Towards Reliability-based decision making in Cyber-Physical Systems

Saideep Nannapaneni, Sankaran Mahadevan
Department of Civil & Environmental Engineering
Vanderbilt University
Nashville, TN 37235, USA
saideep.nannapaneni@vanderbilt.edu
sankaran.mahadevan@vanderbilt.edu

Subhav Pradhan, Abhishek Dubey
Institute for Software Integrated Systems
Department of Electrical Engineering & Computer Science
Vanderbilt University
Nashville, TN 37235, USA
subhav.m.pradhan@vanderbilt.edu
abhishek.dubey@vanderbilt.edu

***Abstract*—Cyber-physical systems (CPS) are systems with a tight integration between the computational (also referred to as software or cyber) and physical (hardware) components. While the reliability evaluation of physical systems is well-understood and well-studied, reliability evaluation of CPS is difficult because software systems do not degrade and follow a well-defined failure model like physical systems. In this paper, we propose a framework for formulating the CPS reliability evaluation as a dependence problem derived from the software component dependences, functional requirements and physical system dependences. We also consider sensor failures, and propose a method for estimating software failures in terms of associated hardware and software inputs. This framework is codified in a domain-specific modeling language, where every system-level function is mapped to a set of required components using functional decomposition and function-component association; this provides details about operational constraints and dependences. We also illustrate how the encoded information can be used to make reconfiguration decisions at runtime. The proposed methodology is demonstrated using a smart parking system, which provides localization and guidance for parking within indoor environments.**

Keywords—Reliability; Self-reconfiguration, Cyber-Physical Systems; Smart Parking;

I. Introduction

Cyber-Physical Systems are increasingly becoming widespread to handle complex tasks in several fields of technology, engineering, and medicine such as smart power systems, smart buildings, smart public transportation, self-driving cars, avionics, medical robotics and smart manufacturing systems. Since CPS are used in several diverse and key applications, it becomes necessary to evaluate their reliability when handling complex tasks because a single failure can result in large financial and safety consequences.

As many CPS operate in real-time, another key requirement is that the analysis be carried out with strong timing constraints [1]. The operation of a CPS can be divided into three possible scenarios: (1) CPS provides the correct solution, within expected time, (2) CPS does not provide correct solution (i.e., incorrect solution or no solution at all), and (3) CPS provides the correct solution, but not within expected time. The last two cases are considered as system-level failure cases of a CPS. Current reliability literature considered only the functional requirements, i.e., CPS is assumed to have failed when no operating configuration is available [2-6]. In this work, we incorporate the timing constraints along with value domain constraints while evaluating the reliability of a CPS.

Software reliability, thus far, has been based on code verification and finding programming errors [7]. In this paper, we propose a new technique for assessing software reliability in terms of the associated hardware and inputs to the software. A lot of literature are available on the reliability estimation of several components in a CPS (software, hardware, sensors) individually but characterization of dependence between components and the manner in which the dependence influences the reliability is a key challenge in CPS. Our approach is based on model-based analysis framework where the system is modeled in a domain specific modeling language [8] in which each system-level function is associated to a component(s) through functional decomposition and function-component association. The availability of required functions can be inferred through monitoring the health of components. To check if a CPS satisfies timing requirements, the uncertainty associated with the analysis time is represented through a probability density function (PDF) and the probability that the analysis time is greater than a pre-defined threshold is computed. The overall reliability is then evaluated as a union of the aforementioned two failure cases. Thus, the developed reliability evaluation can be used in the selection of a design from a set of design alternatives.

When the system in a working configuration fails, the system needs to be re-configured for its continued operation. One possible approach for re-configuration is to enumerate all possible configurations and develop several re-configuration criteria depending on the failures of components. This is an explicit encoding approach that works in CPS with a few working configurations but as the system becomes complex, the number of working configurations increase exponentially. Therefore, it becomes essential to choose a reconfiguration in an automated manner. In this paper, we propose to find the configuration with the highest reliability near the current configuration by formulating it as an optimization problem.

The contributions of this paper can be summarized as – (1) Reliability analysis framework for CPS and initial design selection, and (2) Presenting our current work that uses the reliability analysis framework for reliability-based runtime re-configuration of CPS.

The remainder of this paper is organized as follows – Section II presents related work, Section III provides the proposed methodologies for reliability analysis for design selection and system re-configuration. The developed methodologies are demonstrated using a Smart Parking system in Section IV, followed by Conclusions and Future Work.

II. Related work

Some existing techniques for CPS reliability evaluation and reconfiguration are surveyed below.

Reliability evaluation: Wu, Huang, Zheng and Li [2] developed a reliability model where Markov models are constructed for each component to estimate the reliability of an Integrated Modular Avionics (IMA) system. Using the component Markov models, the probability that all the components reached a failed state is computed. A Markov imbedded system (MIS) is used in [3] to model dependence between components in a smart power grid. Given the functional and non-functional modes, the probability that the system is in one of the functional modes is computed. In [4], a phased-mission system model, which consists of Markov models for individual components and a binary decision diagram (BDD) is used to analyze the reliability of a fuel management system in an aircraft. Li and Kang [5] developed a reliability framework through a weighted reliability metric using individual component reliabilities and the performance metric of the CPS considering service, cyber security, resilience, elasticity and vulnerability. Wu and Kaiser [6] developed FARE (Failure Analysis and Reliability Estimation), a data-driven approach for reliability evaluation using historical data, accelerated life testing data and real-world data. The literature so far considered only operational requirements but in this work, we seek to consider both operational and timing requirements for reliability estimation of a CPS.

Reconfiguration analysis: Wu and Kaiser [9] developed ARIS (Autonomic Reliability Improvement System), a data-centric runtime monitoring system that conducts automated evaluation at multiple stages, provides real-time feedback and self-tunes the system for reliability improvement. A dynamic re-configuration through a distributed policy-based framework in mobile autonomous systems in [10]. Shankaran et al [11] developed RACE (Resource Allocation and Control Engine) for managing performance of applications in distributed real-time embedded systems. RACE monitors the resource usage, infrastructure performance and the reconfiguration is based on control algorithms. Pradhan et al [12] developed a self-adaptive and resilient Deployment and Configuration (D&C) infrastructure for highly dynamic component-based CPS operating in resource-constrained environments. Reconfiguration analyses have thus far been policy-based (where failures and corrective actions are detailed before system operation) or expert-based or considered downtime and resources at runtime. In this work, we investigate reliability-based reconfiguration at runtime.

III. Reliability evaluation

In this section, different types of redundancies are discussed followed by reliability analysis framework for design selection and reconfiguration.

Active and Passive redundancy: In active redundant systems, the redundant components are also in operation, and when the original component fails, the redundant components can be used without any downtime. In passive redundant systems, the redundant components are only used when the original component ceases to work. In this case, there exists some downtime between the loss of original component and the start of the redundant component. Consider a system with two TMR (Triple Modular Redundancy) components where one acts as the original and the other as a backup component. This system is associated with both active and passive redundancy. The active redundancy arises in that three components are present in each TMR component and passive redundancy arises because the backup component is used only if the original TMR component fails. Fig. 1 provides a graphical illustration of active and passive redundancy.

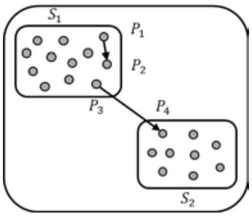

Fig. 1. Figure showing transitions between active and passive configurations

Each small rectangle in Fig. 1 (S_1, S_2) represents a set of active configurations. Configuration changes within (S_1 or S_2) represent active redundancy (P_1 to P_2) as opposed to changes across S_1, S_2 which represent passive redundancy (P_3 to P_4). The bigger rectangle that encompasses all smaller rectangles represent all possible configurations in that system (both active and passive). When estimating the overall reliability, both active and passive redundancies are treated alike. However, for re-configuration, passive re-configurations incur additional costs and downtime due to the reconfiguration process unlike active re-configurations, which do not cause any downtime. Therefore, that additional cost should also be included in reconfiguration analysis.

A. Reliability analysis and design selection of a CPS

The first step in choosing a design alternative is to model the system using a modeling language to capture the component-component interactions. Let the failure events (mentioned in Section I) corresponding to the two failure cases be represented as W, T respectively. Therefore, the overall failure probability for the system is defined as

$$P_f^s = \Pr(W \cup T) = \Pr(W) + \Pr(T) - \Pr(W \cap T) \qquad (1)$$

It should be noted that $W \cap T = \emptyset$ and therefore, $\Pr(W \cap T) = 0$ because as mentioned above, W and T refer to the non-availability and availability of valid configurations and therefore are mutually exclusive. The probability for each failure case and overall failure probability is computed below.

Failure case 1: *CPS does not provide correct solutions*

CPS has multiple subsystems such as a software system, physical system, sensor network, and a communication system; their reliability evaluation are discussed below.

1) Reliability of the sensor network: Let $\zeta(S)$ and $\zeta(S_S)$ represent the total number of sensors and minimum required number of sensors for operation. The number of possible combinations for selecting $\zeta(S_S)$ sensors from $\zeta(S)$ is given as $\binom{\zeta(S)}{\zeta(S_S)}$. All possible combinations are enumerated and reliability for each combination is estimated using the reliability information of the individual sensors, which can then be used for reliability of overall sensor network.

2) Reliability of a distributed software system: When a software application is designed, the ranges of inputs are chosen and the application is designed, tested and validated in these ranges. In this work, we assume that rigorous testing of the software application has been carried out (e.g., software used in critical systems such as avionics and nuclear power plants) and that when the inputs to the CPS are within these ranges, the software always works without failure. When the inputs go beyond the designed ranges, then the software is assumed to fail. Note that the software probability is independent of time, as opposed to hardware probability, which increases with time. Note that by assuming that software always within given input ranges, an upper bound of the reliability estimate is obtained.

Let $\boldsymbol{I} = \{I_1, I_2 \ldots I_N\}$ represent the N inputs to the software system. Let $\boldsymbol{I_L}$ and $\boldsymbol{I_U}$ represent the lower and upper bounds of the nominal values of those N variables for which the software is designed. In cases when $\boldsymbol{I} < \boldsymbol{I_L}$ or $\boldsymbol{I} > \boldsymbol{I_U}$**,** the software application is assumed to fail. The failure probability of the distributed software system (DS) is analyzed under two conditions – (1) when $\boldsymbol{I} < \boldsymbol{I_L}$ or $\boldsymbol{I} > \boldsymbol{I_U}$, and (2) $\boldsymbol{I_L} \leq \boldsymbol{I} \leq \boldsymbol{I_U}$**.** The overall failure probability, using theorem of total probability [13], can be computed as

$$\begin{aligned} P_f(DS) = {} & P_f(DS|\boldsymbol{I} < \boldsymbol{I_L} \cup \boldsymbol{I} > \boldsymbol{I_U})\mathrm{Pr}(\boldsymbol{I} < \boldsymbol{I_L} \cup \boldsymbol{I} \\ & > \boldsymbol{I_U}) + P_f(DS|\boldsymbol{I_L} \leq \boldsymbol{I} \leq \boldsymbol{I_U})\mathrm{Pr}(\boldsymbol{I_L} \\ & \leq \boldsymbol{I} \leq \boldsymbol{I_U}) \qquad (2) \\ = {} & \mathrm{Pr}(\boldsymbol{I} < \boldsymbol{I_L} \cup \boldsymbol{I} > \boldsymbol{I_U}) + P_f(DS|\boldsymbol{I_L} \leq \boldsymbol{I} \leq \\ & \boldsymbol{I_U})(1\text{- } \mathrm{Pr}(\boldsymbol{I} < \boldsymbol{I_L} \text{ or } \boldsymbol{I} > \boldsymbol{I_U})) \end{aligned}$$

In (2), $\mathrm{Pr}(.)$ refers to the probability function. The probability that the inputs are beyond the bounds can be obtained through aggregation of information from historical records, simulation data. The failure probability (P_f) when the inputs are within the bounds can be written as

$$P_f(DS|\boldsymbol{I_L} \leq \boldsymbol{I} \leq \boldsymbol{I_U}) = P_f(NN \cap S_N|\boldsymbol{I_L} \leq \boldsymbol{I} \leq \boldsymbol{I_U}) \qquad (3)$$

where S_N is the minimum set of computational nodes required to carry out the distributed software application and NN is the communication between these nodes. Let $\zeta(DS)$ represent the total number of computational nodes and $\zeta(S_N)$ represent the minimum number of computational nodes required. Thus, all $\binom{\zeta(DS)}{\zeta(S_N)}$ combinations are enumerated and failure probability is evaluated for each combination.

For illustration, the reliability computations for commonly used synchronous request-reply software architectures are described below. In Fig. 2, C_n, $n \in N$ refer to software applications. Fig. 2(a) can be understood as "C_2 implies C_1" and "C_1 implies C_2". Assume that the software applications are hosted only on one hardware without any redundancy. Let the hardware associated with C_1, C_2 and communication between C_1, C_2 be represented as H_1, H_2 and H_{12}. Using Boolean notation, the reliability expression is given as $(H_1 \wedge H_{12}) \wedge (H_2 \wedge H_{12})$ which can be simplified to $(H_1 \wedge H_2 \wedge H_{12})$. Therefore, the reliability is equal to $R_{H_1} \times R_{H_{12}} \times R_{H_2}$.

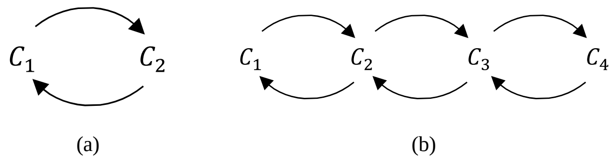

Fig. 2. Examples of synchronous software architectures

Fig. 2(b) shows a complex chain of synchronous request-pattern whose reliability expression is given as $(H_1 \wedge H_{12}) \wedge (H_2 \wedge H_{12}) \wedge (H_2 \wedge H_{23}) \wedge (H_3 \wedge H_{23}) \wedge (H_4 \wedge H_{34}) \wedge (H_3 \wedge H_{34})$; this can be simplified to $(H_1 \wedge H_2 \wedge H_3 \wedge H_4 \wedge H_{12} \wedge H_{23} \wedge H_{34})$. Thus, the reliability can be computed as $R_{H_1} \times R_{H_2} \times R_{H_3} \times R_{H_4} \times R_{H_{12}} \times R_{H_{23}} \times R_{H_{34}}$.

Consider the software architecture in Fig. 2(a) but assume each of software applications C_1 and C_2 is hosted on two systems connected as shown in Fig. 3.

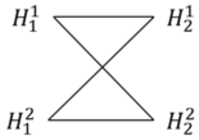

Fig. 3. Example of synchronous software architecture with redundancy

Four operational paths are available by connecting one of $\{H_1^1, H_1^2\}$ to one of $\{H_2^1, H_2^2\}$. The reliability of the path associated with H_1^1 and H_2^1 is given as $R_{H_1^1} \times R_{H_2^1} \times R_{H_{12}^{11}}$, where $R_{H_{12}^{11}}$ refers to the reliability of communication system between H_1^1 and H_2^1. Similarly, the reliability of all the four paths can be computed and represented as R_1, R_2, R_3 and R_4. Note that only one of four paths is required for its operation; therefore, the four paths are connected in parallel. The reliability that at least one path is available is equal to $1 - \prod_1^4(1 - R_n)$. Note that in some cases, the software applications on different nodes may be dependent on each other, i.e., the failure of a node may result in a faulty output or no output which could result in failure of other nodes; resulting in a cascade of nodal failures.

3) Reliability of the physical system: The outputs of the software system are a set of control actions to be taken on the physical system. Using functional decomposition and function-component association, a control action can be mapped to a set of component(s), which can be used to construct the reliability block diagram to assess its reliability. The reliability information on the individual hardware components is used for reliability estimation of the overall hardware system [14].

4) Reliability of communication systems (sensor-to-software, node-to-node and software-to-hardware): The wireless communication system requires hardware components such as network adapters. The common ways of actuation are through pneumatics, hydraulics. Therefore, the reliability of the communication system depends on the reliability of the hardware components. With available reliability information, the reliability of the communication system can be assessed.

After estimating the reliability of sensor network (R_S), the software system (R_{DS}), the physical system (R_H) and communication system (R_{CS}), the probability of no available configuration is given as

$$\Pr(W) = 1 - R_S \times R_{DS} \times R_H \times R_{CS} \quad (4)$$

<u>Failure case 2</u>: *CPS provides correct solution, but not within expected time*

Two approaches (a black box and a white box approach) to estimate the variation in the analysis time are discussed below.

1) *Black box approach*: In this approach, the CPS is considered as a black box, i.e. the architecture of CPS is unknown. The CPS is run multiple times and a PDF for the analysis time is constructed. Given the PDF, the failure probability that the analysis time is greater than a pre-defined threshold value (T_{th}) can be calculated. One possible threshold value could be the inverse of the frequency at which sensor data is collected by the CPS ($T_{th} = 1/F$).

2) White box approach: In the white box approach, the architecture of the CPS is known. Since a CPS consists of multiple subsystems, the overall analysis time is a summation of the analysis time of all the subsystems (sensor-to-software communication, node processing time etc.). Given the PDFs of times at each stage, the PDF of the overall analysis time (A_T) can be estimated using Monte Carlo sampling. Given a pre-defined threshold value, the failure probability can be estimated (5). One benefit of white box approach is that it becomes possible to identify the subsystem that has a high contribution to the failure probability and modify the design to reduce it.

$$\Pr(T) = \Pr(A_T > T_{th}) \quad (5)$$

One should note that the failure analysis associated with timing requirements is performed separately for every configuration. Thus, the failure probability of every configuration is the sum of failure probabilities of both failures cases. The overall failure probability of CPS is calculated using the failure probabilities of all configuration points. By applying the proposed reliability analysis methodology to several design alternatives, the one with the highest reliability can be chosen.

B. *Real-time re-configuration*

When the CPS in a current configuration fails, the system needs to be reconfigured for its continued operation. Depending on the failed component(s), some configurations that share the common failed component(s) become unavailable. Several re-configuration strategies can be used such as minimization of downtime or loss of utility but in this work, we investigate the maximization of reliability. Each configuration has an associated operation cost, which corresponds to the cost for operating several components in that configuration. Hence, the re-configuration is made such that the reliability of the new configuration is maximal while the operation cost is less than a predefined threshold.

Our approach to system reconfiguration relies on the concept of configuration space and configuration points [15]. A configuration space represents system description with respect to different components and their associated functionality, resource availability, resource requirements, and deployment constraints. A configuration space can contain multiple configuration points, where a configuration point represents a valid deployment of the system such that all functionalities are satisfied and there are no resource or deployment constraint violations. In this approach, reconfiguring a system means transitioning from one configuration point to another. To compute a new configuration point, we formulate the problem as a Satisfiability Modulo Theories (SMT) problem [16]. We are currently working on formulating the optimization problem to maximize the reliability and obtain the reconfiguration point by adding appropriate constraints. The re-configuration process is shown in Fig. 4.

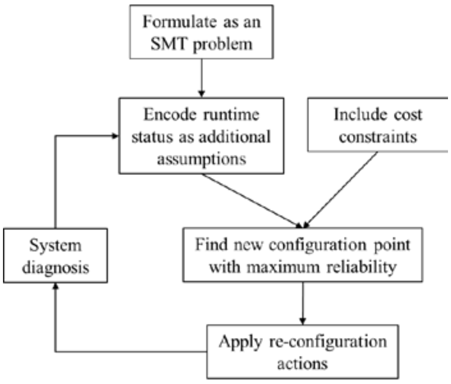
Formulate as an SMT problem
Encode runtime status as additional assumptions
Include cost constraints
System diagnosis
Find new configuration point with maximum reliability
Apply re-configuration actions

Fig. 4. Run-time system re-configuration analysis framework

Though we are working on a single-objective optimization problem in terms of reliability, if necessary, a multi-objective optimization problem can also be defined to maximize the reliability and minimize the operation cost. In such cases, techniques such as Pareto-front [17] can be used to obtain the reconfiguration point. It should be noted that it is computationally very expensive and not affordable in real time to find the configuration with the highest reliability over all possible configurations (global maximum); therefore, as part of our ongoing research effort, we focus on local maximum near the current configuration as the reconfiguration point. In order to enforce the concept of locality to the SMT problem, we make assumptions about the components that have not failed.

IV. Illustration Example: Smart Parking System

Fig. 5 shows a Smart Parking system, which is an automated system that provides localization and guidance in indoor parking structures. We first describe the smart parking system and later present its reliability computation.

A. *System Description*

The components associated with this system include – (1) a DecaWave sensor system, (2) a distributed system for indoor localization, and (3) automobile that requires parking assistance. When an automobile arrives, a parking space is allocated to the automobile, and the software system guides the automobile towards the allocated parking space. The width and length of a parking space are assumed as 2.5 m and 5 m respectively. The overall dimensions of the parking structure are assumed as 17.5 m x 15 m x 2.5 m.

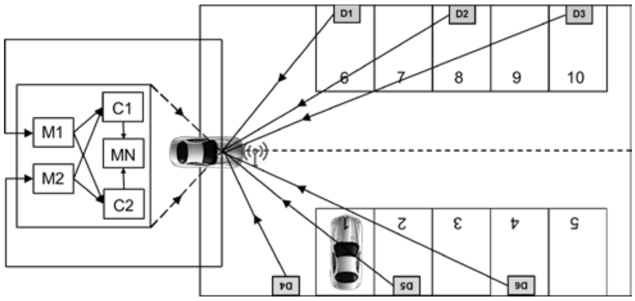

Fig. 5. Illustration of a smart parking system

DecaWave (sensor) system: In this work, we use DecaWave technology due to their high effectiveness for indoor localization with an accuracy of about 10-15 cm [18]. The DecaWave technology is based on Time of Arrival (TOA) or Time Difference of Arrival (TDOA), and uses the Ultra-Wide-Band (UWB) technology, where signals are transmitted under multiple bandwidths for shorter duration as opposed to Radio-frequency identification (RFID) systems, which operate in narrow bandwidths, and comparatively longer duration [18]. In TOA systems, the time difference between the signal transmission and receipt is obtained, and given the time difference, the distance is computed as the signals travel at the speed of light. When an automobile arrives, a DecaWave signal receiver is mounted on it to receive the transmitted signals. The transmitters are mounted on the ceiling at the locations provided in Fig. 5. The frequency of data collection is assumed at 50 Hz, i.e., the time period between successive measurements is 20 ms. Theoretically, data from three transmitters is required for indoor localization but to improve the accuracy, data from at four transmitters is used.

Distributed software system: The software system consists of three elements – (1) two monitor nodes (M_1, M_2), (2) two computational nodes (C_1, C_2), and (3) a master node (MN) to integrate the outputs from individual computational nodes. The two monitor nodes record the health of the computational nodes and DecaWave transmitters, and distribute the sensor data to the computational nodes for analysis. The output of the master node is the estimated location of the automobile. At least one monitor node, one computational node and the master node is required for the system to be operational.

B. Reliability analysis

Each battery-operated DecaWave sensor is assumed to fail when the battery fails or any hardware on the sensor (such as memory) fails. The uncertainty in the time for receiving the data is represented using a Gaussian distribution. The two monitor nodes are assumed different; the primary node (M_1) is less expensive and less reliable compared to the backup node (M_2). All the data used for several parameters in this example are assumed for the sake of illustration. The network bandwidth is assumed as 10 Mbps and the amount of communication data between several components is assumed as 1 KB (equal to 8Kb). The communication time is assumed to vary linearly with the amount of data. A Gaussian distribution is used to represent its variation with a standard deviation of 10%. The processing times of the monitor, computational and master nodes are assumed 3, 6, and 6 times the sensor-to-monitor communication time respectively. The MTTF data (simulated) for all the components and the time at each stage of analysis (such as sensor-to-software communication, node processing, node-to-node communication and node-to-physical system communication) are provided in Tables I and II respectively. Note that the values in Table II represent the analysis times in one analysis cycle (20 ms).

TABLE I. FAILURE RATE DATA FOR ALL COMPONENTS (SIMULATED)

Component	Failure rate	Operation Cost
DecaWave transmitter (F_T)	6 months	10
DecaWave receiver (F_R)	6 months	10
Network communicator (F_{NC})	18 months	10
Primary Monitor node (F_{M_1})	18 months	100
Backup Monitor node (F_{M_2})	24 months	150
First Computational node (C_1, F_{C_1})	24 months	100
Second Computational node (C_2, F_{C_2})	18 months	150
Master node (F_{MN})	24 months	150

TABLE II. TIME AT EACH STAGE OF ANALYSIS

Components	Time
Receiver – Monitor nodes (T_{RM})	$N(0.78, 0.078)$ ms
Monitor nodes (T_M)	$N(2.343, 0.2343)$ ms
Monitor – Computational nodes (T_{MC})	$N(0.78, 0.078)$ ms
Computational nodes (T_C)	$N(4.687, 0.4687)$ ms
Computational – Master nodes (T_{CMN})	$N(0.78, 0.078)$ ms
Master node (T_{MN})	$N(4.687, 0.4687)$ ms
Master node – Physical system (T_{MNP})	$N(0.78, 0.078)$ ms

The overall analysis time for localization is given as the summation of T_{RM}, T_M, T_{MC}, T_C T_{CMN}, T_{MN}, T_{MNP} (Table II) which also results in a Gaussian distribution; the parameters (mean and standard deviation) can be computed as 14.843 and 0.72. Since data is available every 20 ms, the threshold value for analysis time is assumed as 20 ms. Assume that the reliability of the CPS is evaluated over a period of 1 day, which equals 432 x 10^4 cycles. Since the DecaWave receivers can be changed whenever a new automobile arrives, the receivers are always assumed to work and not included in reliability analysis. Also, the duration of any particular automobile that requires parking is small compared to total time (1 day); therefore, the automobile is also not included in reliability estimation. Let the operational cost threshold be 600 units. The operation costs for various components are provided in Table I. The overall failure probability, considering both operational and timing requirements, is approximately equal to 0.017. Thus, the overall reliability of the system considering all possible configurations is 0.983.

C. Reliability-based real-time re-configuration

Let D_1, D_3, D_4, D_6 be the DecaWave transmitters, M_1, C_1, C_2 be working in the initial configuration. The reliability of the initial configuration can be computed as 0.965, considering both failure cases. For illustration, let sensor D_4 and M_1 fail. As mentioned in Section V, the re-configuration is modeled as an SMT problem with failure of M_1 and D_4 as additional constraints. In the remaining possible configurations, the configuration with the highest reliability and within the operational cost constraints is chosen as the re-configuration point. The re-configuration with the highest reliability would be to replace M_1 with M_2 and choose either D_2, or D_5 to replace D_4. However, operating M_2 and both the computational nodes violates the cost constraints. Therefore, one of the two computational nodes should be used even though the processing time increases, which increases the failure probability. Between C_1 and C_2, C_2 is chosen due to its higher MTTF value and therefore higher reliability.

The initial processing time of the computational nodes is assumed to be 6 times the communication time, when both nodes are working. In case when only one node is working, then the processing time is higher and assumed at 7 times the communication time. The failure probabilities associated with the operational and time requirements in the new configuration are 0.0315 and 0.0326; therefore, the overall failure probability and reliability are 0.0641 and 0.9359 respectively. The goal accomplished by this problem, even though straightforward, is the illustration of the reliability evaluation in CPS and reliability-based re-configuration subject to cost constraints.

V. Conclusion and Future Work

This paper developed a framework for reliability analysis, which can be used for design selection and re-configuration in CPS. Real-time CPS are associated with both operational and timing constraints. The failures related to the physical system and sensors are modeled using their failure rates whereas the software is always assumed to work when its inputs are within designed ranges and fail if they are beyond the designed ranges. The uncertainty in the analysis time at each stage of analysis is represented using a PDF. Given individual distributions, the PDF of overall analysis time is obtained through Monte Carlo sampling, which can then be used to estimate the probability that the analysis time is greater than a threshold. The overall failure probability of a CPS is therefore the union of the failure probabilities associated with both operational and timing constraints. Re-configuration is modeled as an optimization problem in the configuration space to choose the configuration with the maximum reliability subject to cost constraints. A smart parking system that provides indoor localization and guidance is used to illustrate the proposed methods.

This paper was primarily focused on reliability analysis for design selection and runtime reconfiguration. Other factors that influence the decision making process include utility and downtime. Increase in redundancy results in a low downtime but increased operational costs. To this end, future work should develop a decision-making framework considering tradeoffs between the utility, reliability and downtime.

Acknowledgment

The work reported in this paper was supported under a grant from Siemens Corporate Technology. Any opinions, findings, and conclusions expressed here are those of the authors and do not necessarily reflect the views of the funding agency.

References

[1] E. A. Lee, “Cyber physical systems: Design challenges,” 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC), 2008, pp. 363-369.

[2] Z. Wu, N. Huang, X. Zheng, and X. Li, “Cyber-physical avionics systems and its reliability evaluation,” IEEE 4th Annual International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2014.

[3] K. Marashi, and S. S. Sarvestani, “Towards comprehensive modeling of reliability for smart grids: Requirements and challenges,” 15th IEEE International Symposium on High-Assurance Systems Engineering (HASE), 2014, pp. 105-112.

[4] X. Sun, N. Huang, B. Wang, and J. Zhou, “Reliability of cyber physical systems assessment of the aircraft fuel management system,” 4th IEEE Annual International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2014, pp. 424-428.

[5] Z. Li, and R. Kang, "Strategy for reliability testing and evaluation of cyber physical systems." IEEE International Conference on Industrial Engineering and Engineering Management, 2015, pp. 1001-1006.

[6] L. Wu, and G. Kaiser, “FARE: A framework for benchmarking reliability of cyber-physical systems,” IEEE Conference on Systems, Applications and Technology Conference (LISAT), 2013, pp. 1-6.

[7] A. K. Verma, S. Ajit, and D. R. Karanki, “Software Reliability,” Reliability and Safety Engineering, 2016, pp. 183-217, Springer London.

[8] S. M. Pradhan, A. Dubey, A. Gokhale, and M. Lehofer, “CHARIOT: A Domain Specific Language for Extensible Cyber-Physical Systems,” 2015.

[9] L. Wu, and G. Kaiser, “An autonomic reliability improvement system for cyber-physical systems,” 14th IEEE International Symposium on High-Assurance Systems Engineering (HASE), 2012, pp. 56-61.

[10] E. Asmare, A. Gopalan, M. Sloman, N. Dulay, and E. Lupu, “Self-management framework for mobile autonomous systems,” Journal of Network and Systems Management, 2012, 20(2), pp.244-275.

[11] N. Shankaran et al, “A framework for (re) deploying components in distributed real-time and embedded systems,” Proceedings of ACM symposium on Applied computing, 2006, pp. 737-738.

[12] S. Pradhan et al, “Towards a Self-adaptive Deployment and Configuration Infrastructure for Cyber-Physical Systems,” ISIS, 2014, 14, p.102.

[13] S. Mahadevan, and A. Haldar, “Probability, reliability and statistical method in engineering design,” John Wiley & Sons, 2000.

[14] S. Nannapaneni et al, “Mission-based reliability prediction in component-based systems,” International Journal of Prognostics and Health Management, Vol 7 (1).

[15] N. Mahadevan, A. Dubey, D. Balasubramanian, and G. Karsai, “Deliberative, search-based mitigation strategies for model-based software health management,” Innovations in Systems and Software Engineering, 2013, 9(4):293–318.

[16] L. De Moura, and N. Bjørner, “Z3: An efficient SMT solver,” Tools and Algorithms for the Construction and Analysis of Systems, Springer Berlin Heidelberg, 2008, pp. 337-340.

[17] A. Konak, D. W. Coit, and A. E. Smith, “Multi-objective optimization using genetic algorithms: A tutorial,” Reliability Engineering & System Safety, 2006, 91(9), pp.992-1007.

[18] H. Liu, H. Darabi, P. Banerjee, and J. Liu, “Survey of wireless indoor positioning techniques and systems,” IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 2007, 37(6), pp.1067-1080.