Supporting Mobile Ubiquitous Applications with Mobility Prediction and Soft Handoff

Marcello Cinque[1] and Stefano Russo[1,2]

[1] Dipartimento di Informatica e Sistemistica
Universita' degli Studi di Napoli Federico II
Via Claudio 21, 80125 - Naples, Italy
{macinque, sterusso}@unina.it
[2] Laboratorio ITEM "Carlo Savy"
Consorzio Interuniversitario Nazionale per l'Informatica
M.S. Angelo, Via Cinthia - 80125 Naples, Italy

Abstract. The increasing success of mobile-enabled, embedded devices is stressing the need for software architectures facing mobility-related issues. This paper proposes a simple yet effective mobility management scheme to ease the development of mobile, ubiquitous applications. The scheme seamlessly handles handoff events and provides ubiquitous applications with both location-awareness and mobility prediction support. An implementation prototype has been developed on real-world, Bluetooth enabled devices. Experimental results are then obtained from the prototype, showing the effectiveness of the proposed scheme.

1 Introduction

Mobility management has widely been recognized as one of the most challenging problems for a seamless integration of embedded, mobile devices (MDs) into the physical world. Such integration is an important step towards the ubiquitous view of computing, where computation resources are spread into small devices which pervasively interact each other "all the time and everywhere" by means of wireless communication infrastructures [1].
One of the key aspects of mobility management is the handling of handoff procedures, i.e., the set of operations that need to be performed to guarantee a MD to be connected with one or more wireless Access Points (APs) while it roams across the ubiquitous environment. Specifically, a handoff procedure is composed of two basic steps: i) the *initiation*, which detects and triggers a handoff event from the old AP to a new one, and ii) the *decision*, where a new AP is selected among the available ones.
Several handoff management schemes have been proposed over the last years, addressing different flavors of wireless networks, from cellular networks, to the wireless Internet. However, when facing mobility-related issues for ubiquitous environments, several new challenges arise which are not generally supported by current software architectures for ubiquitous applications.
First, to achieve the "all the time and everywhere" view of mobility, handoff

management should provide high connection availability to each MD. Second, ubiquitous devices typically offer limited computation and storing capabilities, and rely upon batteries. The mobility management support should thus take into account MDs and APs constrained resources by managing the handoff in a lightweight fashion. Third, ubiquitous applications would greatly benefit from a handoff management architecture able to provide mobility prediction. The ability of predicting both the handoff event and the next MD location enables to implement proactive resource allocation schemes which can have a significant impact on the overall performance.
Connection availability and mobility prediction can be obtained by implementing "soft" handoff procedures, where the MD is always connected to more than one AP, in order to minimize unavailability periods and to oversee the movements. However, this type of strategy may involve unacceptable resource consumption at both MD and AP sides.
This paper addresses these problems by proposing a novel, hybrid approach to handoff management, which requires the MD to be connected to a single AP, while guaranteeing soft handoffs and providing mobility prediction. The novel contribution, namely "Octopus", is a lightweight handoff scheme which extends our previous Last Second Soft Handoff Scheme (LSSH) [2, 3]. In particular, even if LSSH provides soft handoffs while reducing unavailability periods (please refer to [2] for a quantitative evaluation), it presents the drawback of long decision periods, which may in turn degrade the accuracy of the location awareness support. Moreover, LSSH does not embody mobility prediction schemes.
The novel handoff scheme has been implemented and integrated in a mobility management architecture, running over Bluetooth wireless networks. Experimental results have been run on the actual implementation, demonstrating how the novel mobility prediction support offered by Octopus can significantly improve the decision latency and the location accuracy.

2 Related Work

Handoff strategies can be classified as reactive and proactive. Reactive strategies, such as [4, 5], look for other available APs only after the current AP signal is lost.On the other hand, proactive strategies continuously monitor channel conditions and start communication-level handoff before losing current AP signal, at the cost of higher battery consumption. Several criteria are based on the Receiver Signal Strength Indicator (RSSI) [6–9].Some of them, such as [6, 9] are based on a fixed threshold mechanism, that is, the handoff is initiated when the RSSI falls below a certain threshold. It is simple to argue how this kind of initiation leads to a poor availability. Indeed, noisy environments and shadowing problems can lead to transient RSSI degradations, which do not strictly require any handoff. Fore this reason, other solutions use a more complicated RSSI processing, such as fuzzy controllers [7], or mobility prediction [8].We can further distinguish two types of handoff: hard handoff, where the MD is connected to only one AP at

time, minimizing signaling overhead but increasing latency and packet losses; and soft handoff that activates the new data path to the destination AP before client disconnection from the origin AP [10]. It is worth noting that none of the mentioned solutions is able to answer to the needs outlined in previous section.

3 Handoff Management and Mobility Prediction

3.1 The LSSH scheme

The LSSH scheme is a hybrid approach that tries to exploit the advantages of both hard and soft solutions. The initiation phase takes place using uniquely the information about the AP currently in use, as in hard handoff, and only in the decision phase multiple connections are established, as in soft handoff.

LSSH initiation. The initiation phase can be performed using diverse sets of information and techniques, such as broken link recognition and AP monitoring through RSSI or other measures and metrics. Our solution is RSSI based, for several reasons: i) it allows the handoff to be proactive, ii) the RSSI parameter is often already provided by the wireless interface, without performing intrusive measures, and iii) RSSI is an indication of the device position with respect to APs; this helps to achieve load balancing on APs depending on device distribution in the environment. Furthermore, locationing techniques can be implemented. According to the LSSH scheme, the initiation has to be performed using only the RSSI of the AP in use. It is thus crucial to carefully discriminate transient signal degradations, from permanent ones. Indeed, transient signal degradations can trigger unnecessary handoff procedures. To this aim, the LSSH scheme adopts the α-count mechanism due to the clear and simple mathematical characterization, the thorough analysis already conducted, and the minimal computational complexity which properly answers lightweight needs [11]. The α-count function $\alpha^{(L)}$ is a count and threshold mechanism. It takes the L-th measured RSSI as an input, then $\alpha^{(L)}$ is incremented by 1 as the current RSSI falls below the threshold S_{RSSI}. Similarly, $\alpha^{(L)}$ is decremented by a positive quantity dec if the L-th measured RSSI is greater than the S_{RSSI}. A handoff is triggered as soon as $\alpha^{(L)}$ becomes greater than a certain threshold α_T. The function $\alpha^{(L)}$ is thus defined as follows:

$$\alpha^{(L)} = \begin{cases} \alpha^{(L-1)} + 1 & \textit{if } RSSI^{(L)} < S_{RSSI} \\ \alpha^{(L-1)} - dec & \textit{if } RSSI^{(L)} \geq S_{RSSI} \textit{ and } \alpha^{(L-1)} - dec > 0 \\ 0 & \textit{if } RSSI^{(L)} \geq S_{RSSI} \textit{ and } \alpha^{(L-1)} - dec \leq 0 \end{cases} \quad (1)$$

In our previous work we outlined how the values of α_T, dec and S_{RSSI} parameters can be tuned in order to achieve a trade-off between early and late handoffs.

LSSH decision During the decision phase, the MD sequentially connects to all the neighboring APs of the old AP. The decision is then taken by evaluating the RSSI of all the links to the neighbors and by choosing the best AP among them.

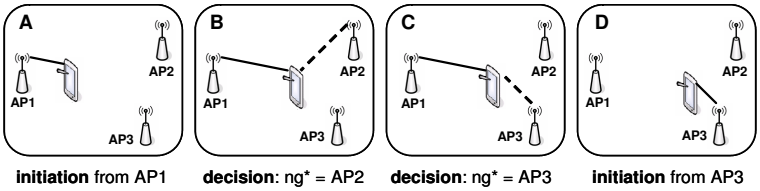

Fig. 1. The LSSH scheme

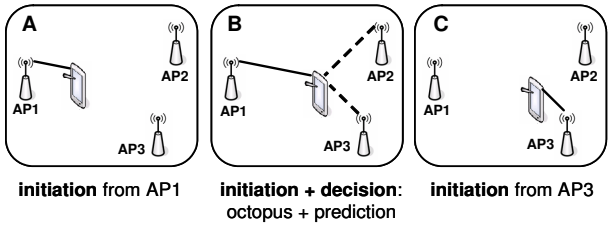

Fig. 2. The Octopus scheme

Let $\{ng_1, ..., ng_n\}$ be the set of neighbors. During the scanning, the scheme keeps track of the best visited AP, let say ng^*. When it connects to ng_i, if the RSSI of ng_i is greater than the ng^* one, then $ng^* = ng_i$. At the end of the scanning, the resulting ng^* is selected as the new AP. It is simple to argue that such sequential scanning may require long decision latencies. As for locationing issues, we assume that a mobile device is in a zone x when it is attached to a AP covering the zone x. Being the RSSI strictly related with the distance between antennas, the scheme enforces devices to be connected to the closest AP. However, even if pathological situations can lead to the selection of a wrong AP, poor values of the signal strength, which are measured on the selected AP, will eventually result in the initiation of a new handoff, thus correcting the error.
For more information on the LSSH scheme, please refer to our previously published work [2]. Figure 1 summarizes the LSSH scheme in the simplistic case of three APs.

3.2 The novel *Octopus* scheme

The Octopus scheme has been introduced to overcome LSSH's main drawbacks, that is, long decision periods, which may affect the locationing accuracy, and the lack of a mobility prediction support able to predict with reasonable anticipation the next AP the device is going to be connected to.

The basic idea behind Octopus is the same of LSSH, i.e., exploiting the advantages of both hard and soft handoff. The main difference lays in the decision phase, which is anticipated and concurrently performed with the initiation. During its normal operation, the MD monitors only one connection, as in hard handoff (panel A in figure 2). When a handoff event becomes probable, the MD

starts to monitor its neighboring APs, as in soft handoff. Specifically it connects, i.e., concurrently attaches "tentacles", to all them, hence the name "octopus" (panel B). During this phase, the device still keeps using the old AP. In addition, thanks to the multiple, concurrent connections, it can predict the AP it is going to be shortly connected to. Finally, when the handoff event is triggered, the MD can quickly decide the next AP, and release connections (panel C).

Octopus Initiation Differently from the LSSH scheme, the Octopus initiation is based on two α-count functions, which are evaluated concurrently on the same RSSI signal. The first function, called $\alpha^{(L)}_{LSSH}$ and based on $\alpha_{T_{LSSH}}$, *dec* and $S_{RSSI_{LSSH}}$ parameters, has the same purpose of the LSSH α-count: it triggers the handoff event as soon as $\alpha^{(L)}_{LSSH}$ becomes greater than $\alpha_{T_{LSSH}}$. The second function, called $\alpha^{(L)}_{Oct}$ and based on $\alpha_{T_{Oct}}$, *dec* and $S_{RSSI_{Oct}}$ parameters, triggers the anticipated decision phase as soon as $\alpha^{(L)}_{Oct}$ becomes greater than $\alpha_{T_{Oct}}$. Since the decision phase has to be triggered before the handoff event, it results:

$$\alpha_{T_{Oct}} = \frac{\alpha_{T_{LSSH}}}{K_T},\ K_T \geq 1 \tag{2}$$

In other terms, the threshold on the $\alpha^{(L)}_{Oct}$ (for the anticipated decision) has to be lower than the threshold on $\alpha^{(L)}_{LSSH}$ (for the initiation). The value of the K_T constant tunes the earliness of the anticipated decision phase: the bigger K_T, the earlier the decision phase will be undertaken. Similarly, it has to be:

$$S_{RSSI_{Oct}} = K_S \cdot S_{RSSI_{LSSH}},\ K_S \geq 1 \tag{3}$$

that is, $\alpha^{(L)}_{Oct}$ has to be less tolerant to RSSI degradations than $\alpha^{(L)}_{LSSH}$.
Figure 3 depicts how the octopus scheme defines two "zones" surrounding every AP: i) the pure *initiation* zone (the first circle surrounding the AP), where both $\alpha^{(L)}_{LSSH}$ and $\alpha^{(L)}_{Oct}$ are below their respective thresholds and where only the source AP RSSI is monitored, and ii) the *decision+initiation* zone (between the first and the second circle surrounding the AP), where $\alpha^{(L)}_{LSSH}$ is below its threshold, while $\alpha^{(L)}_{Oct}$ already reached its threshold, and where the monitoring of the neighboring APs is performed. In figure it is evidenced that $\alpha^{(L)}_{Oct}$ increases faster than $\alpha^{(L)}_{LSSH}$. This is due to the fact that $S_{RSSI_{Oct}} \geq S_{RSSI_{LSSH}}$, or equivalently, to the lower tolerance that $\alpha^{(L)}_{Oct}$ has with respect to RSSI degradations.

Octopus Decision and Mobility Prediction Support The decision phase is performed concurrently with the initiation phase. This way, the final decision is already available once the initiation phase ends, hence reducing the decision latency. During the decision phase, multiple connections are created to the neighboring APs $\{ng_1, ..., ng_n\}$. Each connection is monitored by a separate worker thread (the tentacle). The i-th worker thread is responsible to periodically i) read the RSSI level of the ng_i neighbor, ii) perform a moving average of the

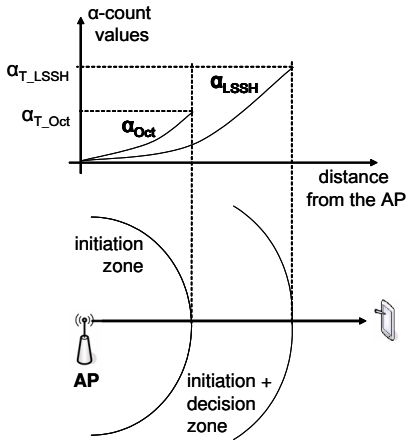

Fig. 3. The zones defined by the Octopus scheme and their relationship with $\alpha^{(L)}_{LSSH}$ and $\alpha^{(L)}_{Oct}$ functions

current reading with past readings (in order to filter out transient degradation phenomena), and iii) store the moving average in a shared structure. A manager thread (the octopus itself) periodically evaluates the best neighbor ng^* by getting the RSSI average of all neighbors from the shared structure. Once the $\alpha^{(L)}_{LSSH}$ triggers the handoff event, the current best neighbor ng_* is selected as the next neighbor. Consequently, all the worker threads are stopped and all the connections to other neighbors are dropped. If the decision cannot be made (e.g., the device movements are too fast to let the octopus create all the needed connections), the LSSH decision is performed as a back up mode.
The manager thread owns the information about the best neighbor during all the decision phase. The best neighbor can of course change during the decision phase, due to natural MD movements. In other terms, the manager thread "follows" device movements and it is thus able to know in advance, i.e., prior to the handoff execution, which is the device direction and hence the next AP that will be likely selected. Therefore, the octopus decision scheme naturally holds precious mobility prediction information, that can be easily provided to applications as soon as the decision phase starts, that is, while the device lays in the *initiation+decision* zone.

4 Experimental Results

This section shows the effectiveness of the novel Octopus scheme as compared to the LSSH scheme. In particular, the main objectives of the experiments are: (i) to show how Octopus practically eliminates the decision latency, and (ii) to demonstrate that Octopus obtains better location accuracy as compared to LSSH. To follow such objective, two are the parameters that need to be mea-

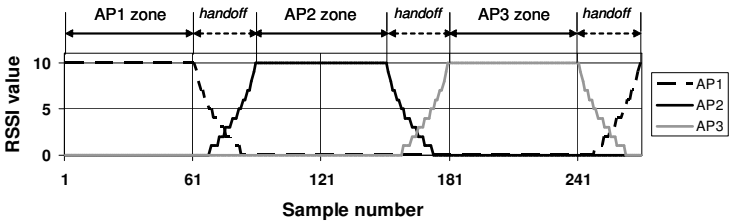

Fig. 4. RSSI reference traces, registered at a 1 m/s speed

sured: the decision time, both with Octopus and with LSSH, and the location estimate accuracy, which can be measured in terms of the percentage of location errors, both with Octopus and with LSSH. The percentage of location errors can be evaluated as:

$$\%\ of\ location\ errors = 100 \cdot \frac{N_{wl}}{N_{req}} \tag{4}$$

where N_{wl} is the number of times that the handoff management scheme (either LSSH or Octopus) returns a wrong location information with respect to the actual device location, and N_{req} is the total number of location requests. This parameter is particularly sensible to the device speed. The faster the device, the more is likely that the location estimate is wrong. In other terms, the faster the device, the less the handoff management scheme is able to follow device movements and to choose the right AP. Our experiments evidence how the Octopus scheme is more robust to device movements than the the LSSH scheme.

4.1 Prototype and Experimental Setting

The Octopus scheme has been implemented and integrated in a preexisting mobility management architecture, which is thoroughly described in our previous work [3]. Please refer to our web site: `www.mobilab.unina.it/Prototypes.htm` if you wish to download the last release of CLM and NCSOCKS including the Octopus scheme. Current implementation has been sufficiently tested only on Bluetooth wireless networks. Experimental results have been thus conducted over such networks.

In order to perform the above mentioned measures, we set up a simple testbed composed of three Bluetooth antennas acting as APs and one roaming, Bluetooth-enabled MD. In order to let each AP have two neighbors, we adopted a triangular topology. This way, every handoff procedure requires a decision between two APs. The distance between the antennas is set to 15 meters. Since we adopted Class 2 Bluetooth devices (with 10 meters transmission range), the overlapping zone between every couple of APs is set to 5 meters. To ease the measurement process at different device speeds, we adopted emulated RSSI readings by exploiting RSSI reference traces. The reference traces have been obtained by measuring actual RSSI values while the MD was moving around the testbed with a 1 m/s

Table 1. LSSH and Octopus decision latency

Handoff scheme	decision latency average (s)	decision latency std. dev. (s)
LSSH	5.279387	4.056980
Octopus	0.000137	0.000012

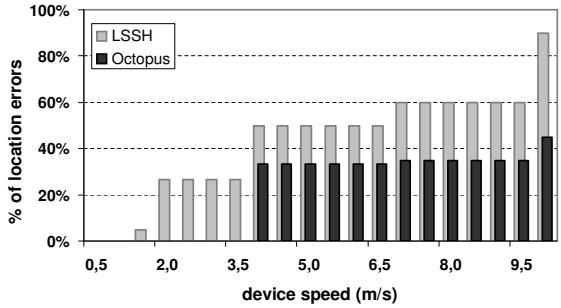

Fig. 5. LSSH and Octopus location accuracy as a function of the device speed

speed. The resulting traces are shown in figure 4. The emulated reading takes place by reading the RSSI value from the registered trace, rather than from the channel. To emulate different speeds, the reading from the traces is performed with different sampling periods. The sampling period is inversely proportional to the device speed. To exemplify, the double the sampling period, the half the emulated speed.

4.2 Results

Table 1 reports the decision latency we obtained with both LSSH and Octopus, with a 1 m/s speed. Due to its anticipated decision strategy, Octopus practically eliminates the decision latency. In particular, Octopus leads to a 99.99% improvement for the decision latency, which only accounts for the time spent by the manager thread to stop all worker threads and to return the last best neighbor estimate. In addition, as confirmed by the standard deviation latency estimates, the Octopus decision latency is by far more predictable than the LSSH decision latency. It is worth mentioning that the high decision latency value obtained for LSSH is particularly influenced by the Bluetooth technology, which involves relatively long connection set-up times.

As for the location accuracy, figure 5 shows the percentage of location errors as a function of the device speed. As expected, the percentage of location errors increases with the device speed. However, the figure clearly shows how Octopus outperforms LSSH by exhibiting a better robustness with respect to device fast movements. Specifically, Octopus starts to exhibit errors (about 30% errors on

Table 2. Octopus decision latency as a function of the device speed

Speed (m/s)	Sampling period (s)	No. of connections	Average latency (s)	
			Initiation	Decision
0,5	2	7	85,415200	0,000144
1	1	14	43,633162	0,000137
2,5	0.4	33	17,945656	0,000144
5	0.2	51	9,955675	0,365707
7,5	0.13333	54	10,061330	0,626858
10	0.1	55	9,859646	1,383985

the total number of location estimates) when the MD speed approaches 4 m/s (e.g., the average speed of a running human being). On the other hand, LSSH starts to deliver wrong estimates even for relatively slow speeds, around 2 m/s. This is basically due to the long decision latency. For relatively high speeds, e.g. from 6 m/s to 10 m/s, Octopus roughly exhibits half the errors of LSSH.

As a last result, table 2 shows the Octopus decision latency as a function of the speed. From a certain speed on (e.g. 5 m/s), the decision latency starts to assume higher values. Fast movements may indeed induce the Octopus decision to fail: for instance, the Octopus fails to establish all the needed connections on time. In these cases, the basic LSSH back-up scheme is adopted, leading to longer decision latencies. However, it is worth noting that performances are good for human walking/running speeds, that is, from 1 m/s up to 4 m/s. This means that the Octopus scheme can be successfully adopted in all those scenarios where the ubiquitous infrastructure "moves" at a human speed, e.g., wearable and portable devices embedded into human activities. However, it is worth recalling that the actual measures are relative to a Bluetooth-based scenario, where the results are partially influenced by long connection set-up times. Hence, the actual numbers (and the speed at which Octopus can successfully operate) depends on the adopted wireless technology. Besides actual numbers, we can reasonably claim that the improvement introduced with Octopus is valid in general terms.

5 Conclusions

This paper presented the driving ideas behind Octopus, a novel mobility prediction and soft handoff support for mobile ubiquitous applications. The novel scheme builds upon a previously proposed scheme, namely LSSH, and improves it by eliminating the need for time-consuming decision periods. This result has been made possible by the integration of mobility prediction, which also leads to the improvement of the locationing accuracy. Such improvements have been quantitatively demonstrated by means of experimental results on a real-world prototype.

Future efforts will concern a thorough evaluation of the Octopus scheme for other widely adopted wireless technologies, such as Wi-FI and ZigBee.

Acknowledgments

This work has been partially supported by the Italian Ministry for Education, University, and Research (MIUR) in the framework of the PRIN project "COMMUTA : Mutant hardware/software components for dynamically reconfigurable distributed systems", and in the framework of the "COSMIC" project "Centro di ricerca sui sistemi Open Source per le applicazioni ed i Servizi MIssion Critical". Authors are grateful to Gabriele Piantadosi and Daniele Zagordi for the precious help they profused in the implementation of the Octopus prototype and related experimental results.

References

1. D. Saha and A. Mukherjee. Pervasive computing: A paradigm for the 21st century. *IEEE Computer, IEEE Computer Society Press*, pages 25 – 31, March 2003.
2. M. Cinque, D. Cotroneo, and S. Russo. Achieving All the Time, Everywhere Access in Next-Generation Mobile Networks. *Mobile Networks and Applications, Special Issues on Mobility of Systems, Users, Data and Computing*, 9(2):29–39, April 2005.
3. M. Cinque, D. Cotroneo, and S. Russo. Mobility Management and Communication Support for Nomadic Applications. *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, Lecture Notes in Computer Science*, 3760:882–900, May 2005.
4. S. Baatz, M. Frank, R. Gopffarth, D. Kassatkine, P. Martini, M. Scheteilg, and A. Vilavaara. Handoff support for mobility with IP over Bluetooth. *Proc. of the 25th Annual IEEE Conf. on Local Computer Networks (LCN 2000)*, November 2000.
5. J. Tourrilhes and C. Carter. P-handoff: A protocol for fine grained peer-to-peer vertical handoff. *Proc. on the 13th IEEE Int. Symposium on Personal, Indoor and Mobile Radio Communcations (PIMRC '02)*, 2002.
6. M. L. George, L. J. Kallidukil, and J. M. Chung. Bluetooth handover control for roaming system applications. *Proc. of the 45th Midwest Symposium on Circuits and Systems. MWSCAS-2002.*, August 2002.
7. G. Bianchi, N. Blefari-Melazzi, M. Holzbock, Y. Fun Hu, A. Jahn, and Ray E. Sheriff. Design and validation of QoS aware mobile internet access procedures for heterogeneous networks. *Mobile Networks and Applications, Special Issues on Mobility of Systems, Users, Data and Computing*, 8(1):11–25, February 2003.
8. P. Bellavista, A. Corradi, and C. Giannelli. Mobility Prediction for Mobile Agent-based Service Continuity in the Wireless Internet. *Proc. of the 1st Int. Workshop on Mobility Aware Technologies and Applications (MATA'04)*, October 2004.
9. S-H Chung, H. Yoon, and J-W Cho. A Fast Handoff Scheme For IP over Bluetooth. *Proc. of 2002 Int. Conf. on Parallel Processing Workshops (ICPPW'02)*, 2002.
10. D. Saha et al. Mobility support in IP: a survey of related protocols. *IEEE Network*, 9(6), June 2004.
11. A. Bondavalli, S. Chiaradonna, F. Di Giandomenico, and F. Grandoni. Threshold-based mechanisms to discriminate transient from intermittent faults. *IEEE Transanction on Computers*, 49(3):230 – 245, March 2000.