Adaptare: Supporting automatic and dependable adaptation in dynamic environments

MÔNICA DIXIT, ANTÓNIO CASIMIRO and PAULO VERISSIMO
University of Lisboa
and
PAOLO LOLLINI and ANDREA BONDAVALLI
University of Firenze

Distributed protocols executing in uncertain environments, like the Internet or ambient computing systems, should dynamically adapt to environment changes in order to preserve Quality of Service (QoS). In earlier work, it was shown that QoS adaptation should be dependable, if correctness of protocol properties is to be maintained. More recently, some ideas concerning specific strategies and methodologies for improving QoS adaptation have been proposed. In this paper we describe *Adaptare* - a complete framework for dependable QoS adaptation. We assume that during its lifetime, a system alternates periods where its temporal behavior is well characterized, with transition periods during which a variation of the environment conditions occurs. Our method is based on the following: if the environment is generically characterized in analytical terms, and we can detect the alternation of these stable and transient phases, we can improve the effectiveness and dependability of QoS adaptation. To prove our point we provide detailed evaluation results of the proposed solutions. Our evaluation is based on synthetic data flows generated from probabilistic distributions, as well as on real data traces collected in various Internet-based environments. We compare our solution with other approaches and we show that *Adaptare* is more complex but is effective, allowing protocols to adapt to the available resources in a dependable way.

Categories and Subject Descriptors: C.0 [**System Architectures**]: ; C.4 [**Performance of Systems**]: —*Modeling techniques*; *Design studies*; *Performance attributes*; G.3 [**Probability and Statistics**]: —*Stochastic processes*

General Terms: Design, Reliability

Additional Key Words and Phrases: Adaptation, dependability, probabilistic analysis, quality of service

1. INTRODUCTION

Computer systems and applications are becoming increasingly distributed, for example, over the Internet. On the other hand, this scenario is complemented by an enormous growth of ambient computing systems, that is, complex embedded sys-

Author's email and address: M. Dixit, A. Casimiro, and P. Verissimo are with University of Lisboa. E-mails: mdixit@di.fc.ul.pt, casim@di.fc.ul.pt, pjv@di.fc.ul.pt.
P. Lollini, and A. Bondavalli are with University of Firenze. E-mails: lollini@unifi.it, bondavalli@unifi.it.

tems which, despite being networked, interact with the environment. The resulting pervasiveness and ubiquity of computing devices leads to complex environments (including networks and computational platforms) that tend to be unpredictable, essentially asynchronous, making it impractical, or even incorrect, to rely on aprioristic time-related bounds. On the other hand, in several application domains (e.g. home and factory automation, interactive services over the Internet, vehicular applications) there are strong requirements for timely operation and increased concerns with dependability assurance. Although hard real-time guarantees cannot be given in such settings, best-effort soft real-time guarantees will not be sufficient in some of the application domains mentioned above.

One possible way to cope with the uncertain timeliness of the environment while meeting dependability constraints, consists in ensuring that applications *adapt* to the available resources, *and* do that in a *dependable* way.

In this paper we build on earlier work that introduced the architectural and functional principles for dependable adaptation [Casimiro and Verissimo 2001]. In essence, the idea behind dependable adaptation lies in two very simple principles: (i) assumed bounds for fundamental variables (e.g. deadlines, timeouts) are not aprioristic constants, but instead adapted throughout the execution; (ii) the adaptation process ensures that, over time, those variables remain within bounds with a known and constant probability. For instance, consider an application or protocol that defines a timeout value based on the assumed message round-trip delay. This application will adapt the timeout during the execution with the objective of ensuring that the probability of receiving timely messages will stay close to some predefined value. Therefore, when message delays increase or decrease, the timeout will also increase or decrease in the exact measure of what is needed to ensure the desired stability of the probability value. In other words, this application will secure a *coverage stability* property.

However, achieving coverage stability is only possible if some limits are imposed (assumed) on how the environment behaves. For instance, if message delays can vary in some arbitrary fashion, then any observation or characterization of the environment will be useless, in the sense that nothing can be inferred with respect to the future behavior. Fortunately, this is not the usual case. There may be instantaneous or short-term variations that are unpredictable and impossible to characterize, but medium to long-term variations typically follow some pattern, allowing to probabilistically characterize the current operational state and derive the bounds that must be used for achieving coverage stability.

In earlier work [Casimiro et al. 2008], we firstly considered that during its lifetime a system alternates periods where its stochastic behavior is well characterized, with transition periods where a variation of the environment conditions occurs. The initial experiments and results were motivating and led to the development of the *Adaptare* framework. Therefore, in this paper we provide: a) a refined and detailed description of the theoretical framework for automatic and dependable adaptation; b) a detailed description of the probabilistic mechanisms that were considered and implemented; c) an extensive set of experimental results that we use to thoroughly evaluate, quantify and compare the benefits of the proposed approach and of the implemented mechanisms.

The *Adaptare* framework is based on the use of statistical formulations for the recognition and characterization of the “state” of the environment. This framework includes phase detection mechanisms, based on statistical goodness-of-fit tests, to perceive the stability of the environment behavior. Additional mechanisms to derive the actual parameters of the distributions are also employed. The paper explains how the chosen methods were implemented, which is relevant to show how dependability constraints are handled in practice.

A fundamental aspect of our work is that we are concerned with dependability objectives. Therefore, *Adaptare* was developed to take dependability-related criteria (the required coverage of an assumption) as input and provide information to support adaptation (the concrete bounds that must be assumed). Moreover, the benefits of the approach were also evaluated from a dependability perspective, measuring the effectively achieved coverage in addition to the improvements in terms of bounds produced by the framework.

The framework was evaluated by performing a number of simulation experiments based on synthetic data flows generated from well-known probabilistic distributions and on real round-trip time (RTT) traces collected in different environments. It was also compared to other approaches, using the real data traces. The results indicate that, in most of the considered environments, our framework was able to reduce the time bounds with respect to the remaining approaches and, at the same time, ensured the required coverage in all cases. Compared to a baseline stochastic (and conservative) approach, as originally proposed in [Casimiro and Verissimo 2001], *Adaptare* performed better in all scenarios, with improvements up to 25%. Based on these results we are able to conclude that *Adaptare* allows achieving dependable adaptation and improved time bounds.

We believe that these results are very important in practical systems. To illustrate this belief with concrete examples, our results are complemented with the description of two cases in which we successfully applied *Adaptare*. First, *Adaptare* was used for timeout-based Failure Detection 0**??**. By ensuring that selected timeouts are dynamically adapted and just as large as necessary to avoid false positives, the Quality of Service (QoS) of the failure detector was improved. The amount of mistakes (accuracy) was directly controlled by the requested coverage and the detection time (speed) was improved through adaptation and optimization of the timeouts. Given that failure detectors are a basic building block of most reliable distributed systems, such improvements can be of paramount relevance. In the second example, *Adaptare* was used to drive the dynamic selection of a timeout value used in a consensus protocol for wireless environments. In this case the the requested coverage was tuned in accordance to percentage of mistakes that would imply excessive retransmissions and negatively affect communication delays. With *Adaptare* it was possible to achieve clear improvements of the overall execution time, which are retained due to the automatic adaptation of the timeout when the number of participants or the network conditions change.

The paper is organized as follows. In the next section we provide a motivation for this work and we discuss related work. Then, Section 3 describes *Adaptare*, which is a framework for dependable adaptation in probabilistic environments. Implementation details are presented in Section 4, while the evaluation of *Adaptare* is discussed in Section 5. Application examples are provided in Section 6 and some conclusions and future perspectives are finally presented in Section 7.

2. MOTIVATION AND RELATED WORK

Current research in the field of adaptive real-time systems uses classical fault tolerance for dependability and QoS management as the workhorse for adaptation. In fault tolerance, the approach is normally based on fairly static assumptions on system structure and possible faults. The assumed system model is typically homogeneous and synchronous [Buttazzo 1997; Kopetz and Bauer 2003; Liu 2000] with a crash-stop fault model. However, this is not appropriate for the distributed and dynamic environments considered in this work, especially when we talk about complex networked embedded systems.

In these dynamic environments, monitoring is of fundamental importance. In fact, developing solutions to deal with problems associated to monitoring and analysing network traffic is an active research area [Tzagkarakis et al. 2008]. Advances in this area are important to support several properties of autonomous systems, like self-organization, self-adaptation, and other self-* properties (see [Babaoglu et al. 2005] for recent work in this area).

Providing QoS guarantees for the communication in spite of the uncertain or probabilistic nature of networks is a problem with a wide scope, which can be addressed from many different perspectives. In the specific field of multimedia applications, for example, significant efforts were made in the last years to develop solutions for QoS adaptation in dynamic best-effort environments such as the Internet, when resource reservation is not possible [Bhatti and Knight 1999; Koliver et al. 2002].

The solution proposed in [Bhatti and Knight 1999] analyzes a snapshot of the network conditions (defined by a set of measured QoS parameters) and defines which application modes are feasible on that context. Therefore, the application can switch to a better operation mode in order to maintain the expected QoS level. Despite the simplicity of the performed analysis, this approach requires the definition of several application-related parameters, such as the set of all possible operation modes of the target application and, for each mode, the accepted boundaries for each QoS parameter. Based on those boundaries and on the measured values (provided by the application or other component), it is possible to define the compatibility of each operation mode with the current network conditions.

In [Koliver et al. 2002], the authors present a more complex model, based on a fuzzy controller for dynamic QoS adaptation. In this approach, data streams are characterized by a set of QoS application parameters, which are mapped into a single quality measure, called quality degree. Based on the difference between the quality degree in the sender and the quality degree perceived by the receiver, the QoS parameters (e.g. bit rate) are adjusted to meet some expected QoS level, according to the fuzzy controller indications. As in [Bhatti and Knight 1999], this approach demands significant a priori configuration: designers must specify not only the parameters to be evaluated, but also their possible ranges, the set of QoS levels, and the mapping between them.

The works described above, although effective, are very application-dependent.

Rather than proposing ready-to-use solutions, they describe methodologies for designing dynamic adaptive multimedia applications. Moreover, those works do not assess the dynamics of the environment. No attempt is made in order to understand how the measured parameters behave, neither to predict any future conditions. In our approach we aim at ensuring a long term coverage of assumptions rather than aiming at maintaining, with no implied assurances, an average acceptable QoS level.

In fact, we are fundamentally concerned with timeliness issues and with improving the dependability of adaptive applications. Therefore, our proposition is for a generic framework, to be used as a plug-and-play solution for a wide range of applications, supporting adaptation while ensuring that the coverage of assumed bounds is satisfied, thus allowing systems to be dependable with respect to how they adapt.

In [Casimiro and Verissimo 2001], the fundamental architectural and functional principles for dependable QoS adaptation were introduced, providing relevant background for the work presented here. In this earlier work we also followed a dependability perspective, analyzing why systems would fail as a result of timing assumptions being violated, as it may happen in environments with weak synchrony. A relevant effect is *decreased coverage* of some time bound, when the number of timing failures goes beyond an assumed limit.

In systems in which time bounds must remain fixed, other properties or variables should be adapted to avoid the decreased coverage of these bounds. For example, the work presented in [Krishnamurthy et al. 2001] prevents the occurrence of timing faults in a replicated distributed service by dynamically selecting the replicas that can satisfy a client's timing requirement. Thus, the set of replicas which serves a client request is adapted according to the replicas responsiveness, in order to meet the client's timing constraints with at least the probability requested by the client, i.e., the required coverage.

Another possible approach to address the undesired effect of decreased coverage, followed in our work, is making the protocols or applications use adaptive time-related bounds, instead of fixed values as usual (communication timeouts, scheduling periods and deadlines, etc.). If done properly, this satisfies a so-called *coverage stability* property, introduced in [Verissimo and Casimiro 2002].

In more practical terms, this means that QoS is no longer expressed as a single value, a time bound to be satisfied, but as a $\langle bound, coverage\rangle$ pair, in which the coverage should remain constant while the bound may vary as a result of adaptation, to meet the conditions of the environment. On the other hand, deciding when and how to adapt depends on how the environment is assumed to behave and on the concrete approaches for monitoring this behavior.

Different assumptions and approaches for monitoring can be considered. The work presented in [Li et al. 1999] applies the optimal control and estimation theory in a QoS adaptation framework, for estimating current values of system parameters based on past measures. The estimated values are then used to determine how the application should be adapted. The main assumption made in this work states that system random disturbances and observation noises are uncorrelated white Gaussian-Markov sequences with zero mean. Relying on these assumption enables the use of a Kalman filter prediction algorithm. The main drawback of this algorithm is the inherent complexity involved in the required matrices inversions and multiplications, which is unavoidable. Comparatively, the framework we propose is of incremental complexity, since it is modular with respect to the number and variety of estimation mechanisms that are used, which can hence be added or removed depending on their usefulness for a given environment.

In [Casimiro and Verissimo 2001] the environment is simply assumed to behave stochastically, but no assumptions are made about the specific probabilistic distribution for communication delays. Therefore, this led to a simple but conservative solution (based on the one-sided inequality of probability theory [Allen 1990]) with respect to the bounds required to guarantee a specified coverage. A similar assumption is used in [Chen et al. 2002], in order to derive timeouts for the configuration of failure detectors.

Interestingly, over the last few years a number of works have addressed the problem of probabilistically characterizing the delays in IP-based networks using real measured data, allowing to conclude that empirically observed delay distributions may be characterized by well-known distributions, such as the Weibull distribution [Hernandez and Phillips 2006], the shifted gamma distribution [Piratla et al. 2004; Corlett et al. 2002], the exponential distribution [Markopoulou et al. 2006] or the truncated normal distribution [Elteto and Molnar 1999]. Based on this, we realized that it would be interesting and appropriate to consider less conservative approaches, by assuming that specific distributions may be identified, thus allowing to achieve better (tighter) time bounds for the same required coverage.

However, some of these works also recognize that probabilistic distributions may change over time (e.g. [Piratla et al. 2004]), depending on the load or other sporadic occurrences, like failures or route changes. Therefore, in order to secure the required dependability attributes, it becomes necessary to detect changes in the distribution and hence use mechanisms for being able to do that. Fortunately, there is also considerable work addressing this problem and well-known approaches and mechanisms (see [Yang et al. 2004] for a nice overview). Among others, we can find approaches based on time-exponentially weighted moving histograms [Menth et al. 2006], on the Kolmogorov-Smirnov test [Elteto and Molnar 1999] or the Mann-Kendall test [Chen et al. 2006]. For instance, in [Elteto and Molnar 1999] the authors apply the Kolmogorov-Smirnov test on RTT data (Round-Trip Time, which is defined as the time required for a message to be sent to some destination and a reply received back) to detect state changes in a delay process. Between these state changes, the process can be assumed to be stationary with constant delay distribution. The stationary assumption was confirmed by the trend analysis test.

All these possibilities motivated the design and development of an efficient and effective generic framework for automatic and dependable adaptation (*Adaptare*). *Adaptare* is a framework specifically characterized by: (i) modularly accommodating various mechanisms to analyze the environment conditions and dealing with several different probabilistic distributions; (ii) automatically determining the most suitable methods and the best fitting probabilistic distributions. Note that this combination allows achieving more accurate characterizations of the environment state, at a given time (i), and along the timeline (ii), a crucial result of our work.

Note also that due to its modularity, the framework can flexibly be extended or modified to incorporate other mechanisms and deal with additional probabilistic distribution. In this paper we consider and present a set of mechanisms that are representative enough and sufficient to illustrate the benefits of *Adaptare*.

3. *ADAPTARE* FRAMEWORK

3.1 Assumptions

As mentioned in the previous section, in this work we advance on previous results by making less conservative and more realistic assumptions about how the environment behaves, in order to achieve improved and still dependable $\langle bound, coverage\rangle$ pairs. Instead of assuming that the environment behaves stochastically but with unknown probabilistic distributions (as done in [Casimiro and Verissimo 2001]), we now make the following assumptions:

—**Interleaved stochastic behavior:** We assume that the environment alternates stable periods, during which it follows some specific probabilistic distribution, with unstable periods, during which the distribution is unknown or cannot be characterized. As discussed in Section 2, this assumption is supported by the results of many recent works (e.g., [Hernandez and Phillips 2006; Papagiannaki et al. 2003; Piratla et al. 2004]).

—**Sufficient stability:** We assume that the dynamics of environment changes is not arbitrarily fast, i.e., there is a minimum duration for stable periods before an unstable period occurs. This is a mandatory assumption for any application that needs to recognize the state of a dynamic environment.

Additionally, we also need to make assumptions concerning application-level behavior and the availability of resources to perform the needed computations (within the framework operation) in support of adaptation decisions.

—**Sufficient activity:** We assume that there is sufficient system activity, allowing enough samples of the stochastic variable under observation to be obtained, as required to feed the phase detection and probabilistic recognition mechanisms. For instance, if message round-trip durations are being observed, then there will be statistically sufficient and independent message transmissions to allow characterizing the state of the environment. Obviously, system activity depends on the application. We believe that this is an acceptable assumption in most practical interactive and reactive systems.

—**Resource availability:** The system has sufficient computational resources (processor, memory, etc), which are needed to execute all the detection and recognition mechanisms implemented in the dependable adaptation framework in a sufficiently fast manner.

Interestingly, it is easy to see that all these assumptions are somehow interdependent. As with any control framework, for a good quality of control it is necessary to ensure that the controller system is sufficiently fast with respect to the dynamics of the controlled system. In our case, the required resources and application activity (sample points) depend on the effective dynamics of the environment. A balance between these three aspects must exist so that it becomes possible to dependably adapt the application.

Fortunately, this is typically the case in the systems we want to address, which usually present intensive interaction patterns (with the ambient or between distributed components) in sufficiently stable environments, hence meeting the requirements implied by these assumptions. Actually, in this paper we go further and provide experimental evidence that this is so: the evaluation presented in Section 5 is aimed at raising evidence that these requirements indeed hold in practical settings. Using traces of real systems' executions we implicitly test the satisfaction of sufficient activity, sufficient stability and interleaved stochastic behavior assumptions. To reason about resource availability, we provide execution measurements in specific computational platforms and conduct a complexity analysis of the frameworks' algorithms.

3.2 Dependability goals

One fundamental objective of this work is to provide the means for adaptive applications to behave dependably despite temporal uncertainties in the operating environment. In these settings, the classical design approach of assuming fixed upper bounds for temporal variables, such as processing duration or communication delay, is typically not a good idea. Either these bounds are set very high to ensure that they are not violated during execution, but then this may have a negative impact on the system performance, or they are made smaller, but then timing faults will occur with potential negative impact on the system correctness.

We argue that in these environments, dependability must be equated through the ability of the system to secure some bounds while adapting to changing conditions. As explained in [Verissimo and Casimiro 2002], for dependable adaptation to be achieved it is sufficient to secure a *coverage stability* property. In concrete terms, this means that given a system property, the effective coverage of this property will stay close to some assumed coverage during the system execution, and that difference is bounded.

For example, a system property could be "The round-trip delay is bounded by T_{RT}". There is a probability that this property will hold during some observation interval (coverage of the assumed property). The coverage of this property will vary due to changes in the environment conditions (e.g., due to load variations over time). On the other hand, the application can adapt the bound T_{RT} to increase or decrease the coverage. The objective of dependable adaptation is to select the adequate bound T_{RT}, so that the effective coverage of the property will be bounded by some specified value. A practical approach, which is the one we adopt in this work, is to ensure that the observed coverage is always higher than the specified one, while the bounds for the random variables are as small as possible.

3.3 Environment recognition and adaptation

Adaptare is a framework for dependable (QoS) adaptation that can be seen as a service composed by two activities: identification of the current environment conditions and QoS adaptation.

—**Environment recognition:** The environment conditions can be inferred by analyzing a real-time data flow representing, for example, the end-to-end message delays in a network. When the analytical description of the data is not known, we need to determine the model that best describes the data. Using statistics, the data may be represented by a cumulative distribution function (CDF), defining the probability distribution of a real random variable X. We note that the data models can be so complex that they cannot be described in terms of simple well-know probabilistic distributions. In the case in which the model that describes the data is known, the problem is reduced to estimating unknown parameters of a known model from the available data.

—**QoS adaptation:** Once the best fitting distribution (together with its parameters) has been identified, its statistics properties can be exploited to find a pair $\langle bound, coverage\rangle$ that will satisfy the objective of keeping a constant coverage of the assumed bound throughout the execution. Compared to the pair $\langle bound, coverage\rangle$ that could be obtained with the method defined in [Casimiro and Verissimo 2001], the new pair is better, since the coverage stability objective can be reached using a lower time bound.

3.4 The adaptation approach

As detailed in Section 3.1, we are assuming an interleaved stochastic behavior of the environment, i.e., we can consider that the system alternates periods during which the conditions of the environment remain fixed (*stable phases*), with periods during which the environment conditions change (*transient phases*). During stable phases, the statistical process that generates the data flow (e.g., end-to-end message delays) is under control and we can compute the corresponding distribution using an appropriate number of samples. On the contrary, if the environment conditions are changing, then the associated statistical process is actually varying and no fixed distribution can describe its real behavior.

Therefore, the system lifetime can be seen as a sequence of alternated phases: a stable phase, during which the distribution is computable, and a transient phase, during which the distribution is changing, moving towards the distribution of the next stable phase, and cannot be computed.

The goal is to adopt a so called *conservative approach* during transient phases, by setting the pair $\langle bound, coverage\rangle$ using the one-sided inequality as in [Casimiro and Verissimo 2001]. It is a conservative bound, but it holds for all the distributions. As soon as the presence of a stable phase is detected, which is equivalent to being able to identify a proper probabilistic distribution, an improved (lower) bound can be computed according to the identified distribution, still ensuring the coverage stability property. In this case we follow a so called *improved approach* for selecting the time bound.

The mechanisms that are used to identify proper probabilistic distributions, and thus stable phases, are based on mathematical and statistical methods for the analysis of sample input data. We call them *phase detection mechanisms*, since they implicitly indicate if the environment is in a transient or in a stable phase. These mechanisms must be continuously executed, processing new incoming data, and producing indications of the actual state of the environment. Consequently, time bounds are also continuously produced, and continuously changing, even if the

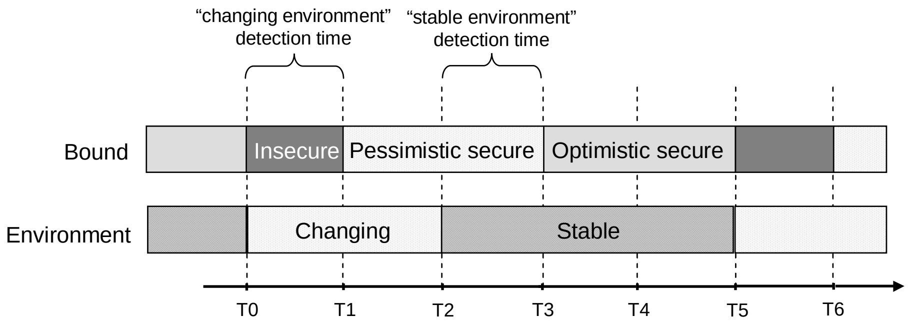

Fig. 1. *Adaptare* operation

environment remains in a stable phase. This directly results from small changes in distribution parameters, steaming from the continuous update of the set of sample values used in the analysis, which does not necessarily imply a phase change.

Figure 1 shows how *Adaptare* operates with the help of a typical example scenario, such as the one identified by the following temporal events:

—Before time T0 the environment is stable.

—At time T1 the phase detection mechanism detects the transient phase and the bound is set to the safe but conservative value as in [Casimiro and Verissimo 2001].

—At time T2 the environment reaches a new stable phase.

—At time T3 the phase detection mechanism starts identifying the stable phase and a new less conservative bound (tailored for the corresponding distribution) can be computed.

—At time T4 the environment conditions start changing again.

—At time T5 the phase detection mechanism detects that the environment is changing and the bound is set to a new safe but conservative value, and so on.

We note that there is an alternation of periods during which the bound is improved and safe (e.g. [T3;T4]), unsafe ([T0;T1]) and safe but conservative ([T1;T3]). Therefore, the effectiveness o *Adaptare* clearly depends on: (i) the "transient environment" detection time, which is the time that it takes to detect that the environment is changing; and (ii) the "stable environment" detection time, which is the time needed to detect that the environment has reached a new stable configuration.

The "transient environment" detection time is particularly important. During the unsafe periods, the environment is changing but the bound is tailored for a particular set of environment conditions that do not hold anymore. In other words, *the "transient environment" detection time has direct impact on the dependability of Adaptare.* In real environments it is very difficult, or even impossible, to a priori quantify the impact of detection time on the achievable dependability. However, it is possible to measure the achieved coverage for verifying if the requirements are

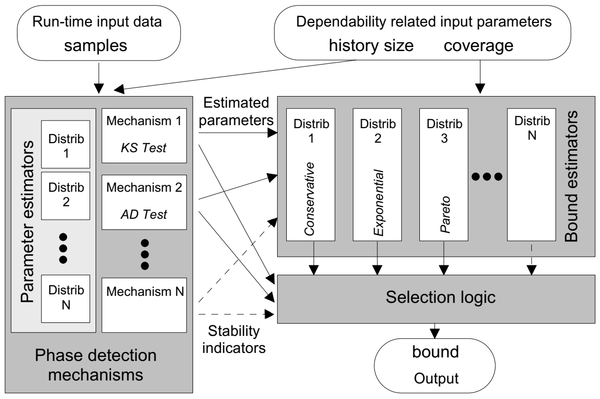

Fig. 2. Schematic view of *Adaptare*

met (which we do in our evaluation) and, provided that the stated assumptions are met, this impact will be negligible. Note that if the ratio between stability periods ([T2;T4]) and unsafe periods ([T0;T1]) is large enough, then the long term assurance of the coverage stability property can be guaranteed. Therefore, to ensure that the impact of unsafe periods becomes negligible, two conditions must be fulfilled:

—The system must be stable for sufficiently long periods of time. This is precisely what we state in the *sufficient stability* assumption (see Section 3.1) and therefore this may be considered as given by assumption.

—Then, the "transient environment" detection time must be just as small as needed to ensure the large ratio mentioned above. This is achieved both by construction (e.g., configuring the phase detection mechanisms to use the smallest possible number of input samples when identifying the state of the environment) and based on the assumption of *sufficient activity* (i.e., there will be enough observation points to ensure a fast detection).

Developing fast mechanisms is thus important to guarantee the maximum possible dependability. We address this issue in Section 5, where we evaluate the impact of the implemented mechanisms on the overall system dependability, which is the most important metric in the context of our work. Measuring actual detection times (for both transient or stable phases) can be regarded as a partial metric and, as mentioned above, is not feasible with the real traces that we have used in our evaluations.

3.5 *Adaptare* architecture

The scheme depicted in Figure 2 shows the architecture of *Adaptare*. It was modeled as a service that:

—Accepts the history size (i.e., the number of collected samples of the random variable under observation) and the required coverage, as dependability related parameters;

—Reads samples (measured delays) as input, using them to fill up the history buffer that is used by the phase detection mechanisms and for the estimation of distribution parameters;

—Provides, as output, a bound that should be used in order to achieve the specified coverage.

Internally, the service admits the use of several phase detection mechanisms, as well as several bound estimators, corresponding to different probabilistic distributions. This openness allows designers to extend the framework with different phase detection mechanisms, as well as probabilistic distributions, which are more suitable for their applications. Because of these possible multiple phase detection mechanisms, several bounds may be selectable as candidate to the output value. Therefore, a selection logic must be implemented to choose only one of the available bounds. The following section addresses the implementation of all these internal mechanisms.

4. *ADAPTARE* IMPLEMENTATION

4.1 Phase detection mechanisms

We implemented two goodness-of-fit (GoF) tests as phase detection mechanisms: the Kolmogorov-Smirnov (KS) test and the Anderson-Darling (AD) test. GoF tests are formal statistical procedures used to assess the underlying distribution of a data set. A stable period with distribution $\hat{D}$ is detected when some GoF tests establish the goodness of fit between the postulated distribution $\hat{D}$ and the evidence contained in the experimental observations [Trivedi 2002]. Both AD and KS are distance tests based on the comparison of the cumulative distribution function (CDF) of the assumed distribution $\hat{D}$ and the empirical distribution function (EDF), which is a CDF built from the input samples. If the assumed distribution is correct, the assumed CDF closely follows the empirical CDF.

The phase detection mechanism based on the KS test [Trivedi 2002] performs the following steps:

(1) Given a sample of size n, order the sample points to satisfy $x_1 \leq x_2 \leq ... \leq x_n$;

(2) Build the empirical distribution function $\hat{F}_n(x)$, for each $x \in \{x_1 \leq x_2 \leq ... \leq x_n\}$:

$$\hat{F}_n(x) = \frac{\text{number of values in the history that are } \leq x}{n}$$

(3) Assume a distribution F with CDF $F_0(x)$ (the parameters of the postulated distribution F are previously estimated using the methods described in Section 4.2);

(4) Compute the KS statistic D_n:

$$D_n = \max_x |\hat{F}_n(x) - F_0(x)|$$

(5) If $D_n \leq d_{n;\alpha}$, the KS test accepts that the sample points follow the assumed distribution F with significance level α, and a stable phase is detected. Otherwise, a transient phase is detected. The significance level defines the probability that a stable phase is wrongly recognized as a transient one. The value of $d_{n;\alpha}$ is obtained from a published table of KS critical values.

The algorithm of the phase detection mechanism that implements the AD test [Stephens 1974] executes the following steps:

(1) Given a sample of size n, order the sample points to satisfy $x_1 \leq x_2 \leq ... \leq x_n$;
(2) Assume a distribution F with CDF $F_0(x)$ (the parameters of the postulated distribution are previously estimated using the methods described in Section 4.2);
(3) Compute the AD statistic A^2:

$$A^2 = -n - S$$

where:

$$S = \sum_{i=1}^{n} \frac{(2i-1)}{n} [\ln F_0(x_i) + \ln(1 - F_0(x_{n+1-i}))]$$

(4) If $A^2 \leq a_{n,\alpha}$, the sample points follow the assumed distribution F with significance level α, and a stable phase is detected. Otherwise, a transient phase is detected. The value of $a_{n,\alpha}$ is obtained from tables of AD critical values.

Both algorithms described above assume a probabilistic distribution F and verify if the given sample points come from this distribution. We are considering five distributions to be tested: exponential, shifted exponential, Pareto, Weibull and uniform distributions. The set of distributions was defined based on other works that address the statistical characterization of network delays, e.g. [Papagiannaki et al. 2003; Hernandez and Phillips 2006; Markopoulou et al. 2006; Bolot 1993; Downey 2001; Tickoo and Sikdar 2004]. However, it is important to note that the framework can be extended with more distributions, depending on the characteristics of the random variable which will be analyzed. If the phase detection mechanisms do not identify a stable phase with any of the tested distributions, they assume that the environment is changing, i.e., a transient phase is detected.

The main advantage of the KS test in comparison to the AD test is that the critical values for the KS statistic are independent of specific distributions: there is a single table of critical values, which is valid for all distributions. However, the test has some limitations: it tends to be more sensitive near the center of the distribution than at the tails, and if the distribution parameters must be estimated from the data to be tested (which is what we do in our implementation), the results can be affected [Jain 1991]. Due to this limitation, there are some works that propose different KS critical values to specific distributions with unknown parameters, which are estimated from the sample. In our experiments with the KS test we used these modified tables for the exponential and shifted exponential [Trivedi 2002], Pareto [Porter et al. 1992] and Weibull [Evans et al. 1989] distributions. Regarding the Pareto distribution, the table of critical values presented in [Porter et al. 1992] is limited to a set of shape parameters: 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, and 4.0. Thus, if the estimated shape parameter is not in the interval $[0.5, 4.0]$, the standard KS table [Trivedi 2002] is used. Otherwise, the estimated parameter is rounded for the closest defined value and modified table is applied. The test for the uniform distribution is also performed using the standard KS table, since we did not find a modified table for this distribution in the current literature.

The major limitation of the KS test is solved when using the AD test: distribution parameters estimated from the sample points do not compromise the results. However, the AD test is only available for a few specific distributions, since the critical values depend on the assumed distribution and there are published tables for only a limited number of distributions. AD critical values for the Weibull distribution can be found in [Stephens 1976]. For the exponential and shifted exponential distributions, we implemented the modified statistic and used the critical values proposed in [Stephens 1974]. Like in the KS test, AD critical values for the Pareto distribution are limited to shape parameters in the interval $[0.5, 4.0]$. Considering that there is no general table of AD critical values (independent of the tested distribution), if the estimated shape parameter is not in this interval, the AD phase detection mechanism does not test the Pareto distribution. Finally, critical values of AD test for the uniform distribution are presented in [Rahman et al. 2006].

For a given input sample, it is possible that a phase detection mechanism identifies more than one distribution, due to similarities between distributions, and uncertainty of the statistical methods and parameters estimation. However, each mechanism must return only one distribution when a stable phase is detected. Thus, in those cases both mechanisms return the detected distribution with the lowest statistic value, which means that the sample data are closer to that distribution.

4.2 Parameters estimation

Both KS and AD tests need to estimate distribution parameters in order to execute their statistical tests. There are various methods, both numerical and graphical, for estimating the parameters of a probability distribution. From a statistical point of view, the method of maximum likelihood estimation (MLE) is considered to be one of the most robust techniques for parameter estimation.

The principle of the MLE method is to select as an estimation of a parameter θ the value for which the observed sample is most "likely" to occur [Trivedi 2002; Balakrishnan and Basu 1995]. We applied this method to estimate exponential, shifted exponential, Pareto and uniform parameters. For the Weibull distribution, the MLE method produces equations that are impossible to solve in closed form: they must be simultaneously solved using iterative algorithms, which have the disadvantage of being very time-consuming. Since execution time is an important factor in our framework, we decided to estimate Weibull parameters through linear regression, instead of using MLE. We implemented the method of least squares, which requires that a straight line be fit to a set of data points, minimizing the sum of the squares of the y-coordinate deviations from it [Trivedi 2002].

Table I presents the equations for parameters estimation, where $\{t_1, t_2, ..., t_n\}$ is the *ordered* sample history. Regarding the linear regression to estimate Weibull parameters, consider that $x_i = \ln(t_i)$, $y_i = \ln(-\ln(1 - F(t_i)))$, and $F(x_i) = (i - 0.3)/(n + 0.4)$ (approximation of the median rankings). These equations are derived from the Weibull CDF, using the method of regression on Y. Due to its complexity, we will not present the derivation process in this document, but a complete explanation can be found in [ReliaSoft 2006].

Table I. Equations for parameters estimation

Distribution	**CDF**	**Parameters estimators**
Exponential	$F_X(t) = 1 - e^{-\lambda t}$	$\hat{\lambda} = \frac{1}{\bar{t}}$
Pareto	$F_X(t) = 1 - (\frac{k}{t})^{\lambda}$	$\hat{k} = t_{min}$ $\hat{\alpha} = \frac{n}{\sum_{i=1}^{n} \ln \frac{t_i}{\hat{k}}}$
Shifted Exponential	$F_X(t) = 1 - e^{\frac{(-t+\gamma)}{\lambda}}$	$\hat{\lambda} = n\frac{(\bar{t}-t_{min})}{n-1}$ $\hat{\gamma} = t_{min} - \frac{\hat{\gamma}}{n}$
Weibull	$F_X(t) = 1 - e^{-(\frac{t}{\lambda})^{\gamma}}$	$\hat{\gamma} = \frac{\sum_{i=1}^{n} x_i y_i - \frac{\sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n}}{\sum_{i=1}^{n} x_i^2 - \frac{(\sum_{i=1}^{n} x_i)^2}{n}}$ $\hat{\alpha} = e^{-\frac{\bar{y}-\hat{\gamma}\bar{t}}{\hat{\gamma}}}$
Uniform	$F_X(t) = \frac{t-b}{b-a}$	$\hat{a} = t_{min}$ $\hat{b} = t_{max}$

4.3 Bound estimators

Depending on the output of the phase detection mechanisms, which consists both in stability indicators and in estimated parameters that characterize a detected distribution, one of the bounds computed by the implemented bound estimators will be selected as the *Adaptare* output. The current implementation has six bound estimators: the conservative one, which is based on the one-sided inequality of probability theory, providing a conservative bound which holds for all probabilistic distributions; and estimators for the exponential, shifted exponential, Pareto, Weibull and uniform distributions.

The bound estimators are derived from the distributions' CDF. We recall that the CDF represents the probability that the random variable X takes on a value less than or equal to x (for every real number x):

$$F_X(x) = P(X \leq x)$$

Our objective is to ensure that a given bound is safe, i.e., that the real delay will be less than or equal to the assumed bound, with a certain probability (the expected coverage). Thus, in the CDF function:

—X is the observed delay;

—x is the assumed bound, provided by the framework;

—$F_X(x) = C$, the expected coverage.

For example, the exponential CDF is:

$$F_X(x) = 1 - e^{-\lambda x} = C$$

For a given coverage C, a safe bound can be computed by isolating x:

Table II. Bound estimators for a required coverage C

Estimator	Minimum time bound t
Conservative	$t = E(D) + \sqrt{\frac{V(D)}{1-C} - V(D)}$
Exponential	$t = \frac{1}{\lambda} \ln \frac{1}{1-C}$
Shifted Exponential	$t = \gamma + \lambda \ln \frac{1}{1-C}$
Pareto	$t = \frac{k}{\sqrt[\alpha]{1-C}}$
Weibull	$t = \lambda \sqrt[\gamma]{\ln \frac{1}{1-C}}$
Uniform	$t = C(b-a) + a$

$$x = \frac{1}{\lambda} \ln \frac{1}{1-C}$$

The same logic was applied to define bound estimators for the other four distributions. The six estimators are presented in Table II.

4.4 Selection Logic

As explained in Section 4.1, the phase detection mechanisms are individually executed and each one indicates the environment condition (transient or stable period) and returns one distribution which best characterizes the analyzed history trace whenever a stable phase is detected. The selection logic receives these results and is responsible for selecting one of them as the output of the framework (see Figure 2). Since our objective is to produce improved safe bounds, *Adaptare* returns the lowest bound to the client application.

5. RESULTS AND EVALUATION

The main objective of our work is to show the possibility of having a component which is able to characterize the current state of the environment and, based on dependability objectives of the application, infer how dependability-related bounds should be adapted.

In the previous sections we have introduced *Adaptare*, a framework for QoS adaptation, describing its objectives, functionalities, implemented methods, and algorithms. This section presents a set of results obtained with this framework, and a systematic evaluation focusing on the following main points:

—Validation of our implementation and demonstration of its correctness by comparing the ability of both mechanisms (based on AD and KS tests, respectively) to characterize the current conditions of the environment using controlled traces, synthetically produced;

—Quantification of *Adaptare*'s achievements by determining the real effectiveness and improvements obtained, using real RTT (round-trip time) traces;

—Quantification of *Adaptare*'s overhead through a complexity analysis of the implemented algorithms and effective latencies using typical computing platforms.

Table III. Parameters used to generate the synthetic data traces

Traces	Exponential (λ)	Shifted Exp (λ, γ)	Pareto (λ, k)	Weibull (λ, γ)	Uniform (a, b)
1	0.8	0.8, 100	1.0, 1.0	1.0, 2.0	10, 20
2	1.6	1.6, 200	1.4, 2.0	1.4, 2.5	20, 40
3	2.4	2.4, 300	1.8, 3.0	1.8, 3.0	30, 60
4	3.2	3.2, 400	2.2, 4.0	2.2, 3.5	50, 100
5	4.0	4.0, 500	2.5, 5.0	2.5, 4.0	70, 140
6	4.8	4.8, 600	2.8, 6.0	2.8, 4.5	80, 160
7	5.6	5.6, 700	3.1, 7.0	3.1, 5.0	100, 200
8	6.4	6.4, 800	3.4, 8.0	3.4, 5.5	200, 400
9	7.2	7.2, 900	3.7, 9.0	3.7, 6.0	300, 600
10	8.0	8.0, 1000	4.0, 10.0	4.0, 6.5	500, 1000

—Performance comparison of *Adaptare* to other well-known solutions for the problem of dynamically adapting time-related variables.

5.1 Analysis of the phase detection mechanisms

In this first part of our evaluation, we tested the our framework using synthetic data traces in order to analyze the functioning of the AD and KS phase detection mechanisms. We generated 10 traces of 3000 samples for each one of the five considered distributions (exponential, shifted exponential, Weibull, Pareto and uniform). The distributions parameters were configured according to Table III.

Adaptare was executed for each trace using the phase detection mechanisms individually. For these executions, we defined the minimum required coverage $C = 98\%$, and history size $h = 30$ sample points. According to the experiments with real traces presented in Section 5.2, this history size ($h = 30$) produces the best overall results. Moreover, both phase detections mechanisms have a parameter called *significance level* (α), which defines the probability of not detecting a stable phase. In our experiments we set $\alpha = 0.05$. We specified four metrics to compare the AD and KS goodness-of-fit tests:

—*Stability detection:* percentage of the sample points that were detected as stable phases.

—*Detection correctness:* percentage of the sample points characterized as stable phases in which the framework detected the correct distribution. Given that in this phase of the evaluation we used synthetic traces generated from known distributions, we are able to quantify the correctness of the mechanisms.

—*Improvement of bounds*: improvement obtained with the adaptive approach in comparison with the conservative one. It represents the percentage of reduction of the conservative bounds, obtained when using the adaptive bounds.

—*Coverage*: achieved coverage, which corresponds to the number of measured samples less than or equal to the computed bound, divided by the total number of samples.

The above metrics can be applied when using synthetic traces generated from stable and fixed distributions. On the other hand, other metrics that could have been considered, like "stable environment" and "transient environment" detection

Table IV. Comparing phase detection mechanisms using synthetic data traces

Distribution	Stability det.		Det. correct.		Improvement		Coverage	
	AD	KS	AD	KS	AD	KS	AD	KS
Exponential	94.53	93.53	98.64	99.19	20.16	19.90	99.58	99.59
Shifted exp.	98.98	99.07	99.18	63.04	1.98	2.12	99.10	98.96
Pareto	76.26	81.75	72.96	78.41	23.62	27.20	98.33	98.25
Weibull	98.43	97.86	79.88	92.63	32.57	35.35	99.41	99.34
Uniform	55.57	48.39	94.82	70.58	9.60	7.50	100.00	100.00

times, would have required synthetic traces with both stable and transient periods. Although it would have been possible to create several synthetic scenarios with varied dynamics, it would not be able to cover all the cases and the evaluation would always be partial. Furthermore, it was enough to perform a basic comparison of the goodness-of-fit tests in steady situations to conclude that there is not a single best goodness-of-fit test for all distributions. This is clear from the average results in Table IV. Actually, the performance of each mechanism depends on its intrinsic features and on the distribution characteristics. Besides, the definition of the best mechanism is related to the evaluation criteria to be verified.

Regarding the considered metrics, we should note that the *detection correctness* metric can only be used with synthetic traces (since we know the actual distribution). Because of this, only the other three metrics are used in the evaluation with real traces, presented in Section 5.2.

The most important result to be observed in these values is that the two main objectives of *Adaptare* were achieved for all distributions by both mechanisms: the bounds were improved (up to 35%) and the minimum required coverage (98%) was secured. In order to fully understand the values presented in Table IV and verify the correctness of the mechanisms to characterize the environment, it is necessary to perform a more detailed analysis, separated by distribution.

In the experiments with exponential traces, both mechanisms reached excellent and very similar results: they detected stability in more than 90% of the sample points, and correctly characterized almost all of these stable points as exponentially distributed. This significant rate of exponential detection allowed a reduction of approximately 20% in the bounds computed by our framework (comparing to the conservative bounds).

Regarding the shifted exponential distribution, almost all points were detected as belonging to stable phases by the two mechanisms. However, the KS mechanism made incorrect characterizations (recognizing other distributions instead of the shifted exponential) in more than 35% of these points. There are at least two factors that can lead to these mistakes: the inherent uncertainty associated with parameters estimation and the probabilistic mechanisms, and the similarities between the tested distributions (e.g. the shifted exponential distribution is an exponential distribution with an extra location parameter, and every exponential distribution is also a Weibull distribution with $\lambda = 1$), which is interesting in order to verify the precision of the implemented mechanisms, but also increases the possibility of inaccurate detection. Despite of all these uncertainties, the characterization was correct in the majority of the sample points. Another important observation in the shifted exponential results is that while there was a high rate of stability detection, the improvement of bounds was not significant (2%), showing that the bounds produced by *Adaptare* to the shifted exponential distribution are very close to the conservative bounds.

The tests with Pareto traces presented the lowest rate of correct detection among all distributions. This is due to the technical limitation explained in Section 4.1: both mechanisms have Pareto critical values only for a small set of shape parameters. Thus, besides the uncertainty already present in the parameter estimation, this estimator is rounded to one of these pre-defined values, compromising the accuracy of the characterization. However, even with this limitation, the improvement of bounds for the Pareto traces was very significant (about 25% in average).

Both mechanisms presented very good results in the experiments with Weibull traces. Excellent rate of stability detection (more than 95%), good distribution characterization (80% - 90%) and the best rate of improvement of bounds (up to 35%). These are the best overall results among the tested distributions.

Finally, the results of the tests using uniform traces show that the stability detection for this distribution is the more conservative: around 50%. The AD test correctly characterized these sample points as uniformly distributed. We did not obtain the same results with the KS test, which is a good example of the KS limitation: estimating parameters from the sample is not adequate when applying the standard test. As described in Section 4.1, we did not find a modified table of KS critical values for testing uniformity, so we decided to use the general KS table. Consequently, this test failed in detecting the uniform distribution in approximately 30% of the stable points. Nevertheless, both mechanisms obtained almost 10% of improvement in the produced bounds.

Detecting wrong distributions may lead to lower coverage, compromising the dependability of *Adaptare*. For example, we observed that the bounds produced with the shifted exponential distribution are very close to the conservative bounds, while the bounds produced when detecting a Weibull distribution are more aggressive, with improvements of at least 30%. Thus, if a trace that follows a shifted exponential distribution is wrongly characterized as a Weibull trace, the produced bounds will be lower than the measured samples, decreasing the coverage. However, our results show that despite the inherent errors in the characterization (due to similarities between distributions, and statistical uncertainties on the parameter estimation), *Adaptare* secures the expected coverage, i.e., the rate of mistakes is low enough to not compromise the dependability of our approach.

From these results, we conclude that *it is possible to define effective mechanisms to detect stable and transient phases and, for the stable ones, correctly characterize the observed probabilistic distribution.*

5.2 Validation using real RTT measurements

This section presents a set of experiments based on real data, in order to prove that our assumption about the interleaved stochastic behavior is realistic for network delays in different environments and demonstrate that *Adaptare* is able to characterize these behaviors and provide applications with improved and dependable time bounds. We selected data traces freely available in the Internet, composed by measurements collected in different wired and wireless networks, and extracted RTT traces from them using the *tcptrace* tool. These RTT traces have been provided as input to *Adaptare* (they constitute the "input samples" of Figure 2). Data traces were gathered from the following sources:

—**Inmotion:** Traceset of TCP transfers between a car traveling at speeds from 5 mph to 75 mph, and an 802.11b access point. The experiments were performed on a traffic free road in the California desert. For our tests, we selected traces from experiments simulating FTP file transfers. A complete description of the environment, hardware and software used to generate this traceset can be found in [Gass et al. 2006]. We downloaded the tcpdump files from [Gass et al. 2005].

—**Umass:** A collection of wireless traces from the University of Puerto Rico. Contains wireless signal strength measurements for Dell and Thinkpad laptops. Tests were performed over distances of 500 feet and one mile. For more information, see [UMass Trace Repository 2006].

—**Dartmouth:** Dataset including tcpdump data for 5 years or more, for over 450 access points and several thousand users at Dartmouth College. We downloaded for our tests a traceset composed by packet headers from wireless packets sniffed in 18 buildings on Dartmouth campus [Kotz et al. 2004]. Experiments details are discussed in [Henderson et al. 2004].

—**LBNL:** Packet traces from two internal network locations at the Lawrence Berkeley National Laboratory (LBNL) in the USA, giving an overview of internal enterprise traffic recorded at a medium-sized site. The packet traces span more than 100 hours, over which activity from a total of several thousand internal hosts appears. These traces are available to download in [Paxson et al. 2007].

—**RON:** Traces containing thousands of latency and loss samples taken on the RON testbed. A RON (Resilient Overlay Network) is an application-layer overlay on top of the existing Internet routing substrate, which has the ability to take advantage of network paths to improve the loss rate, latency, or throughput perceived by data transfers. More information can be found in [Andersen et al. 2001]. The dataset is available in [RON 2001].

For the tests with real traces, the phase detection mechanisms were executed in parallel, as illustrated in Figure 2. The most important parameter that must be defined by the framework's clients is the history size. This parameter states the number of recently collected sample points that are analyzed in the environment characterization, exhibiting a high impact on the results. For this reason, we conduced a set of experiments with different history sizes for all traces. For these experiments we set the required coverage to $C = 0.98$, and selected four traces from each data source, with different sizes (10000 to 50000 sample points).

The average results for each data source are presented in Figures 3, 4, and 5. Figure 3 shows the average stability detection. Each sample in a trace is characterized by *Adaptare* as either stable or unstable, thus the percentage of stable points in the total of samples determines the stability detection for a given trace. Figure 4 shows the improvement of bounds, which quantifies the reduction in the time bounds achieved when using *Adaptare*, in comparison to the base line conservative solution proposed in [Casimiro and Verissimo 2001]. The average coverage is presented in Figure 5. Each sample in the trace is matched against the bound assumed at that moment, and if the RTT is higher than the bound, a timing fault is

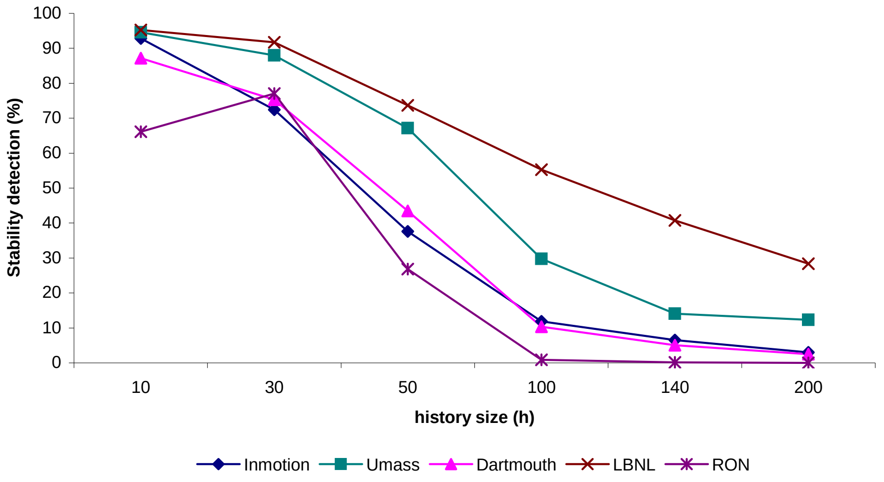

Fig. 3. Stability detection using different history sizes

accounted. The coverage of a given trace is defined by the proportion of non-faulty samples (those that do not lead to timing faults), over the total number of samples.

Average values are representative enough in our evaluation because stability detection and coverage are measures of the entire trace, not individual samples. To achieve a more consolidated view, the presented results are averaged over the four values obtained for each of the traces of the given source, which were very similar. Within each trace we observed significant differences between minimum and maximum bounds. However, and for all traces, our experiments shown that a confidence interval of 95% for the computed time bounds deviates from the average at most by 2%, making the average a proper indicative of the overall results.

The effects of increasing the history size are perceptible on the obtained values: Figure 3 shows that the larger the history size is, the lower is the rate of detected stability points. Noting that the capacity of correctly identifying stable phases is dependent on the correlation between the sample points (they need to be collected within a sufficiently small interval of time), this result highlights the interdependence between the "sufficient stability" and the "sufficient activity" assumptions. If there is no sufficient activity (a requirement exacerbated by using large histories), then the environment is required to be stable for longer periods (otherwise it appears as unstable more easily).

Intuitively, what happens is that we are dealing with traces from real environments, subject to uncertain statistical processes, making it increasingly difficult to fit a large set of samples into one specific distribution. Taking as given that data correlation is good enough for all the considered history sizes (i.e., both the "sufficient stability" and the "sufficient activity" assumptions hold), the inability of detecting stable phases when using higher history sizes is explained by the fact that in real environments (as considered in this work) the stochastic behavior is not "pure", i.e. it does not strictly follow one well-defined probability distribution, but

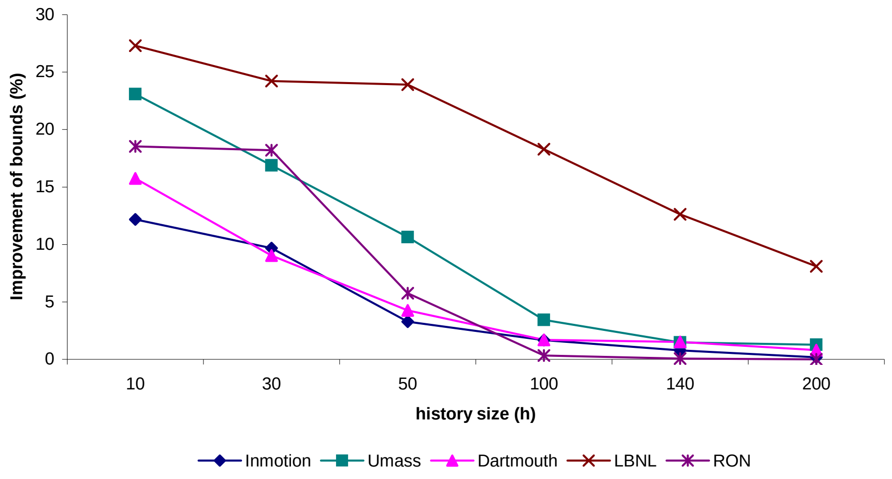

Fig. 4. Improvement of bounds using different history sizes

approximates some distribution(s). Therefore, fitting a big history into one specific distribution becomes impossible, which does not mean the environment is not stable, but rather that we are looking for "pure" (stable) distributions, that indeed are not there. On the other hand, fitting a small history into a specific probability distribution that is close to the real distribution becomes feasible, allowing the detection of stable phases (that exist in reality, but for approximate distributions). This is the reason why the selection of an adequate history size has a considerable impact on the results and is crucial.

For large history sizes, the outcome of the lower rate of detected stability points is that the average improvement of bounds is modest (Figure 4) – *Adaptare* computes the conservative bound most of the time.

On the other hand, if the history size is too small, detection mechanisms will tend to incorrectly identify stable periods that do not correspond to the real environment conditions. This erroneous behavior will lead *Adaptare* to possibly select lower bounds than it should (depending on the identified distribution and the actual state of the environment), which may compromise the dependability of our approach. This effect can be observed in our experiments with $h = 10$: in the majority of the traces, they have the highest rate of stability detection (Figure 3) and consequently the best improvement of bounds (Figure 4), but the coverage is not secured (Figure 5).

For all performed experiments, the best results were obtained with history size $h = 30$: highest improvement of bounds, securing the minimum required coverage.

These results suggest that *there are real environments in which our assumptions hold*. In fact, although it is impossible to verify each assumption alone, the fact that we observe a consistent improvement of the average bounds while securing the required dependability, provides sufficient evidence. In these environments, *Adaptare* can be successfully applied in order to make estimations through a probabilistic

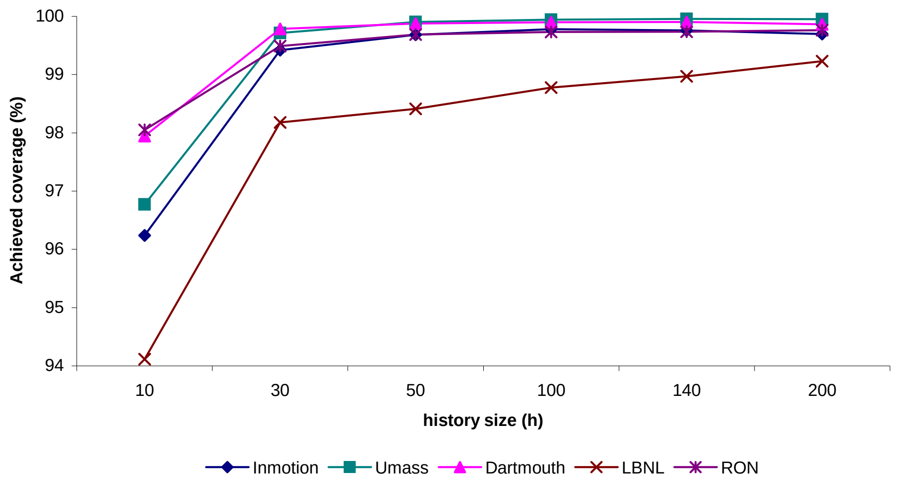

Fig. 5. Achieved coverage using different history sizes

analysis of historical data and provide applications with better information related to the environment behavior (e.g., network delays), while ensuring the same level of dependability.

5.3 Complexity analysis

One essential factor to be considered in our work is the complexity of the implemented algorithms. Although we are assuming that there are enough resources to perform the necessary computations (see Section 3.1), it is important to demonstrate that *Adaptare* has an acceptable complexity in order to reach its objectives. This section presents an informal analysis of the framework complexity and some measures of its execution time.

We show that the current implementation of *Adaptare* presents a complexity $O(m \times d \times h \log(h))$, where m is the number of phase detection mechanisms based on GoF tests, d is the number of considered distributions, and h is the history size. We realize that because our implementation uses small values of m, d, and h, this asymptotic analysis does not provide enough information about execution time. Nevertheless, we believe it is important to give an intuition of the impact of our algorithms and the influence of each variable in the framework execution. In any case, the asymptotic analysis is complemented with real measurements of the *Adaptare* framework performance, also presented ahead.

The general framework algorithm is described in Figure 6. Assuming that line 2 is a single operation, the complexity of the entire algorithm would be $O(m)$.

However, the execution of a phase detection mechanism (line 2) is not composed by one single operation, as shown in Figure 7. For each distribution, the parameters are estimated (line 3) and the verification if the given sample fits in the assumed distribution is performed (line 4).

Regarding parameters estimation, the exponential distribution has only one pa-

```
input : coverage, history size, measured samples
output: New computed bound

1 foreach phase detection mechanism M do
2     Get distribution from M
3 end
4 Select one new bound according to the selection logic
5 Return new bound
```

Fig. 6. *Adaptare*'s algorithm

```
input : measured samples
output: detected distribution

 1 detected distribution = none
 2 foreach distribution D do
 3     Estimate D parameters
 4     if samples fit in D (with its parameters) then
 5         Update detected distribution according to the mechanism's logic
 6     end
 7 end
 8 if detected distribution = none then
 9     return transient phase
10 else
11     return detected distribution
12 end
```

Fig. 7. Phase detection mechanism's algorithm

rameter (λ), which is estimated as the inverse of the sample mean. This estimation has a complexity that is linear with the history size ($O(h)$). The shifted exponential parameters (λ and γ) and the Weibull parameters (α and γ) are estimated using linear regression, which also has linear complexity with the history size $O(h)$. The k parameter of the Pareto distribution is estimated as the smallest sample value and the α parameter is computed using a MLE formula. Both estimations are performed with complexity $O(h)$. Finally, the uniform parameters a and b are estimated as the minimum and maximum sample values, leading to a $O(h)$ complexity.

For now, we will assume that the conditional test in line 4 takes one operation. Thus, given that there are d distributions, and estimating the parameters of each tested distribution takes linear time $O(h)$, the complexity of this algorithm is $O(d \times h)$. In our implementation, the conditional test in line 4 is performed by GoF tests.

A general algorithm of GoF tests is presented in Figure 8 (the specific algorithms of the two GoF tests implemented in *Adaptare* were described in Section 4.1). For both mechanisms, in order to calculate the GoF statistic (line 2), it is necessary to sort the input sample. Our implementation uses a modified mergesort with complexity $O(h \log(h))$ to sort the measured data (line 1), where h is the history size. Moreover, the formulas presented in Section 4.1 to compute the GoF statistics are calculated in linear time ($O(h)$). Thus, performing a Gof test takes $h \log(h) + h$ operations, which is reduced to an asymptotic complexity of $O(h \log(h))$.

After analyzing the complexity of each part of the framework execution, we

```
input : assumed distribution with estimated parameters, measured samples
output: if the measures samples fit in the assumed distribution

1 Sort measured samples
2 Calculate GoF statistic $S$ for assumed distribution $\hat{D}$
3 Get critical value $c_{h,\alpha}$
4 if $S \leq c_{h,\alpha}$ then
5     return true
6 else
7     return false
8 end
```

Fig. 8. GoF test's algorithm

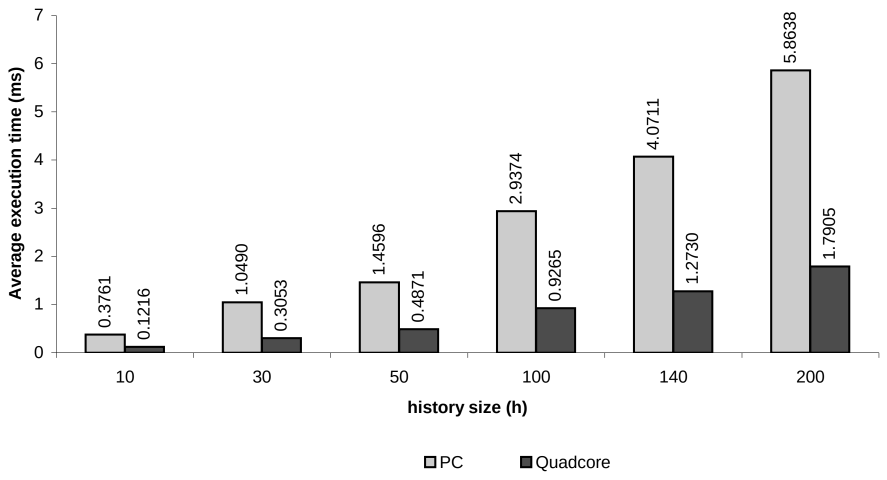

Fig. 9. Measured execution time

can conclude that since each phase detection mechanism M is executed for each distribution D, and the GoF tests are executed for all distributions, after estimating their parameters, the execution time of the current implementation of *Adaptare* is a function of $m \times d \times (h + h\log(h))$. Thus, the asymptotic complexity is $O(m \times d \times h\log(h))$, as previously mentioned. Considering a fixed number of mechanisms and distributions (currently, $m = 2$ and $d = 5$), the actual complexity of the framework execution is $O(h\log(h))$. It is important to note that this complexity is imposed by the sort operation required by both mechanisms.

To conclude our analysis, we measured the time that *Adaptare* takes to compute a new bound, for different history sizes. These measurements were performed using the current implementation of the framework (composed by two phase detection mechanisms and five distributions, as described in Section 4), which was executed in two different platforms: (i) a Dell Optiplex GX520 PC, with 2GB of RAM and one 2.8GHz Pentium 4 processor, and (ii) a 64-bit 2.3GHz quadcore Xeon machine.

The initial point (time 0) of the measure interval is when the framework has the sample trace available in a list, and the final point is defined immediately after a new bound is produced. To execute these tests we varied the history size from 10 to 200 and used an Inmotion trace composed by 8200 points. *Adaptare* was executed for each new sample point added to the history, computing approximately 8000 new bounds. Figure 9 presents the average execution time of the framework, for each considered history size, using both platforms.

The analysis of the execution time can be helpful in order to determine how a certain application should use the framework. In average, *Adaptare* takes one to five milliseconds to analyse the sample trace and compute a new bound, depending on the history size and on the execution platform. We believe that these are reasonable values for many practical applications. Note that these values reflect the user level / application perception of the framework execution time, which includes all typical overheads (interference of other tasks, context switches, etc.), usual in multiprocess systems.

Nevertheless, when considering embedded applications or other systems with constrained resources, these values may be a considerable trade-off for the improvements that may be achieved. In these cases, we propose two simple and combinable measures that a designer can take in order to to reduce the overhead of executing *Adaptare*. The first measure consists in executing the framework more sparingly. Ideally, a new bound would be computed whenever a new sample is added to the history. However, if new data is added to the history too frequently, it will be wise to execute *Adaptare* more parsimoniously, in fixed intervals, or when a reasonable number of n new values are added to the history. The second measure consists in disabling the recognition of some probabilistic distributions, which can be safely done without reducing the performance of *Adaptare* when it is known that these distributions are never detected. With this measure, the execution times in Figure 9, which correspond to a fully-fledged framework detecting five different distributions, could be significantly reduced.

5.4 Comparing *Adaptare* to other adaptive solutions

Adaptare was designed to be easily integrated with client applications, driving their adaptation while guaranteeing the required dependability level. To achieve this, *Adaptare* requires the execution of complex and costly analysis mechanisms, as discussed in the previous section. Therefore, it is important to evaluate the relative benefits of *Adaptare* in comparison to simpler and cheaper mechanisms, which we do in this section.

We selected two simpler approaches to compare with: the RTT estimation algorithm performed by TCP, which has constant execution time ($O(1)$), and a method based on the WINMEAN estimator plus a safety margin, of linear complexity $O(h)$. Recall that *Adaptare*'s execution has a complexity of $O(hlog(h))$, dictated by the sorting algorithm executed within the phase detection mechanisms.

The TCP protocol formally defines a simple method for setting the retransmission timer of each non-acknowledged packet, based on the observed RTTs and their variations. The algorithm, presented in Figure 10, was originated by Jacobson in [Jacobson 1988] and is formalized in RFC2988. In this section we refer to it as the

```
**output**: New retransmission timer RTO
**consts**: $k = 4$, $\alpha = 1/8$, $\beta = 1/4$

1 Wait for ACK of sent message $m$
2 if $ACK$ is not received within $RTO$ then
3     Retransmit message $m$
4     $RTO = RTO * 2$
5 else
6     $R = RTT$ of sent message
7     if $R$ is the first measured $RTT$ then
8         $SRTT = R$
9         $RTTVAR = R/2$
10        $RTO = SRTT + k * RTTVAR$
11    else
12        $RTTVAR = (1 - \beta) * RTTVAR + \beta * |SRTT - R|$
13        $SRTT = (1 - \alpha) * SRTT + \alpha * R$
14        $RTO = SRTT + k * RTTVAR$
15    end
16 end
17 Return RTO
```

Fig. 10. TCP's algorithm to compute the retransmission timer

```
impute: measured samples
output: new estimator
consts: $\alpha = 1/4$

1 $mean$ = samples average
2 $safetyMargin = lastMargin + \alpha * (|lastSample - lastEstimator| - lastMargin)$
3 $estimator = mean + safetyMargin$
4 $lastEstimator = estimator$
5 $lastMargin = safetyMargin$
5 Return estimator
```

Fig. 11. WinMean-Jac estimation algorithm

TCP-RTT approach.

The algorithm of based on the WINMEAN estimator plus a safety margin is presented in Figure 11. The WINMEAN estimator simply computes the history average, and the safety margin is also based in the Jacobson algorithm [Jacobson 1988]. This approach is applied to adjust failure detectors timeouts in [Nunes and Jansch-Porto 2004] and [Falai and Bondavalli 2005], and it will be referred as the *WinMean-Jac* approach.

For the comparative experiments we configured *Adaptare* with $C = 98\%$ and $h = 30$, considering the positive results previously achieved with this history size. Then, we applied the real traces from the five different sources as input for the different approaches, collecting the bounds produced by all of them, based on the same input. The results are summarized in figures 12 and 13. Figure 12 compares the average bounds achieved with the three solutions and Figure 13 allows the

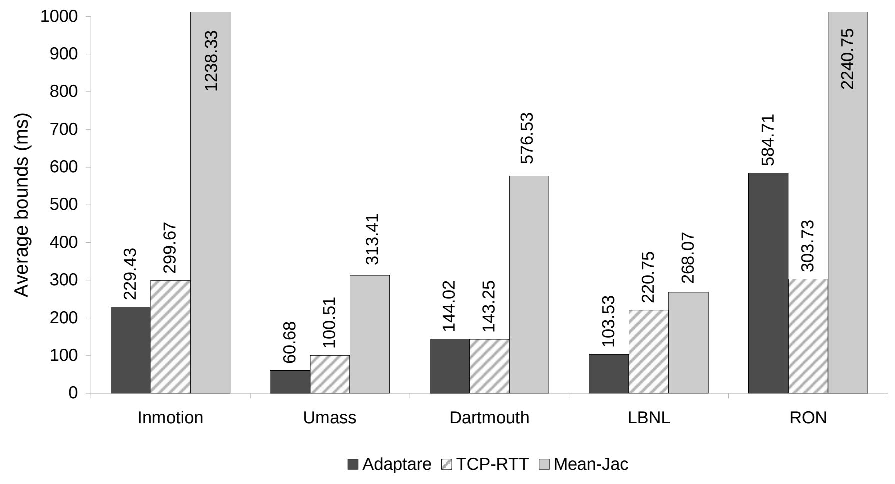

Fig. 12. Comparing computed bounds from different solutions

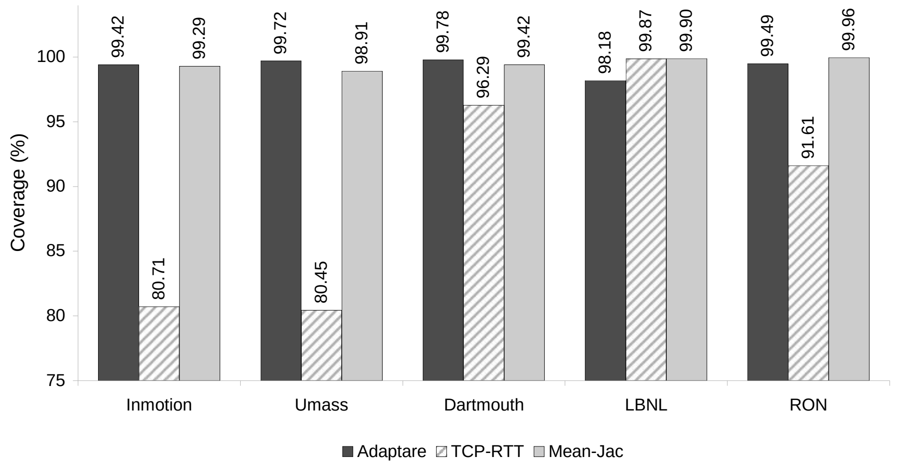

Fig. 13. Comparing achieved coverage from different solutions

comparison of achieved coverage.

When comparing *Adaptare* to *WinMean-Jac*, it is possible to verify that *Adaptare* was able to provide significantly lower bounds (reductions roughly ranging between 50% to 80%) without any negative impact on the achieved coverage. In fact, in most cases the coverage provided by *Adaptare* was even better than the coverage provided by *WinMean-Jac*. In any case, and for both solutions, the required minimum coverage of 98% was always secured.

The results achieved when applying the *TCP-RTT* approach were noticeable different from those achieved with *WinMean-Jac*. Regarding the provided bounds, *TCP-RTT* was much closer to *Adaptare*. More specifically, *Adaptare* was able to provide smaller bounds in three cases (Inmotion, Umass and LBNL), while *TCP-RTT* was better in the case of the RON trace. No significant difference was observed for the Dartmouth trace. However, when comparing *Adaptare* to *TCP-RTT* in terms of the achieved coverage, the results clearly indicate that *TCP-RTT* leads to significantly lower coverage values (except in the case of LBNL), that is, produces much more timing faults than those produces with *Adaptare*.

In order to correctly interpret these results, it is necessary to retain that if the objective was to merely achieve small bounds, then an aggressive method would achieve that easily. However, if bounds are too small, this will typically have negative effects. For instance, in the case of the TCP protocol, every timing fault resulting from a badly dimensioned timeout implies one unnecessary retransmission, and a consequent waste of network resources. Therefore, for comparison purposes, it is necessary to consider both the achieved bounds and the achieved coverage, altogether.

In general, *Adaptare* was able to compute the best bounds and secure the expected dependability level. These results indicate that *Adaptare* is indeed able to properly react to environment changes and compute more precise bounds than with the other two approaches. In one case *TCP-RTT* was able to provide smaller bounds, but at a possibly high cost, given that 10% of them resulted in timing faults. Our conclusion is that among the three adaptive methods evaluated in this section, *Adaptare* is the best solution for adaptive systems, in particular for those with dependability concerns.

The price to pay for such optimized solution is that it requires more resources, in terms of storage space and particularly in terms of memory operations. The *TCP-RTT* approach is based only on the last measured RTT and estimation error, so it does not need any extra storage. The bound computation is straightforward, requiring a total of 15 arithmetic operations. The *WinMean-Jac* approach is slightly more complex. The average computation performed by the WINMEAN estimator requires h arithmetic operations, while the Jacobson safety margin is calculated in constant time, using 5 arithmetic operations. Thus, this approach executes $h + 5$ operations and requires storage space for keeping the history of samples. In contrast, *Adaptare* is a complex solution. The number of operations executed for the estimation of distribution parameters, by phase detection mechanisms, and so on, is clearly much higher than in simple approaches. Regarding execution overhead, and as presented in Section 5.3, *Adaptare* leads to overheads in the millisecond order, while the execution overhead of simple approaches like *TCP-RTT* or *WinMean-Jac* is comparably insignificant.

Overall, deciding which approach is the better will necessarily depend on the application objectives and requirements and on the amount of available resources.

6. EXAMPLES OF ADAPTIVE APPLICATIONS USING *ADAPTARE*

Two examples of practical applications that use *Adaptare* to drive the adaptation process are presented in [Dixit and Casimiro 2010] and [Dixit et al. 2011].

[Dixit and Casimiro 2010] introduces *Adaptare-FD*, an adaptive failure detector that is built over *Adaptare*. The failure detector is configured by setting the minimum required coverage and a lower bound for the average mistake recurrence time (the interval between two consecutive false suspicions). The coverage parameter is particularly relevant in the context of this paper, as it is used to configure *Adaptare* and has direct influence on the timeouts that are used during the execution. These timeouts are dynamically adjusted based on *Adaptare*'s output, allowing the failure detector to perform as well as possible, depending on the actual state of the network.

The performed evaluation compared *Adaptare-FD* to a set of 12 timeout-based adaptive failure detectors. The experiments were executed in the PlanetLab testbed ([Spring et al. 2006]), and the evaluation metrics were timeout, mistake recurrence time, average mistake duration, total mistake duration and coverage. The results shown that *Adaptare-FD* performed better than the remaining detectors, especially in more dynamic environments. One particularly significant observation was that the tradeoff between the mistake recurrence time and the mistake duration is significantly reduced with *Adaptare-FD*.

The work presented in [Dixit et al. 2011] proposes an adaptive version of a randomized consensus algorithm for wireless ad-hoc networks, using *Adaptare*. The algorithm uses a timeout to decide whether, and when, a broadcast must be retransmitted. The static version uses a fixed timeout, which must be set at deployment time and requires some previous knowledge, or measurements, of the operational environment. On the other hand, in the adaptive consensus this timeout is adjusted based on *Adaptare*'s indications. In this application, the coverage required to *Adaptare* is defined to optimize the trade-off between waiting time (before retransmitting a message) and network load (also with eventual implications on latency). Timeouts may be aggressive (small) to ensure fast termination (based on retransmissions after the timeout), but not too aggressive, to avoid excessive retransmissions and the consequent network overload and increased latency.

The static and the adaptive versions of this consensus algorithm were compared in terms of generated network load (number of broadcasts sent by each process) and latency (execution time). The experiments were carried out on the Emulab testbed ([White et al. 2002]), with a varying number of processes participating in the consensus, and considering both normal network conditions and overload conditions. The evaluation showed that the adaptive algorithm was able to perform well in the various conditions due to the automatic adjustment of the timeout guided by *Adaptare*, whereas the static version would perform badly when some conditions changed (particularly when increasing the number of processes).

Those results illustrate the applicability of *Adaptare*, and the benefits of using more elaborated adaptation solutions in order to improve system's performance and dependability.

7. CONCLUSIONS

In this paper we addressed the problem of supporting adaptive systems and applications in stochastic environments, from a dependability perspective: maintaining the correctness of system properties after adaptation. We started from the observation that the scene in open distributed applications such as those running in IP-based and/or complex embedded systems, is subject important factors of change: (i) those under traditionally weak models, such as asynchronous, face increasing performance demands, sometimes forcing designers to resort to hidden timing/synchrony assumptions, which are potential causes of failures; (ii) those under traditionally strong models, such as hard real-time or synchronous, face increasing demands for dynamic behavior or configuration.

This is leading to increased threats to system correctness, such as failures due to impossibility results in asynchronous systems, or failures due to missed deadlines and bounds in synchronous ones, respectively. Clearly, the key issue in these problems is the lack, or loss thereof, of coverage of time-related assumptions.

We proposed to overcome these problems by securing coverage stability. With this, we aim at ensuring automatic adaptation of time-sensitive applications whilst safeguarding correctness, despite variations of aprioristic assumptions about time. Therefore, while other works addressing adaptive systems are mainly concerned with performance, our main concern is dependability.

We advanced on previous work by leveraging on the assumption that i) a system alternates stable periods, during which the environment characteristics are fixed, and unstable periods, in which a variation of the environment conditions occurs, and ii) that the mode changes can be detected. Based on that, we proposed and evaluated *Adaptare* - a general framework for Automatic and Dependable Adaptation in ProbabilisTic environments. Our implementation is composed by two phase detection mechanisms based on statistical tests, which try to characterize the monitored environment using five different probability distributions. However, *Adaptare* is a generic and open framework that may be extended by designers with other mechanisms and distributions, depending on their needs. The characterization performed by this framework allows to dynamically compute improved (tighter) time bounds when a stable phase is detected, and conservative but still dependable bounds during transient phases.

The methodology used in our evaluation was based on experiments with synthetic data traces generated from specific distributions, and real data traces available in the Internet, which were collected by different applications, operating in different environments. The experiments with synthetic traces were useful to validate the correctness of the implemented phase detection mechanisms, while the results using real data traces proved that with *Adaptare* it was possible to compute bounds in the order of 10% to 15% lower than the bounds produced by the conservative approach, securing the required coverage in all cases. *Adaptare* was also compared to simpler approaches to evaluate the relative performance benefits. The achievable benefits were clear, since *Adaptare* performs dependably and still allows to achieve significant improvements in terms of computed bounds. The achieved improvements also allow to conclude that the fundamental assumptions on which the framework is based were satisfied by the real data traces used in the experiments.

The costs incurred by *Adaptare* to provide these benefits were also addressed in the paper. We presented a complexity analysis and we evaluated the execution overhead of our implementation of *Adaptare*, allowing us to conclude that the incurred costs can be acceptable in many practical systems. For systems with resource constraints we mentioned two simple ways of reducing the overhead of *Adaptare*.

The fundamental conclusion to derive from the full set of experiments reported in this paper is that it is possible to define effective mechanisms to detect stable and transient phases and, for the stable ones, correctly characterize the observed probabilistic distribution. Because of that, *Adaptare* constitutes a promising approach to achieve dependable adaptation and, at the same time, obtain improved time bounds. This improvement is relevant in the implementation of practical systems, as we illustrated in the paper.

Our work definitely contributes to supporting automatic adaptation of time-sensitive applications whilst safeguarding correctness, despite variations of aprioristic assumptions about time. We believe this is an important contribution to the dependability of emerging complex networked embedded systems characterizing ambient computing environments.

ACKNOWLEDGMENTS

This work was partially supported by the EC, through project IST-FP6-STREP-26979 (HIDENETS), and by the FCT, through the Multiannual and CMU-Portugal programmes.

REFERENCES

Allen, A. O. 1990. *Probability, statistics, and queueing theory with computer science applications.* Academic Press Professional, Inc., San Diego, CA, USA.

Andersen, D., Balakrishnan, H., Kaashoek, F., and Morris, R. 2001. Resilient overlay networks. *SIGOPS Oper. Syst. Rev. 35,* 5, 131–145.

Babaoglu, Ö., Jelasity, M., Montresor, A., Fetzer, C., Leonardi, S., van Moorsel, A. P. A., and van Steen, M., Eds. 2005. *Self-star Properties in Complex Information Systems, Conceptual and Practical Foundations.* Lecture Notes in Computer Science, vol. 3460. Springer.

Balakrishnan, N. and Basu, A. 1995. *The exponential distribution: Theory, methods and applications.* CRC Press.

Bhatti, S. N. and Knight, G. 1999. Enabling qos adaptation decisions for internet applications. *Comput. Netw. 31,* 7, 669–692.

Bolot, J.-C. 1993. Characterizing end-to-end packet delay and loss in the internet. *Journal of High Speed Networks 2*, 305–323.

Buttazzo, G. C. 1997. Hard real-time computing systems. Kluwer Academic Publishers.

Casimiro, A., Lollini, P., Dixit, M., Bondavalli, A., and Verissimo, P. 2008. A Framework for Dependable QoS Adaptation in Probabilistic Environments. In *23rd ACM Symposium on Applied Computing, Dependable and Adaptive Distributed Systems Track.* Fortaleza, Ceara, Brazil, 2192–2196.

Casimiro, A. and Verissimo, P. 2001. Using the Timely Computing Base for Dependable QoS Adaptation. In *Proceedings of the 20th IEEE Symposium on Reliable Distributed Systems.* New Orleans, USA, 208–217.

Chen, K.-T., Jiang, J.-W., Huang, P., Chu, H.-H., Lei, C.-L., and Chen, W.-C. 2006. Identifying mmorpg bots: a traffic analysis approach. In *ACE '06: Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology.* ACM, New York, NY, USA, 4.

Chen, W., Toueg, S., and Aguilera, M. K. 2002. On the quality of service of failure detectors. *IEEE Trans. Comput. 51,* 1, 13–32.

Corlett, A., Pullin, D., and Sargood, S. 2002. Statistics of one-way internet packet delays. http://www.potaroo.net/ietf/all-ids/draft-corlett-statistics-of-packet-delays-00.txt.

Dixit, M. and Casimiro, A. 2010. Adaptare-FD: A dependability-oriented adaptive failure detector. In *To appear in Proceedings of the 29th IEEE Symposium on Reliable Distributed Systems*. Delhi, India.

Dixit, M., Moniz, H., and Casimiro, A. 2011. Timeout Adaptive Consensus: Improving Performance through Adaptation. In *Submitted to the 26th ACM Symposium on Applied Computing, Dependable and Adaptive Distributed Systems Track*. TaiChung, Taiwan.

Downey, A. B. 2001. Evidence for long-tailed distributions in the internet. In *IMW '01: Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*. ACM, New York, NY, USA, 229–241.

Elteto, T. and Molnar, S. 1999. On the distribution of round-trip delays in tcp/ip networks. In *Local Computer Networks, 1999. LCN '99. Conference on*. 172–181.

Evans, J. W., Johnson, R. A., and Green, D. W. 1989. Two- and three-parameter weibull goodness-of-fit tests. FPL-RP-493, 1989; Forest Products Laboratory Research Paper.

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

Gass, R., Scott, J., and Diot, C. 2005. CRAWDAD trace set cambridge/inmotion/tcp (v. 2005-10-01). Downloaded from http://crawdad.cs.dartmouth.edu/cambridge/inmotion/tcp.

Gass, R., Scott, J., and Diot, C. 2006. Measurements of in-motion 802.11 networking. In *WMCSA '06: Proceedings of the Seventh IEEE Workshop on Mobile Computing Systems & Applications*. Washington, DC, USA, 69–74.

Henderson, T., Kotz, D., and Abyzov, I. 2004. The changing usage of a mature campus-wide wireless network. In *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking*. 187–201.

Hernandez, J. A. and Phillips, I. W. 2006. Weibull mixture model to characterise end-to-end internet delay at coarse time-scales. *IEE Proc. Communications 153,* 2 (Apr.), 295–304.

Jacobson, V. 1988. Congestion avoidance and control. *Innovations in Internetworking*, 273–288.

Jain, R. 1991. *The Art of Computer Systems Performance Analysis*. John Wiley and Sons.

Koliver, C., Nahrstedt, K., Farines, J.-M., Fraga, J. D. S., and Sandri, S. A. 2002. Specification, mapping and control for qos adaptation. *Real-Time Syst. 23,* 1/2, 143–174.

Kopetz, H. and Bauer, G. 2003. The Time-Triggered Architecture. *Proceedings of the IEEE 91,* 1 (Jan), 112–126.

Kotz, D., Henderson, T., and Abyzov, I. 2004. CRAWDAD trace set dartmouth/campus/tcpdump (v. 2004-11-09). Downloaded from http://crawdad.cs.dartmouth.edu/dartmouth/campus/tcpdump.

Krishnamurthy, S., Sanders, W. H., and Cukier, M. 2001. A dynamic replica selection algorithm for tolerating timing faults. In *2001 International Conference on Dependable Systems and Networks*. 107–116.

Li, B., Xu, D., and Nahrstedt, K. 1999. Optimal state prediction for feedback-based qos adaptations. In *IEEE IWQoS '99. Proceedings of the Seventh International Workshop on Quality of Service*. IEEE, London, UK, 37–46.

Liu, J. 2000. Real-time systems. Prentice Hall.

Markopoulou, A., Tobagi, F. A., and Karam, M. J. 2006. Loss and delay measurements of internet backbones. *Computer Communications 29,* 10 (June), 1590–1604.

Menth, M., Milbrandt, J., and Junker, J. 2006. Time-exponentially weighted moving histograms (TEWMH) for application in adaptive systems. In *Proceedings of the Global Telecommunications Conference (GLOBECOM '06)*. 1–6.

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

Papagiannaki, K., Moon, S., Fraleigh, C., Thiran, P., and Diot, C. 2003. Measurement and analysis of single-hop delay on an ip backbone network. *IEEE Journal on Selected Areas in Communications. Special Issue on Internet and WWW Measurement, Mapping, and Modeling 21,* 6 (Aug.), 908–921.

PAXSON, V., PANG, R., ALLMAN, M., BENNETT, M., LEE, J., AND TIERNEY, B. 2007. lbl-internal.20041004-1303.port001.dump.anon (package). http://imdc.datcat.org/package/1-507R-8=lbl-internal.20041004-1303.port001.dump.anon.

PIRATLA, N., JAYASUMANA, A., AND SMITH, H. 2004. Overcoming the effects of correlation in packet delay measurements using inter-packet gaps. In *Proceedings of the 12th IEEE International Conference on Networks (ICON 2004)*. 233–238.

PORTER, J.E., I., COLEMAN, J., AND MOORE, A. Mar 1992. Modified ks, ad, and c-vm tests for the pareto distribution with unknown location and scale parameters. *Reliability, IEEE Transactions on 41,* 1, 112–117.

RAHMAN, M., PEARSON, L. M., AND HEIEN, H. C. 2006. A modified anderson-darling test for uniformity. *Bulletin of the Malaysian Mathematical Sciences Society 29,* 1, 11–16.

RELIASOFT. 2006. Using rank regression on y to calculate the parameters of the weibull distribution - reliasoft corporation. http://www.weibull.com/LifeDataWeb/estimation_of_the_weibull_parameter.htm.

RON, M. 2001. Ron - resilient overlay networks. http://nms.csail.mit.edu/ron.

SPRING, N., PETERSON, L., BAVIER, A., AND PAI, V. 2006. Using planetlab for network research: myths, realities, and best practices. *SIGOPS Oper. Syst. Rev. 40,* 1, 17–24.

STEPHENS, M. A. 1974. Edf statistics for goodness of fit and some comparisons. *Journal of the American Statistical Association 69,* 347 (sep), 730–737.

STEPHENS, M. A. 1976. Asymptotic results for goodness-of-fit statistics with unknown parameters. *Annals of Statistics 4*, 357–369.

TICKOO, O. AND SIKDAR, B. 2004. Queueing analysis and delay mitigation in ieee 802.11 random access mac based wireless networks. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*. Vol. 2. 1404–1413 vol.2.

TRIVEDI, K. S. 2002. *Probability and Statistics with Reliability, Queuing and Computer Science Applications.* John Wiley and Sons.

TZAGKARAKIS, G., PAPADAPOULI, M., AND TSAKALIDES, P. 2008. Trend forecasting based on singular spectrumanalysis of traffic workload in a large-scale wireless lan.

UMASS TRACE REPOSITORY. 2006. UPRM wireless traces. Downloaded from http://traces.cs.umass.edu/index.php/Network/Network.

VERISSIMO, P. AND CASIMIRO, A. 2002. The Timely Computing Base model and architecture. *Transactions on Computers - Special Issue on Asynchronous Real-Time Systems 51,* 8 (Aug.), 916–930.

WHITE, B., LEPREAU, J., STOLLER, L., RICCI, R., GURUPRASAD, S., NEWBOLD, M., HIBLER, M., BARB, C., AND JOGLEKAR, A. 2002. An integrated experimental environment for distributed systems and networks. In *5th Symposium on Operating Systems Design and Implementation.* Boston, MA, USA, 255–270.

YANG, M., LI, X. R., CHEN, H., AND RAO, N. S. V. 2004. Predicting internet end-to-end delay: an overview. In *Proc. of the 36th IEEE Southeastern Symposium on Systems Theory.* 210–214.