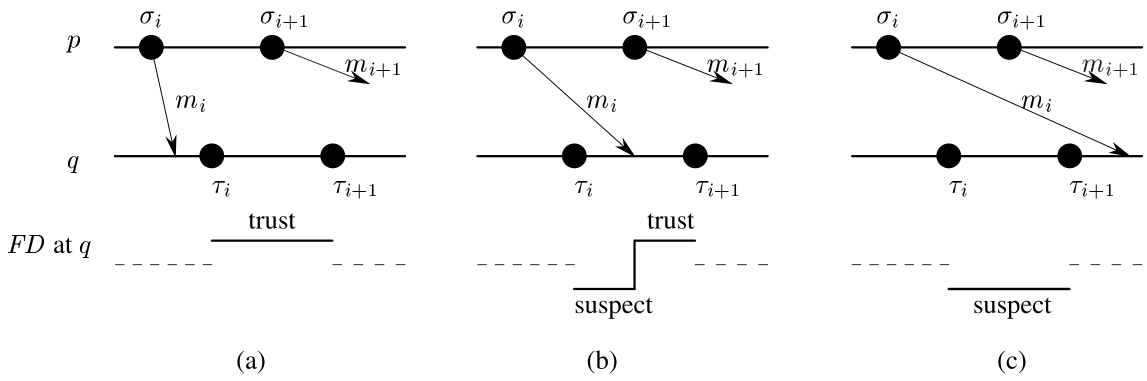

Figure 5: Three scenarios of the failure detector output in one interval $[\tau_i, \tau_{i+1})$

Process p:

for all $i \geq 1$, at time $\sigma_i = i \cdot \eta$, send heartbeat m_i to q;

Process q:

Initialization: $output = S$; {suspect p initially}

for all $i \geq 1$, at time $\tau_i = \sigma_i + \delta$:
  if did not receive m_j with $j \geq i$ **then** $output \leftarrow S$; {suspect p if no fresh message is received}

upon receive message m_j at time $t \in [\tau_i, \tau_{i+1})$:
  if $j \geq i$ **then** $output \leftarrow T$; {trust p when some fresh message is received}

Figure 6: Failure detector algorithm NFD-S with parameters η and δ (clocks are synchronized)

time period $[\tau_i, \tau_{i+1})$. At time τ_i, q checks whether it has received some message m_j with $j \geq i$. If so, q trusts p during the entire period $[\tau_i, \tau_{i+1})$ (Fig. 5 (a)). If not, q starts suspecting p. If at some time before τ_{i+1}, q receives some message m_j with $j \geq i$ then q starts trusting p from that time until τ_{i+1}. (Fig. 5 (b)). If by time τ_{i+1}, q has not received any message m_j with $j \geq i$, then q suspects p during the entire period $[\tau_i, \tau_{i+1})$ (Fig. 5 (c)). This procedure is repeated for every time period. The detailed algorithm with parameters η and δ is denoted by NFD-S, and is given in Fig. 6.[9]

Note that from time τ_i to τ_{i+1}, only messages m_j with $j \geq i$ can affect the output of the failure detector. For this reason, τ_i is called a *freshness point*: from time τ_i to τ_{i+1}, messages m_j with $j \geq i$ are *still fresh* (useful). With this algorithm, q trusts p at time t if and only if q received a message that is still fresh at time t.

[9]This version of the algorithm is convenient for illustrating the main idea and for performing the analysis. We have omitted some obvious optimizations.

3.3 The QoS Analysis of the Algorithm

We now give the QoS of the algorithm (the analysis is given in Appendix B). We assume that the link from p to q satisfies the following *message independence* property: the behaviors of any two heartbeat messages sent by p are independent.[10] Henceforth, let $\tau_0 \stackrel{\text{def}}{=} 0$, and $\tau_i = \sigma_i + \delta$ for $i \geq 1$ (as in line 3 of the algorithm).

We first formalize the intuition behind freshness points and fresh messages:

Lemma 2 *For all $i \geq 0$ and all time $t \in [\tau_i, \tau_{i+1})$, q trusts p at time t if and only if q has received some message m_j with $j \geq i$ by time t.*

The following definitions are for runs where p does not crashes.

Definition 1

(1) For any $i \geq 1$, let k be the smallest integer such that for all $j \geq i + k$, m_j is sent at or after time τ_i.

(2) For any $i \geq 1$, let $p_j(x)$ be the probability that q does not receive message m_{i+j} by time $\tau_i + x$, for every $j \geq 0$ and every $x \geq 0$; let $p_0 = p_0(0)$.

(3) For any $i \geq 2$, let q_0 be the probability that q receives message m_{i-1} before time τ_i.

(4) For any $i \geq 1$, let $u(x)$ be the probability that q suspects p at time $\tau_i + x$, for every $x \in [0, \eta)$.

(5) For any $i \geq 2$, let p_S be the probability that an S-transition occurs at time τ_i.

The above definitions are given in terms of i, a positive integer. Proposition 3, however, shows that they are actually independent of i.

Proposition 3 *(1) $k = \lceil \delta/\eta \rceil$. (2) For all $j \geq 0$ and for all $x \geq 0$, $p_j(x) = p_L + (1 - p_L)Pr(D > \delta + x - j\eta)$. (3) $q_0 = (1 - p_L)Pr(D < \delta + \eta)$. (4) For all $x \in [0, \eta)$, $u(x) = \prod_{j=0}^{k} p_j(x)$. (5) $p_S = q_0 \cdot u(0)$.*

By definition, if $p_0 = 0$ then for every $i \geq 1$, the probability that q receives m_i by time τ_i is 1. Thus, if $p_0 = 0$ then, with probability one, q trusts p forever after time τ_1. Similarly, it is easy to see that if $q_0 = 0$ then, with probability one, q suspects p forever. So $p_0 = 0$ and $q_0 = 0$ are degenerated cases of no interest. We henceforth assume that $p_0 > 0$ and $q_0 > 0$.

The following theorem summarizes our QoS analysis of the new failure detector algorithm.

[10] In practice, this holds only if consecutive heartbeats are sent more than some Δ time units apart, where Δ depends on the system. So assuming that the behavior of heartbeats are independent is equivalent to assuming that $\eta > \Delta$.

Theorem 4 *Consider a system with synchronized clocks, where the probability of message losses is p_L, and the distribution of message delays is $P(D \le x)$. The failure detector NFD-S of Fig. 6 with parameters η and δ has the following properties.*
(1) The detection time is bounded:

$$T_D \le \delta + \eta. \tag{3.1}$$

(2) The average mistake recurrence time is:

$$E(T_{MR}) = \frac{\eta}{p_S}. \tag{3.2}$$

(3) The average mistake duration is:

$$E(T_M) = \frac{\int_0^\eta u(x)\,dx}{p_S}. \tag{3.3}$$

From $E(T_{MR})$ and $E(T_M)$ given in the theorem above, we can easily derive the other accuracy measures using Theorem 1. For example, we can get the query accuracy probability $P_A = 1 - E(T_M)/E(T_{MR}) = 1 - 1/\eta \cdot \int_0^\eta u(x)\,dx$.

Theorem 4 (1) shows an important property of the algorithm: the detection time is bounded, and the bound does not depend on the behavior of message delays and losses.

In Sections 4, 5 and 6, we show how to use Theorem 4 to compute the failure detector parameters, so that the failure detector satisfies some QoS requirements (given by an application).

3.4 An Optimality Result

Among all failure detectors that send heartbeats at the same rate and satisfy the same upper bound on the detection time, the new algorithm provides the best query accuracy probability. More precisely, let $\mathcal{C}$ be the class of failure detector algorithms A such that in every run of A, process p sends heartbeats to q every η time units and A satisfies $T_D \le T_D^U$ for some constant T_D^U. Let A^* be the instance of the new failure detector algorithm NFD-S with parameters η and $\delta = T_D^U - \eta$. By part (1) of Theorem 4, we know that $A^* \in \mathcal{C}$. We can show that

Theorem 5 *For any $A \in \mathcal{C}$, let P_A be the query accuracy probability of A. Let P_A^* be the query accuracy probability of A^*. Then $P_A^* \ge P_A$.*

The theorem is a consequence of the following important property of algorithm A^*. Consider any algorithm $A \in \mathcal{C}$. Let r be any failure-free run of A^*, and r' be any failure-free run of A in which the heartbeat delays and losses are exactly as in r. We can show that if q suspects p at time t in r, then q also suspects p at time t in r'. With this property, it is easy to see that the probability that q trusts p at a random time in A^* must be at least as high as the probability that q trusts p at a random time in any $A \in \mathcal{C}$. The detailed proof is given in Appendix C.

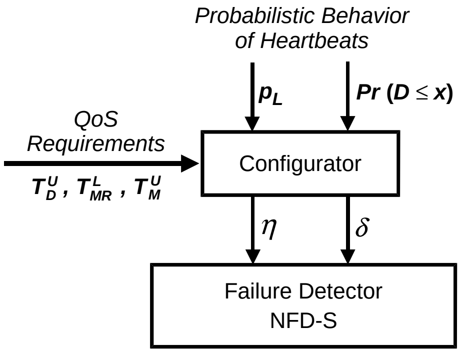

Figure 7: Meeting QoS requirements with NFD-S. The probabilistic behavior of heartbeats is given, and clocks are synchronized

4 Configuring the Failure Detector to Satisfy QoS Requirements

Suppose we are given a set of failure detector QoS requirements (the QoS requirements could be given by the application that uses this failure detector). We now show how to compute the parameters η and δ of our failure detector algorithm, so that these requirements are satisfied. We assume that (a) the local clocks of processes are synchronized, and (b) one knows the probabilistic behavior of the messages, i.e., the message loss probability p_L and the distribution of message delays $Pr(D \le x)$. In Sections 5 and 6, we consider the cases when these assumptions do not hold.

We assume that the QoS requirements are expressed using the primary metrics. More precisely, a set of QoS requirements is a tuple (T_D^U, T_{MR}^L, T_M^U) of positive numbers, where T_D^U is an upper bound on the detection time, T_{MR}^L is a lower bound on the average mistake recurrence time, and T_M^U is an upper bound on the average mistake duration. In other words, the requirements are that:[11]

$$T_D \le T_D^U,\ E(T_{MR}) \ge T_{MR}^L,\ E(T_M) \le T_M^U. \tag{4.4}$$

Our goal, illustrated in Fig. 7, is to find a configuration procedure that takes as inputs (a) the QoS requirements, namely T_D^U, T_{MR}^L, T_M^U, and (b) the probabilistic behavior of the heartbeat messages, namely p_L and $Pr(D \le x)$, and outputs the failure detector parameters η and δ so that the failure detector satisfies the QoS requirements in (4.4). Furthermore, to minimize the network bandwidth taken by the failure detector, we want a configuration procedure that finds the largest intersending interval η that satisfy these QoS requirements.

Using Theorem 4, our goal can be stated as a mathematical programming problem:

$$\text{maximize} \quad \eta$$

[11]Note that the bounds on the primary metrics $E(T_{MR})$ and $E(T_M)$ also impose bounds on the derived metrics, according to Theorem 1. More precisely, we have $\lambda_M \le 1/T_{MR}^L$, $P_A \ge (T_{MR}^L - T_M^U)/T_{MR}^L$, $E(T_G) \ge T_{MR}^L - T_M^U$, and $E(T_{FG}) \ge (T_{MR}^L - T_M^U)/2$.

$$\text{subject to} \quad \delta + \eta \le T_D^U \tag{4.5}$$

$$\frac{\eta}{p_S} \ge T_{MR}^L \tag{4.6}$$

$$\frac{\int_0^\eta u(x)\,dx}{p_S} \le T_M^U \tag{4.7}$$

where the values of $u(x)$ and p_S are given by Proposition 3. Solving this problem is hard, so instead we show how to find some η and δ that satisfy (4.5)–(4.7) (but the η that we find may not be the largest possible). To do so, we replace (4.7) with a simpler and stronger constraint, and then compute the optimal solution of this modified problem (see Appendix D). We obtain the following procedure to find η and δ:

- *Step 1*: Compute $q_0' = (1 - p_L)Pr(D < T_D^U)$, and let $\eta_{\max} = q_0' T_M^U$. If $\eta_{\max} = 0$, then output "QoS cannot be achieved" and stop; else continue.

- *Step 2*: Let
$$f(\eta) = \frac{\eta}{q_0' \prod_{j=1}^{\lceil T_D^U/\eta \rceil - 1} [p_L + (1 - p_L)Pr(D > T_D^U - j\eta)]}. \tag{4.8}$$
Find the largest $\eta \le \eta_{\max}$ such that $f(\eta) \ge T_{MR}^L$. Such an η always exists. To find such an η, we can use a simple numerical method, such as binary search (this works because when η decreases, $f(\eta)$ increases exponentially fast).

- *Step 3*: Set $\delta = T_D^U - \eta$, and output η and δ.

Theorem 6 *Consider a system in which clocks are synchronized, and the probabilistic behavior of messages is known. Suppose we are given a set of QoS requirements as in (4.4). The above procedure has two possible outcomes: (1) It outputs η and δ. In this case, with parameters η and δ the failure detector NFD-S of Fig. 6 satisfies the given QoS requirements. (2) It outputs "QoS cannot be achieved". In this case,* no *failure detector can achieve the given QoS requirements.*

As an example of the configuration procedure of the failure detector, suppose we have the following QoS requirements: (a) a crash failure is detected within 30 seconds, i.e., $T_D^U = 30\,s$; (b) on average, the failure detector makes at most one mistake per month, i.e., $T_{MR}^L = 30 \text{ days} = 2\,592\,000\,s$; (c) on average, the failure detector corrects its mistakes within one minute, i.e. $T_M^U = 60\,s$. Assume that the message loss probability is $p_L = 0.01$, the distribution of message delay D is exponential, and the average message delay $E(D)$ is $0.02\,s$. By inputting these numbers into the configuration procedure, we get $\delta = 20.03\,s$ and $\eta = 9.97\,s$. With these parameters, our failure detector satisfies the given QoS requirements.

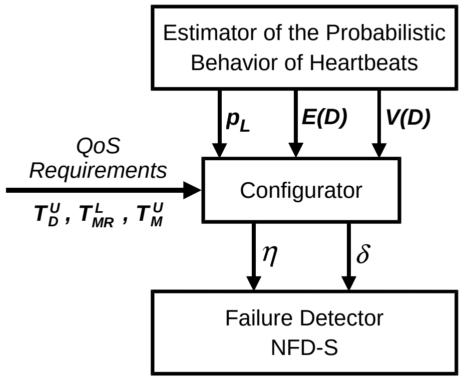

Figure 8: Meeting QoS requirements with NFD-S. The probabilistic behavior of heartbeats is not known, and clocks are synchronized

5 Dealing with Unknown Message Behavior

In Section 4, our procedure to compute the parameters η and δ of NFD-S to meet some QoS requirements assumed that one knows the probability p_L of message loss and the distribution $Pr(D \le x)$ of message delays. This assumption is not unrealistic, but in some systems the probabilistic behavior of messages may not be known. In that case, it is still possible to compute η and δ, as we now explain. We proceed in two steps: (1) we first show how to compute η and δ using only p_L, $E(D)$ and $V(D)$ (recall that $E(D)$ and $V(D)$ are the expected value and variance of message delays, respectively); (2) we then show how to estimate p_L, $E(D)$ and $V(D)$. In this section we still assume that local clocks are synchronized (we drop this assumption in the next section). See Fig. 8.

Computing Failure Detector Parameters η and δ Using p_L, $E(D)$ and $V(D)$. With $E(D)$ and $V(D)$, we can bound $Pr(D > t)$ using the following *One-Sided Inequality* of probability theory (e.g., see [4], p.79): For any random variable D with a finite expected value and a finite variance,

$$Pr(D > t) \le \frac{V(D)}{V(D) + (t - E(D))^2}, \text{ for all } t > E(D). \tag{5.9}$$

With this, we can derive the following bounds on the QoS metrics of algorithm NFD-S.

Theorem 7 *Consider a system with synchronized clocks and assume $\delta > E(D)$. For algorithm NFD-S, we have $E(T_{MR}) \ge \eta/\beta$ and $E(T_M) \le \eta/\gamma$, where*

$$\beta = \prod_{j=0}^{k_0} \frac{V(D) + p_L(\delta - E(D) - j\eta)^2}{V(D) + (\delta - E(D) - j\eta)^2}, \quad k_0 = \lceil(\delta - E(D))/\eta\rceil - 1,$$

and

$$\gamma = \frac{(1 - p_L)(\delta - E(D) + \eta)^2}{V(D) + (\delta - E(D) + \eta)^2}.$$

Note that in Theorem 7 we assume that $\delta > E(D)$, where δ is a parameter of NFD-S. This assumption is reasonable because if $\delta \leq E(D)$ then NFD-S would generate a false suspicion every time the heartbeat message is delayed by more than the average message delay. But then, NFD-S would make too many mistakes to be a useful failure detector.

Theorem 7 can be used to compute the parameters η and δ of the failure detector NFD-S, so that it satisfies the QoS requirements given in (4.4). Recall that these QoS requirements are given as a tuple (T_D^U, T_{MR}^L, T_M^U), where T_D^U is an upper bound on the worst-case detection time, T_{MR}^L is a lower bound on the average mistake recurrence time, and T_M^U is an upper bound on the average mistake duration. The configuration procedure is given below. This procedure assumes that $T_D^U > E(D)$, i.e., the required detection time is greater than the average message delay (a reasonable assumption).

- *Step 1*: Compute $\gamma' = (1 - p_L)(T_D^U - E(D))^2/(V(D) + (T_D^U - E(D))^2)$ and let $\eta_{\max} = \min(\gamma' T_M^U, T_D^U - E(D))$. If $\eta_{\max} = 0$, then output "QoS cannot be achieved" and stop; else continue.

- *Step 2*: Let
$$f(\eta) = \eta \cdot \prod_{j=1}^{\lceil (T_D^U - E(D))/\eta \rceil - 1} \frac{V(D) + (T_D^U - E(D) - j\eta)^2}{V(D) + p_L(T_D^U - E(D) - j\eta)^2}. \tag{5.10}$$
Find the largest $\eta \leq \eta_{\max}$ such that $f(\eta) \geq T_{MR}^L$. Such an η always exists.

- *Step 3*: Set $\delta = T_D^U - \eta$, and output η and δ.

Notice that the above procedure does not use the distribution $Pr(D \leq x)$ of message delays; it only uses p_L, $E(D)$ and $V(D)$.

Theorem 8 *Consider a system in which clocks are synchronized, and the probabilistic behavior of messages is not known. Suppose we are given a set of QoS requirements as in (4.4), and suppose $T_D^U > E(D)$. The above procedure has two possible outcomes: (1) It outputs η and δ. In this case, with parameters η and δ the failure detector NFD-S of Fig. 6 satisfies the given QoS requirements. (2) It outputs "QoS cannot be achieved". In this case,* no *failure detector can achieve the given QoS requirements.*

The above configuration procedure works when the distribution of the message delay D is not known (only $E(D)$ and $V(D)$ are known). To illustrate this procedure, we take the same example as in Section 4, except that we do not assume that the distribution of D is exponential. Specifically, suppose that the failure detector QoS requirements are that: (a) a crash failure is detected within 30 seconds, i.e., $T_D^U = 30\,s$; (b) on average, the failure detector makes at most one mistake per month, i.e., $T_{MR}^L = 30$ days $= 2\,592\,000\,s$; (c) on average, the failure detector corrects its mistakes within one minute, i.e. $T_M^U = 60\,s$. Assume that the message loss probability is $p_L = 0.01$, the average message delay $E(D)$ is $0.02\,s$, and the variance $V(D)$ is also 0.02. By inputting these numbers into the configuration procedure, we get $\delta = 20.29\,s$ and $\eta = 9.71\,s$. With these parameters, failure detector NFD-S satisfies the given QoS requirements. Note that when we go from the case that the distribution of D is known (example of Section 4) to the case that D is not known, η decreases from $9.97\,s$ to $9.71\,s$. This corresponds to a slight increase in the heartbeat sending rate (in order to achieve the same given QoS).

Estimating p_L, $E(D)$ and $V(D)$. It is easy to estimate p_L, $E(D)$ and $V(D)$ using heartbeat messages. For example, to estimate p_L, one can use the sequence numbers of the heartbeat messages to count the number of "missing" heartbeats, and then divide this count by the highest sequence number received so far. To estimate $E(D)$ and $V(D)$, we use the synchronized clocks as follows: When p sends a heartbeat m, p timestamps m with the sending time S, and when q receives m, q records the receipt time A. In this way, $A - S$ is the delay of m. We then compute the average and variance of $A - S$ for multiple past heartbeat messages, and thus obtain accurate estimates for $E(D)$ and $V(D)$.

6 Dealing with Unknown Message Behavior *and* Unsynchronized Clocks

So far, we assumed that the clocks of p and q are synchronized. More precisely, in the algorithm NFD-S of Fig. 6, q sets the freshness points τ_i's by shifting the sending times of heartbeats by a constant. When clocks are not synchronized, the local sending times of heartbeats at p cannot be used by q to set the τ_i's, and thus q needs to do it in a different way. The basic idea is that q sets the τ_i's by shifting the *expected arrival times* of the heartbeats, and q estimates the expected arrival times accurately (to compute these estimates, q does *not* need synchronized clocks).

6.1 NFD-U: an Algorithm that Uses Expected Arrival Times

We now present NFD-U, a new failure detector algorithm for systems with unsynchronized clocks. The new algorithm is very similar to NFD-S; the only difference is that q now sets the τ_i's by shifting the *expected arrival times* of the heartbeats, rather than the *sending times* of heartbeats. We assume that local clocks do not drift with respect to real time, i.e., they accurately measure time intervals (our algorithm and results can be easily generalized to the case where local clocks have small bounded drifts with respect to real time). Let σ_i denote the sending time of m_i with respect to q's local clock. Then, the expected arrival time of m_i at q is $EA_i = \sigma_i + E(D)$, where $E(D)$ is the expected message delay. Assume that q knows the EA_i's (we will soon show how q can accurately estimate them). To set the τ_i's, q shifts the EA_i's forward by α time units (i.e., $\tau_i = EA_i + \alpha$), where α is a new failure detector parameter that replaces δ. The intuition here is that EA_i is the time when m_i is expected to be received, and α is a slack added to EA_i to accommodate the possible extra delay or loss of m_i.

Figure 9 shows the whole algorithm, denoted by NFD-U. We restructured the algorithm a little, to

Process p: {using p's local clock}

1 for all $i \geq 1$, at time $i \cdot \eta$, send heartbeat m_i to q;

Process q: {using q's local clock}

2 Initialization:
3 $\quad \tau_0 = 0$;
4 $\quad \ell = -1$; $\quad$ {ℓ keeps the largest sequence number in all messages q received so far}

5 upon $\tau_{\ell+1} =$ the current time: $\quad$ {if the current time reaches $\tau_{\ell+1}$, then none of the messages received is still fresh}
6 $\quad output \leftarrow S$; $\quad$ {suspect p since no message received is still fresh at this time}

7 upon receive message m_j at time t:
8 $\quad$ **if** $j > \ell$ **then** $\quad$ {received a message with a higher sequence number}
9 $\quad\quad \ell \leftarrow j$;
10 $\quad\quad \tau_{\ell+1} \leftarrow EA_{\ell+1} + \alpha$; $\quad$ {set the next freshness point $\tau_{\ell+1}$ using the expected arrival time of $m_{\ell+1}$}
11 $\quad\quad$ **if** $t < \tau_{\ell+1}$ **then** $output \leftarrow T$; $\quad$ {trust p since m_ℓ is still fresh at time t}

Figure 9: Failure detector algorithm NFD-U with parameters η and α (clocks are not synchronized, but the EA_i's are known)

show explicitly when q uses the EA_i's. Variable ℓ keeps the largest heartbeat sequence number received so far, and $\tau_{\ell+1}$ refers to the "next" freshness point. Note that when q updates ℓ, it also changes $\tau_{\ell+1}$. If the local clock of q ever reaches time $\tau_{\ell+1}$ (an event which might never happen), then at this time none of the heartbeats received is still fresh, and so q starts suspecting p (lines 5–6). When q receives m_j, it checks whether this is a new heartbeat ($j > \ell$) and in this case, (1) q updates ℓ, (2) q sets the next freshness point $\tau_{\ell+1}$ to $EA_{\ell+1} + \alpha$, and (3) q trusts p if the current time is less than $\tau_{\ell+1}$ (lines 9–11). Note that this algorithm is identical to NFD-S, except in the way in which q sets the τ_i's. In particular, for any time t, let i be so that $t \in [\tau_i, \tau_{i+1})$; then with NFD-U q trusts p at time t if and only if q has received heartbeat m_i or higher.

6.2 Analysis and Configuration of NFD-U

NFD-U and NFD-S differ only in the way they set the τ_i's: in NFD-S, $\tau_i = \sigma_i + \delta$, while in NFD-U, $\tau_i = EA_i + \alpha = \sigma_i + E(D) + \alpha$ (the last equality holds because $EA_i = \sigma_i + E(D)$). Thus, the QoS analysis of NFD-U is obtained by simply replacing δ with $E(D) + \alpha$ in Proposition 3, Theorem 4 and Theorem 7.

To configure the parameters η and α of NFD-U to meet some QoS requirements, we use a method similar to the one in Section 5. We proceed in two steps: (1) we first show how to compute η and α using only p_L and $V(D)$ (note that $E(D)$ is not used); (2) we then show how to estimate p_L and $V(D)$. See Fig. 10.

Computing Failure Detector Parameters η and α using p_L and $V(D)$. By replacing δ with $E(D)+\alpha$ in Theorem 7, we obtain the following bounds on the accuracy metrics of NFD-U:

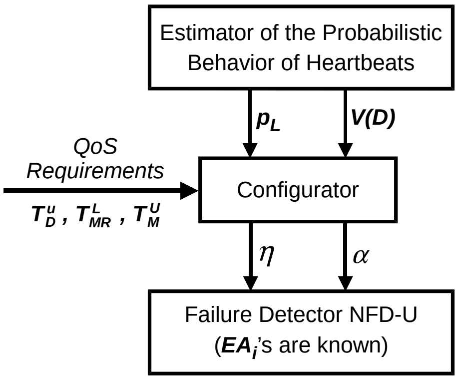

Figure 10: Meeting QoS requirements with NFD-U. The probabilistic behavior of heartbeats is not known; clocks are *not* synchronized, but they are drift-free

Theorem 9 *Consider a system with drift-free clocks and assume $\alpha > 0$. For algorithm NFD-U, we have $E(T_{MR}) \geq \eta/\beta$ and $E(T_M) \leq \eta/\gamma$, where*

$$\beta = \prod_{j=0}^{k_0} \frac{V(D) + p_L(\alpha - j\eta)^2}{V(D) + (\alpha - j\eta)^2}, \ \ k_0 = \lceil \alpha/\eta \rceil - 1, \text{ and } \gamma = \frac{(1-p_L)(\alpha+\eta)^2}{V(D) + (\alpha+\eta)^2}.$$

Note that the bounds given in Theorem 9 uses only p_L and $V(D)$; on the other hand, $E(D)$ is *not* used.

Theorem 9 can be used to compute the parameters η and α of the failure detector NFD-U, so that it satisfies some QoS requirements. We assume the QoS requirements are given as a tuple (T_D^u, T_{MR}^L, T_M^U) of positive numbers. The requirements are that:

$$T_D \leq T_D^u + E(D), \ E(T_{MR}) \geq T_{MR}^L, \ E(T_M) \leq T_M^U. \tag{6.11}$$

Note that the upper bound on the detection time T_D is *not* T_D^u, but T_D^u plus the *unknown* average message delay $E(D)$. So, the actual upper bound T_D^U on the detection time is $T_D^u + E(D)$. In other words, the QoS requirement on detection time is not absolute as in (4.4), but relative to $E(D)$. This is justified as follows. Note that when local clocks are not synchronized and only one-way messages are used, an absolute bound T_D^U on detection time cannot be enforced by any nontrivial failure detector. Moreover, it is reasonable to specify an upper bound requirement relative to the average delay $E(D)$ of a heartbeat. In fact, a failure detector that guarantees to detect crashes faster than $E(D)$ makes too many mistakes to be useful.

The following is the configuration procedure for algorithm NFD-U, modified from the one in Section 5.

- *Step 1*: Compute $\gamma' = (1-p_L)(T_D^u)^2/(V(D) + (T_D^u)^2)$ and let $\eta_{\max} = \min(\gamma' T_M^U, T_D^u)$. If $\eta_{\max} = 0$, then output "QoS cannot be achieved" and stop; else continue.

- *Step 2*: Let

$$f(\eta) = \eta \cdot \prod_{j=1}^{\lceil T_D^u/\eta \rceil - 1} \frac{V(D) + (T_D^u - j\eta)^2}{V(D) + p_L (T_D^u - j\eta)^2}. \tag{6.12}$$

  Find the largest $\eta \leq \eta_{\max}$ such that $f(\eta) \geq T_{MR}^L$. Such an η always exists.

- *Step 3*: Set $\alpha = T_D^u - \eta$, and output η and α.

Theorem 10 *Consider a system with unsynchronized, drift-free clocks, where the probabilistic behavior of messages is not known. Suppose we are given a set of QoS requirements as in (6.11). The above procedure has two possible outcomes: (1) It outputs η and α. In this case, with parameters η and α the failure detector NFD-U of Fig. 9 satisfies the given QoS requirements. (2) It outputs "QoS cannot be achieved". In this case,* no *failure detector can achieve the given QoS requirements.*

Estimating p_L and $V(D)$. When local clocks are not synchronized, we can estimate p_L and $V(D)$ using the procedure of Section 5. To estimate p_L, this procedure did not use clocks, and so it works just as before. For $V(D)$, the procedure did use clocks, but it works even though the clocks are not synchronized. To see why, recall that the procedure estimates $V(D)$ by computing the variance of $A - S$ of multiple heartbeat messages, where A is the time (with respect to q's local clock time) when q receives a message m, and S is the time (with respect to p's local clock time) when p sends m. When clocks are not synchronized, $A - S$ is not the actual delay of m, but rather the delay of m plus a *constant*, namely, the skew between the clocks of p and q. Thus the variance of $A - S$ is the same as the variance $V(D)$ of message delays.

6.3 NFD-E: an Algorithm that Uses *Estimates* of Expected Arrival Times

Failure detector NFD-U (Fig. 9) assumes that q knows the exact value of all the EA_i's (the expected arrival time of messages). In practice, q may not know such values, and needs to estimate them. To do so, every time q executes line 10 of algorithm NFD-U in Fig. 9, q considers the n most recent heartbeat messages (for some n), denoted $m'_1, \ldots, m'_n$. Let $s_1, \ldots, s_n$ be the sequence numbers of such messages and $A'_1, \ldots, A'_n$ be their receipt times according to q's local clock. Then $EA_{\ell+1}$ can be estimated by:

$$EA_{\ell+1} \approx \frac{1}{n}\left(\sum_{i=1}^{n} A'_i - \eta s_i\right) + (\ell+1)\eta. \tag{6.13}$$

Intuitively, this formula first "normalizes" each A'_i by shifting it backward in time by ηs_i. Then it computes the average of the normalized A'_is. Finally, it shifts forward the computed average by $(\ell+1)\eta$. It is easy to see that this is a good estimate of $EA_{\ell+1}$. We denote by NFD-E the algorithm obtained from Fig. 9 by replacing $EA_{\ell+1}$ with this estimate. Our simulations show that NFD-E and NFD-U are practically indistinguishable for values of n as low as 30. Thus, for large values of n, the configuration procedure for NFD-U can also be used to configure NFD-E. See Fig. 11.

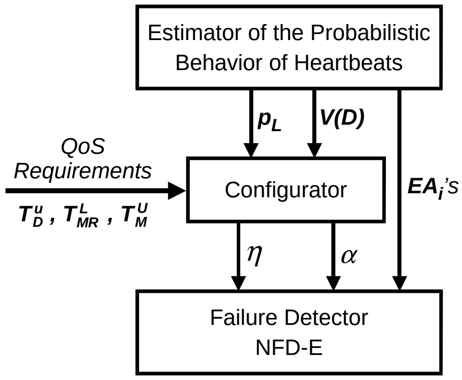

Figure 11: Meeting QoS requirements with NFD-E (same as with NFD-U, except that the expected arrival times EA_i's of heartbeats are estimated)

7 Simulation Results

We simulate both the new failure detector algorithm that we developed and the simple algorithm commonly used in practice (as described in Section 1.2). In particular, (a) we simulate the algorithm NFD-S (the one with synchronized clocks), and show that the simulation results validate our QoS analysis of NFD-S in Section 3.3; (b) we simulate the algorithm NFD-E (the one without synchronized clocks that estimates the expected arrival times), and show that it provides essentially the same QoS as NFD-S; and (c) we simulate the simple algorithm and compare it to the new algorithms NFD-S and NFD-E, and show that the new algorithms provide a much better accuracy than the simple algorithm.

The settings of the simulations are as follows. For the purpose of comparison, we normalize the intersending time η of heartbeat messages in both the new algorithm and the simple algorithm to 1. The message loss probability p_L is set to 0.01. The message delay D follows the exponential distribution (i.e., $Pr(D \leq x) = 1 - e^{-x/E(D)}$ for all $x \geq 0$). We choose the exponential distribution because of the following two reasons: first, it has the characteristic that a large portion of messages have fairly short delays while a small portion of messages have large delays, which is also the characteristic of message delays in many practical systems [14]; second, it has a simple analytical representation which allows us to compare the simulation results with the analytical results given in Theorem 4. The average message delay $E(D)$ is set to 0.02, which is a small value compared to the intersending time η. This corresponds to a system in which message delays are in the order of tens of milliseconds (typical for messages transmitted over the Internet), while heartbeat messages are sent every few seconds. Note that since D follows an exponential distribution, the standard deviation is $\sigma(D) = E(D) = 0.02$, and the variance is $V(D) = \sigma(D)^2 = 4 \times 10^{-4}$.

To compare the accuracy of different algorithms, we first set their parameters so that: (a) they send messages at the same rate (recall that $\eta = 1$), and (b) they satisfy the same bound T_D^U on the detection

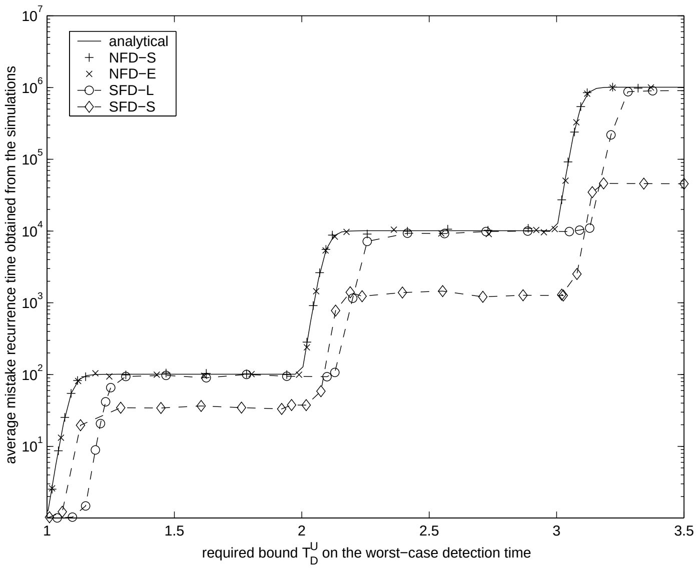

Figure 12: The average mistake recurrence times obtained by: (a) simulating the new algorithms NFD-S and NFD-E (shown by $+$ and $\times$), (b) simulating the simple algorithm (shown by -○- and -◇-), and (c) plotting the analytical formula for $E(T_{MR})$ of the new algorithm NFD-S (shown by —).

time. We simulated runs for values of T_D^U ranging from 1 to 3.5, and for each value of T_D^U, we measured the accuracy of the failure detectors in terms of the average mistake recurrence time $E(T_{MR})$ and the average mistake duration $E(T_M)$. For each value of T_D^U, we plotted $E(T_{MR})$ by considering a run with 500 mistake recurrence intervals, and computing the average length of these intervals. We do not show the plots for $E(T_M)$ because the $E(T_M)$ of all the algorithms were similar and bounded above by approximately $\eta = 1$.

7.1 Simulation Results of NFD-S and NFD-E

To ensure that NFD-S meets the given upper bound T_D^U on the detection time, we set δ to $T_D^U - \eta = T_D^U - 1$ (as prescribed by Theorem 4 (1)). In algorithm NFD-E, we choose to estimate the expected arrival time using the most recent 32 heartbeat messages. To ensure NFD-E meets the given upper bound T_D^U, we set $\alpha = T_D^U - E(D) - \eta = T_D^U - 1.02$.

In Fig. 12, we show the simulation results for algorithms NFD-S and NFD-E, together with the analytical formula of $E(T_{MR})$ derived in Section 3.3. These results show that: (a) the accuracy of algorithms NFD-S and NFD-E are very similar, and (b) the simulation results of both algorithms match the analytical formula for $E(T_{MR})$.

7.2 Simulation Results of the Simple Algorithm

The simple algorithm has no upper bounds on the detection time. However, such an upper bound can be guaranteed with a simple modification: the general idea is to discard heartbeats which have very large delays. More precisely, the modified algorithm has another parameter, the *cutoff time* c, such that all heartbeats delayed by more than c time units, called slow heartbeats, are discarded.[12] With this modification, the detection time T_D is at most $T_D \leq c + TO$.

Given a bound T_D^U on the detection time, there is a tradeoff in setting the cutoff time c and the timeout value TO: the larger the cutoff time c, the smaller the number of slow heartbeats being discarded, but the shorter the timeout value TO, and vice versa. In our simulations, we choose two cutoff times $c = 0.16$ and $c = 0.08$, i.e., 8 and 4 times the average message delay, respectively. The timeout TO is set to $T_D^U - c$. The algorithm with $c = 0.16$ is denoted by SFD-L, and the one with $c = 0.08$ is denoted by SFD-S.

The simulation results on the average mistake recurrence times of SFD-L and SFD-S (Fig. 12) show that the accuracy of the new algorithms (with or without synchronized clocks) is better — sometimes by an order of magnitude — than the accuracy of the simple algorithm. Intuitively, this is because the use of a cutoff time to bound the detection time in the simple algorithm is detrimental to its accuracy: if the simple algorithm uses a large cutoff time, then it must use a small timeout value, and this decreases the accuracy of the failure detector; if it uses a small cutoff time, then it discards more heartbeats, and this is equivalent to an increase in the message loss probability; this in turn also decreases the accuracy of the failure detector (a detailed explanation of the simulation results can be found in [13]).

8 Concluding Remarks

An Adaptive Failure Detector. In this paper, we assumed that the probabilistic behavior of heartbeat messages does not change. In some networks, this may not be the case. For instance, a corporate network may have one behavior during working hours (when the message traffic is high), and a completely different behavior during lunch time or at night (when the system is mostly idle): During peak hours, the heartbeat messages may have a higher loss rate, a higher expected delay, and a higher variance of delay, than during off-peak hours. Such networks require a failure detector that *adapts* to the changing conditions, i.e., it dynamically reconfigures itself to meet some given QoS requirements.

It turns out that our failure detectors can be made adaptive, as we now explain. For the case when clocks are synchronized, we make NFD-S adaptive by periodically reexecuting the configuration outlined in Fig. 8. The basic idea is to periodically run the estimator, which uses the n most recent heartbeats to estimate the *current* values of p_L, $E(D)$ and $V(D)$. These estimates are then fed into the configurator to recompute the new failure detector parameters η and δ.

[12]This assumes that the algorithm can detect slow messages; this is not easy when local clocks are not synchronized, but a *fail-aware datagram service* [18] may be used.

Similarly, when clocks are not synchronized, we can make NFD-E adaptive by periodically reexecuting the configuration outlined in Fig. 11. The only difference here is that the estimator also outputs EA_i — the estimated arrival time of the next heartbeat — which is input into the failure detector NFD-E.

The above adaptive algorithms form the core of a failure detection service that is currently being implemented and evaluated [15]. This service is intended to be shared among many different concurrent applications, each with a different set of QoS requirements. The failure detector in this architecture dynamically adapts itself not only to changes in the network condition, but also to changes in the current set of QoS demands (as new applications are started and old ones terminate).

Acknowledgments

We would like to thank Carole Delporte-Gallet, Hugues Fauconnier and anonymous referees of the conference version of this paper for their useful comments which helped us improve the paper.

References

[1] M. K. Aguilera, W. Chen, and S. Toueg. Using the heartbeat failure detector for quiescent reliable communication and consensus in partitionable networks. *Theoretical Computer Science*, 220(1):3–30, June 1999.

[2] M. K. Aguilera, W. Chen, and S. Toueg. Failure detection and consensus in the crash-recovery model. *Distributed Computing*, 13(2):99–125, Apr. 2000.

[3] M. K. Aguilera, W. Chen, and S. Toueg. On quiescent reliable communication. *SIAM Journal on Computing*, 29(6):2040–2073, Apr. 2000.

[4] A. O. Allen. *Probability, Statistics, and Queueing Theory with Computer Science Applications*. Academic Press, 2nd edition, 1990.

[5] Y. Amir, D. Dolev, S. Kramer, and D. Malkhi. Transis: a communication sub-system for high availability. In *Proceedings of the 22nd Annual International Symposium on Fault-Tolerant Computing*, pages 76–84, Boston, July 1992.

[6] K. Arvind. Probabilistic clock synchronization in distributed systems. *IEEE Transactions on Parallel and Distributed Systems*, 5(5):475–487, May 1994.

[7] O. Babaoğlu, R. Davoli, L.-A. Giachini, and M. G. Baker. Relacs: a communications infrastructure for constructing reliable applications in large-scale distributed systems, 1994. BROADCAST Project deliverable report, Department of Computing Science, University of Newcastle upon Tyne, UK.

[8] P. Billingsley. *Probability and Measure*. John Wiley & Sons, 3rd edition, 1995.

[9] K. P. Birman and R. van Renesse, editors. *Reliable Distributed Computing with the Isis Toolkit*. IEEE Computer Society Press, 1993.

[10] R. Braden, editor. *Requirements for Internet Hosts-Communication Layers*. RFC 1122, Oct. 1989.

[11] T. D. Chandra, V. Hadzilacos, S. Toueg, and B. Charron-Bost. On the impossibility of group membership. In *Proceedings of the 15th ACM Symposium on Principles of Distributed Computing*, pages 322–330, May 1996.

[12] T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267, Mar. 1996. A preliminary version appeared in *Proceedings of the 10th ACM Symposium on Principles of Distributed Computing*, Aug., 1991, 325–340.

[13] W. Chen. *On the Quality of Service of Failure Detectors*. PhD thesis, Cornell University, May 2000. available at www.cs.cornell.edu/home/weichen/research/mypapers/thesis.ps.gz.

[14] F. Cristian. Probabilistic clock synchronization. *Distributed Computing*, 3(3):146–158, 1989.

[15] B. Deianov and S. Toueg. Failure detector service for dependable computing (fast abstract). In *Proceedings of the 2000 International Conference on Dependable Systems and Networks*, pages B14–B15. IEEE Computer Society, June 2000.

[16] D. Dolev, R. Friedman, I. Keidar, and D. Malkhi. Failure detectors in omission failure environments. Technical Report 96-1608, Department of Computer Science, Cornell University, Ithaca, New York, Sept. 1996.

[17] C. Fetzer and F. Cristian. Fail-aware failure detectors. In *Proceedings of the 15th Symposium on Reliable Distributed Systems*, pages 200–209, Oct. 1996.

[18] C. Fetzer and F. Cristian. A fail-aware datagram service. In *2nd Annual Workshop on Fault-Tolerant Parallel and Distributed Systems*, Apr. 1997.

[19] M. G. Gouda and T. M. McGuire. Accelerated heartbeat protocols. In *Proceedings of the 18th International Conference on Distributed Computing Systems*, May 1998.

[20] R. Guerraoui, M. Larrea, and A. Schiper. Non blocking atomic commitment with an unreliable failure detector. In *Proceedings of the 14th IEEE Symposium on Reliable Distributed Systems*, pages 41–50, Sept. 1995.

[21] M. G. Hayden. *The Ensemble System*. PhD thesis, Department of Computer Science, Cornell University, Jan. 1998.

[22] L. E. Moser, P. M. Melliar-Smith, D. A. Argarwal, R. K. Budhia, and C. A. Lingley-Papadopoulos. Totem: A fault-tolerant multicast group communication system. *Commun. ACM*, 39(4):54–63, Apr. 1996.

[23] G. F. Pfister. *In Search of Clusters*. Prentice-Hall, Inc., 2nd edition, 1998.

[24] M. Raynal and F. Tronel. Group membership failure detection: a simple protocol and its probabilistic analysis. *Distributed Systems Engineering Journal*, 6(3):95–102, 1999.

[25] S. M. Ross. *Stochastic Processes*. John Wiley & Sons, 1983.

[26] K. Sigman. *Stationary Marked Point Processes, an Intuitive Approach*. Chapman & Hall, 1995.

[27] R. van Renesse, K. P. Birman, and S. Maffeis. Horus: a flexible group communication system. *Commun. ACM*, 39(4):76–83, Apr. 1996.

[28] R. van Renesse, Y. Minsky, and M. Hayden. A gossip-style failure detection service. In *Proceedings of Middleware'98*, Sept. 1998.

[29] P. Veríssimo and M. Raynal. Time, clocks and temporal order. In S. Krakowiak and S. K. Shrivastava, editors, *Recent Advances in Distributed Systems*, chapter 1. Springer-Verlag, 2000. to appear.

Appendix

A Main Symbols Used in the Paper

System model

p_L	message loss probability	Section 3.1, page 11
D	message delay (random variable)	Section 3.1, page 11

Primary QoS metrics

T_D	detection time (random variable)	Section 2.2, page 8
T_{MR}	mistake recurrence time (random variable)	Section 2.2, page 8
T_M	mistake duration (random variable)	Section 2.2, page 8

Derived QoS metrics

λ_M	average mistake rate	Section 2.3, page 9
P_A	query accuracy probability	Section 2.3, page 9
T_G	good period duration (random variable)	Section 2.3, page 9
T_{FG}	forward good period duration (random variable)	Section 2.3, page 9

QoS requirements

T_D^U	upper bound on detection time	Section 4, page 15
T_{MR}^L	lower bound on average mistake recurrence time	Section 4, page 15
T_M^U	upper bound on average mistake duration	Section 4, page 15
T_D^u	upper bound on detection time relative to the average message delay E(D)	Section 6.2, page 21

Failure detector algorithms

NFD-S	algorithm with synchronized clocks	Section 3.2, page 12
NFD-U	algorithm with unsynchronized clocks and known expected arrival times of heartbeats	Section 6.1, page 19
NFD-E	algorithm with unsynchronized clocks and estimates of expected arrival times of heartbeats	Section 6.3, page 22

Variables of failure detector algorithms

σ_i	sending time of the i-th heartbeat at p	Section 3.2, page 11
τ_i	time of the i-th freshness point at q	Section 3.2, page 11
EA_i	expected arrival time of the i-th heartbeat at q (in algorithm NFD-U)	Section 6.1, page 19

Parameters of failure detector algorithms

η	intersending time between two consecutive heartbeats	Section 3.2, page 11
δ	shift of freshness point w.r.t. heartbeat sending time, i.e., $\tau_i = \sigma_i + \delta$	Section 3.2, page 11
α	shift of freshness point w.r.t. heartbeat expected arrival time, i.e., $\tau_i = \sigma_i + E(D) + \alpha$	Section 6.1, page 19

B Proof of Theorem 4

Some of the following lemmata and propositions are easy and thus their proofs are omitted here.

Proof of Lemma 2. Fix an $i \geq 0$ and a time $t \in [\tau_i, \tau_{i+1})$. Suppose first that q has received some message m_j with $j \geq i$ by time t. Let $t' \leq t$ be the time when m_j is received. Choose i' such that $t' \in [\tau_{i'}, \tau_{i'+1})$. Thus $i' \leq i \leq j$. According to line 6 of the algorithm, q trusts p at time t'. For every τ_ℓ in the period $(t', t]$, since m_j is received at t' and $\ell \leq i \leq j$, the output of the failure detector does not change to S, according to lines 3–4. Therefore, q trusts p at time t.

Suppose now that q has not received any message m_j with $j \geq i$ by time t. Then at time τ_i, q suspects p according to lines 3–4. During the period $(\tau_i, t]$, since no message m_j with $j \geq i$ is received, the output of the failure detector does not change to T. So q suspects p at time t. □

We now analyze the accuracy metrics of the algorithm NFD-S, and to do so we assume that p does not crash.

Proposition 11 *(1) An S-transition can only occur at time τ_i for some $i \geq 2$, and it occurs at τ_i if and only if message m_{i-1} is received by q before time τ_i and no message m_j with $j \geq i$ is received by q by time τ_i; (2) Lemma 2 remains true if $j \geq i$ in the statement is replaced by $i \leq j \leq i + k$; (3) part (1) above remains true if $j \geq i$ in the statement is replaced by $i \leq j < i + k$.*

Note that part (1) of the above proposition guarantees that during any bounded time period, there is only a finite number of transitions of failure detector output.

Proposition 12 $u(0) \geq p_0^k$*, and for all* $x \in [0, \eta)$, $u(0) \geq u(x)$.

Lemma 13 *(1) If $p_0 = 0$, then with probability one q trusts p forever after time τ_1; (2) If $q_0 = 0$, then with probability one q suspects p forever; (3) If $p_0 > 0$ and $q_0 > 0$, then with probability one the failure detector at q has an infinite number of transitions.*

The above lemma factors out the special case in which $p_0 = 0$ or $q_0 = 0$. We call this special case the *degenerated case*. From now on, we only consider the *nondegenerated case* in which $p_0 > 0$ and $q_0 > 0$, and only consider runs in which the output of the failure detector has an infinite number of transitions.

Lemma 14 $P_{_A} = 1 - \frac{1}{\eta}\int_0^\eta u(x)\,dx$.
Proof. For all $i \geq 1$, let P_i be the probability that at any random time $T \in [\tau_i, \tau_{i+1})$, q is suspecting p. Note that T is uniformly distributed on $[\tau_i, \tau_{i+1})$ with density $1/(\tau_{i+1} - \tau_i) = 1/\eta$. Thus

$$P_i \;=\; \frac{1}{\eta}\int_{\tau_i}^{\tau_{i+1}} u(x - \tau_i)\,dx = \frac{1}{\eta}\int_0^\eta u(x)\,dx.$$

Note that the value of P_i does not depend on i. Let this value be P. Thus we have that $P_{_A}$, the probability that q trusts p at a random time, is $1 - P$. This shows the lemma. □

We now analyze the average mistake recurrence time $E(T_{_{MR}})$ of the failure detector. We will show that

Lemma 15 $E(T_{_{MR}}) = \eta/p_{_S}$.

If, at each time point τ_i with $i \geq 2$, the test of whether an S-transition occurs were an independent Bernoulli trial, then the above result would be very easy to obtain: $p_{_S}$ is the probability of success in one Bernoulli trial, i.e. an S-transition occurs at τ_i, and η is the time between two Bernoulli trials, and so $\eta/p_{_S}$ is the expected time between two successful Bernoulli trials, which is just the expected time between two S-transitions. Unfortunately, this is not the case because the tests of whether S-transitions occur at τ_i's are not independent. In fact, by Proposition 11, the event that an S-transition occurs at τ_i dependents on the behavior of messages $m_i, \ldots, m_{i+k-1}$. Thus two such events may depend on the behavior of common messages, and so they are not independent in general.

To deal with this, we use some results in *renewal theory*, a branch in the theory of stochastic processes. Besides proving Lemma 15, the analysis also reveals an important property of the failure detector output: each recurrence interval between two consecutive S-transitions is independent of other recurrence intervals.

The analysis proceeds as follows. We first introduce the concept of a *renewal process*. A more formal account can be found in any standard textbook on stochastic processes (see for example Chapter 3 of [25]). Let $\{(T_n, R_n), n = 1, 2, \ldots\}$ be a sequence of random variable pairs such that (1) a nonnegative T_n denotes the time between the $(n-1)$-th and n-th occurrences of some recurrent event A, i.e., event A occurs at time $t_1 = T_1, t_2 = T_1 + T_2, t_3 = T_1 + T_2 + T_3, \ldots$; and (2) R_n can be interpreted as the reward associated with the n-th occurrence of event A. A *delayed renewal reward process* is such a sequence satisfying: (1) The pairs $(T_n, R_n), n \geq 1$ are mutually independent; and (2) The pairs $(T_n, R_n), n \geq 2$ are identically distributed. If $\{R_n\}$ is omitted, then the above process $\{T_n, n \geq 1\}$ is called a *delayed renewal process*. Such processes are well studied in the literature, and are known to have some nice properties.

Now consider S-transitions of the failure detector output as the recurrent events. Let $T_{_{MR,n}}$ be the random variable representing the time that elapses from the $(n-1)$-th S-transition to the n-th S-transition (as a convention consider time 0 to be the time at which the 0-th S-transition occurs). Let $T_{_{M,n}}$ be the random variable representing the time that elapses from the $(n-1)$-th S-transition to the n-th T-transition. Thus $T_{_{M,n}} \leq T_{_{MR,n}}$ for all $n \geq 1$.

Lemma 16 $\{(T_{MR,n}, T_{M,n}), n = 1, 2, \ldots\}$ *is a delayed renewal reward process.*

We omit the proof of the lemma here since it is technical and lengthy. It can be found in [13].

It is immediate from the above lemma that for any $n \geq 2$, the joint probability distribution of $(T_{MR,n}, T_{M,n})$ is identical to that of (T_{MR}, T_M). This provides a simple way to analyze the QoS metrics T_{MR}, T_M and T_G. Moreover, any delayed renewal reward process is ergodic (see for example Section 2.6 of [26]), so Theorem 1 is applicable to our failure detector.

Proof of Lemma 15. For all $i \geq 2$, let A_i be the event that an S-transition occurs at time τ_i. By definition and Proposition 12, we have that $Pr(A_i) = p_S = q_0 \cdot u(0) \geq q_0 \cdot p_0^k$. Since in the nondegenerated case $q_0 > 0$ and $p_0 > 0$, we have $Pr(A_i) > 0$. By Proposition 11 (3), A_i is also the event that m_{i-1} is received before time τ_i but no message m_j with $i \leq j < i + k$ is received by time τ_i. This implies that A_i only depends on messages m_j with $i - 1 \leq j < i + k$, which in turn implies that for every $j \in \{2, \ldots, k + 2\}$, events $A_{i(k+1)+j}, i \geq 0$ are independent.

For $j \in \{2, \ldots, k + 2\}$, let B_j be the set of time points $\{\tau_{i(k+1)+j} : i = 0, 1, \ldots\}$. Obvious $B_j, j \geq 0$ is a partition of all time points $\tau_i, i \geq 2$. Let $N_j(t)$ be the random variable representing the number of S-transitions that occur at times in B_j by time t. Let $N(t)$ be the random variable representing the number of S-transitions by time t. Thus $N(t) = \sum_{j=2}^{k+2} N_j(t)$.

Consider $N_j(t)$ for some $j \in \{2, \ldots, k + 2\}$. For $t \geq \tau_j$, the number of time points in B_j that are no greater than t is $\lfloor (t - \tau_j)/((k + 1)\eta) \rfloor + 1$. From the above, we know that at each of these time points, there is an *independent* probability of p_S that an S-transition occurs. Therefore, the average number of S-transitions at these time points by time $t \geq \tau_j$ is given by

$$E(N_j(t)) = p_S \left(\left\lfloor \frac{t - \tau_j}{(k + 1)\eta} \right\rfloor + 1 \right).$$

Hence, we have for $t \geq \tau_{k+2}$,

$$E(N(t)) = \sum_{j=2}^{k+2} p_S \left(\left\lfloor \frac{t - \tau_j}{(k + 1)\eta} \right\rfloor + 1 \right).$$

By Lemma 16, $\{T_{MR,n}, n \geq 1\}$ is a delayed renewal process. Then by the Elementary Renewal Theorem (see for example [25] p.61),

$$\begin{aligned} E(T_{MR}) &= \lim_{t\to\infty} \frac{t}{E(N(t))} = \lim_{t\to\infty} \frac{t}{\sum_{j=2}^{k+2} p_S \left(\left\lfloor \frac{t-\tau_j}{(k+1)\eta} \right\rfloor + 1 \right)} \\ &= \frac{1}{\sum_{j=2}^{k+2} p_S \lim_{t\to\infty} \left(\left\lfloor \frac{t-\tau_j}{(k+1)\eta} \right\rfloor + 1 \right)/t} = \frac{1}{\sum_{j=2}^{k+2} \frac{p_S}{(k+1)\eta}} = \frac{\eta}{p_S}. \end{aligned}$$

□

By Lemma 15, we know that $0 < E(T_{MR}) < \infty$. Then we can apply Theorem 1 to obtain results on other metrics by our results on P_A and $E(T_{MR})$.

The above is the analysis on the accuracy metrics of the new failure detector. We now give the bound on the detection time T_D.

Lemma 17 $T_D \le \delta + \eta$. *Moreover, the inequality is tight when* $q_0 > 0$*, and* T_D *is always 0 when* $q_0 = 0$.

Proof. Suppose that process p crashes at time t. Let m_i be the last heartbeat message sent by p before p crashes. By definition, m_i is sent at time σ_i, and $\sigma_i \le t$. Since no messages with sequence number greater than i are sent by p, q does not receive these messages. Thus by Lemma 2, for all $t' \in [\tau_{i+1}, \infty)$, q suspects p at time t'. So the detection time is at most $\tau_{i+1} - t = \sigma_i + \delta + \eta - t \le \delta + \eta$.

When $q_0 > 0$, with nonzero probability m_i is received before τ_{i+1} and thus q trusts p just before τ_{i+1}.[13] In these cases, the detection time is $\sigma_i + \delta + \eta - t$. Since the time t when p crashes can be arbitrarily close to σ_i, we have that the bound $\delta + \eta$ is tight. When $q_0 = 0$, similar to Lemma 13 (2), we can see that in runs in which p crashes q also suspects p forever. Therefore T_D is always 0. □

Proof of Theorem 4. Parts (1) and (2) of the theorem are direct from Lemmata 17 and 15. Part (3) is derived from the relation between $E(T_M)$, P_A and $E(T_{MR})$ as given in part (2) of Theorem 1, and the results on P_A and $E(T_{MR})$ as given in Lemmata 14 and 15.

C Proof of Theorem 5

A *message delay pattern* P_D is a sequence $\{d_1, d_2, d_3, \ldots\}$ with $d_i \in (0, \infty]$ representing the delay time of the i-th message sent by p; $d_i = \infty$ means that the i-th message is lost. The distribution of message delay patterns are governed by the message loss probability p_L and the message delay time D, and thus it is the same for all algorithms in $\mathcal{C}$.

We first consider a subclass $\mathcal{C}'$ of $\mathcal{C}$ such that for any algorithm $A \in \mathcal{C}'$, in any run of A process p sends messages to q at times $\eta, 2\eta, 3\eta, \ldots$, just as in A^*. For any algorithm in $\mathcal{C}'$, a message delay pattern completely determines the time and the order at which q receives messages in failure-free runs. For A^*, this means that a message delay pattern uniquely determines a failure-free run of A^*. For some other algorithm $A \in \mathcal{C}'$, if A is nondeterministic, then A may have different failure-free runs with the same message delay pattern.

Lemma 18 *Given any message delay pattern* P_D*, let* r^* *be the failure-free run of* A^* *with* P_D*, and let* r *be a failure-free run of some algorithm* $A \in \mathcal{C}'$ *with* P_D*. Then for every time* $t \ge T_D^U$*, if* q *suspects* p *at time* t *in run* r^**, then* q *suspects* p *at time* t *in run* r.

Proof. Suppose that in run r^* of A^*, q suspects p at time $t \ge T_D^U$. Note that $T_D^U = \eta + \delta = \tau_1$, so $t \ge \tau_1$. Suppose $t \in [\tau_i, \tau_{i+1})$ for some $i \ge 1$. By Lemma 2, in run r^* q does not receive any message m_j with $j \ge i$ by time t. Since in run r p sends messages at the same times as in run r^*, and both runs have the same message delay pattern P_D, then in run r, by time t q does not receive any message sent by p at time $j\eta$ with $j \ge i$.

[13] Even though q_0 is defined with respect to runs in which p does not crash, it also applies to runs in which p crashes, since the system behavior before p crashes is independent of if and when p crashes in the future.

Consider first that $t \in (\tau_i, \tau_{i+1})$. Suppose for a contradiction that q trusts p at time t in run r. Let $\epsilon = t - \tau_i$, and let $t' = (i-1)\eta + \epsilon/2$. Thus $\epsilon > 0$. We now construct a new run r' of A as follows: (a) p crashes at time t'; (b) before t', p sends the same messages at the same times as in run r; this is possible because p's state before its crash is independent of its crash in the future; (c) the delays and losses of all messages sent before t' are the same as in run r; this is possible because the message loss and delay behaviors of messages sent by p are independent of p's crash. Note that for messages to be sent after time t', in run r none of them is received by q by time t, and in run r' they are not sent since p crashes at time t'. Therefore, in run r' up to time t, q receives the same messages at the same times as in run r. Thus q cannot distinguish run r' from r at time t, and so q trusts p at time t in run r' as in run r. The detection time in run r', however, is at least $t - t' = (\tau_i + \epsilon) - ((i-1)\eta + \epsilon/2) = \eta + \delta + \epsilon/2 = T_D^U + \epsilon/2 > T_D^U$, contradicting the assumption that A satisfies $T_D \le T_D^U$.

Now suppose $t = \tau_i$. Since the failure detector output is right continuous, there exists $\epsilon > 0$ such that q suspects p in the period $(t, t+\epsilon)$ in run r^*. Then by the above argument, q suspects p in the period $(t, t+\epsilon)$ in run r. By the right continuity again, we have that q suspects p at time t in run r. $\Box$

Corollary 19 *For any $A \in \mathcal{C}'$, let P_A be the query accuracy probability of A. Let P_A^* be the query accuracy probability of A^*. Then $P_A^* \ge P_A$.*

Proof (Sketch). We first fix a message delay pattern P_D. For the run r^* of A^* and any run r of A with message delay pattern P_D, Lemma 18 shows that for any time $t \ge T_D^U$, if q suspects p in r^* at time t, then q suspects p in r at time t. Thus, given a fixed message delay pattern P_D, at any random time t, the probability that q trusts p at time t in algorithm A^* is at least as high as the probability that q trusts p at time t in algorithm A. So $P_A^* \ge P_A$ given a fixed message delay pattern P_D. When summing (or integrating) both sides of the inequality over all message delay patterns according to their distribution, we have $P_A^* \ge P_A$. $\Box$

The above corollary shows that the new algorithm A^* has the best query accuracy probability in $\mathcal{C}'$, the class of failure detector algorithms in which p sends messages at exactly the same times as in A^*. We now generalize this result to class $\mathcal{C}$, where p still sends messages every η time units, but it may do so at times different from those in A^*.

A *message sending pattern* P_S is a sequence of time points $\{\sigma_1, \sigma_2, \sigma_3, \ldots\}$ at which p sends messages. The message sending pattern is determined by the algorithm. For any algorithm $A \in \mathcal{C}$, its message sending pattern is in the form $\{s, s+\eta, s+2\eta, \ldots\}$ for some $s \in [0, \infty)$. Different runs of algorithm A may have different message sending patterns due to the possible nondeterminism of A. Let A_s^* be the algorithm in which p sends heartbeat messages according to the sending pattern $\{s, s+\eta, s+2\eta, \ldots\}$, and q behaves the same way as in A^*. Thus A_s^* is a shifted version of A^*, and so the behavior of the failure detector output in A_s^* is also a shifted version of that of A^*. Since the behaviors of the two failure detectors only differ in some initial period, their steady state behaviors are the same. Therefore the QoS metrics of A_s^* and A^* are the same. In particular, they have the same query accuracy probability.

Proof of Theorem 5 (Sketch). We first fix a message sending pattern $P_S = \{s, s+\eta, s+2\eta, \ldots\}$. For any algorithm $A \in \mathcal{C}$, consider the runs of A with the sending pattern P_S. In these runs p sends messages at exactly the same times as in algorithm A_s^*. By the similar argument as in Lemma 18 and Corollary 19, we can show that the query accuracy probability of A_s^* is at least as high as the query accuracy probability of A given the message sending pattern P_S. Since A_s^* and A^* have the same query accuracy probability, we have $P_{_A}^* \geq P_{_A}$ given the message sending pattern P_S. Since P_S is arbitrary, we thus have $P_{_A}^* \geq P_{_A}$. □

D Proof of Theorem 6

Proposition 20 *If $p_0 > 0$ and $q_0 > 0$ (the nondegenerated case), then $E(T_{_M}) \leq \eta/q_0$.*

Proof. By Proposition 12, we have for all $x \in [0, \eta)$, $u(0) \geq u(x)$. Thus by equality (3.3), we have

$$E(T_{_M}) = \frac{\int_0^\eta u(x)\,dx}{p_{_S}} \leq \frac{\int_0^\eta u(0)\,dx}{q_0 u(0)} = \frac{\eta}{q_0}.$$

□

Proof of Theorem 6. We prove the theorem in the following three parts.

(1) Suppose that the procedure outputs parameters η and δ. Then by step 3 we have $T_{_D}^U = \eta + \delta$. By part (1) of Theorem 4, $T_{_D} \leq T_{_D}^U$ is satisfied. By step 1 and Proposition 3, $q_0' = (1 - p_{_L})Pr(D < \eta + \delta) = q_0$ (note that $q_0 > 0$: otherwise $\eta_{\max} = 0$ and the procedure cannot find $\eta \geq \Delta$). Consider first that $p_0 > 0$. Then by Proposition 20, $E(T_{_M}) \leq \eta/q_0 \leq \eta_{\max}/q_0 = q_0 T_{_M}^U/q_0 = T_{_M}^U$. So $E(T_{_M}) \leq T_{_M}^U$ is satisfied. Note that

$$\begin{aligned}
&\prod_{j=1}^{\lceil T_D^U/\eta \rceil - 1} [p_{_L} + (1 - p_{_L})Pr(D > T_{_D}^U - j\eta)] \\
= &\prod_{j=1}^{\lceil (\eta+\delta)/\eta \rceil - 1} [p_{_L} + (1 - p_{_L})Pr(D > \eta + \delta - j\eta)] \\
= &\prod_{j=0}^{\lceil \delta/\eta \rceil - 1} [p_{_L} + (1 - p_{_L})Pr(D > \delta - j\eta)] \\
= &\prod_{j=0}^{\lceil \delta/\eta \rceil} [p_{_L} + (1 - p_{_L})Pr(D > \delta - j\eta)] = u(0).
\end{aligned}$$

Thus $f(\eta) = \eta/(q_0 u(0)) = \eta/p_{_S} = E(T_{_{MR}})$, by equality (3.2). By step 2, $f(\eta) \geq T_{_{MR}}^L$, and so $E(T_{_{MR}}) \geq T_{_{MR}}^L$ is satisfied.

Consider now that $p_0 = 0$. By Theorem 4, in failure-free runs, the failure detector keeps trusting p after time τ_1, and so $E(T_{_{MR}}) = \infty$ and $E(T_{_M}) = 0$. Thus the requirements in (4.4) are also satisfied.

(2) Suppose that the procedure outputs "QoS cannot be achieved". Then the procedure stops at step 1, and thus $\eta_{\max} = 0$. This implies $q_0' = 0$ (since $T_{_M}^U > 0$), which in turn implies that either $p_{_L} = 1$ or

$Pr(D < T_D^U) = 0$. This means that, in such a system, no message is received within T_D^U time units after it is sent. Then to satisfy $T_D \leq T_D^U$, we claim that at any time $t > T_D^U$, any failure detector has to suspect p. In fact, since all messages q has received by time t are sent before time $t - T_D^U$, q does not obtain any information about whether p crashes at time $t - T_D^U$. Thus, to satisfy $T_D \leq T_D^U$, q has to suspect p at time t. Hence, for any failure detector, we have $E(T_M) = \infty$, and thus it fails to satisfy $E(T_M) \leq T_M^U$. Therefore, no failure detector can satisfy the given QoS in this case.

(3) We now show that the procedure only has two possible outcomes: it either outputs parameters η and δ, or outputs "QoS cannot be achieved". To show this, it is enough to show that if step 1 of the procedure succeeds, then step 2 always exceeds in finding an η such that $f(\eta) \geq T_{MR}^L$.

Let $r(x) = p_L + (1 - p_L)Pr(D > T_D^U - x)$, and $s(\eta) = \prod_{j=1}^{\lceil T_D^U/\eta \rceil - 1} r(j\eta)$. Thus $f(\eta) = \eta/(q_0' s(\eta))$. It is easy to see that for all x, $0 \leq r(x) \leq 1$, and for all $x_1 \leq x_2$, $r(x_1) \leq r(x_2)$.

We first claim that there exists an $\xi > 0$ such that $r(\xi) < 1$. Indeed, since $\eta_{\max} > 0$, we have $0 \leq p_L < 1$ and $Pr(D < T_D^U) > 0$. Thus there must be an $\xi > 0$ such that $Pr(D \leq T_D^U - \xi) > 0$. Otherwise, we would have that for all $\xi > 0$, $Pr(D \leq T_D^U - \xi) = 0$, and since the probability measure is continuous from below (see, e.g., Theorem 2.1 (i) of [8]), we would have $Pr(D < T_D^U) = 0$. Therefore, we have $r(\xi) = p_L + (1 - p_L)Pr(D > T_D^U - \xi) = p_L + (1 - p_L)(1 - Pr(D \leq T_D^U - \xi)) < 1$.

Let $\epsilon = 1 - r(\xi)$. Thus $0 < \epsilon \leq 1$. Let $\eta_1 = \min(\xi, \eta_{\max}, T_D^U/2)$. Let $\eta_i = \eta_1/i$ for $i = 1, 2, 3, \ldots$. We have

$$s(\eta_i) = \prod_{j=1}^{\lceil iT_D^U/\eta_1 \rceil - 1} r(j/i \cdot \eta_1) \leq \prod_{j=1}^{i} r(j/i \cdot \eta_1) \leq \prod_{j=1}^{i} r(\eta_1) \leq \prod_{j=1}^{i} r(\xi) = (1 - \epsilon)^i$$

Therefore, $f(\eta_i) = \eta_i/(q_0' s(\eta_i)) \geq \eta_1/(q_0' i(1 - \epsilon)^i) \to \infty$ when $i \to \infty$. Hence, there is always some η_i such that $f(\eta_i) \geq T_{MR}^L$. We also see that when i increases linearly (η_i decreases linearly), $f(\eta_i)$ increases exponentially. $\square$