Bordel Sánchez et al. EURASIP Journal on Wireless Communications and Networking (2016) 2016:228
DOI 10.1186/s13638-016-0731-0

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Predictive algorithms for mobility and device lifecycle management in Cyber-Physical Systems

Borja Bordel Sánchez[1*], Ramón Alcarria[2], Diego Sánchez de Rivera[1] and Alvaro Sánchez-Picot[1]

Abstract

Cyber-Physical Systems (CPS) are often composed of a great number of mobile, wireless networked devices. In order to guarantee the system performing, management policies focused on becoming transparent to high-level applications, the changes in the hardware platform have to be implemented. However, traditional reactive methodologies and basic proposed predictive solutions are not valid either due to the extremely dynamical behavior of CPS or because the high number of involved devices prevents fulfill the timing requirements. Therefore, in this paper, we present an advance predictive solution for managing the mobility and device lifecycle, being able to meet all requirements of CPS. The solution is based on an infinite loop, which calculates, in each iteration, a sequence of future system states using a CPS simulator and interpolation algorithms. Furthermore, an experimental validation is provided in order to determine the performing of the proposed solution.

Keywords: Cyber-Physical Systems, Mobility management, Device lifecycle, Predictive models, CPS simulation, Interpolation algorithms

1 Introduction

CPS are the next generation of engineered systems in which computing, communication, and control technologies are tightly integrated [1]. In theory, any type of device could be part of a Cyber-Physical System: large or miniaturized, wired or wireless, fixed or mobile, etc. However, most practical discussions about the characteristics of CPS conclude that they are composed of a great number of wireless [2], embedded [3], mobile [2] devices.

Wireless mobile devices allow creating a more flexible network architecture than other types of devices [4]. But, on the contrary, these devices tend to interact and communicate opportunistically [4], so the access to them is not guarantee, in general. Nevertheless, most CPS applications (such as energy infrastructures, transport systems, or medical instruments) require a permanent and transparent access to the hardware capabilities, so mobility and device lifecycle management policies are essential in CPS.

Besides, this challenging situation has turn more complex due to the appearance of concepts such as the Cyber-Physical Internet [5]. These scenarios consider the possibility of deploying various communicated CPS in the same geographical area, so mobile devices not only can move inside the coverage area of one CPS but also they can move to other CPS and later return (or not).

In traditional systems, device lifecycle and mobility are supported by means of reactive techniques, i.e., the system makes decisions or actions when a certain change occurs [6]. However, these techniques suppose the changes in the system are continuous and slow, so there is enough time to determine and execute the proper actions before a fatal error occurs. For example, in Long-Term Evolution (LTE) networks [7], a handover is executed when quality or power measures from a user equipment go below a certain limit. Then, it is suppose these measures are not going to change abruptly, so there is time to execute the handover (although it is a long and complicated process) before losing the

* Correspondence: bbordel@dit.upm.es
[1]Department of Telematics Systems Engineering, Universidad Politécnica de Madrid, Avenida Complutense n° 30, 28040 Madrid, Spain
Full list of author information is available at the end of the article

connectivity definitively. However, CPS present an extremely dynamical behavior, and the system state may change rapid and randomly [8]. For example, in underwater military applications [9], the medium can degrade so fast that there is no enough time to react properly before losing access to hardware devices, since a relevant change in the signal quality is detected. Thus, reactive techniques are not valid in CPS, as the decision and actions are calculated considering situations which may change strongly during the calculation process.

On the other hand, a small number of basic predictive solutions are also available. In these proposals, the system predicts the future changes and executes the appropriate actions before they occur. However, they are based on heavy simulators which are useful in simple scenarios but which fail when used in CPS due to the high number of involved devices. In these cases, the needed time to simulate a certain future situation is higher than remaining time to really reach it. Then, the essential timing requirement of predictive solutions is not fulfilled.

Therefore, in this paper, we propose a predictive algorithm for device lifecycle and mobility management in CPS, being able to predict the future states of the system and calculate and execute the appropriate actions before future changes or fatal errors occur. The proposed algorithm is based on a CPS simulator and interpolation functions being able to calculate the future system states. These results feed a decision-making module, which is not part of the proposed predictive algorithms, but which is required to manage the mobility and the device lifecycle. The proposed solution presents a flexible definition, so it could be adapted to any application scenario of CPS (low-rate networks, intelligent devices, etc.). Moreover, an experimental validation is provided, considering a particular implementation of the proposed algorithm. The proposed experiment proves that in more than 95 % of cases, the algorithm manages the system changes successfully.

The rest of the paper is organized as follows: Section 2 introduces the state of the art in mobility and device lifecycle in CPS. Section 3 presents the application scenario and proposes the predictive algorithm. Section 4 describes a particular implementation of the algorithm, used in the experimental validation. Finally, Sections 5 and 6 explain some results of this experimental validation and the conclusions of our work.

2 State of the art

Works on managing the device lifecycle in CPS are not common. In general, papers which treat this topic are focused on configuring the whole CPS, not only on managing the hardware platform. Then, in general, particular details about hardware issues (such as controlling the remaining battery charge in wireless devices) are not addressed.

In this area, different types of proposals are found. Some works are focused on the use of artificial intelligence algorithms being able of automatically adding, removing, and configuring components in CPS [10]. Others describe special tools (such as middleware layers) which, at the end, need the human intervention [11]. Works describing algorithms where each device generates a description file when a change occurs in the system, which is processed by the rest of the platform, may be also found [12]. Finally, papers discussing the installation of an intelligent scheduler in each device, communicating with a central core where decisions are taken are also available [13, 14].

However, as can be seen, all the previous proposals follow a reactive scheme which is not valid in our general scenario (it must be noted that, sometimes, these solutions could be valid, if some assumptions are considered).

The topic of mobility support in CPS has received more attention than device lifecycle, although it is not a main research line nowadays. In general, two types of mobility may be defined in CPS: intra-system and inter-system.

In intra-system mobility scenarios, devices move inside the coverage area of one CPS. Proposal about this topic delegate the mobility control to the underlying network [15]. In these works, devices are clients of a mobile network through which a data service (representing the CPS applications) is provided. These solutions present a good performing; however, as we said in the introduction, nowadays concepts such as the Cyber-Physical Internet have to be considered, and this scheme cannot cover the requirements of these scenarios. Then, mobility problem in CPS is focused on inter-system mobility.

In the case of inter-system mobility, works used to propose application-specific mobile frameworks (based on SOA, for example), adapted to the particular mobility requirements of their scenario [16, 17]. Moreover, some authors incorporate GPS transponders into devices, in order to provide geographical data to a computational core. Then, the core may determine if devices are or not in the coverage area [18]. Nevertheless, all these proposals (including the cited for the intra-system case) follow a reactive scheme and are valid in very restrictive scenarios (as we said previously).

Finally, relating to inter-system mobility management, predictive schemes have been also proposed [19]. In these solutions, predictive algorithms are based on simulators which use numerical integration functions (one for each device) [20]. Then, the resulting algorithms present an order of complexity of $o(n^2)$ for both: the number of considered devices and the number of

instants for which the system state is calculated. Thus, in scenarios where hundreds of devices are included, and/or where the dynamics requires to simulate the system state in several time instants, the needed time to simulate a certain future situation may be higher than remaining time to really reach it.

Therefore, basically, it is necessary to design a predictive algorithm being able to manage both the device lifecycle and the mobility and to predict the future system states before it reaches them. The solution should be as general as possible and should allow including a high number of devices and obtaining the system state in as many time instants as needed. For that, we propose an algorithm based on an infinite loop, where four steps are repeated in each iteration: (i) data about the real system state from hardware devices; (ii) a CPS simulator obtains a certain number of future system states using the data acquired; (iii) the rest of the system states are obtained using interpolation techniques; and (iv) the decision-making algorithms execute the adequate actions based on the predicted future system states. The use of interpolation techniques, as we are seeing, introduces a greater error in the predicted future states. However, they reduce strongly the needed time to obtain a sequence of future states. In the experimental validation section we are proving the resulting solution allow predicting and managing correctly the changes in the system, despite the additional error introduced by interpolation techniques.

3 Proposal: a predictive algorithm based on CPS simulation

In this section, we analyze the general characteristics of the base scenario, show the functional architecture for the proposed solution and describe the proposed predictive algorithm.

3.1 Base scenario, functional architecture

As we said in Section 2, we are looking for an algorithm as general as possible. However, CPS may be deployed in very different scenarios: from small-scale applications to large-scale solutions [21]. Then, a deep analysis of device lifecycle and mobility in CPS should be based on the definition and characterization of several different deployments (manufacturing [7], irrigation [22], etc.), architectures, implementations, and/or use cases. Nevertheless, in our case, we are going to work over some general characteristics (common to most cases), which are

1. Various small-scale CPS are deployed in the same area. Besides, the high-level applications and processes associated to one CPS are not modified or affected by changes in the hardware platform.
2. The communication among the different CPS is supported by the so-called Cyber-Physical Internet [6], to which CPS are connected through an "Inter-system services interface."
3. Two types of CPS are defined: internal-CPS and border-CPS. Internal-CPS present a coverage area completely surrounded by the coverage areas of other CPS. In border-CPS, the limits of the coverage area represent the geographical limits of the Cyber-Physical Internet.
4. The movement of the mobile devices can be defined as follows: (a) *free*, when there is not external control; (b) *controlled*, when it is possible to plan the and route for them and; (c) *limited* to one or several specific routes independently if they are free or controlled.
5. Only predictable changes are considered. In real systems, both predictable and unpredictable changes occur: devices run out of battery charge or leave a coverage area (which are predictable changes), but also they get broken down or the tasks they are executing get blocked (which are unpredictable changes). Unpredictable events have to be addressed using reactive techniques [23], which are not discussed in this work but which must complement the proposed solution in commercial deployments.

The first and second points focus the mobility problem on inter-system mobility. As we said, valid solutions for intra-system mobility may be found, and mobility problem in CPS is centered almost exclusively on inter-system mobility. The third point refers to the geographic limitation of the small-scale CPS and their underlying network which, in general, does not have worldwide coverage. Figure 1 shows graphically the resulting network architecture from the first three points.

The fourth point is directly related to the great diversity of devices which can be considered: from robots to wearable devices. Each one of these devices presents a different movement and the proposed solution should consider all of them.

Finally, the fifth point limits the scope of the proposed solution to predictable situations. These situations represent the majority of changes in the regular operation of a CPS (or, even, in traditional communication networks [24]). However, traditional reactive techniques (such as timers) may complement the proposed predictive solution, especially in commercial deployments where unpredicted system failures might occur and have to be managed.

The functional architecture for supporting the main proposal of this paper (the predictive algorithms, see Section 3.2) can be seen in Fig. 2. Various components may be distinguished.

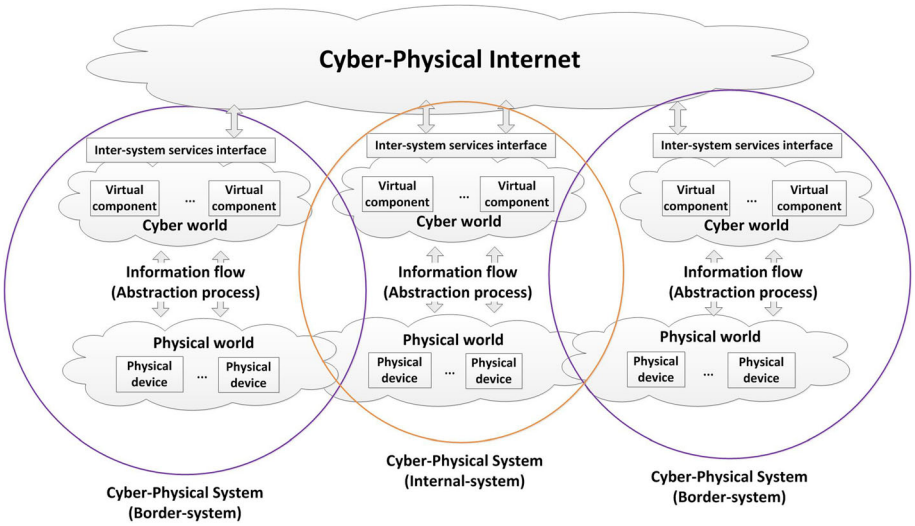

Fig. 1 Network architecture of CPS and Cyber-Physical Internet

Peripherals: They include all the components which support the task execution in a CPS (sensors, actuators, processors, etc.). They communicate through an appropriate middleware (such as a serial port) with their associate controllers.

Hardware controllers: It refers to all the components which manage and control the operation of the peripherals. In particular, they include all the necessary drivers to operate the peripherals. Each hardware controller has associated a certain number of peripherals. A hardware device which implements both peripherals and, at least, one hardware controller is called self-manage device or intelligent device.

Hardware manager: The entire hardware platform is managed from this component. In particular, it includes the module which implements the proposed algorithms (the *mobility and device lifecycle predictive manager*). It also implements the decision-making algorithms, executes the transactions for system reconfiguration, and makes independent changes in the hardware platform from the high-level applications or processes.

3.2 A predictive algorithm for device lifecycle and mobility management

At the start of the system, in the *mobility and device lifecycle predictive manager* of the *hardware manager*

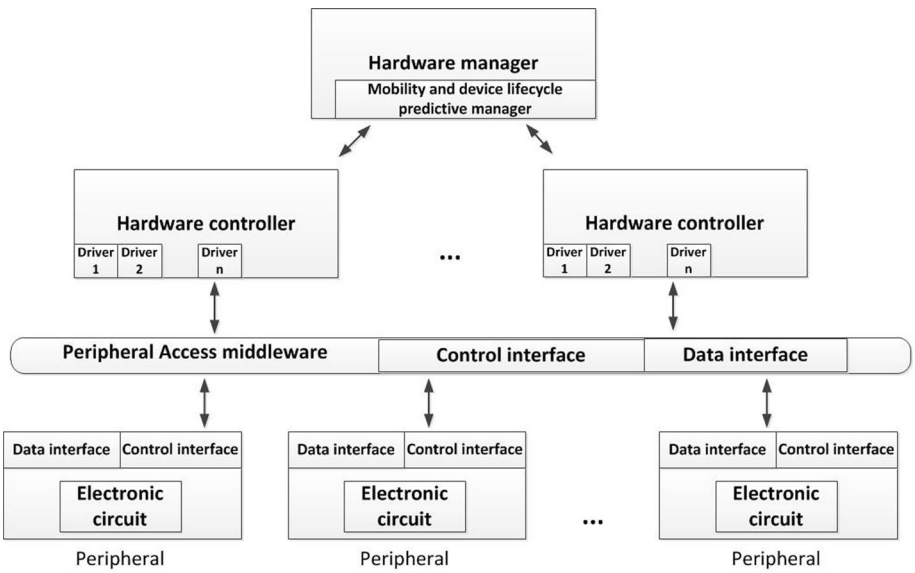

Fig. 2 Functional architecture

showed in Fig. 2, a model M of the underlying hardware platform is created. In this model, the ith device is represented by a collection m_i of n_i relevant parameters (1).

$$m_i = \left\{\mathrm{id}_i,\ \mathrm{type}_i,\ \mathrm{param}_1,\ \ldots,\ \mathrm{param}_{n_i}\right\} \quad (1)$$

The device model m_i includes a unique identification id_i and a type identifier type_i. The type identifier takes values from a set T, where each value represents a different class of device (2). The type identifier also indicates the pattern which must follow the collection m_i in each case. Examples of possible relevant parameters are the geographical position or the battery charge level.

$$T = \left\{\mathrm{device\_class}_j \quad j = 1, \ldots, Q\right\} \quad (2)$$

Then, the entire model M is a list of collections of parameters m_i (3). In order to allow its mathematical treatment (for example, in the interpolation functions), this model can also be expressed as matrix S (4). We are calling this matrix representing the value of all important parameters in the system "system state matrix" or, for simplicity, "system state."

$$M = \{m_1, m_2, \ldots, m_k\} \quad (3)$$

$$S = \begin{pmatrix} m_1 \\ \ldots \\ m_k \end{pmatrix} \quad (4)$$

In (2) and (3), k represents the total number of devices in the CPS. As, in general, not all device models m_i have the same number of relevant parameters n_i. In system state, matrix dimensions are homogenized by adding as many zeros as needed (these additional zeros allow the mathematical treatment of the system state matrix but are ignored in the decision-making algorithms).

In the *mobility and device lifecycle predictive manager*, a simulator being able to simulate CPS scenarios is also included. This simulator will take the system state in a certain instant $t = t_0$, S_{t_0}, and will obtain the system state h seconds later, S_{t_0+h}. Additional zeros added to homogenize dimensions in the system state matrix will not be taken into account.

Two additional important parameters are the time step h and the total simulated time T_{sim}. Then, the predictive algorithm must obtain the system state each h seconds, between the current time t_0 and $t_0 + T_{\mathrm{sim}}$. In order to obtain useful information, the calculation should finish before the first time step passes, i.e., before $t = t_0 + h$.

In previous proposals about predictive mobility management systems, all the needed future states are obtained by means of the CPS simulator. However, in complex systems where a great number of devices are considered, and/or where several time instants have to be obtained (i.e., the time step h is very small), timing conditions cannot be fulfilled, i.e., the calculations do not finish before $t = t_0 + h$. As a solution, we propose to generate some future states, instead of through the CPS simulator using interpolation techniques (which are much faster and lighter, although they present a greater error). Then, three different types of system states may be found:

- Intra-state: It represents the real state of the system in a certain instant $t = t_0$. In this state, the values of the parameters in the model are obtained from the physical devices through a data acquisition process. The cited data acquisition process could be based on a periodical transmission of information from the physical devices to the management unit; or in a solution where the management unit requests the devices for that information. In both cases, solutions based on JSON objects or XML description files are the most adequate (although any other could be used). Intra-state is the most precise way of describing the system; however, it is also the most costly (in time and processing capabilities). These states will be noted as S^I or I. At least, one intra-state has to be composed at the beginning of each iteration in the infinite loop.
- Predictive state: It refers the states obtained from the simulator. To obtain a predictive state, it is necessary to dispose, first, of an intra-state. From one intra-state, we may be able to obtain as many predictive states as desired. However, the precision in the prediction goes down, as we try to predict farther temporal instant is farther than what we desire to predict. On the other hand, the time needed to calculate a predictive state is lower than necessary to acquire an intra-state. These states will be noted as S^P or P.
- Interpolated state: It refers to the states obtained by interpolating two predictive states or one intra-state and one predictive state. These states are really fast to obtain but present the biggest error. This error goes up when more temporal distance exists between the states interpolated. These states will be noted as S^B or B.

The proposed algorithm is based on the definition of a sequence of predictive and interpolated states, describing the system in collection of future instants $\{t_0, t_0 + h, \ldots, t_0 + T_{\mathrm{sim}}\}$. Every sequence of states must start with an intra-state. And, later, a predictive or interpolated state is obtained for each time step, according to a predesigned pattern. The selected pattern might be very varied and should be adapted to the specific application scenario. In general, a balance between precision, calculation, speed, and the amount of data transmitted must be achieved. Figure 3 shows some possible schemes, depending on the considered scenario.

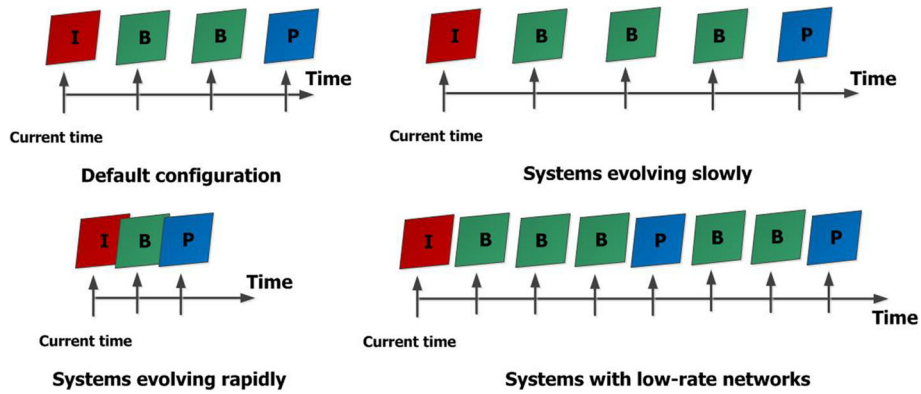

Fig. 3 Possible sequences of states

Given a certain time step h and a total simulated time T_{sim}, the precision of the sequence of states goes up, when the number of predictive states included increases. However, the calculation time also increases and, perhaps, the timing condition cannot be met. On the contrary, if more interpolated states are considered, the calculation time will go down strongly. Nevertheless, the error in the predicted states will be greater, and some changes could not be predicted. We argue that it is possible to find a pattern adapted to the considered CPS, such that it allows fulfilling the timing condition and the additional introduced error does not affect the correct prediction of changes in the system. In Sections 4 and 5, we prove that.

On the other hand, the precision in the predicted states also goes up when the time step h decreases. Nonetheless, in this case, meeting the timing condition (i.e., the calculation time must be smaller than the time step) gets more complicated. Finally, T_{sim} may be used as a design parameter. In general, if the considered CPS presents a very variable behavior, the simulation time T_{sim} will tend to be small (as well as the time step).

Then, in order to clarify the proposed algorithm, a numerical example is provided. In a CPS, four equal devices are available. Each device is represented by three relevant parameters. Numerically, the four models get described by

$$m_1 = \{1,\ 1,\ 9,8,7\} \quad m_2 = \{2,\ 1,\ 9,8,7\}$$
$$m_3 = \{3,\ 1,\ 9,8,7\} \quad m_4 = \{4,\ 1,\ 9,8,7\}$$

These models are ordered as a matrix, which corresponds with the first intra-state obtained:

$$S = \begin{pmatrix} 1 & 1 & 9 & 8 & 7 \\ 2 & 1 & 9 & 8 & 7 \\ 3 & 1 & 9 & 8 & 7 \\ 4 & 1 & 9 & 8 & 7 \end{pmatrix} = S^I_{t_0}$$

Using a CPS simulator, a predictive stet is obtained:

$$S^P_{t_0+T} = \begin{pmatrix} 1 & 1 & 6 & 8 & 5 \\ 2 & 1 & 1 & 0 & 0 \\ 3 & 1 & 2 & 6 & 6 \\ 4 & 1 & 9 & 8 & 7 \end{pmatrix}$$

And, finally, using matrix interpolation techniques, at least one interpolated state is calculated:

$$S^B_{t_0+\frac{T}{2}} = \begin{pmatrix} 1 & 1 & 8 & 8 & 6 \\ 2 & 1 & 5 & 4 & 4 \\ 3 & 1 & 6 & 7 & 6 \\ 4 & 1 & 9 & 8 & 7 \end{pmatrix}$$

Then, the situation of the devices for each temporal instant is obtained by reconstructing the device models from the previous matrixes

$$t = \frac{T}{2} \qquad m_1 = \{1,\ 1,\ 8,8,6\} \quad m_2 = \{2,\ 1,9,4,4\}$$
$$m_3 = \{3,\ 1,\ 6,7,6\} \quad m_4 = \{4,\ 1,\ 9,8,7\}$$

$$t = T \qquad m_1 = \{1,\ 1,\ 6,8,5\} \quad m_2 = \{2,\ 1,\ 1,0,0\}$$
$$m_3 = \{3,\ 1,\ 2,6,6\} \quad m_4 = \{4,\ 1,\ 9,8,7\}$$

As we said, the proposed sequence of predictive and interpolated states must be repeated in an infinite loop, which in each iteration starts with obtaining an intra-state. This new intra-state could be obtained for an instant posterior to the calculated interpolated and predictive states. However, as we said, obtaining an intra-state is a very costly task in time. Then, a temporal gap could appear in the management activities due to the delay of the intra-state generation process. In order to avoid that, the new intra-state may be acquired for a moment before the end of the current sequence (see Fig. 4).

Algorithm 1 Management algorithm

Input: List of devices in the hardware platform L
Pattern of predictive and interpolated states SS
Create a state matrix S
for each device dev in L **do**
Obtain the values of the relevant parameters m of dev
Incorporate m into S
end for
Add zeros in S to homogenize the dimensions
while true **do**
Create a three-dimensional array $S3$
$S3$ is equal to the sequence of predictive and interpolated states calculated with the pattern SS and the intra-state S
Evaluate and execute the next management actions considering the array S3
Wait until the obtaining time of the next intra-state
if differential acquisition is available **then**
Create a state matrix S
for each device dev in L **do**
Obtain the differential values of the relevant parameters m^* of dev
The real values of the relevant parameters m are reconstructed from SS and m^*
end for
else
Create a state matrix S
for each device dev in L **do**
Obtain the values of the relevant parameters m of dev
end for
end if
Incorporate m into S
Add zeros in S to homogenize the dimensions
end while

Moreover, in the second case, and if devices in the hardware platform present a certain level of intelligence, the intra-state may be generated using differential acquisition (see Fig. 4). In differential acquisition, each device is able to predict (and interpolate) its future states. Then, in order to reduce the required time to collect the information about the real device state, they only transmit to the *hardware manager* the differences between one of the last predictive or interpolated states in the sequence and the reality. Later, the *hardware manager* reconstructs the complete intra-state. This solution is in general much more efficient (especially if information compression is also enabled) and, overall, faster (which helps to meet the timing condition).

Then, considering all previous discussion, Algorithm 1 presents the general management algorithm.

In summary, Algorithm 1 creates the first intra-state and calculates the sequence of interpolated and

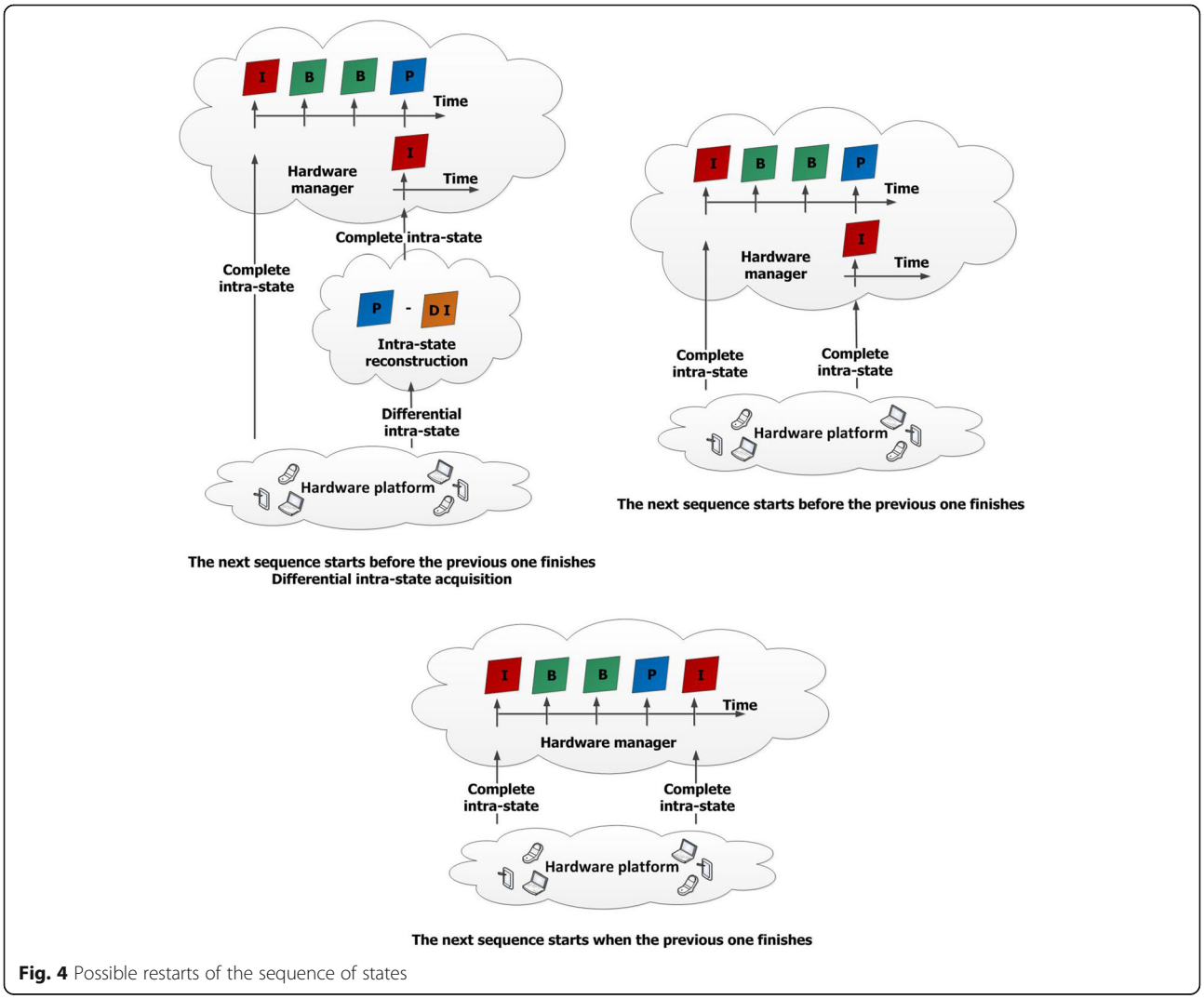

Fig. 4 Possible restarts of the sequence of states

predictive states using the obtained intra-state and given the desired pattern. Then, it calls the decision-making process with the calculated sequence of future states and performs the appropriate actions. Finally, it waits until it is necessary to obtain the next intra-state, and then acquires it (using differential acquisition if available).

In Algorithm 1, two procedures remain undefined: the decision-making function and the calculation process of the future states (including the simulation and interpolation process). With respect to the first function (decision-making), it is totally independent of the proposed predictive algorithms (see Section 3.3). Concerning the second function (calculation of the future states), Algorithm 2 represents its implementation.

Algorithm 2 Calculation of the sequence of predictive and interpolated states

Input: Intra-state at $t = t_0$, $S^I_{t_0}$

Scheme of predictive and interpolated states SS

Output: Sequence of predictive and interpolated states SQ

Extract the first element $instant_{pred}$ of SS

Extract the second element $instant_{interp}$ of SS

Create a three-dimensional array SQ
Create a state matrix S
S is equal to $S^I_{t_0}$
Create an integer i equal to one
for each instant $inst$ in $instant_{pred}$ **do**
  Configure the CPS simulator with the scenario described by S
  Run the simulation until $t = inst$
  Extract the final system state of the simulation S^P_{inst}
  Add S^P_{inst} to SQ
  L_i is the i-th list in $instant_{interp}$
  Interpolate the state in the instants indicated in L_i considering the states S^P_{inst} and S
  $S^B_{L_i}$ is equal to the obtained interpolated states
  Add $S^B_{L_i}$ to SQ
  S is equal to S^P_{inst}
  Increment i in one unit
end for

As can be seen in Algorithm 2, two parameters are taken as input: the original intra-state and the scheme of predictive and interpolated states. This scheme has the following structure (5).

$$\mathrm{SS} = \left\{ \mathit{instant}_{\mathrm{pred}} ,\ \mathrm{instant}_{\mathrm{interp}} \right\} \tag{5}$$

SS is a list of two elements where the fist element, $\mathrm{instant}_{\mathrm{pred}}$, is a list of the temporal instants for which a predictive state must be calculated. The second element, $\mathrm{instant}_{\mathrm{interp}}$, is a list of lists indicating the structure of the interpolated states (6).

$$\mathrm{instant}_{\mathrm{interp}} = \left\{ \{T_1, \ldots, T_q\}_i ,\ i = 1, \ldots, \mathrm{length}\left(\mathrm{instant}_{\mathrm{pred}}\right) \right\} \tag{6}$$

In $\mathrm{instant}_{\mathrm{interp}}$, each list indicates the temporal instants for which an interpolated state must be calculated. The first list indicates the instants between the first intra-state and the first predictive state, the second list indicates the instants between the first and the second predictive states, etc.

Considering Algorithm 2, a first numerical evaluation of the impact of using interpolated states in the calculation time may be done. As we said, simulators present an order of complexity of $o(n^2)$ for both, the number of considered devices and the number of instants for which the system state is calculated. On the contrary, interpolation algorithms present an order of complexity of $o(n)$ for both, the number of considered devices and the number of instants for which the system state is calculated. Then, we consider the needed time to calculate an intra-state for a certain CPS T_I, the required time to calculate one predictive state in the same CPS T_P, and, finally, the needed time to calculate one interpolated state in that CPS T_B. If we suppose the selected pattern for the sequence of future states is the *default configuration* (see Fig. 3), the time employed in the calculation process if interpolated states are not considered is (7). In (7) T_{3_P} is the calculation time for three predictive states.

$$T_I + T_{3\text{–}P} = T_I + 3^2 T_P \tag{7}$$

In the same situation, if interpolated states are considered, the needed time is (8).

$$T_I + T_P + T_{2\text{–}B} = T_I + T_P + 2T_B \tag{8}$$

In the general case, $T_I > T_P > T_B$. In order to obtain a first evaluation, we imagine that $T_I \approx 3T_P$ and $T_P \approx T_B$. In those circumstances, the use of interpolated states reduces the calculation time around 50 %. Then, it is clear that the use of interpolated states helps to fulfill the timing condition.

On the other hand, it is important to remark that interpolation techniques also present some disadvantages. As we said, interpolated states have a low precision, although they can be calculated in a very fast way. The second part helps to fulfill timing requirements, but the first idea refers to the need of more complex decision-making algorithms (depending on the case). If interpolation techniques are correctly designed and the CPS presents a regular behavior, then the loss of precision does not affect significantly the mobility and device lifecycle management. However, in CPS whose behavior tends to be very dynamical and changing (or if interpolation techniques are not correctly planned), the loss of precision might be remarkable and the decision-making modules should implement mechanism to operate in those conditions (what complicates the algorithm design and programming).

Finally, in Algorithm 2, any of the available CPS or Internet-of-Things (IoT) simulators is valid. For example, the simulator CyPhySim [20, 25] proposed by the Berkeley University could be employed. Other options are the NS3 simulator [26], the SimpleIoTsimulator suite [27], or the junction of MATLAB and Simulink [28]. Relating to the interpolation process, also any of the available techniques may be used: from linear interpolation [29] to cubic splines [30].

3.3 Decision-making and predictive solutions

Once a sequence of future states is generated, the decision-making algorithms evaluate the information and decide to execute the appropriate actions. Many possible decision-making algorithms may be employed in the predictive algorithms. On the most efficient case, intelligent decision-making [31] would be used. However, in this work, we have designed a more simple strategy.

Algorithm 3 Decision making

Input: Sequence of predictive and interpolated states SQ

List of relevant changes LC

for each change C in LC **do**
  if SQ predicts that C occurs **then**
    Initiate the proper transaction
    Put the transaction in pending
    Invoke asynchronous service as which returns when the change really occurs
  end if
end for each
Wait until the obtaining time of the next intra-state
Cancel all pending transactions and release all allocations not used

callback as
  Accept the initiated transaction
end callback

In our proposal, in each sequence of future states, the decision-making algorithm looks for the relevant changes defined in the system (such as a device running out of battery charge). Then, if any of them is found, the system will initiate the transaction for the system reconfiguration immediately. Once all the parameters have been negotiated, all data transmitted, etc., the transaction is pending. In the moment when the change really occurs, the transaction is finally accepted and the CPS is reconfigured. If, at the end, the change never happens, the transaction is canceled. Algorithm 3 presents the decision-making function specifically designed for this work.

On the other hand, in advanced scenarios, decision-making module may also change the scheme of interpolated and predictive states. As we said, CPS present a very dynamical behavior. In some circumstances, the selected scheme for the interpolated and predictive states at first might not be adequate when the CPS evolves. In those cases, the decision-making algorithms may implement mechanism to detect the increase in the number of unpredicted changes and to reconfigure the scheme of states in order to be more effective.

4 Practical implementation and experimental validation

For the first practical implementation of the proposed solution, we have selected as deployment scenario a laboratory at the Technical University of Madrid. The laboratory is organized in three rooms (see Fig. 5), being deployed a different CPS (i.e., a different *hardware manager*) in each room.

In this scenario, three predictable relevant changes can occur: (a) devices move from the coverage area of one CPS to the coverage area of another, (b) devices leave all the deployed CPS, or (c) devices run out of battery and become unavailable. In the first case, a handover must be executed. In the second and third case, if necessary, tasks being executed by the device have to be delegated or canceled. In advance scenarios, techniques for avoiding the predicted events could also be employed. For example, in applications involving humans, messages to prevent them from leaving the system could be sent. These advanced solutions, however, are not considered in this work.

In CPS, processes are not assigned to only one device. Instead, processes are divided in several tasks which are executed in the hardware platform. For this reason, if one device is transferred to other CPS, the whole context (process) cannot be transferred. Then, handovers in our CPS are managed as a case of IPv6 mobility [32] and

Fig. 5 Deployment scenario

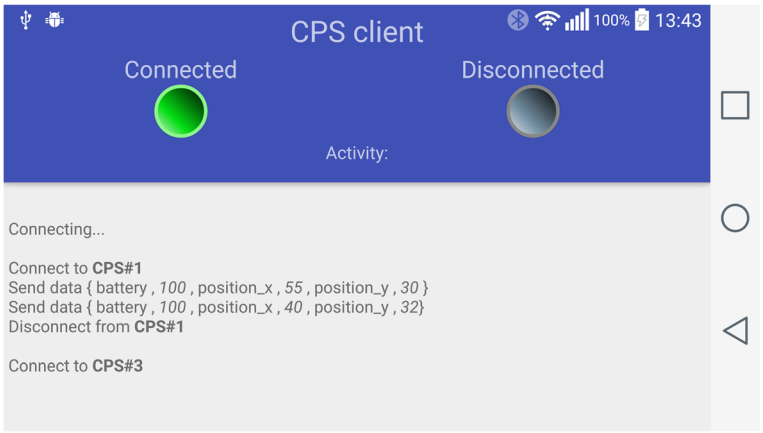

Fig. 6 CPS client application

not as traditional handovers from mobile networks. In that way, the target CPS acts only as a proxy, connecting the source CPS with the transferred device.

In the case that the device leaves all CPS or runs out of battery charge, if the estimated time to finish the assigned tasks is greater than the remaining time until the device becomes unavailable, resources are allocated in other device or devices, and the necessary data are transmitted to them. When the device becomes unavailable, the new execution is activated. If, at the end, the device remains available, the allocated resources are released.

We performed a system deployment in order to validate the proposed solution. For that, the hardware platform in the described CPS is made of various smartphones and tablets, where an application called *CPS Client* was installed. This application (see Fig. 6) allows the device to be part of hardware platform when pressing the *connect* button and releases the device by pressing the *disconnect* button. When acting as an element of the hardware platform, the device tries always to be connected and sends periodically the needed data to the *hardware manager* to create the mandatory intra-states. It also shows some information about the activity in the system.

In this scenario, differential acquisition is not available, and the employed simulator is an integration of the NS-3 simulator and the suites MATLAB and Simulink [33]. The pattern of predictive and interpolated states is as follows (9) (time given in seconds). As can be seen, $h = 1\ s$ and $T_{\text{sim}} = 3\ s$. In order to select these values, we tried to not commit errors up to the maximum typical error considered in engineering. We considered people walking to have a medium speed of $1.1\,\frac{m}{s}$ [34]. Then, in a room with 30 m long, in 3 s a person has varied its situation in more than a 10 % (the typical limit to consider a value negligible in engineering).

$$S^I_{t=0} \quad S^B_{t=1} \quad S^B_{t=2} \quad S^P_{t=3} \tag{9}$$

In the proposed scheme of future system states, two interpolated states have been considered. Walking people tend to follow a very predictable path and batteries are discharged slowly, so various interpolated states may be included without committing great errors. It must be noted that a matrix cubic spline technique was implemented as interpolation technology.

In this scenario, thirty-six (36) people are provided with the *CPS Client* and are requested to operate in the coverage area of the CPS. They also could leave that zone. The human behavior perfectly represents the complicated dynamic of CPS. Data about the number of managed changes, the success rate and the calculation time were collected.

5 Results and discussion

More than 400 relevant changes were registered while performing the experimental validation. Figure 7 shows the distribution of the registered changes, depending on

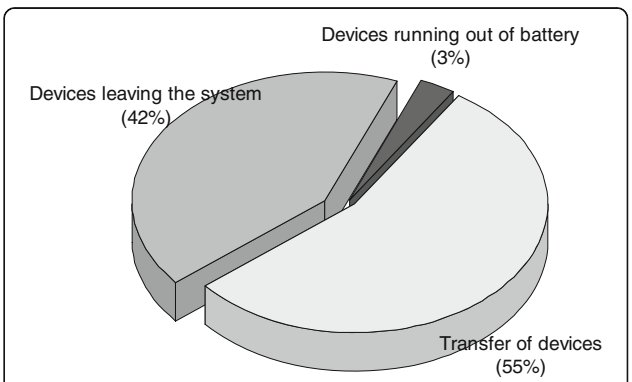

Fig. 7 Classification of the registered events into types

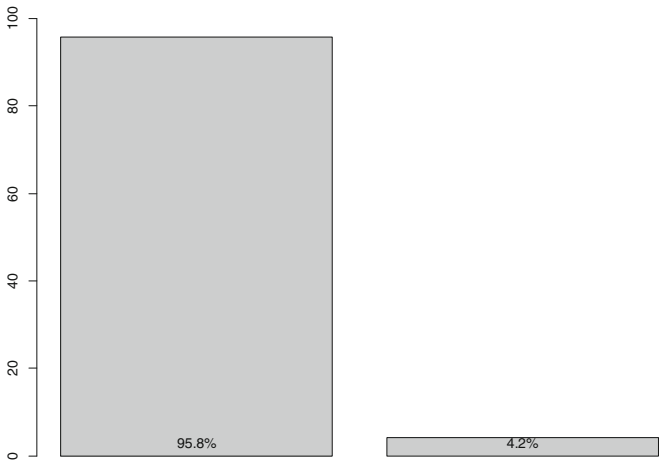

Fig. 8 Aggregated success rate

their type. As can be seen, most of them were handovers (55 %). This fact agrees with the scenario configuration, where three different CPS were deployed. On the other hand, changes about devices running out of battery charge were few (3 %), as most devices had batteries with enough capacity to be independent for dozens of hours.

Most of these changes were successfully managed, i.e., the expected actions are taken when the change occurs (and only if the change occurs). The aggregated success rate is slightly over 95 % (see Fig. 8). With respect to the wrongly managed events (around a 4 %), all errors are due to changes unpredicted by the simulator, so the management algorithm failed. The underlying cause of these misinterpretations is the dynamical model employed to represent the system evolution, which for some phenomena (such as the batteries discharge process) implements laws which only are approximated mathematical functions due to the complex behavior of these phenomena. For example, typically the battery charge is overestimated, and devices may turn off before executing the adequate actions. Advanced CPS models [35] would solve this situation.

In conclusion, the proposed predictive algorithm, where patterns include interpolated states, clearly allows the correct management of changes in the hardware platform (despite the greater error included).

In fact, around the 4 % of events are unpredicted by the simulator. However, some additional considerations should be given to Fig. 8. If the success rate is disaggregated into the three event types, some relevant facts might be exposed (see Fig. 9).

First, as can be seen, the success rate in the case of transferring a device to a second CPS (a handover) is higher than the aggregated rate. If we consider a device leaving the entire deployment, the success rate is, more or less, equal to the aggregated rate. These high values for the success rate may be associated to a good model for the device movement in the simulator. Then, most displacements are correctly predicted.

On the other hand, success rate in the case that a device runs out of battery charge is much fewer (around a 30 % fewer) than the aggregated rate. Clearly, that is due to the complexity associated to the battery charge simulation. Generic models are employed in these simulations, but they poorly fit the real evolution of battery charge. Then, an important percentage of the events of this type are not predicted. Nevertheless, as this type of events only represents the 3 % of the total amount events, the impact in the aggregated success rate is limited.

Finally, Fig. 10 represents the histogram of the calculation time employed in obtaining each one of the 250 sequences of future states generated during the experimental validation. As can be seen, the most

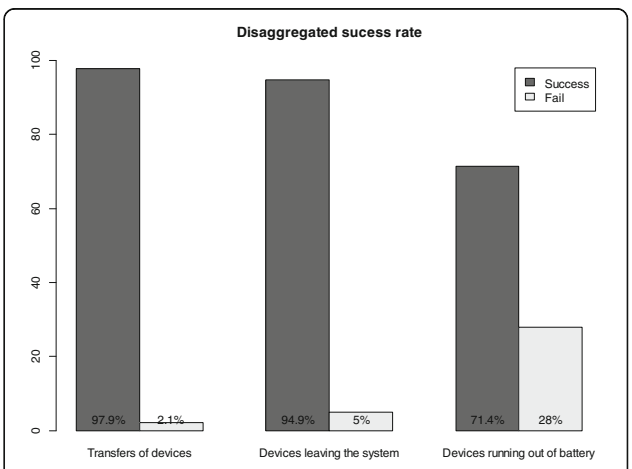

Fig. 9 Disaggregated success rate

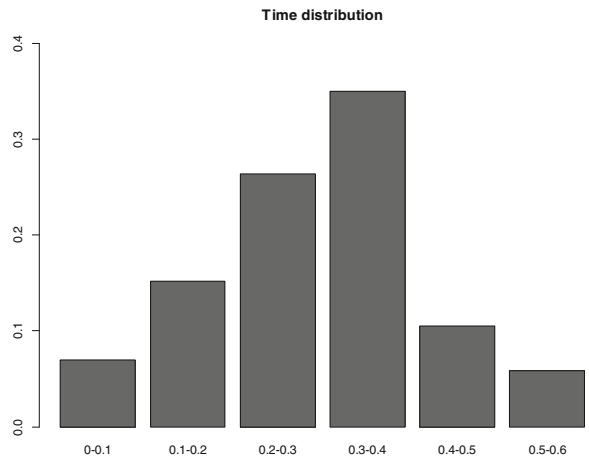

Fig. 10 Calculation time distribution

probable time is the interval of 0.3–0.4 s, so it is clearly smaller than the time step ($h = 1\ s$). Besides the maximum calculation time was 0.57 s, which continuous being smaller than the time step. In conclusion, the proposed solution meets the timing condition.

6 Conclusions

Most practical discussions about the characteristics of Cyber-Physical systems (CPS) conclude that they are composed of wireless, embedded, mobile devices. Thus, techniques for mobility and device lifecycle management are necessary. Traditional reactive solutions are not valid in CPS, as they present a complex, and sometimes random, dynamic. The proposed basic predictive solution cannot be employed as they do not fulfill the timing requirements. Therefore, in this paper, we propose an advanced predictive technique for managing the mobility and device lifecycle, being able to meet all requirements of CPS. The solution is based on an infinite loop, which calculates, in each iteration, a sequence of future system states using a CPS simulator and interpolation algorithms.

As we saw, the obtained success rate is higher than the 95 %, so the proposed solution correctly manages the changes in the hardware platform. However, most complicated elements (such as the battery charge) present difficulties to be simulated correctly. Besides, timing requirements are comfortably met.

The proposed solution is valid for all types of CPS; however, improved simulation models are needed and tools for generating them automatically are also necessary. Obtaining these technologies will determine the future commercial success of our proposal. Moreover, reactive techniques which complement the proposed predictive solution are necessary, in order to create a really useful hardware management algorithm for CPS.

Funding
The research leading to these results has received funding from the Ministry of Economy and Competitiveness through SEMOLA project (TEC2015-68284-R) and from the Autonomous Region of Madrid through MOSI-AGIL-CM project (grant P2013/ICE-3019, co-funded by EU Structural Funds FSE and FEDER).

Authors' contributions
The contributions described in this work are distributed among the authors in the way that follows: all the authors conceived and designed the solution; BB and RA wrote the paper; DSdeR performed the experiments; and ÁS-P programmed the simulator. All authors read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]Department of Telematics Systems Engineering, Universidad Politécnica de Madrid, Avenida Complutense n° 30, 28040 Madrid, Spain. [2]Department of Topographic Engineering and Cartography, Universidad Politécnica de Madrid, Campus Sur, 28031 Madrid, Spain.

Received: 1 June 2016 Accepted: 15 September 2016
Published online: 22 September 2016

References

1. KD Kim, PR Kumar, Cyber-Physical Systems: a perspective at the centennial. Proc. IEEE **100**(Special Centennial Issue), 1287–1308 (2012)
2. K Wan, KL Man, D Hughes, Specification, analyzing challenges and approaches for Cyber-Physical Systems (CPS). Eng. Lett. **18**(3), 308 (2010)
3. M Jiménez, R Palomera, I Couvertier, *Introduction to embedded systems* (Springer, New York 2013). ISBN: 978-1-4614-3142-8. doi: 10.1007/978-1-4614-3143-5
4. S Aram, I Khosa, E Pasero. Conserving energy through neural prediction of sensed data. JoWUA. **6**(1), 74-97 (2015)
5. A Koubâa, B Andersson, *A vision of cyber-physical internet. In 8th International Workshop on Real-Time Networks (RTN'09)* (Instituto Politécnico do Porto. Porto, 2009). http://hdl.handle.net/10400.22/3837
6. A Bindel, P Conway, L Justham, A West, New lifecycle monitoring system for electronic manufacturing with embedded wireless components. Circuit World **36**(2), 33–39 (2010)
7. B. Bordel Sanchez, Á. Sánchez-Picot, & D. Sanchez De Rivera, Using 5G technologies in the Internet of things handovers, problems and challenges. in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2015 9th International Conference on* (IEEE, 2015), pp. 364-369.
8. B Bordel Sánchez, R Alcarria, D Martín, T Robles, TF4SM: a framework for developing traceability solutions in small manufacturing companies. Sensors **15**(11), 29478–29510 (2015)
9. V Katiyar, N Chand, N Chauhan, Recent advances and future trends in wireless sensor networks. Int. J. Appl. Eng. Res. **1**(3), 330 (2010)
10. N. Keddis, G. Kainz, C. Buckl, & A. Knoll, Towards adaptable manufacturing systems. in *Industrial Technology (ICIT), 2013 IEEE International Conference on* (IEEE, 2013), pp. 1410-1415
11. DD Hoang, HY Paik, CK Kim, Service-oriented middleware architectures for Cyber-Physical Systems. Int. J. Comput. Sci. Netw. Secur **12**(1), 79–87 (2012)
12. U Mönks, H Trsek, L Dürkop, V Geneiß, V Lohweg, Assisting the design of sensor and information fusion systems. Procedia Technol **15**, 35–45 (2014)
13. T. Dillon, V. Potdar, J. Singh, & A. Talevski, Cyber-Physical Systems: providing quality of service (QoS) in a heterogeneous systems-of-systems environment. in *Digital Ecosystems and Technologies Conference (DEST), 2011 Proceedings of the 5th IEEE International Conference on* (IEEE, 2011), pp. 330-335.
14. TS Dillon, H Zhuge, C Wu, J Singh, E Chang, Web-of-things framework for Cyber-Physical Systems. Concurrency Comput. Pract. Exp **23**(9), 905–923 (2011)
15. D. Work, A. Bayen, & Q. Jacobson, Automotive cyber physical systems in the context of human mobility. in *National Workshop on high-confidence automotive Cyber-Physical Systems* (Troy, Berkeley University 2008), http://bayen.eecs.berkeley.edu/sites/default/files/conferences/cps1.pdf

16. V. C. M. X. Hu, W. Leung, B.C. Du, P. Seet, & P. Nasiopoulos, *A service oriented mobile social networking platform for disaster situations in Proc. 46th HICSS* (2013), pp. 136–145
17. X Hu, T Chu, H Chan, V Leung, Vita: a crowdsensing-oriented mobile Cyber-Physical System. Emerg. Top. Comput. IEEE Trans **1**(1), 148–165 (2013)
18. C Fok, A Petz, D Stovall, N Paine, C Julien, S Vishwanath, *Pharos: a testbed for mobile Cyber-Physical Systems*. (The University of Texas, The Center for Advanced Research in Software Engineering, Austin, 2011). TR-ARiSE-2011-001
19. J Fink, A Ribeiro, V Kumar, Robust control for mobility and wireless communication in Cyber-Physical Systems with application to robot teams. Proc. IEEE **100**(1), 164–178 (2012)
20. E.A. Lee, M. Niknami, T.S. Nouidui, & M. Wetter, Modeling and simulating Cyber-Physical Systems using CyPhySim. in *Proceedings of the 12th International Conference on Embedded Software* (IEEE Press, 2015), pp. 115-124
21. R Poovendran, K Sampigethaya, SKS Gupta, I Lee, KV Prasad, D Corman, JL Paunicka, Special issue on Cyber-Physical Systems [scanning the issue]. Proc. IEEE **100**(1), 6–12 (2012)
22. T Robles, R Alcarria, D Martın, M Navarro, R Calero, S Iglesias, M López, An IoT based reference architecture for smart water management processes. J. Wirel. Mob. Netw. Ubiquit. Comput. Dependable Appl **6**(1), 4–23 (2015)
23. F-Y Leu, H-L Chen, C-C Cheng, Improving multi-path congestion control for event-driven wireless sensor networks by using TDMA. J. Internet Serv. Inf. Secur **5**(4), 1–19 (2015)
24. P Satam, H Alipour, Y Al-Nashif, S Hariri, Anomaly behavior analysis of DNS protocol. J. Internet Serv. Inf. Secur **5**(4), 85–97 (2015)
25. C. Brooks, E. A. Lee, D. Lorenzetti, T. S. Nouidui, & M. Wetter, M. CyPhySim: a Cyber-Physical Systems simulator. in *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control* (ACM, 2015), pp. 301-302
26. NS3 simulator webpage. Available online: https://www.nsnam.org/ Accessed on 25 May 2016
27. SimpleIoTsimulator webpage. Available online: http://www.smplsft.com/SimpleIoTSimulator.html Accessed on 25 May 2016
28. CM Ong, *Dynamic simulation of electric machinery: using MATLAB/Simulink*, vol. 5 (Prentice Hall PTR, Upper Saddle River, 1998)
29. G. Barequet, & M. Sharir, Piecewise-linear interpolation between polygonal slices. in *Proceedings of the tenth annual symposium on Computational geometry* (ACM, 1994), pp. 93-102
30. C De Boor, *A practical guide to splines*, vol. 27 (Springer, New York, 1978), p. 325
31. NK Janjua, FK Hussain, OK Hussain, Semantic information and knowledge integration through argumentative reasoning to support intelligent decision making. Inf. Syst. Front. **15**(2), 167–192 (2013)
32. D Johnson, C Perkins, J Arkko, *Mobility support in IPv6 (No. RFC 3775)*, 2004
33. Dmitry Kachan. Integration of NS-3 with MATLAB/Simulink. Available online: http://epubl.ltu.se/1653-0187/2010/062/LTU-PB-EX-10062-SE.pdf. Accesed 25 May 2016
34. Y Feng, K Liu, Q Qian, F Wang, X Fu, Public-transportation-assisted data delivery scheme in vehicular delay tolerant networks. Int. J. Distrib. Sens. Netw. (2012)
35. D Martín, J García Guzmán, J Urbano, A Amescua, Modeling software development practices using reusable project patterns: a case study. J. Softw. Evol. Process **26**(3), 339–349 (2014)