Flexible and Dynamic Replication Control for Interdependent Distributed Real-Time Embedded Systems

Luís Nogueira, Luís Miguel Pinho, Jorge Coelho

Flexible and Dynamic Replication Control for Interdependent Distributed Real-Time Embedded Systems

Luís Nogueira[1], Luís Miguel Pinho[1], and Jorge Coelho[2]

[1] CISTER Research Centre
[2] LIACC
School of Engineering, Polytechnic Institute of Porto, Portugal
{lmn,lmp,jmn}@isep.ipp.pt

Abstract. Replication is a proven concept for increasing the availability of distributed systems. However, actively replicating every software component in distributed embedded systems may not be a feasible approach. Not only the available resources are often limited, but also the imposed overhead could significantly degrade the system's performance.
This paper proposes heuristics to dynamically determine which components to replicate based on their significance to the system as a whole, its consequent number of passive replicas, and where to place those replicas in the network. The activation of passive replicas is coordinated through a fast convergence protocol that reduces the complexity of the needed interactions among nodes until a new collective global service solution is determined.

1 Introduction

The highly dynamic and unpredictable nature of open distributed real-time embedded systems can lead to a highly volatile environment where QoS provision needs to adapt seamlessly to changing resource levels [1].Some of the difficulties arise from the fact that the mix of independently developed applications and their aggregate resource and timing requirements are unknown until runtime but, still, a timely answer to events must be provided in order to guarantee a desired level of performance. Our previous work [2] applied concepts of cooperative QoS-aware computing to address such challenges, emerging as a promising distributed computing paradigm to face the stringent demands on resources and performance of new embedded real-time systems. Available software components can be shared among different services and can be adapted at runtime to varying operational conditions, enhancing the efficiency in the use of the available resources.

Nevertheless, it is imperative to accept that failures can and will occur, even in meticulously designed systems, and design proper measures to counter those failures [3]. Software replication has some advantages over other fault tolerance solutions in distributed environments, providing the shortest recovery delays, it is less intrusive with respect to execution time, it scales much better, and is relatively generic and transparent to the application domain [4]. However, actively replicating all software components,

independently of their significance to the overall system, may be infeasible in some embedded systems due to the scale of their timing, cost, and resource constraints [5].

This paper is then motivated by the need to develop a flexible and cost-effective fault tolerance solution with a significant lower overhead compared to a strict active redundancy-based approach. The term *cost-effective* implies that we want to achieve a high error coverage with the minimum amount of redundancy. The paper proposes low runtime complexity heuristics to (i) dynamically determine which components to replicate based on their significance to the system as a whole; (ii) determine a number of replicas proportional to the components' significance degree; and (iii) select the location of those replicas based on collected information about the nodes' availability as the system progresses. To quantitatively study the effectiveness of the proposed approach an extensive number of simulation runs was analysed. The results show that even simple heuristics with low runtime complexity can achieve a reasonably higher system's availability than static offline decisions when lower replication ratios are imposed due to resource or cost limitations.

One of the advantages of passive replication is that it can be implemented without the use of complex replica consistency protocols [6, 7]. Nevertheless, consider the case where the quality of the produced output of a particular component depends not only on the amount and type of used resources but also on the quality of the inputs being sent by other components in the system [8]. If a primary replica is found to be faulty, a new primary must be elected from the set of passive backup ones and the execution restarted from the last saved state. However, it is not guaranteed that the new primary will be able to locally reserve the needed resources to output the same QoS level that was being produced by the old primary. In such cases, the need of coordination arises in order to preserve the correct functionality of the service's distributed execution [9, 10]. This paper proposes a distributed coordination protocol that rapidly converges to a new globally consistent service solution by (i) reducing the needed interactions among nodes; and (ii) compensating for a decrease in input quality by an increase in the amount of used resources in key components in interdependency graphs.

2 System model

We understand a service $S = \{c_1, c_2, \ldots, c_n\}$ as a set of software components c_i being cooperatively executed by a coalition of nodes [2]. Each component c_i is defined by its functionality, is able to send and receive messages, is available at a certain point of the network, and has a set of QoS parameters that can be changed in order to adapt service provisioning to a dynamically changing environment. Each subset of QoS parameters that relates to a single aspect of service quality is named as a *QoS dimension*. Each of these QoS dimensions has different resource requirements for each possible level of service quality. We make the reasonable assumption that services' execution modes associated with higher QoS levels require higher resource amounts.

Users provide a single specification of their own range of acceptable QoS quality levels Q for a complete service S, ranging from a desired QoS level $L_{desired}$ to the maximum tolerable service degradation, specified by a minimum acceptable QoS level $L_{minimum}$, without having to understand the individual components that make up the

service. Nodes dynamically group themselves into a new coalition, cooperatively allocating resources to each new service and establishing an initial Service Level Agreement (SLA) that maximises the satisfaction of the user's QoS constraints associated with the new service while minimises the impact on the global system's QoS caused by the new service's arrival [2]. Within a coalition, each component $c_i \in S$ will then be executed at a QoS level $L_{minimum} \leq Q^i_{val} \leq L_{desired}$ at a node n_i. This relation is represented by a triple (n_i, c_i, Q^i_{val}).

There may exist QoS interdependencies among two or more of the multiple QoS dimensions of a service S, both within a component and among components that may be in the same or in different nodes. Given two QoS dimensions, Q_a and Q_b, a QoS dimension Q_a is said to be dependent on another dimension Q_b if a change along the dimension Q_b will increase the needed resource demand to achieve the quality level previously achieved along Q_a [11]. Furthermore, we consider the existence of feasible QoS regions [8]. A region of output quality $[q(O)_1, q(O)_2]$ is defined as the QoS level that can be provided by a component when provided with sufficient input quality and resources. Within a QoS region, it may be possible to keep the current output quality level by compensating a decreased input quality by an increase in the amount of used resources or vice versa.

The set of QoS interdependencies among components $c_i \in S$ is represented as a connected graph $\mathcal{G}_\mathcal{S} = (\mathcal{V}_S, \mathcal{E}_S)$, on top of the service's distribution graph, where each vertex $v_i \in \mathcal{V}_\mathcal{S}$ represents a component c_i and a directed edge $e_i \in \mathcal{E}_\mathcal{S}$ from c_j to c_k indicates that c_k is functionally dependent on c_j. Within $\mathcal{G}_\mathcal{S} = (\mathcal{V}_S, \mathcal{E}_S)$, we call *cut-vertex* to a component $c_i \in \mathcal{V}_\mathcal{S}$, if the removal of that component divides $\mathcal{G}_S$ in two separate connected graphs.

A component c_i is only aware of the set of inputs $I_{c_i} = \{(c_j, Q^j_{val}), \ldots, (c_k, Q^k_{val})\}$ describing the quality of all of its inputs coming from precedent components in $\mathcal{G}_\mathcal{W}$ and the set of outputs $O_{c_i} = \{(c_l, Q^l_{val}), \ldots, (c_p, Q^p_{val})\}$, describing the quality of all of its outputs sent to its successor components in $\mathcal{G}_\mathcal{W}$. As such, no global knowledge is required for coordinating the activation of a backup replica after a failure of a primary component c_i.

3 Towards a flexible and adaptive replication control

The possibility of partial failures is a fundamental characteristic of distributed applications, even more so in open environments. A sub-domain of reliability, fault tolerance aims at allowing a system to survive in spite of faults, *i.e.* after a fault has occurred, by means of redundancy. In this paper, we consider a failure to be when a software component stops producing output.

Replication is an effective way to achieve fault tolerance for such type of failure [12]. In fault-tolerant real-time systems, using active replication schemes, where several replicas run simultaneously, has been common [13]. Even if errors are detected in some of the replicas, the non-erroneous ones will still be able to produce results within the deadlines. On the negative side, running several replicas simultaneously is costly and can be infeasible or undesirable in distributed embedded systems [5] due to the limited resource availability and excessive overhead. Thus, a different approach is needed.

Passive replication [14] minimises resource consumption by only activating redundant replicas in case of failures, as typically providing and applying state updates is less resource demanding than requesting execution. As such, passive replication is appealing for soft real-time systems that cannot afford the cost of maintaining active replicas and tolerate an increased recovery time [15]. Nevertheless, it may still be possible to tolerate faults within deadlines, thus improving the system's reliability without using a more resource consuming fault-tolerance mechanism [16].

However, most of the existing solutions for passive fault tolerance are usually designed and configured at design time, explicitly and statically identifying the most critical components and their number of replicas, lacking the needed flexibility to handle the runtime dynamics of open distributed real-time embedded systems [6]. Distributed real-time embedded systems often consist of several independently developed components, shared across applications and whose criticality may evolve dynamically during the course of computation. As such, offline decisions on the number and allocation of replicas may be inadequate after the system has been executing for some time.

Consequently, the problem consists in finding a replication scheme which minimises the probability of failure of the most important components without replicating every software component. This involves the study of mechanisms to determine which components should be replicated, the quantity of replicas to be made, and where to deploy such replicas [17]. As such, the benefits of replication in open dynamic resource-constrained environments are a complex function of the number of replicas, the placement of those replicas, the selected replica consistency protocol, and the availability and performance characteristics of the nodes and networks composing the system. Since replica consistency protocols are relatively well understood [18, 7, 6], we will not consider them in the remainder of this paper.

Thus, assuming that a mechanism exists for keeping passive replicas consistent, how can we make use of passive replication for increasing the reliability of distributed resource-constrained embedded systems where it may not be possible to replicate every available component? Our approach is based on the concept of significance, a value associated to each component which reflects the effects of its failure on the overall system. Intuitively, the more a component c_i has other components depending on it, the more it is significant to the system as a whole. Then, the significance degree w_i of a component c_i can be computed as the aggregation of the interdependencies of other components on it, determining the usefulness of its outputs to all the components which depend on it to perform their tasks.

More formally, given $S_{\mathcal{G}} = \{\mathcal{G}_1, \ldots, \mathcal{G}_n\}$, the set of connected graphs of interdependencies between components for a given system, and $\mathcal{O}_{\mathcal{G}_j}(c_i)$, the out-degree of a node $c_i \in \mathcal{G}_j$, the significance wi of c_i is given by Equation 1.

$$w_i = \sum_{k=1}^{n} \mathcal{O}_{\mathcal{G}_k}(c_i) \tag{1}$$

Once the significance of each component to the system has been estimated, the decision on which components to replicate and the correspondent number of passive replicas must be taken. Equation 2 determines a number of replicas for a component c_i which is directly proportional to the component's significance degree w_i and to the

maximum number of possible replicas max_{c_i}[3] and inversely proportional to the sum of the significance degree of all components in the system W.

$$n^{c_i} = \left\lceil \frac{w_i * max_{c_i}}{W} \right\rceil \tag{2}$$

The next step is to determine a strategy for placing those replicas in the network. Consider the effects of placing replicas on unreliable nodes. The resulting unreliability of those replicas will usually require replica consistency protocols to work harder [6], increasing network traffic and processing overheads. Thus, not only will the system's performance suffer but its availability may actually decrease, despite the increased number of available components through replication [18]. However, an optimal replica placement in a distributed system can be classified as a NP-hard discrete location problem. Consequently, several heuristic strategies which do not have a guarantee in terms of solution quality or running time, but provide a robust approach to obtaining a high quality solution to problems of a realistic size in reasonable time have been investigated, independently of the followed replication approach [19, 20]. Nevertheless, it is our belief that static offline approaches are inadequate for open real-time systems, where the environment dynamically changes as the system progresses. As such, a placement of a replica which was correct when a service was started may be incorrect after it has been executing for some time.

Two gross measures of the reliability of a node are its Mean Time To Failure (MTTF) and its Mean Time To Recovery (MTTR) [17]. We propose to use those measures to dynamically allocate the set of replicas of a component c_i based on the expected availability of nodes in the system. The utility $0 \leq u_k^{r_j^i} \leq 1$ of allocating a passive replica r_j^i of a component c_i to a node n_k is then defined by the probability of its availability during the system's execution, given by Equation 3. Utilities range from zero, the value of a completely unavailable node, to one, the value of a totally available node.

$$u_k^{r_j^i} = \frac{MTTF_k}{MTTF_k + MTTR_k} \tag{3}$$

Having the utility of each possible allocation, the probability of failure of a given set of replicas $R_i = r_1^i, r_2^i, \ldots, r_{n^{c_i}}^i$ is determined by Equation 4.

$$F(R_i) = (1 - u_1^i) * (1 - u_2^i) * \ldots * (1 - u_{n^{c_i}}^i) \tag{4}$$

The system will then allocate the set of replicas $R_i = r_1^i, r_2^i, \ldots, r_{n^{c_i}}^i$ such that its probability of failure $F(R_i)$ is minimal among all the possible allocation sets. In order to keep this allocation as up-to-date as possible, nodes have to be monitored as the system runs. If reliability of a replica set strays outside a predefined tolerance value a reconfiguration of the allocation set should be performed.

[3] max_{c_i} is given by the number of nodes in a heterogeneous environment which have the needed type of resources to execute the component c_i.

4 Coordinated activation of passive backup replicas

While passive replication is appealing for systems that cannot afford the cost of maintaining active replicas, the requirement to provide both high availability, strong state consistency, and satisfactory response times during the non-failure cases is conflicting in many ways. In fact, response times perceived by applications will depend on the time taken by the primary replica to synchronise its state with that of the slowest backup replica, even if low complexity replica consistency protocols are used [6, 7]

To overcome this limitation a possibility is for the backup replicas' state to be made consistent only during a failure's recovery, which significantly improves response times and saves resources during the non-failure cases. We recognise, however, that extra time must be spent to activate a new primary due to the significantly weaker consistency model. The problem is even more challenging when activating replicas in interdependent cooperative coalitions where the output produced by a component may depend not only on the amount and type of used resources but also on the quality of the received inputs [21]. Nevertheless, the complexity of the needed interactions among nodes until a new collective global service solution is determined can be reduced through a fast convergence protocol, while benefiting from a better performance on non-failure cases by only updating the backup replicas' state on a failure of a primary one.

Ideally, whenever a primary components fails it is elected as a new primary a backup which is able to obtain the needed resources to output the QoS level that was being produced by the old primary replica. However, due to the heterogeneity and dynamically varying workloads of nodes in the system, it is not guaranteed that at least one of the backups will be able to locally reserve the needed resources to output such quality level. Such feasibility is determined by the anytime local QoS optimisation algorithm of [2], which aims to minimise the impact of the activation of a new component on the currently provided QoS level of previously activated components at a particular node.

Thus, whenever the required QoS level cannot be assured by the new primary replica there is a need to ensure that individual substitutions of a component will produce a globally acceptable solution for the entire distributed interdependent service [22]. While there has been a great deal of research in several aspects of runtime coordination in embedded real-time systems [23–26], to the best of our knowledge we are the first to address the specific problem of coordinating the activation of passive replicas in interdependent distributed environments with real-time constraints. Here, the term *coordinated activation* refers to the ability of a distributed system to invoke adaptive actions on multiple nodes in a coordinated manner so as to achieve a new service configuration.

Without any central coordination entity, the collective adaptation behaviour must emerge from local interactions among components. This is typically accomplished by the exchange of multiple messages to ensure that all involved components make the same decision about whether and how to adapt [26]. One main challenge is controlling this exchange of information in order to achieve a convergence to a globally consistent solution. It may be difficult to predict the exact behaviour of the system taken as a whole due to the large number of possible non-deterministic ways in which the system can behave [27]. Whenever real-time decision making is in order, a timely answer to events suggests that after some finite and bounded time the global adaptation process

converges to a consistent solution. We propose to achieve a time-bounded convergence to a global solution through a regulated decentralised coordination protocol defined by the following three phases:

1. **New primary selection.** Let Q^i_{val} be the QoS level that was being outputted by the primary replica of component $c_i \in S$ that has failed. If no passive replica of c_i is able to output the same QoS level, select the one which is able to output the QoS level $Q^i_{val'} < Q^i_{val}$ closer to Q^i_{val}. A coordination message is sent to affected partners in the coalition.
2. **Local adaptation.** Affected partners, executing any interdependent component $c_j \in S$, become aware of the new output values $Q^i_{val'}$ of c_i and recompute their local set of SLAs using the anytime QoS optimisation approach of [2]. We assume that coalition partners are willing to collaborate in order to achieve a global coalition's consistency, even if this might reduce the utility of their local optimisations.
3. **Coordinated adaptation.** Coalition partners affected by the decrease to $Q^i_{val'}$ in the path to the next cut-vertex $c_c \in S$ may be able to continue to output their current QoS level despite the downgraded input by compensating with an increased resource usage while others may not. If the next cut-vertex c_c is unable maintain its current QoS level then all the precedent components c_j which are compensating their downgraded inputs with an increased resource usage can downgrade to $Q^j_{val'}$ since their effort is useless.

Note that, if a cut-vertex c_c, despite the change in the current quality of some or all of its inputs, is able to maintain its current QoS level there is no need to further propagate the required coordination along the dependency graph $\mathcal{G}_\mathcal{S}$, reducing the needed time to achieve a new acceptable global solution. On the other hand, if c_j is forced to downgrade its outputs due to the lower quality of its inputs, global resource usage is optimised by aborting useless compensations at precedent components in $\mathcal{G}_\mathcal{S}$ and the coordination request is propagated. Thus, the core idea behind the proposed decentralised coordination model is to support distributed systems composed of autonomous individual nodes working without any central control but still producing the desired function as a whole. The proposed approach is sufficient to reach a new acceptable, non-optimal, service configuration in a time-bounded manner.

A formal validation of the properties and correctness of the proposed coordination model, as well as a detailed example of its operation, are available in [28].

5 Evaluation

An application that captures, compresses and transmits frames of video to end users, which may use a diversity of end devices and have different sets of QoS preferences, was used to evaluate the efficiency of the proposed passive replication mechanism with coordinated activations, with a special attention being devoted to introduce a high variability in the characteristics of the considered scenarios. The application is composed by a set of components to collect the data, a set of compression components to gather and compress the data sent from multiple sources, a set of transmission components to

transmit the data over the network, a set of decompression components to convert the data into the user's specified format, and a set of components to display the data in the end device [2].

The number of simultaneous nodes in the system randomly varied, in each simulation run, from 10 to 100. For each node, the type and amount of available resources, creating a distributed heterogeneous environment. Nodes failed and recovered according to their MTTF and MTTR reliability values, which were randomly assigned when the nodes were created (it was ensured that each node had an availability between 60% and 99%). Each node was running a prototype implementation of the CooperatES framework [29], with a fixed set of mappings between requested QoS levels and resource requirements. At randomly selected nodes, new service requests from 5 to 20 simultaneous users were randomly generated, dynamically generating different amounts of load and resource availability. Based on each user's service request, coalitions of 4 to 20 components were formed [2] and a randomly percentage of the connections among those components was selected as a QoS interdependency.

In order to assess the efficiency of the proposed dynamic replication control as opposed to an offline static replication in dynamic resource-constrained environments, we considered the number of coalitions which where able to recover from failures and conclude their cooperative executions as a function of the used replication ratio. The reported results were observed from multiple and independent simulation runs, with initial conditions and parameters, but different seeds for the random values used to drive the simulations, obtaining independent and identically distributed variables. The mean values of all generated samples were used to produce the charts.

In the first study, we evaluated the achieved system's availability with the proposed dynamic replication control based on components' significance and with a static offline approach in which the components to replicate and their number of replicas is fixed by the system's designer at a coalition's initialisation phase [17]. At each simulation run, if the primary replica of a component c_i failed during operation, a new primary was selected among the set of passive backups. If this was not possible, all the coalitions depending on c_i were aborted. In this study, replicas were also randomly allocated among eligible nodes with the dynamic replication control policy.

Figure 1 clearly shows that our strategy is more accurate to determine and replicate the most significant components than a static offline one, particularly with lower replication ratios. Thus, when lower replication ratios are imposed due to resource or cost limitations, a higher availability can be achieved if the selection of which components to replicate and their number of replicas depends on their significance to the system as a whole. In open and dynamic environments, such significance can be determined online as the aggregation of all the other components that depend on a particular component to perform their tasks.

A second study evaluated the impact of the selected replicas' placement strategy on the achieved system's availability for a given replication ratio. The study compared the performance of the proposed allocation heuristic based on collected information about the nodes' availability as the system evolves with a random policy in which the placement of the generated replicas is fixed offline [30]. The decision on which components to replicate and their number of replicas followed the same dynamic and static

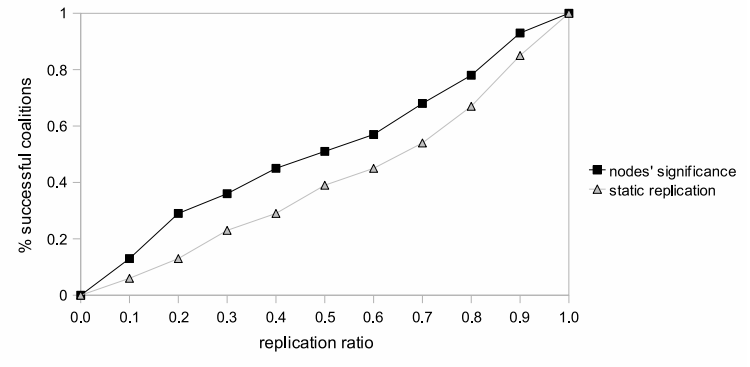

Fig. 1. Impact of the chosen replication control strategy on the system's availability

approaches of the first study. For the dynamic allocation strategy, a tolerance value for the availability of each replica set was randomly generated at each simulation run. If this tolerance was surpassed, a reassignment of replicas was performed. The results were plotted in Figure 2.

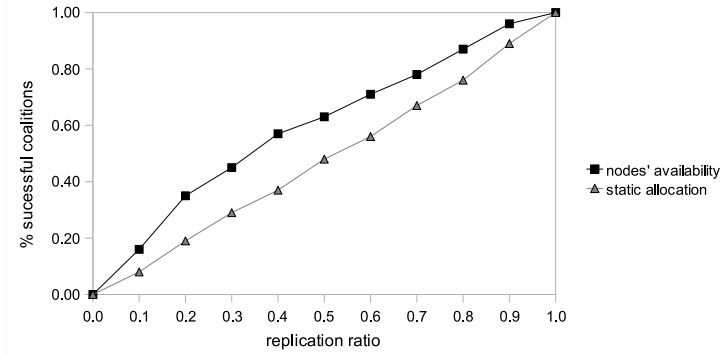

Fig. 2. Impact of the chosen replica allocation strategy on the system's availability

It is then possible to conclude that the location of replicas is a relevant factor for the system's availability as a whole. The proposed dynamic replicate allocation that takes into account the nodes' reliability over time always achieves a better performance than an offline static allocation policy in open and dynamic environments. Furthermore, a comparison of Figures 1 and 2 shows that even though an improvement in availability can be achieved by increasing the replication ratio, the impact of replicas' placement is quite significant.

A third study evaluated the efficiency of the proposed coordinated activation of interdependent passive replicas in comparison to a typical centralised coordination approach [31] in which a system-wide controller coordinates resource allocations among multiple nodes. The average results of all simulation runs for the different coalition sizes and percentages of interdependencies among components are plotted in Figure 3. As expected, both coordination approaches need more time as the complexity of the service's topology increases. Nevertheless, the proposed decentralised coordination model is faster to determine the overall coordination result in all the evaluated services' topologies, needing approximately 75% of the time spent by the centralised near-optimal model.

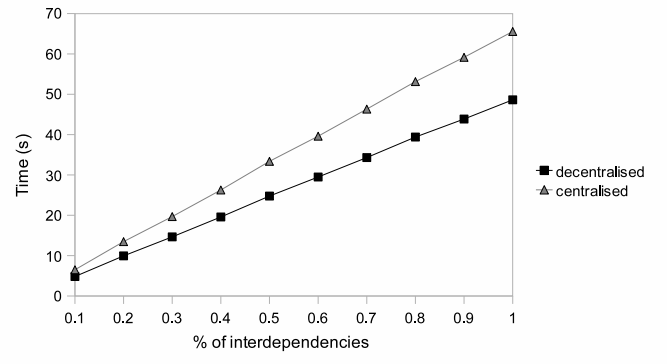

Fig. 3. Time for a coordinated replica activation

6 Conclusions

The availability and performance of open distributed embedded system is significantly affected by the choice of the replication control strategy and placement of the generated replicas. Due to its low resource consumption, passive replication is appealing for embedded real-time systems that cannot afford the cost of maintaining active replicas and need not assure hard real-time performance. The proposed heuristics based on the components' significance to the overall system and on nodes' reliability history have a low runtime complexity and achieve a reasonably higher system's availability than static offline decisions, particularly when lower replication ratios are imposed due to resource or cost limitations.

Another challenge is to transfer the state of a distributed service to a new globally acceptable configuration whenever a new elected primary cannot provide the same QoS level that was being outputted by the old primary that was found to be faulty. The proposed distributed coordination model reduces the complexity of the needed interactions among nodes and is faster to converge to a globally acceptable solution than a traditional centralised coordination approach.

Acknowledgements

This work was supported by FCT through the CISTER Research Unit - FCT UI 608 and the research project CooperatES - PTDC/EIA/71624/2006

References

1. Anane, R.: Autonomic behaviour in qos management. In: Proceedings of the Third International Conference on Autonomic and Autonomous Systems, Athens, Greece (June 2007) 57
2. Nogueira, L., Pinho, L.M.: Time-bounded distributed qos-aware service configuration in heterogeneous cooperative environments. Journal of Parallel and Distributed Computing **69**(6) (June 2009) 491–507
3. Kopetz, H.: Real-Time Systems: Design Principles for Distributed Embedded Applications. Kluwer (1997)
4. Guerraoui, R., Schiper, A.: Software-based replication for fault tolerance. IEE Computer **30** (1997) 68–74
5. Cai, Z., Kumar, V., Cooper, B.F., Eisenhauer, G., Schwan, K., Strom, R.E.: Utility-driven proactive management of availability in enterprise-scale information flows. In: Proceedings of the ACM/IFIP/USENIX 2006 International Conference on Middleware, Springer-Verlag (2006) 382–403
6. Rubel, P., Gillen, M., Loyall, J., Schantz, R., Gokhale, A., Balasubramanian, J., Paulos, A., Narasimhan, P.: Fault tolerant approaches for distributed real-time and embedded systems. In: Proceedings of the 2007 Military Communications Conference, Orlando, Florida, USA (October 2007) 1–8
7. de Juan-Marin, R., Decker, H., Munoz-Esco, F.D.: Revisiting hot passive replication. In: Proceedings of the 2nd International Conference on Availability, Reliability and Security. (April 2007) 93–102
8. Shankar, M., de Miguel, M., Liu, J.W.S.: An end-to-end qos management architecture. In: Proceedings of the 5th IEEE Real-Time Technology and Applications Symposium, Washington, DC, USA, IEEE Computer Society (1999) 176–191
9. Allen, G., Dramlitsch, T., Foster, I., Karonis, N.T., Ripeanu, M., Seidel, E., Toonen, B.: Supporting efficient execution in heterogeneous distributed computing environments with cactus and globus. In: Proceedings of the 2001 ACM/IEEE conference on Supercomputing. (November 2001) 52–52
10. Ensink, B., Adve, V.: Coordinating adaptations in distributed systems. In: Proceedings of the 24th International Conference on Distributed Computing Systems, Tokyo, Japan (March 2004) 446–455
11. Rajkumar, R., Lee, C., Lehoczky, J., Siewiorek, D.: A resource allocation model for qos management. In: Proceedings of the 18th IEEE Real-Time Systems Symposium, IEEE Computer Society (1997) 298
12. Powell, D., ed.: A generic fault-tolerant architecture for real-time dependable systems. Kluwer Academic Publishers, Norwell, MA, USA (2001)
13. Pinho, L.M., Vasques, F., Wellings, A.: Replication management in reliable real-time systems. Real-Time Systems **26**(3) (2004) 261–296
14. Budhiraja, N., Marzullo, K., Schneider, F.B., Toueg, S.: The primary-backup approach. Distributed systems (2nd Ed.) (1993) 199–216

15. Balasubramanian, J., Tambe, S., Lu, C., Gokhale, A., Gill, C., Schmidt, D.C.: Adaptive failover for real-time middleware with passive replication. In: Proceedings of the 15th IEEE Real-Time and Embedded Technology and Applications Symposium, IEEE Computer Society (April 2009) 118–127
16. Tjora, A., Skavhaug, A.: A general mathematical model for run-time distributions in a passively replicated fault tolerant system. In: Proceedings of the 15th Euromicro Conference on Real-Time Systems, Porto,Portugal (July 2003) 295–301
17. Little, M.C., McCue, D.: The replica management system: a scheme for flexible and dynamic replication. In: Proceedings of the 2nd International Workshop on Configurable Distributed Systems. (April 1994) 46–57
18. Little, M.C.: Object Replication in a Distributed System. PhD thesis, Department of Computing Science, Newcastle University (September 1991)
19. Streichert, T., Glaß, M., Wanka, R., Haubelt, C., Teich, J.: Topology-aware replica placement in fault-tolerant embedded networks. In: Proceedings of the 21st International Conference on Architecture of Computing Systems, Dresden, Germany (February 2008) 23–37
20. Fu, W., Xiao, N., Lu, X.: A quantitative survey on qos-aware replica placement. In: Proceedings of the 7th International Conference on Grid and Cooperative Computing, Shenzhen, China (October 2008) 281–286
21. Nogueira, L., Pinho, L.M.: Dynamic qos adaptation of inter-dependent task sets in cooperative embedded systems. In: Proceedings of the 2nd ACM International Conference on Autonomic Computing and Communication Systems, Turin,Italy (September 2008) 97
22. Gelernter, D., Carriero, N.: Coordination languages and their significance. Communications of the ACM **35**(2) (1992) 96–107
23. Dorigo, M., Caro, G.D.: The ant colony optimization meta-heuristic. New ideas in optimization (1999) 11–32
24. De Wolf, T., Holvoet, T.: Towards autonomic computing: agent-based modelling, dynamical systems analysis, and decentralised control. Proceedings of the IEEE International Conference on Industrial Informatics (August 2003) 470–479
25. Boutilier, C., Das, R., Kephart, J.O., Tesauro, G., Walsh, W.E.: Cooperative negotiation in autonomic systems using incremental utility elicitation. In: In Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence, Acapulco,Mexico (August 2003) 89–97
26. Dowling, J., Haridi, S.: in Reinforcement Learning: Theory and Applications. In: Decentralized Reinforcement Learning for the Online Optimization of Distributed Systems. I-Tech Education and Publishing, Vienna, Austria (2008) 142–167
27. Serugendo, G.D.M.: Handbook of Research on Nature Inspired Computing for Economy and Management. In: Autonomous Systems with Emergent Behaviour. Idea Group, Inc., Hershey-PA, USA (September 2006) 429–443
28. Nogueira, L., Pinho, L.M., Coelho, J.: Towards a flexible and dynamic replication control for distributed real-time embedded systems with qos interdependencies. Technical report, CISTER Research Centre, Available at http://www.cister.isep.ipp.pt/docs/ (February 2010)
29. Pinho, L.M., Nogueira, L., Barbosa, R.: An ada framework for qos-aware applications. In: Proceedings of the 10th Ada-Europe International Conference on Reliable Software Technologies, York, UK (June 2005) 25–38
30. On, G., Schmitt, J., Steinmetz, R.: Quality of availability : replica placement for widely distributed systems. In: Proceedings of the 11th International Workshop on Quality of Service, Monterey, CA (June 2003) 325–342
31. Rohloff, K., Schantz, R., Gabay, Y.: High-level dynamic resource management for distributed, real-time embedded systems. In: Proceedings of the 2007 Summer Computer Simulation Conference. (July 2007) 749–756