Towards Middleware-Based Cooperation Topologies for the Next Generation of CPS

http://dx.doi.org/10.3991/ijoe.v8iS4.2273

I. Etxeberria-Agiriano[1], I. Calvo[1], A. Noguero[2] and E. Zulueta[1]
[1] University of the Basque Country (UPV/EHU), Vitoria-Gasteiz, Spain
[2] Tecnalia, Bilbao, Spain

***Abstract*—Cyber-Physical Systems (CPS) integrate embedded computers that control physical processes. Application domains for CPS may be found in intelligent buildings, healthcare, transportation and factory automation, among many others. Typically, they are based on low profile computing elements, such as sensors and actuators that must communicate to carry out complex tasks. They must address certain issues such as managing available resources and service redundancy, as well as solving heterogeneity. In particular, managing communication issues can be relatively complex. In this scenario, middleware technologies can help developers in the design of state-of-the-art CPS. This work describes the design principles of CPS that require cooperation. More specifically, it presents a generic family of logical information exchange and cooperation topologies capable of adapting dynamically to changes in the environment. These topologies may be implemented on top of several middleware specifications as a means for managing distributed resources and service redundancy of CPS at run-time.**

***Index Terms*— CPS, Middleware, Fault-Tolerance, Energy Efficiency.**

I. INTRODUCTION

The term "Cyber-Physical Systems" (CPS) was coined around 2006 by researchers from different disciplines, mainly real-time systems, network communications, hybrid systems and control systems (see Fig. 1). Nowadays, CPS are becoming a hot research topic being funded with a growing number of projects granted by different organizations [1, 2].

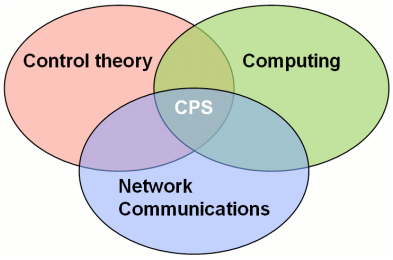

Figure 1. Main disciplines involved in CPS.

CPS can be considered the next step forward in computing, involving control/computing co-design [3]. More specifically, CPS are integrations of computation and physical processes [4]. They include some relevant characteristics such as taking care of computation performing time and the intrinsic concurrency. Additional characterization of CPS is provided by Shi et al. [5]. They describe current research on the subject with some paradigmatic applications. Wu et al. [6] review CPS platforms and wireless sensor networks. They identify several current technical challenges including Quality of Service (QoS) requirements.

Unfortunately, there are still a surprisingly small amount of theory and tools that help designers to build the next generation systems in an efficient way [3]. An interesting review of the different technologies and approaches used to build these systems may be found in [7, 8].

Regarding the application domains, base technologies are needed to build large-scale safety-critical CPS correctly, affordably, flexibly and on schedule [9]. Some of these domains include intelligent buildings, healthcare, transportation systems, process control, factory automation or electrical power grids, just to name a few [5, 7] (see Fig. 2).

During the last years distributed embedded systems have increased in size. They have shifted from centralized small applications, based on real-time operating systems capable of handling multitasking and basic operations locally, to large computer-controlled systems, such as those found in nation-wide power grids, to supply power to billions of devices simultaneously or world-wide communication networks [3, 4]. These new systems require the development of solid theory and semantics that ease its construction. Thus, some of the main abstractions introduced by the scheduling theories developed in the early 1970s, such as RMS (*Rate Monotonic Scheduling*) or EDF (*Earliest Deadline First*), are still used to build the control application software. However, these abstractions frequently do not suit the models of the physical entities behind [4]. On top of that, it is not a matter of over scaling the resulting systems: adopted solutions must be efficient, applying Ockham's razor.

In this scenario, the designers of CPS face several challenges. Namely, (1) heterogeneous network technologies must communicate efficiently; (2) individual elements may fail but the whole system must be reliable and able to perform correctly in a degraded reconfigured mode; (3) maintenance corrective actions could be delayed without service disruption when they are not critical; (4) since energy consumption can be a limitation in locations with difficult access, nodes with higher energy load or easier to replace can be identified and chosen to carry out the computations more frequently.

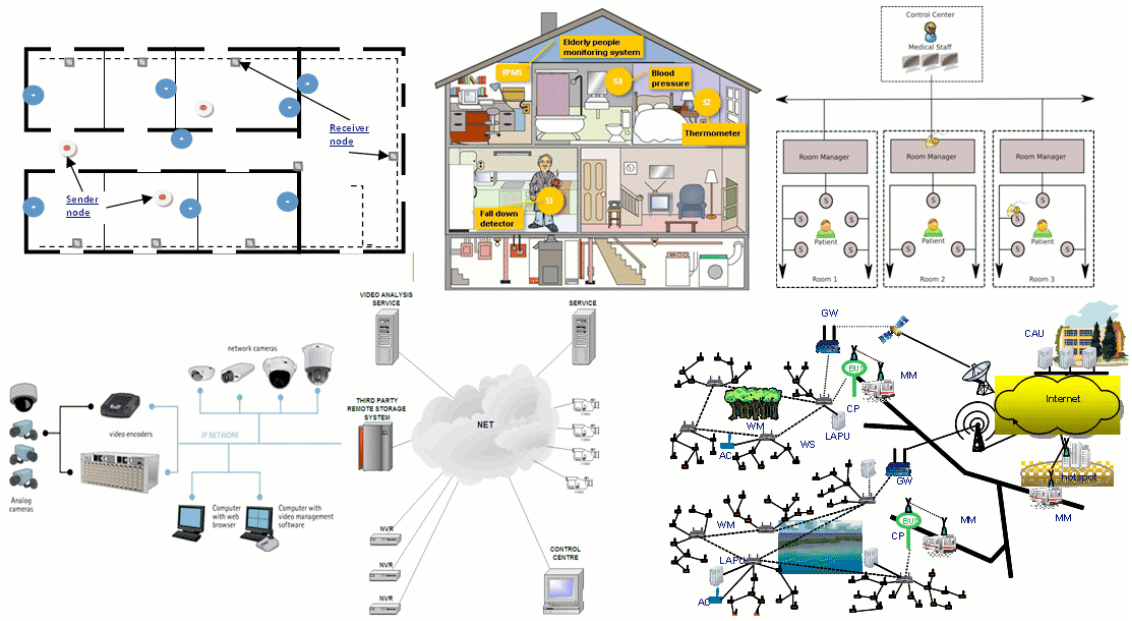

Figure 2. Different types of CPS applications.

Communication issues related to CPS are of special interest in this work. Some authors such as [4] propose the adoption of new radical design approaches that match the specific requirements of CPS; other authors, such as [10] recommend using more pragmatic approaches, at least in short to medium term. In particular, [10] propose the use of worldwide standard technologies such as IP (Internet Protocols) and IEEE 802.11 (WiFi) for CPS. Although these standards are relatively poor in terms of efficiency and Quality of Service (QoS), several patches have been proposed for IP (such as Integrated Services, Differentiated Services, etc) as well as for WiFi (e.g. IEEE 802.11e extension) to enhance their performance. Actually, these technologies have beaten in acceptance more sophisticated technologies that provide higher performance.

As heterogeneous CPS grow, the use of middleware solutions is desirable to reduce the design complexity of programming communications issues, especially in heterogeneous environments. In fact, the economic benefits of using middleware in complex distributed applications produce up to 50% decrease in software development and costs [11]. Some examples of well established distribution middleware technologies allowing the integration of control devices are CORBA [12, 13, 14], ICE [15], OPC [16, 17] and Web Services [18]. More recently the OMG specified DDS (Data Distribution Service) [19, 20, 21]. DDS is a very promising middleware technology that follows the publisher/subscriber paradigm. One of the main features of DDS is that it provides several mechanisms to set and manage a broad number of QoS parameters in real-time applications.

These middleware standards allow building CPS applications on top of the well extended TCP/IP stack. These technologies can hide the low-level implementation details facilitating the construction of the new applications. Middleware can serve as the backbone or software bus for building CPS applications across many domains. An early example of using standard middleware in CPS can be found in [22], where the time-triggered paradigm was applied to sensor networks. In this work, physical nodes (sensors and actuators) were not connected to the distributed system directly but by means of CORBA gateways that encapsulate clusters of objects.

However, generic middleware technologies present two main drawbacks; on one side they tend to be excessive and source of performance overhead for CPS [23], and on the other, they do not match some of the special requirements of this kind of systems [10], such as providing abstractions that represent the entities found in CPS. This is why several high-level middleware architectures that adapt better to CPS have been proposed.

This work focuses on high level middleware services aimed at simplifying the construction of large CPS. In particular, it presents a set of logical cooperating topologies that may be used to manage the resources of the distributed systems and redundancy of distributed services. These topologies allow the creation of structures that can be set up during the system bootstrap discovery process. These structures may evolve at run-time with nodes that join or leave the system dynamically. State changes could happen as a result of a modification in the functionality of the system, changes at the availability of the resources or failures of one device. Communication efficiency and other aspects can influence and modify this relationship but, for the design of the CPS, these requirements need only be configured. Some preliminary simulations results that show the application of these structures for load sharing can be found in [24].

Software migration and dynamic software updates are desirable services. An approach to dynamically update software in CPS using DDS is proposed in [25]. Dynamic reconfiguration is particularly important in applications with high availability requirements.

The reminder of this paper is structured as follows. Section II describes the features of CPS with a potential to profit from cooperation with QoS requirements, setting

the base cooperation; Section III introduces and analyzes the construction of the logical cooperating topologies proposed in this work. In Section IV a scenario is presented to illustrate the application of the generic mechanisms under study; Finally, Section V concludes and discusses future work.

II. COOPERATION REQUIREMENTS AT CPS

Due to their nature CPS introduce particular safety and reliability requirements which are qualitatively different from those found in general purpose computing. According to [4], new computing and networking abstractions are needed to deal with the entities used in CPS, since they must be able to represent the passage of time and concurrency which are intrinsic to the physical world. In addition, as CPS do not operate in controlled environments, they must be designed in a robust way so they are capable to adapt to subsystem failures.

In addition, most interesting CPS are cooperative systems in which networking technologies play a key role. Unfortunately, most widely used networking technologies introduce a great deal of timing variability. As a consequence, two main approaches can be followed: (1) Using less widely accepted network technologies, like CAN or FlexRay, typically confined geographically to local area networks [26]; or (2) assuming lower performance and proposing extended abstractions that integrate in the existing network infrastructures and reference models [10].

By cooperation we understand the capability of a distributed system with autonomous subsystems to dynamically decide which components will carry out a certain task in order to optimize response time, energy consumption or fulfill a QoS policy.

A. Features of CPS

Some of the most relevant features of CPS have been identified by [5, 26, 27]. Namely, these are:

- *Dealing with time*: Since CPS deal with physical processes, time and concurrency become relevant issues. In addition, CPS present strict timing restrictions to response times.
- *Close integration*: CPS are highly coupled systems that may involve large number of devices.
- *Solving heterogeneity*: Most CPS use a mixture of a high variety of technologies and platforms that include different operating systems, programming languages and network technologies.
- *Use of the resources*: Typically, CPS are implemented over devices with low resources in terms of CPU, memory, network bandwidth and energy consumption. Consequently, the management of these resources is a key issue.
- *Dynamic reconfiguration and reorganization*: CPS should provide the capability to dynamically reorganize and reconfigure in order to adapt to changes in the physical world or to changing requirements.
- *Dependability and robustness*: CPS must be reliable even in adverse situations, since the security and safety of people and investments can be affected by its malfunction. Sometimes, they must be certified.

B. Cooperation in CPS with QoS requirements

CPS involve a number of physically distributed elements: sensors, actuators, processing/memory units and communication devices, many of them with a low profile. Frequently the amount of required elements can be established, each with a number of duties statically assigned in advance. However, many resources are mostly idle while others cannot produce high quality results on time. For such cases the implementation of cooperation mechanisms is proposed.

- *Autonomy*: For many CPS the autonomy of their subsystems is a highly desirable feature. Certainly, not depending on the accurate functioning of the whole system reduces security risks coming from human errors, natural disasters and human attacks. This is particularly true when communications are involved and some subsystems are temporarily unavailable.
- *Fault tolerance*: Static CPS may be highly reliable and predictable but after the failure of any of the physical elements the whole system may fail. Restoring back the system to a working situation can be a hard task requiring identifying the failed elements and replacing them, often with service disruption. To prevent such situations single points of failure must be avoided. Replication, fault detection and fault tolerance mechanisms are often used. Services are distributed along the CPS and backup physical elements are present to dynamically replace similar devices when their behavior is not the expected one. Faulty elements can after be substituted without the urgency of restoring a disrupted service.
- *Scalability:* Another common problem of CPS is the difficulty to provide smooth scalability. Changing the extension of the phenomenon being controlled may require duplicating all the elements. Systems become unnecessarily large and this size often implies higher difficulties.
- *Soft Real-Time Systems*: Cooperation can be appropriate for soft real-time systems. Indeed, hard real-time systems must guarantee response times under all failure free situations. Even under failures the behavior in such systems must be highly predictable. Many physical phenomena do not require such degree of reliability and soft real-time solutions will be more convenient. On top of that functional parts that are not safety critical can often be distinguished. These are often more sensible to hardware costs.

C. Basis for Cooperation in CPS

To develop middleware allowing generic cooperation in CPS some preliminary concepts must be taken into account.

- *Control over Communications*: CPS communications must be well coordinated. It is desirable that different components communicate with a common reference of priority and other QoS aspects. This is a key factor to allow growth without increasing system complexity.
- *Communication Paradigm*: The Client/Server communications paradigm has been extensively used, especially in combination with TCP/IP. It is very common to find real-time features in CPS and as such Publisher/Subscriber paradigms have advantages in the distribution of data, especially when QoS features can be controlled.

- *Resource Replication*: The replication of some resources can be used to implement fault tolerance mechanisms. It also allows exploiting the parallel execution when dealing with processing and memory elements.
- *Resource Availability Modeling*: Truly dynamic systems often include resource discovery mechanisms to identify the potential candidates for cooperation. This cooperation depends on the nature of the resource. For example, a battery can be considered available when it allows a normal functioning and unavailable when it runs out. A quantitative value of how charged the battery is can also be distinguished, as a notion of load information. However, in order to reduce the number of transitions and therefore the information update needs, state categories will be associated with resource states rather than quantitative values.
- *Process Migration*: The dynamic execution of CPS processes may involve the code mobility, data distribution or process migration. Software components may be transferred from one device to another in reaction to resource availability.

D. Middleware architectures for CPS

Middleware is typically organized in a hierarchy of several layers [28]. The model presented in [10] proposes an additional layer, the so-called Cyber-Physical Layer, which includes an abstract description of the properties and nature of cyber-physical data. This layer, situated on top of the application layer, should provide services for the lower layer protocols to support an efficient cross-layer design of the underlying application and communication protocols. Fig. 3 reproduces this new layer and its interaction with some of the OSI layers.

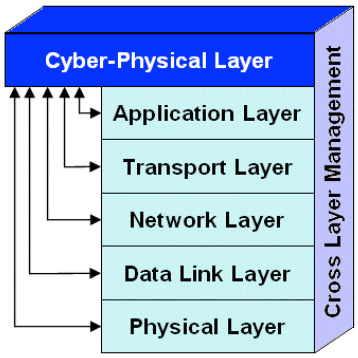

Figure 3. Cyber-Physical Layer [10].

In [29, 30] the authors present the FTT-CORBA middleware architecture aimed at synchronizing the task activations of a distributed system according to a plan that may be changed at run-time; tasks are wrapped as CORBA methods activated by a central node, the Orchestrator, over a LAN. This architecture can be used at CPS.

Several authors have proposed different solutions for CPS on top of DDS. For example, in [31] the authors present a discovery method on top of DDS to be used in CPS. Also, in [32] a proprietary distribution service aimed at CPS is presented. These methods may be used in cooperation strategies to provide fault tolerance and reconfiguration of CPS [33].

Alternative middleware architectures built on top of Web services can also be found in the literature. An example is the WebMed architecture [34], designed with a service-oriented view point to support CPS applications. It enables access to the underlying smart devices and allows the integration of their specific functionality with other software services.

E. Infrastructure Modeling of CPS

CPS require a close interaction with the underlying infrastructure. However, most of the architectures do not model the available resources to allow making reconfiguration decisions. Some models only consider this issue partially. For example, traditional scheduling algorithms for distributed systems focus on the assignment of a set of tasks in a set of CPUs. Frequently, the use of other infrastructure resources or the interaction with the environment are partially analyzed or not considered at all. In CPS, it is necessary to simultaneously consider the computing and communication processing together with the physical system and their interactions [35]. An example of this interaction is the management of the energy consumption as described in [36]. More detailed models for describing the whole underlying infrastructure are required with algorithms to make decisions according to its current state.

III. LOGICAL COOPERATION TOPOLOGIES

In this section some logical topologies and their use in CPS for cooperation purposes are described. It must be noted that these topologies may be implemented on top of different distribution middleware specifications. Hence, the current approach abstracts the underlying physical communication technologies, e.g. Ethernet or WiFi.

Theses topologies represent associations between nodes, which are abstractions of autonomous devices that can be replicated. These topologies will be used to maintain state information that allows quick reaction to cooperation necessities. They may be used to provide fault-tolerance mechanisms as well as resource management at CPS.

Depending on the final application the use of one simple topology may be sufficient, for example, when a small number of participating nodes are present. More sophisticated topologies will typically involve more communication traffic but appropriate information according to the requirements. The use of one topology or another may similarly be established statically or changed dynamically depending on the number of nodes and the nature of the system.

A. Reliable Friend

The basic logical cooperation topology is called *Reliable Friend* (RF). A similar structure can be found in [37]. In this topology each node establishes a best-effort compromise to carry out some service when necessary to another node called its **official sender** (OSe) and gets the compromise to carry out a service from another node, the **official receiver** (ORe), according to a friendship paradigm. This relationship is described in Fig. 4, where circles represent nodes and the thick arrows represent this official relationship, or compromise to provide a service.

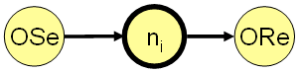

Figure 4. Official relationship sender-receiver for a node n_i.

This relationship can be established dynamically as nodes incorporate to the system at bootstrap. This may be achieved by exploiting the discovery service provided by many middleware specifications. Fig. 5 illustrates this construction process.

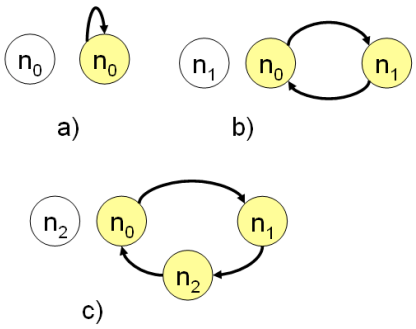

Figure 5. Dynamic construction of the Official relationship.

In 5.a only n_0 is present and acts as OSe and ORe of itself with no practical use apart from the construction initiation. In 5.b a second node n_1 is incorporated to the structure therefore forming a two-node official structure. In 5.c a third node n_2 incorporates and it forms a logical cooperation ring, being node n_1 its OSe and node n_0 its ORe.

Node numbering has been used to describe the vision of the official relationship for a node, n_i, as illustrated in Fig. 6. It is noted that this simple vision from the perspective of each node of only one OSe and one ORe maintains regardless of the number of participating nodes.

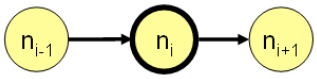

Figure 6. Official relationship from a node's point of view.

Nodes may be incorporated to the structure, establishing a logical cooperating ring relationship. Fig. 7 depicts this logical official structure for a system with 10 nodes. As a reminder, the thick arrows represent the cooperation compromise between nodes, a logical link between nodes.

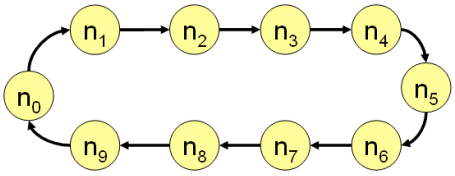

Figure 7. Reliable Friend 10 node Official Structure.

B. *Temporal Structure*

Depending on the nature of the participating resources there will be some circumstances that will make some of the nodes temporarily incapable of fulfilling the official compromise. On the basic case two possible states are considered: **Available** or able to fulfill compromises and **Unavailable**, or temporarily unable to fulfill compromises. The simultaneous presence of multiple degrees of availability can be allowed as in the case of different battery loads.

In those cases where some nodes are not available, a temporal structure is dynamically maintained. Each node keeps the identity of the first available node found following node the official structure, the **temporal receiver** (TRe). Obviously, due to information propagation delays this information may be out of date. A node may have more than one **temporal sender** (TSe) or nodes temporarily relying on it.

From the point of view of a node denoted **Me** in general there are only four possible situations, described in Fig. 8 with Available nodes shaded in light color, Unavailable nodes shaded in dark color and the rest, not shaded, are unknown or irrelevant.

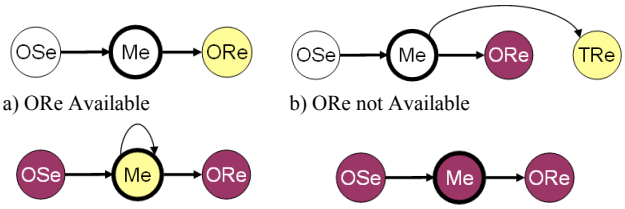

Figure 8. Temporal Structure different situations for Me.

The scheme in 8.a illustrates a situation where the ORe of a node Me is Available, so that it will be trusted. In 8.b the ORe has become Unavailable and a temporal receiver (TRe) is Available. The information exchange mechanism presented here provides the identity of the first Available node found following the official structure, so that it is known by node Me. In 8.c only one node is Available, Me. Only this node is aware of the fact that only one Available node is left in the whole system. When this node becomes Unavailable, the situation in 8.d will be reached and the information exchange mechanism will update that no Available node can be found in the whole system. All nodes are aware of that condition.

Fig. 9 illustrates the snapshot of a RF structure with three Unavailable nodes represented with dark circles and the rest Available, represented in light circles. Thin arrows represent the temporal compromise relationship. In this example, n_1 and n_2 both rely on n_3, their TRe. Additionally n_4, n_5 and n_6 they all share the same TRe, i.e. n_7.

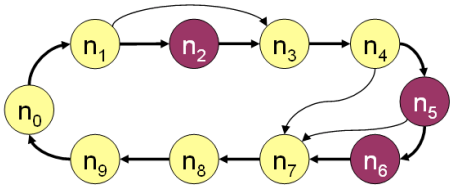

Figure 9. Reliable Friend Temporal Structure snapshot.

C. *Bidirectional Reliable Friend*

Sometimes having just one candidate may not be sufficient, especially in CPS requiring a quick response. When this ring structure results in an accumulation of several Unavailable nodes it is convenient to have more choice on cooperation candidates.

An alternative to the basic RF structure contemplates maintaining for each node two official candidates, resulting in the *Bidirectional Reliable Friend* (BRF) logical cooperation topology. Fig. 10 depicts this official

structure for 10 participating nodes, representing this official relationship with bidirectional thick arrows.

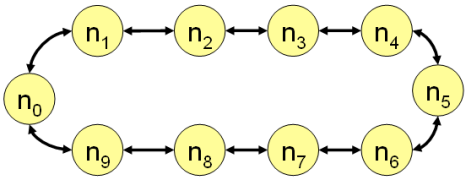

Figure 10. BRF Official 10 node structure with bidirectional arrows.

In this BRF cooperation topology the basic solution is to maintain the previously described temporal structure in both senses. For the scenario shown Fig. 9 the corresponding BRF temporal structure is depicted in Fig. 11, where every node has two TRe. The official structure has been omitted. When only one node remains Available in the whole system every node will be aware of this fact as they all keep a duplicated TRe.

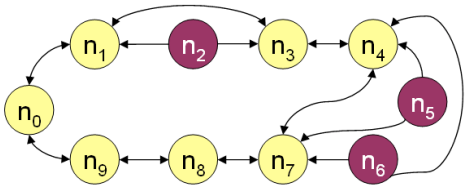

Figure 11. BRF Temporal Structure snapshot.

D. *Simplified Bidirectional Reliable Friend*

In order to reduce the message exchange activity of Unavailable nodes, in the *Simplified Bidirectional Reliable Friend* (SBRF) cooperation topology Unavailable nodes only keep the identity of one temporal candidate. This candidate has been chosen to be the topologically closest in either direction or the one on the clockwise sense when two possibilities exist. Fig. 12 illustrates a scenario with 10 participating nodes and four dark shaded Unavailable nodes.

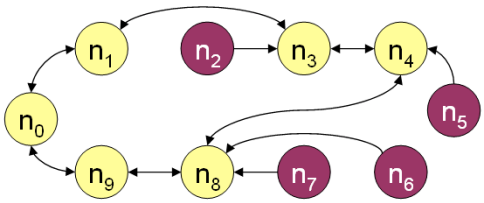

Figure 12. SBRF Temporal Structure snapshot.

E. *Multi-Level Reliable Friend*

When the concept of availability may be categorized in different degrees of load the *Multi-Level Reliable Friend* (MLRF) can be appropriate to dynamically maintain the temporal structure. The convention adopted here is to use load of a specific resource as CPU workload, so that low load means Available. However, for some resources, such as batteries, is just the opposite, since batteries would be highly Available at the highest load value.

Thus, at the beginning a low threshold th_0 will be established to separate Available A_0 and Unavailable U_0 nodes as illustrated in Fig. 13. According to the convention, the higher the load is the less Available a node will be. When all nodes are Unavailable in U_0 state the threshold will be redefined to th_1 so that node with a load in between th_1 and th_0 are now considered Available, A_1. Depending on the presence of one or many nodes with a load lower than th_1 a categorization down to set the threshold in th_0 can be carried out. This operations require the participation of all nodes and publish/subscribe mechanisms are appropriate for the propagation of this information.

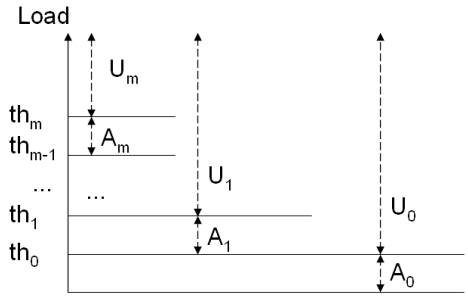

Figure 13. Multi-Level RF load categorizations.

F. *Multi-State Reliable Friend*

Similar to the MLRF, when different degrees of load can be distinguished quantitatively, boundaries can be set to establish load states. Nodes with different load states may coexist at the same time in the so called *Multi State Reliable Friend* (MSRF) cooperation topology. With a similar ring logical official structure nodes keep the identity of the next node with a lower load state if such a node exists or one at the same load state, possibly itself. A snapshot of the temporal structure of a system with 10 nodes is shown in Fig. 14. The convention adopted is similar to that of MLRF, which means that a node in a state labeled 5 means is aiming at sending workload to a node with less workload, e.g. 3 or ideally 0.

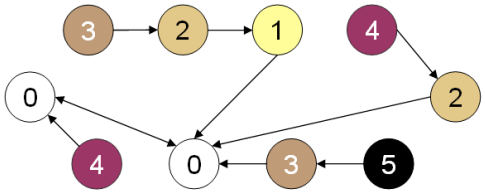

Figure 14. MSRF temporal structure snapshot.

Intuitively, nodes may get loaded with no further constraint. No load balancing is necessary and that is why load state differences between nodes appear. Yet, when a node anticipates that some cooperation can show advantageous it has solid candidates for reconfiguration.

Compared to previous cooperation topologies, this information exchange scheme requires more communications for its maintenance. The advantage is that a load gradient is obtained so that when necessary we can visit nodes following this structure and each hop reduces at least one degree until we reach the node with the lowest load in the system.

G. *Multiple Reliable Friends*

The combination of coexisting Reliable Friend structures may extend the power of this kind of cooperation forming multiple Reliable Friend structures. Two general approaches are considered: (i) the combination of different RF structures within the same hardware system dealing with information on different resources (e.g. batteries, processing power or memory) called *Multi-Purpose RF* (MPRF) and (ii) the combination

of different RF structures dealing with the same aspect, called *Multi RF* (MRF).

In MPRF approaches a single information exchange schema may be utilized to exchange and maintain this information.

In MRF approaches individual RF structures form autonomous groups that can collaborate with others in a second degree of collaboration.

IV. CONCLUSIONS AND FUTURE WORK

This paper proposes the integration of some state information exchange mechanisms in Cyber Physical Systems (CPS). More specifically, various generic state information exchange cooperation topologies are described. The topology choice for a particular system depends on the meaning given to the concept of state and the characteristics of cooperation.

These information exchange mechanisms can be useful in CPS built out of COTS middleware components in order to provide reconfiguration mechanisms for fault tolerance and resource management in a scalable manner.

Currently, the proposed cooperation topologies are being implemented on top of DDS middleware, namely OpenSplice, exploiting its discovery mechanisms. This is consistent with the proposition of using a single communication backbone middleware for the whole system with control over the QoS, since DDS provides a logical software bus.

In our current implementation the official and temporal structures are maintained in a network of computers simulating state changes.

Envisaged future work includes the implementation of all the proposed information exchange mechanisms, and individual and combined performance evaluation, together with the development of solid middleware architecture with an API allowing its use on various CPS scenarios.

The authors are also analyzing a case study for applying the proposed topologies and high-level middleware, consisting of a security surveillance system with cooperation requirements similar to the applications found in [29, 38]. More specifically, the considered case study consists of a large building with multiple surveillance subsystems installed in several strategic points. Each surveillance subsystem has optional sensors including temperature, pressure, noise and vibrations; smart embedded cameras; batteries preventing service disruption when electricity is cut; memory to store some minutes of video of different qualities. These devices can communicate with each other as well as with remote headquarters. They can work in different modes, namely, standby, remote control and standalone autonomous. Cameras can capture raw images and video and process them locally using demanding compression algorithms or send them over so other subsystems are responsible for compressing, storing or even sending them to remote locations.

REFERENCES

[1] W. G. Gilroy, "NSF funds Cyber-Physical Systems Project," Available at: http://newsinfo.nd.edu/news/17248-nsf-funds-cyber-phys/

[2] National Science Foundation Program Solicitation, Available at: http://www.nsf.gov/pubs/2008/nsf08611/nsf08611.pdf.

[3] W. Wolf, "Cyber-physical Systems," *Computer*, vol. 42, no. 3, pp. 88–89, March 2009. http://dx.doi.org/10.1109/MC.2009.81

[4] E. A. Lee, "Cyber Physical Systems: Design Challenges," *11th IEEE Symp. Object Oriented Real-Time Distributed Computing* (ISORC 2008), pp. 363-369.

[5] J. Shi, J. Wan and H. Y. Hui Suo, "A Survey of Cyber-Physical Systems," *Intl. Conf. on Wireless Communications and Signal Processing* (WCSP), 2011.

[6] F-J. Wu, Y-F. Kao and Y-C. Tseng, "From wireless sensor networks towards cyber physical systems," *Pervasive and Mobile Computing* (2011). http://dx.doi.org/10.1016/j.pmcj.2011.03.003

[7] K.D. Kim, P.R. Kumar, "Cyber-Physical Systems: A Perspective at the Centennial," *Proc. of the IEEE* (Centennial-Issue), vol. 100, pp. 1287-1308, May 2012.

[8] P. Marwedel, "Embedded and cyber-physical systems in a nutshell," *DAC.COM Knowledge Center Article*, 2010.

[9] A. Koubâa, B. Andersson, "A Vision of Cyber-Physical Internet," Proc. of the Workshop of Real-Time Networks (RTN 2009), Satellite Workshop to (ECRTS 2009), pp. 1-6, July 2009.

[10] R. Rajkumar, I. Lee, L. Sha, J. Stankovic, "Cyber-physical systems: The next computing revolution," *Design Automation Conference (DAC), 2010 47th ACM/IEEE* , vol., no., pp.731-736, 13-18 June 2010.

[11] T. Pearson, "Save time and money with COTS middleware for network equipment," www.commsdesign.com/printableArticle/?articleID=174402378. Nov 2005.

[12] OMG, Object Management Group, "Common Object Request Broker Architecture: Core Specification," Version 3.0.3, March 2004.

[13] OMG, Object Management Group, "Notification Service Specification," Version 1.1, October 2004.

[14] Sanz R., M. Alonso, "CORBA for Control Systems," Annual Reviews in Control, No 25, 2001, pp. 169-181. http://dx.doi.org/10.1016/S1367-5788(01)00016-5

[15] Henning, M., "A new approach to object oriented middleware," IEEE Internet Computing, Vol. 8, Issue 1, pp. 66-75, 2004. http://dx.doi.org/10.1109/MIC.2004.1260706

[16] OPC foundation, http://www.opcfoundation.org/.

[17] F. Perez, D. Orive, M. Marcos, E. Estévez, G. Morán, and I. Calvo, "Access to Process Data with OPC-DA using IEC61499 Service Interface Function Blocks," *14th IEEE Intl. Conf. ETFA*-2009, Palma de Mallorca, Spain, Sep. 2009.

[18] F. Jammes and H. Smith, "Service-oriented paradigms in industrial automation," *IEEE Trans. Industrial Informatics*, vol. 1, issue 1, pp. 62-70, Feb. 2005. http://dx.doi.org/10.1109/TII.2005.844419

[19] OMG, Object Management Group, "Data Distribution Service for Real-time Systems," v1.2, June 2007.

[20] J.A. Dianes, M. Díaz and B. Rubio, "Using standards to integrate soft real-time components into dynamic distributed architectures," *Computer Standards & Interfaces*, Vol. 34, Issue 2, Feb. 2012, pp. 238-262. http://dx.doi.org/10.1016/j.csi.2011.10.002

[21] Ryll, M. and S. Ratchev, "Application of the Data Distribution Service for Flexible Manufacturing Automation," Proc. of World Academy of Science, Engineering and Technology, vol. 31, July 2008, pp. 178-185.

[22] W. Elmenreich, "Time-triggered smart transducer networks," *IEEE Transactions on Industrial Informatics*, vol. 2, no. 3, pp.192-199, Aug. 2006. http://dx.doi.org/10.1109/TII.2006.873991

[23] A. Dabholkar, A. Gokhale, "An Approach to Middleware Specialization for Cyber Physical Systems," *Proc. of the 29th IEEE Intl. Conf. on Distributed Computing Systems* Workhops, 2009. http://dx.doi.org/10.1109/ICDCSW.2009.70

[24] I. Etxeberria-Agiriano, I. Calvo and E. Zulueta, "Simulation of Various Candidate Selection Strategies for Migration in Distributed Systems," *16th Intl. Conf. on Soft Computing*, 2010, pp. 338-345.

[25] M.J. Park, D. K. Kim, W-T. Kim and S-M. Park, "Dynamic Software Updates in Cyber-Physical Systems," *Intl. Conf. on Information and Communication Technology Convergence* (ICTC 2010), pp. 425-426.

[26] E.A. Lee, "Cyber-Physical Systems – Are Computing Foundations Adequate?," Position Paper for NSF Workshop on Cyber-Physical Systems: Research Motivation, Techniques and Roadmap (2006).

[27] I. Calvo, I. Etxeberria-Agiriano and A. Noguero, "Distribution Middleware Technologies for Cyber Physical Systems," *Proc. of the Remote Engineering & Virtual Instrumentation (REV-2012)*, pp. 298-301, July 2012.

[28] Schmidt, D.C. "Middleware for real-time and embedded systems," *Communications of the ACM* 45(6), 43-48 (2002). http://dx.doi.org/10.1145/508448.508472

[29] I. Calvo, L. Almeida, F. Pérez, A. Noguero and M. Marcos, "Supporting a reconfigurable real-time service-oriented middleware with FTT-CORBA," *15th IEEE Intl. Conf. of the Emerging Technologies and Factory Automation* (*ETFA*-2010), Bilbao, Spain, pp. 1-8, Sep. 2010.

[30] A. Noguero and I. Calvo, "A Time-Triggered Data Distribution Service for FTT-CORBA," *Proc. of the Emerging Technologies and Factory Automation* (ETFA-2012), Sept. 2012, 1-8 (to be P.).

[31] S.H. Lee, J.H. Kim, W.T. Kim and J.C. Ryou "Communication Entities Discovery in Complex CPS System," Control and Automation, and Energy System Engineering - Communications in Computer and Information Science, Springer Heidelber, Berlin, Vol. 256, pp. 213-219, 2011.

[32] W. Kang, K. Kapitanova, S. H. Son, "RDDS: A Real-Time Data Distribution Service for Cyber-Physical Systems," *IEEE Trans. on Industrial Informatics*, vol. 8, no. 2, pp. 393-405, May 2012. http://dx.doi.org/10.1109/TII.2012.2183878

[33] I. Etxeberria-Agiriano, I. Calvo, A. Noguero and E. Zulueta, "Configurable Cooperative Middleware for the Next Generation of CPS," *Proc. of the Remote Engineering & Virtual Instrumentation (REV-2012)*, pp. 315-319, July 2012.

[34] D. Hoang, H. Paik and C. Kim, "Service-Oriented Middleware Architectures for Cyber-Physical Systems," *Intl. Journal of Computer Science and Network Security*, Vol. 12, no. 1, Jan. 2012.

[35] M. Lindberg and K. E. A□rzén, "Feedback control of cyber-physical systems with multi resource dependencies and model uncertainties," in *Proc. of the 31st IEEE Real-Time Systems Symposium*, Dec 2010.

[36] L. Parolini, B. Sinopoli, B.H. Krogh, Z. Wang, "A Cyber–Physical Systems Approach to Data Center Modeling and Control for Energy Efficiency," *Proc. of the IEEE*, vol.100, no. 1, pp. 254-268, Jan. 2012. http://dx.doi.org/10.1109/JPROC.2011.2161244

[37] I. Echeverria and M.C. Woodward, "'A Reliable Friend': a Method for Maintaining the Load Information in a Distributed Computer System," *3rd Intl. Conf. Software Engineering for Real Time Systems*, 1991, pp. 63-68.

[38] M. Jovanovic and B. Rinne, "Middleware for Dynamic Reconfiguration in Distributed Camera Systems," *5th Workshop on Intelligent Solutions in Embedded Systems*, 2007, pp. 139-150. http://dx.doi.org/10.1109/WISES.2007.4408495

AUTHORS

Ismael Etxeberria-Agiriano is with the University College of Engineering of Vitoria-Gasteiz, Department of Computer Languages and Systems, University of the Basque Country (UPV/EHU), as Senior Lecturer, (e-mail: ismael.etxeberria@ehu.es).

Isidro Calvo is with the University College of Engineering of Vitoria-Gasteiz, Department of Systems Engineering and Automatic Control, University of the Basque Country (UPV/EHU), Spain, as Senior Lecturer (email: isidro.calvo@ehu.es).

Adrián Noguero is with the Software Systems Engineering division of Tecnalia, as a Software Engineer, (e-mail: adrian.noguero@tecnalia.com).

Ekaitz Zulueta Guerrero is with the University College of Engineering of Vitoria-Gasteiz, Department of Systems Engineering and Automatic Control, University of the Basque Country (UPV/EHU), Spain, as Senior Lecturer (email: ekaitz.zulueta@ehu.es)

This work was supported in part by the ARTEMIS JU through the iLand project (grant no. 10026), the Basque Government (Saiotek) under project S-PE11UN061 and the University of the Basque Country (UPV/EHU) through grant EHU11/35. It is an extended and modified version of a paper presented at the International Conference on Remote Engineering & Virtual Instrumentation (REV2012), held at University of Deusto, Bilbao, Spain, July 4-6, 2012. Received 14 September 2012. Published as resubmitted by the authors 28 November 2012.