Improving the Dependability of Self-Adaptive Cyber Physical System With Formal Compositional Contract

Peng Zhou, Decheng Zuo, Kunmean Hou, Zhan Zhang, and Jian Dong

***Abstract*—To adapt to the uncertain environment smartly and timely, cyber physical systems (CPSs) have to interact with the physical world in a decentralized but rigorous, organized way. Guaranteeing the timing reliability is key to achieve consensus on the order of distributed events, as well as dependable cooperative decision processing. Based on our hierarchically decentralized compositional self-adaptive framework, we propose a formal compositional reliability-contract-based solution to guarantee the timing reliability of event observation and decision processing in a large-scale, geographically distributed CPS. As the prophetic decision may not fit the local situation well because of the uncertainties, we propose a gradual contract optimization solution to refine the dependability, timeliness, and energy consumption. Following the seven proposed composition schemes, we employ the nondominated sorting genetic algorithm II (NSGA-II) algorithm to optimize arrangement of decision. Moreover, a topology-aware time reserving solution is applied to improve the resilience of processing time and to tolerance timing failures. Both simulation results and real-world testing are introduced to evaluate the efficacy of our proposal. We believe that the formal compositional contract will be a competitive CPS solution to analyze requirements and optimize the self-adaptation decision at runtime.**

***Index Terms*—Compositional contract, cyber physical systems (CPSs), model@run.time, NSGA-II, self-adaptation, timing reliability.**

NOMENCLATURE

Notation

τ_v^{bcet}	Best case execution time of actor *v*.
τ_v^{wcet}	Worst case execution time of actor *v*.
τ_v^{aet}	Actual execution time of actor *v*s.
T^r	Requirement of execution time.
$T(g_{\mathrm{DAAN}})$	Execution time of solution g_{DAAN}.
R^r	Reliability requirement of the decision solution.
p_v^{cp}	Reliability capability of actor v.
p_χ^r	Reliability requirement of activity χ.
λ	Failure rate.
ε	Atomic event.
χ_o	Observation activity (complex event).
χ_a	Action activity (complex event).
f_o	Fitness function of objective o.
DWVCG	Graph view of the topology of CPS.
$\mathrm{DAAN}_{\mathrm{con}}$	Workflow graph of contract.
(v_p, v_s)	Path between v_p and v_s.
$[v_p, v_s)$	Path between v_p and v_s, v_p is included.
$(v_p, v_s]$	Path between v_p and v_s, v_s is included.

Manuscript received October 7, 2018; revised December 19, 2018; accepted July 14, 2019. Date of publication September 17, 2019; date of current version August 31, 2020. This work was supported in part by the National High Technology Development 863 Program of China under Grant 2013AA01A215, in part by the National Natural Science Foundation of China under Grant 61173020, and in part by the State Key Laboratory of High-end Server & Storage Technology Fund under 2014HSSA05. Associate Editor: Y. Dai. *(Corresponding author: Zhan Zhang.)*

P. Zhou is with the Harbin Institute of Technology, Harbin 150001, China, and Université Clermont Auvergne, 63001 Clermont-Ferrand, France (e-mail: zhoupeng@ftcl.hit.edu.cn).

D. Zuo, Z. Zhang, and J. Dong are with the School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China (e-mail: zuodc@hit.edu.cn; zz@ftcl.hit.edu.cn; dan@hit.edu.cn).

K. Hou is with the LIMOS, UMR 6158 CNRS, Université Clermont Auvergne, 63001 Clermont-Ferrand, France (e-mail: kunmean.hou@gmail.com).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TR.2019.2930009

I. INTRODUCTION

CYBER physical system (CPS) has been widely applied in various domains such as smart manufacture [1], smart transportation [2], precision agriculture [3]. It has been regarded as a next revolution of technology that can rival the contribution of the Internet [4]. However, it is still a serious challenge to guarantee the runtime safety and dependability. To respond to the changeable environment timely and dependably, a timing reliable CPS should quickly make proper decisions and process them at the right time at the proper speed. To overcome the long transmission delay and response in time, prophetic decisions are made. However, such decisions may become inopportune for local environment because of various uncertainties, i.e., the changed environment, the random failure of subsystems, which may lead to great loss and even serious damage. It urgently needs a systematic solution to instruct the local subsystems to deal with uncertainties without distorting the global requirements of decisions, and to improve the controllable and predictable of behavior.

CPS should process the decisions in a distributed way, and the activities should be taken at the right time and in the right order. However, reaching consensuses takes time, which increases the risk of deadline missing. Timing information plays a unique role in CPS (for reasoning and coordinating the progress). Current timing-related solutions are mainly based on the assumption of global reference time, where all subsystems have consistent absolute reference time. However, it is full of challenges to

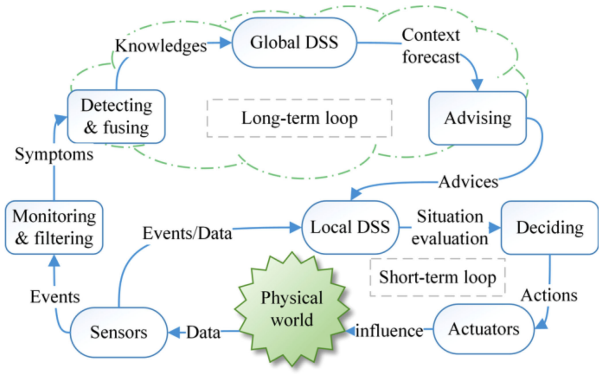


Fig. 1. Multiterm self-adaptation loop of CPS.

maintain the global reference time in a large-scale, geographically distributed CPS because of various interferences such as jitter of oscillators and asymmetrical communication medium [5], [6]. The ambiguous timing information may mislead the CPS into making wrong decisions and disturb the cooperation of subsystems. It is necessary to guarantee the timing reliability of events and the timeliness of activities.

A. Shortages of Current CPS Approaches

Currently, there are two main types of CPS approach, one is centralized approach [7]–[9], another is decentralized approach [i.e., multiagent system (MAS)-based CPS [10], [11]]. Unfortunately, these approaches cannot well meet the requirements of fast, intelligent, and safe reaction. Centralized approaches generally use a central controller to coordinate the decision process. However, the central controller is a single point of failure (SPoF) and a performance bottleneck [12]. With the increasing scale, centralized approaches also suffer serious time synchronization issue [13] and high feedback delay issue. The accumulated timing error rapidly increases with the number of hops between sensors and central decision support system (DSS), which in return limits the scalability of CPS.

On the other hand, the decentralized CPS has high information-sharing overhead, and its subsystems generally are short-sighted because of limited resources. As lack of global arbitrator, the subsystems are easy to imbalance in the resources (i.e., the phenomenon of hot spot and energy hole in wireless sensor network (WSN) [14]). Moreover, the subsystems tend to make self-centered decisions for some reason, i.e., reserving energy and resources for some purposes [15], occupying more resources [16], which undermines the cooperation. What is worse, subsystems may be misled by the inconsistent information and make conflict decisions, and even harms each other's interests [17].

B. Brief Introduction of Our Approach and the Role of Contract

To overcome the shortages of current approaches, we have proposed a hierarchically decentralized compositional self-adaptive framework [18]. In our framework, as shown in Fig. 1, the DSSs make and refine long-term adaptation of advices (i.e., the prophetic decisions) based on the comprehensive (but generally lagging) information. The local DSS can refine the advices based on the newest events, and select the feasible decisions as well as the backup plans. The actuators and sensors cooperatively process the composed decision in a decentralized way, and they can dynamically adjust the decision process flow to improve the timing reliability of the process.

To balance the controllability against the autonomy, we proposed a runtime programmable and executable specification called contact. The formal contact can instruct the subsystems in achieving consensus on the global goal and their duties, as well their cooperators. Meanwhile, contact decouples the procedure of decision making, decision control, and decision processing. CPS can make advices on remote, resource-rich subsystems (i.e., Cloud system), and process the proper decision on resource-limited subsystems (i.e., sensors and actuators). Based on the contract, we proposed a gradual optimization solution to refine the contract at runtime, and to adapt to the changeable environment better.

From the view of the degree of the autonomy, there are two types of contract in our approach. One is the contract between global DSS and local DSSs, which is named advice for distinction. Another is named decision, which is the contract among the local DSS, the actuators, and the sensors. Compared with the decision, the advice is more negotiable, the local DSS can selectively accept the reasonable advices and refine the items and the thresholds in the advices. While the decision is more similar to the concept of command, it could only be slightly adjusted, such as the thresholds of activities and the speed of processing. The contract contains several types of items; in this article, we mainly focus on the requirement of timing reliability. Meanwhile, there are also two types of activity, one is observation and another is action. We use the term activity to refer both them. Generally, the formal process flow of the adaptation loop includes four periods, which are

1) data collection (DC) period;
2) advice refinement (AR) period;
3) actions (AC) period;
4) feedback period (as feedback can be regarded as a DC period in next loop, we will not specially discuss it in this article).

In this article, we mainly focus on AR and AC period.

The remainder of this article is organized as follows. Section II briefly summarizes the formal process flow of CPS and reliability strategies. Section III introduces the formal compositional safety contract and composition schemas. Section IV states the problem of runtime optimization of contract solution and presents our gradual contract optimization solution in detail. In Section V, the simulation results show the positive effectiveness of additional waiting time, and the effects of applying different composition schemas; the testing results on real-world system are also presented in this section. Section VI presents the discussion and further work.

II. Outline of the Formal Process Flow of CPS and Reliability Strategies

Self-adaptation loops, such as the MAPE-K (Monitor, Analyze, Plan, Execute, Knowledge) loop [19], are the common

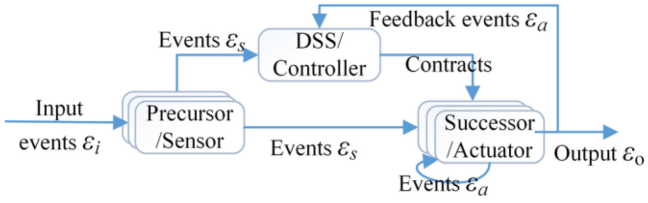


Fig. 2. Generic feedback process flow of self-similar CPS.

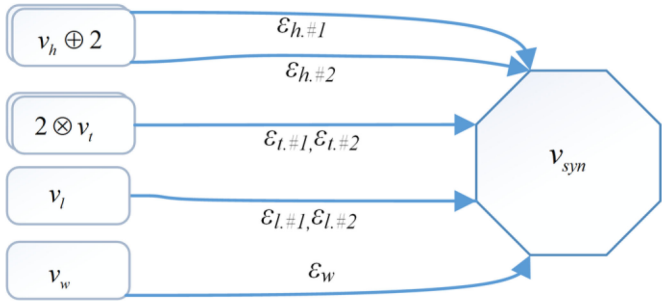


Fig. 3. Generic schemes of redundancy strategies.

solution to build CPS to automatically interact with the changeable environment. A generic CPS constrains three types of primary subsystems, which are sensors, DSS(s), and actuators. The interoperations among these subsystems can be simplified as the multiterm feedback loops, such as the long-term adaptation loop at global system level, the short-term adaptation loop at local system level, and the control loop at subsystem level (i.e., the control loop in actuator). The generic feedback loop schema is shown in Fig. 2.

In the self-adaptation loop, the sensors monitor the input event/signal ε_i and send the event ε_s to both the DSS and the actuators. If ε_i is an emergent event, the successors/actuators could take actions ε_a immediately without waiting the contracts from the DSS and generate output physical event ε_o. As lack of global information, such actions may be short-sighted, the successor should notify both the DSS and neighbor actuators about its newest action ε_a, so that these subsystems can adjust their decision in time. If the ε_s is a nonemergent event, the actuators should wait for the advices from DSS. After receiving the prophetic advices (i.e., the decision with trigger conditions) from the global DSS, the local DSS and the actuators could selectively take advices or refine the advices according to their newest contexts.

A. Formal Process Flow and Specification of Adaptation Loop

From the view of information loop, the abstract adaptation loop can be simplified as a sequence of triggered events. Hence, we formalize the abstract loop with *transition system* (TS) and ignore the detailed decision making and action control in this article.

Definition 1: (TS) $< V, v_0, \sum, T >$ where V is a finite set of actors (i.e., the states set in general TS), v_0 is the initial actor; $\sum$ is a finite set of events, and $\varepsilon_i, \varepsilon_s, \varepsilon_a, \varepsilon_o \in \sum$; and the transition set $T \subseteq V \times \sum \to V$ is a partial deterministic transition function. The sequence of triggered events is the *trajectory* (i.e., $\varepsilon_i \to \varepsilon_s \to \varepsilon_a \to \varepsilon_o$). Generally, the trajectory of all events is a graph. We denote a transition $T = v_p \xrightarrow{\varepsilon} v_s$ by $\varepsilon_{(v_p,v_s)} = (v_p, \varepsilon, v_s)$. For any actor $\forall v \in V$, it should take an action α to process the input events and generate output events, i.e., sensor should preprocess the data. Then the actor v takes a new action to send the output events. We call these actions activities and denote them with the union $A = V \cup T$. To simplify, we use $[v_p, \varepsilon, v_s)$ to denote the union $v_p, \varepsilon_{(v_p,v_s)}$, $(v_p, \varepsilon, v_s] = \varepsilon_{(v_p,v_s)}, v_s$ and $[v_p, \varepsilon, v_s] = v_p, \varepsilon_{(v_p,v_s)}, v_s$.

Note that $v \in V$ is a logical subsystem formalized with a Mealy finite-state machine. For more detailed formal definitions refer to this article [18]. An actor v receives an output event and processes it with several internal transitions, and then sends a new event to other actors, which triggers an external transition. In this article, we focus on the external transition (the event flow of the adaptation loop) and ignore the detailed internal transitions.

For $\forall v \in V$, it contains a specification about its properties $s_v =< \tau^{\text{bcet}}, \tau^{\text{mean}}, \tau^{\text{wcet}}, p^{\text{cp}} >$, where $\tau^{\text{bcet}} < \tau^{\text{mean}} < \tau^{\text{wcet}}$. To the sensors, τ^{bcet}, τ^{wcet}, and τ^{mean} are the best case, the worst case, and the mean observation time, respectively. To the actuators, τ^{bcet}, τ^{wcet}, and τ^{mean} are the execution time at the fastest, the slowest, and the safest speed, respectively. p_v^{cp} is the reliability of actors.

B. Spatial and Temporal Redundancy Strategies

As CPS can directly affect the physical world, any failure may lead to unpredictable damage. It is extremely important to improve the timing and reliability requirements of self-adaptation decisions and feedback loop. The traditional dependability measures include the spatial and temporal redundancy strategies. To CPS, the available redundancy strategies include redundant actors, which are denoted as $n \otimes v$; redundant transmission links, which are denoted as $n \otimes T$; redundant information $n \otimes \varepsilon$; and redoing activities, which are denoted as $v \oplus n$; retransmission, which is denoted as $\varepsilon \oplus n$; where n is the copies of redundant operations.

For example, CPS can apply multiple redundancy strategies, which is shown in Fig. 3. The actor v_h can redo at most one time $(v_h \oplus 2)$ and send the two events ε_h with two different links $(T \oplus 2)$ (if redo operation occurs). Two redundant actors $v_t(2 \otimes v_t)$ are arranged to process subdecision and send events ε_t with one link. One actor v_l is arranged and send the same event ε_l twice with the one link $(\varepsilon_l \oplus 2)$. Actor v_w processes and sends without any redundancy strategy.

III. Formal Compositional Safety Contract

To safely adapt to the dynamic environment, the actors should observe events in time, and take the right actions at the right time at the proper speed in a distributed manner. In other words, CPS needs to guarantee the timing reliability of activities, the timeliness of decisions, and the efficiency of distributed consensus.

Roughly speaking, there are three directions to improve timeliness and timing reliability. One natural solution is improving

the precision of the clock and related clock synchronization algorithms. However, these methods generally are costly [6]. Another solution is applying distributed (detection) algorithms to achieve consensus on the timing order of events before making decisions. However, the convergence rate of distributed consensus algorithms is low [20], which introduces additional delay and increases the risk of deadline missing. The third solution is reducing the number of transmission hops such as using unmanned aerial vehicle (UAV) to collect data in WSN [21]. From the perspective of event observation, our solution belongs to the third type. Compared to using mobile subsystems, our contract-based solution can improve the timing reliability without adding additional high-cost devices.

Contract has been applied in the design period to guarantee consistency between requirements and design or testing for a long time such as design contract [22], timing contract [23], and integration testing contract [24]. However, contract-based runtime solution is rare. Self-adaptive CPS should organize proper subsystems and verify the satisfaction of decision requirements at runtime, which is similar to the procedure of software design. Hence we introduce a formal contract to regulate the cooperation among subsystems. The harder issue is how to guarantee the consistency under the situation that both (the requirements of) decisions and (the properties of) actors change dynamically. Our formal contract can instruct the subsystems to achieve consensus on the adaptation goals and the requirements by gradually refining the decisions and arrangements according to the newest observations. In our approach, the global DSS just specifies the type id of candidates and the global requirements of advices. The local DSS can refine the ranges of these requirements, and even reject the advices. The sensors and actuators can flexibly arrange proper successors at runtime. Meanwhile, these subsystems can smartly apply spatial and temporal redundancy strategies by arranging different number of successors.

A. Formal Contract and the Hierarchically Decentralized Process Flow

As it is hard to accurately predict the processing time of each activity, a safe contract should be resilient in the processing time of single activity but strict in the global processing time of decisions. Our contracts can be regarded as a complex-event trigger graph, the subsystems can adjust its process speed according to the accumulated residual time. The formal definition is shown as follows.

Definition 2: (*generic contract*) Let $\mathrm{Con} = (\chi_o, \chi_a, R, X)$ be a contract, where X is a set of complex activities and $A \subset X$ (*A* is the set of atomic activities defined in **TS**); $\chi_o \in X$ is the observation activity, which is the trigger condition of the contract; $\chi_a \in X$ is the triggered action with the termination conditions; the runtime requirements set $R =< R_\chi, R_{\mathrm{Con}} >$ is the requirement for the contract, where $\forall \chi \in X$, it has a set of constraints $< \mathrm{Tid}_\chi, r_\chi >$, Tid_χ is the type id or the universal unique IDentifier (UUID) of actor (type id is a substring of the UUID, we use type id to represent both of them without explicit statement), and $r_\chi \in R_\chi$ is the runtime decomposable requirements of the χ; R_{Con} is the global requirements of the contract.

The formal basic contract in Backus–Naur Form (BNF) is described as follows:

$$\mathrm{Con} ::= (\chi_o \Rightarrow \chi_a)|(\chi_a)$$

$$\chi_o ::= \varepsilon|(\neg\chi_o)|(\chi_o \wedge \chi_o)|(\chi_o \vee \chi_o)|(\chi_o \Rightarrow \chi_o) \\ |(\chi_o||\chi_o)|(\chi_o, \chi_o)$$

$$\chi_a ::= \varepsilon|(< \mathrm{Tid}, \perp\chi_o >)|(\chi_a \Rightarrow \chi_a)|(\chi_a \Rightarrow \chi_o)|(\chi_a \parallel \chi_a) \\ |(\neg\chi_a)|(\chi_a \wedge \chi_a)|(\chi_a \vee \chi_a)|(\chi_a||\chi_a)|(\chi_a, \chi_a)$$

$$\varepsilon ::= < \mathrm{Tid}, \mathrm{Cdt} >$$

$$\mathrm{Cdt} ::= \mathrm{comparison\_op}, \mathrm{threshold}|\mathrm{Cdt} \vee \mathrm{Cdt}|\mathrm{Cdt} \wedge \mathrm{Cdt}|\neg\mathrm{Cdt}$$

$$\mathrm{comparison\_op} ::= \{>, <, =, != , \leq, \geq\}$$

where Tid is the type id, $\Theta ::= \{\wedge, \vee, \neg\}$ is the Boolean operator, $\Rightarrow$ is the serial operator, and $||$ is the parallel operator.

1) $\chi_{o1} \Rightarrow \chi_{o2}$ represents that χ_{o1} must be observed before χ_{o2}. Note that $\chi_{o1} \Rightarrow \chi_{o2}$ does not imply that $\mathrm{actor}_{\chi_{o1}}$ should directly send its observation to $\mathrm{actor}_{\chi_{o2}}$, the two actors could send their observations to a third actor to check the timing order (we denote the third actor as the synchronization vertex in Section IV).
2) $\chi_1 \Rightarrow \chi_{a2}$ implies that action χ_{a2} can only be taken after the activity χ_1 has been successful processed (actor_{χ_1} should send the output event to $\mathrm{actor}_{\chi_{o2}}$).
3) $(\chi_{o1} \wedge \chi_{o2}) \Rightarrow \chi_3$ represents that both χ_{o1} and χ_{o2} should be observed before activity χ_3, but χ_1 and χ_2 have no timing constraint.
4) We use $\mathrm{Tid}, \perp\chi_o$ to denote that the actor v_{Tid_t} takes an action ε until the termination conditions χ_o has been observed (Note that timeout is also an observation event χ_o).
5) $\chi_{a1} \parallel \chi_{a2}$ represents that actions χ_{a1} and χ_{a2} are taken in parallel. Note that, $(\chi_a \Rightarrow \chi_2) \parallel (\neg\chi_a \Rightarrow \chi_3)$ represents "if χ_a successful then doχ_2, else if χ_a fails then do χ_3", but to simplify optimization, $\chi_a \Rightarrow \chi_2$ and $\neg\chi_a \Rightarrow \chi_3$ can be optimized independently. We first optimize one branch with higher probability, i.e., we can optimize the branches $\chi_1 \Rightarrow \chi_a \Rightarrow \chi_2$ and $\neg\chi_a \Rightarrow \chi_3$ separately, and then combine the results. To simplify, we ignore the contract with if-else conditions in this article.

B. Items of the Contract (the Runtime Specification)

To adapt to the dynamic environment, CPS should organize proper actors in the right order and build a temporary team to process the decision. It is necessary to provide specifications to instruct the actors to take proper activities and coordinate their behavior.

Definition 3: (*The safety items of contract*)

1) $\forall$Con, we have a specification of timing and reliability requirement $R_{\mathrm{con}} =< T^v_{\mathrm{DAAN}}, T^r_{\mathrm{AC}}, T_p, R^r_{\mathrm{AC}} >$, where

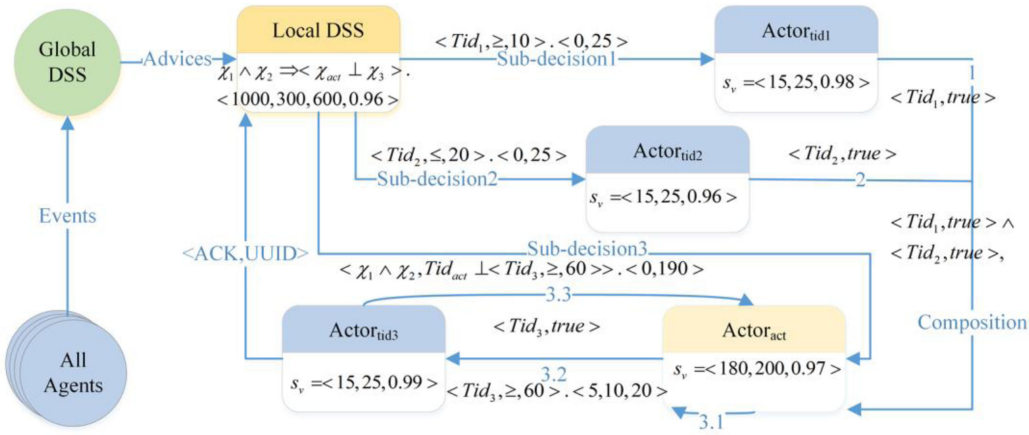


Fig. 4. Hierarchically decentralized process flow.

T^v_{DAAN} is the term of validity of contract, T^r_{AC} is the upper bound of global execution time, T_p is the execution period, and R^r_{AC} is the low bound of global reliability.

2) $\forall \chi \in \mathrm{Con}$, we have a decomposable requirement specification $R_\chi =< \tau^w, \tau^d, p^r_\chi, [, \tau^p] >$, where $\tau^w \in \Re$ is the waiting time before the actor v is triggered, $\tau^d \geq \tau_v^{\mathrm{bcet}} + \tau^w$ is the deadline of the activity that is the duration from the time when the trigger event has been observed to the time when the activity is finished, τ^p is the period of periodic activities. p^r_χ is the low bound of the reliability of each activity. Actors could adjust the process speed [i.e., via dynamic voltage and frequency scaling (DVFS)] to guarantee that the activities are started and finished in the time window $[0, \tau^d - \tau^w]$. If the host actor v_c cannot feed back to its precursor in time, the precursor v_p will try to use the backup solution and rearrange another alternative actor to process the activity. Notice that, τ^w is mainly determined by the precursor and $\tau^w < 0$ implies that the progress has lagged.
3) $\forall \chi \in \mathrm{Con}$, the precursor can apply redundancy strategies, try best to guarantee $1 - \prod (1 - p^{cp}_{v_\chi}) \geq p^r_\chi$ and process the activities in time. Note $p^r_\chi \neq R^r_{\mathrm{AC}}$. The CPS will try best to process the decision as long as there exists a solution that meet T^r_{AC} even though it does not meet R^r_{AC}.

Example 1: An example of abstracted process flow is shown in Fig. 4. The global DSS drafts a contract $\chi_1 \wedge \chi_2, < \chi_{\mathrm{act}} \perp \chi_3 >$ and its corresponding safety specification, $\mathrm{Con}_s =< 1000, 300, 600, 0.98 >$. $\mathrm{Actor}_{\mathrm{tid1}}$, $\mathrm{Actor}_{\mathrm{tid2}}$, $\mathrm{Actor}_{\mathrm{act}}$, and $\mathrm{Actor}_{\mathrm{tid3}}$ are the selected actors to process the contract. $s_v \in S_a$ is the properties of these actors (i.e., s_v of $\mathrm{Actor}_{\mathrm{tid1}}$ is $< \tau^{\mathrm{bcet}}, \tau^{\mathrm{wcet}}, p^{\mathrm{cp}} >=< 15, 25, 0.98 >$). The local DSS decomposes the contract, and sends the subdecisions to related actors. To $\mathrm{Actor}_{\mathrm{tid1}}$, it receives $< \mathrm{Tid}_1, \geq, 10 > . < 0, 25 >$, which represents that $\mathrm{Actor}_{\mathrm{tid1}}$ should take action immediately $(\tau^w = 0)$ and send the output event in less than $\tau^d = 25$. In this case, the output event is $< \chi_1, \mathrm{true} >$, which represents that the observation result is true $[\mathrm{value}(\chi_1) \geq 10]$. According to the subdecisions $\chi_1 \wedge \chi_2, \mathrm{Tid}_{\mathrm{act}} \perp < \mathrm{Tid}_3, \geq 60 >$, $\mathrm{Actor}_{\mathrm{act}}$ waits the observation results of χ_1 and χ_2, then decides to take actions based on the values of final composed trigger $\chi_1 \wedge \chi_2$. In this case, the final result is True, $\mathrm{Actor}_{\mathrm{act}}$ takes actions and terminates until the event $< \mathrm{Tid}_3, \geq 60 >$ is observed.

C. Composition Schemas of Activities

To improve the reliability and to accelerate processing, CPS can arrange different types and different numbers of actors, and compose them with proper schemas. In our approach, we classify seven types of the composition schema, which are shown in Table I. Four of them are normal functional composition schemas, which include logical composition, timing composition, parallel composition, and serial composition. Both logical composition and timing composition have strict timing requirements. The events of logical composition should be generated at almost the same time and the events of timing composition should be generated in the right order. While the activities of parallel composition and serial composition have loose timing constraints, we can transform a parallel composition into a serial composition if the processing time is rich, or transform a serial composition to a parallel composition to save time. Another three types of composition schemas are for redundancy composition, which include the proactive temporal redundancy, the remedial temporal redundancy, and the spatial redundancy. Note that, we can recursively apply the composition schema in Table I, and these rules can also be applied to calculate the decomposed requirements.

In the logical composition for observations, all subobservations can be processed in parallel. If any subobservation fails, the compositional observation fails (indeed, we can accelerate the calculation of the compositional observation at AC period with short-circuit evaluation, i.e., false $\wedge \chi_{o2} \rightarrow$ false, but we cannot achieve it at AR period). In the timing composition, all subactivities are serially processed, and if any subactivity fails the compositional activity fails. In the parallel composition for actions, all subactions should be serially processed, and if any subaction fails the compositional action fails.

TABLE I
TIME AND RELIABILITY CALCULATION RULE OF THE COMPOSITIONAL SCHEMAS

Composition schema	Time (budget)	Expected/real reliability
Logical composition (for observations) $\chi_{o1}\Theta\chi_{o2}, \Theta := \{\wedge, \vee\}$	$\tau(\chi)=\max_{\chi_o\in\chi}(\tau_{\chi_o})$	$R(\chi)=\prod_{\chi_o\in\chi} p^{cp}_{\chi_o}$
Timing composition (for activities) $\chi_p \Rightarrow \chi_s$	$\tau(\chi)=\sum_{\chi\in\chi_p\to\chi_s}(\tau_\chi)$	$R(\chi)=\prod_{\chi} p^{cp}_{\chi}$
Parallel composition (for actions) $\chi_{\alpha_1}\|\chi_{\alpha_2}$	$\tau(\chi)=\max_{\chi_\alpha\in\chi}(\tau_{\chi_\alpha})$	$R(\chi)=\prod_{\chi_\alpha\in\chi} p^{cp}_{\chi_\alpha}$
Serial composition (for actions) $\chi_{\alpha_1};\chi_{\alpha_2}$	$\tau(\chi)=\sum_{\max_{\chi_\alpha\in\chi}} \tau_{\chi_\alpha}$	$R(\chi)=\prod_{\chi_\alpha\in\chi} p^{cp}_{\chi_\alpha}$
Proactive temporal redundancy (for observations) $\chi_o \oplus n$	$\tau(\chi_o \oplus n)=\tau^d+\tau^w-k*\tau_\alpha \quad k=1\cdots n$	$R(\chi_o \oplus n)=1-(1-p^{cp})\wedge n$
Remedial temporal redundancy (for activities) $\chi \oplus n$ (redo)	$\tau(\chi \oplus n)=n*\tau^d-(n-1)*\tau^w$	$R=0$,if χ fails; $R=1$ if χ successes before timeout
Spatial redundancy (for activities) $n \otimes \chi$	$\tau(n\otimes\chi)=\max_{1\le k\le n}(Top-k_{desc}(\tau_{v_k})) \quad k=1\cdots n$	$R(\chi)=\sum_k^n C_n^k (p_\chi^{cp})^k (1-p_\chi^{cp})^{n-k}$

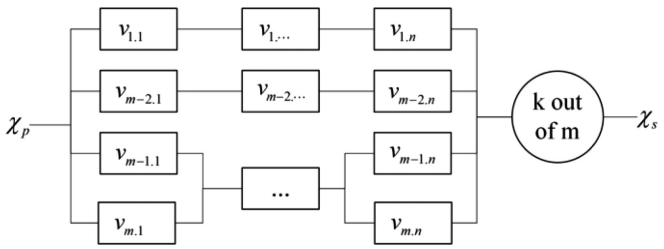


Fig. 5. Generic spatial redundancy schema.

In the proactive temporal redundancy, $\chi_o \oplus n$: actor can process the activity k times and reduce the time window to $[\tau^w - k \times \tau^{\mathrm{rt}}, \tau^d]$, where τ^{rt} is the real execution time at runtime, $k = 1, \ldots, n$ is the first success activity. By the way, it is unsafe to improve the reliability of actions with proactive temporal redundancy because it may lead to over-operation.

In the remedial temporal redundancy for activities $\chi \oplus n$ (redo), if the activity fails unexpectedly in the first execution, actor should take remedial action without waiting, to catch up with the normal progress. It takes the actor τ^d to be aware of the failure (i.e., timeout of feedback). Suppose the activity is successful after n times redo, then the time budget is $\tau^{\mathrm{aet}}(\chi \oplus n)_{\mathrm{rtr}} = \tau^d + (n-1) \times (\tau^d - \tau^w) = n \times \tau^d - (n-1) \times \tau^w$.

In the spatial redundancy for activities $n \otimes \chi$, which is shown in Fig. 5, we can apply three types of strategies.

1) Time first strategy: the precursor arranges n actors to process the activities in parallel and accept the first return result $(k = 1)$.
2) Time-reliability balance strategy: the precursor also arranges n actors to process the activities, and does not process next activities until the first k results are observed.
3) Reliability first strategy: system does not process next activities until all n results are observed $(k = n)$.

The time budget of three strategies can be calculated with the equation $\tau^{\mathrm{aet}}(n \otimes \chi) = \max(\mathrm{Top} - k_{\mathrm{desc}1\le k\le n}(\tau^{\mathrm{aet}}_{\chi\#k}))$. $\mathrm{Top} - k_{\mathrm{desc}1\le k\le n}$ is the top-k ranking in a descending order. The expected reliability can be recursively calculated as a series-parallel system with $R(\chi) = \sum_k^n C_n^k (p_\chi^{\mathrm{cp}})^k (1 - p_\chi^{\mathrm{cp}})^{n-k}$.

IV. OPTIMIZATION OF SAFETY CONTRACT AND SELF-ADAPTIVE DECISION PROCESSING

CPS contains various redundant (heterogeneous) actors, massive possible arrangements are available for one given contract. To overcome the issues of inefficient autonomous self-organization and uncertainties, we should avoid premature optimization and take full advantage of the hierarchically decentralized architecture, and design a systematic solution to minimize with resource budget and balance the workload at different periods.

In a given safety contract, CPS should guarantee the feasibility of contract and try best to improve reliability with fewest resources. In detail, the global DSS should check the rationality of T^r and generate some basic advices. The local DSS checks the fitness of advices, then optimizes the arrangement based on these advices. At the AC period, the precursor actor takes its own activities, and arranges the proper (number of) successor actors according to the progress and qualities of successors. Meanwhile, all involved actors should update the temporal information and maintenance of the correctness of contract.

A. Formal Problem Statement

1) Presupposition: The randomly deployed actors, especially the sensor and actuators, are connected with the (wireless) mesh network. Suppose that the network is a connected graph with no routing cycles, the topology can be regarded as a doubly weighted vertex-colored graph $\mathrm{DWVCG} = (V(c), E, W_v, W_e)$, where the vertex represents the actors and the edge represents the connection between actors, the color represents the role of actors. For each $v \in V$, we have a weight set $< \tau^{\mathrm{bcet}}, \tau^{\mathrm{mean}}, \tau^{\mathrm{wcet}}, p^{\mathrm{cp}} > \in W_v$; and for each $e \in E$, we have weight $< \mathrm{hop}^{\mathrm{mean}}, p^{\mathrm{mean}}_{\mathrm{hop}} > \in W_e$, where $\mathrm{hop}^{\mathrm{mean}}$ is the mean number of transmission based on the recent statistics, $p^{\mathrm{mean}}_{\mathrm{hop}}$ is the mean successful transmission rate of each hop.

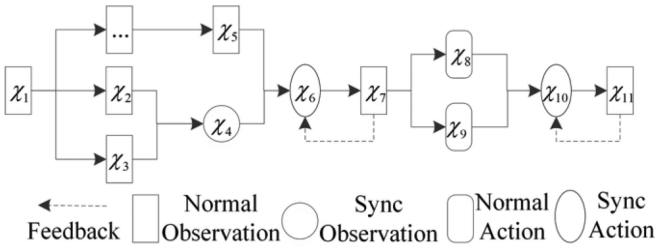


Fig. 6. Abstract decision process flow at AC period.

We use (v_p, v_s) to denote the all available path between vertex $v_p \in V$ and $v_s \in V$, where $v_p \neq v_s$. And $\min(v_p, v_s)$ is the shortest path between v_p and v_s.

Definition 4. (redundant actors): $\forall v \in V$, it has a color to represent the type id of the actors, where color(v) is a unique identification and color$(v) \in N$. For $\forall v_i, v_j \in V$, the two actors are redundant iff color$(v_i) =$ color(v_j).

Definition 5. (redundant path): Two paths (v_m, v_i) and (v_n, v_j) are redundant $(v_m, v_i) \simeq (v_n, v_j)$, iff color$(v_i) =$ color(v_j) and color$(v_m) =$ color(v_n) and color$(v_i) \neq$ color(v_m) where $v_i, v_j, v_m, v_n \in V$ and $v_i \neq v_j \vee v_m \neq v_n$.

Meanwhile, the fully expanded TS (the decision process flow) can be regarded as a special directed acyclic activity network DAAN $= (V_\chi, E_\chi, R_{\text{Con}})$ with synchronous operation, where R_{Con} is the requirement of the contract. A typical DAAN is the decision process flow without the edge of feedback, which is shown in Fig. 6, where the dashed-line arrows is the edge of feedback, and DAAN is the figure without the dashed-line arrows. Take the subcontract $(\chi_6 \perp \chi_7) \Rightarrow (\chi_8 || \chi_9)$ as example, χ_7 is the termination condition of χ_6, actions χ_8 and χ_9 are triggered when the action χ_6 is finished. To save time, the actor of χ_7 could send the termination event to χ_6, as well as the trigger event to χ_8 and χ_9.

As shown in Fig. 6, there are following four types of vertexes:

1) The normal observation vertexes that are distributed (physical and cyber) events observers.
2) The synchronous observation vertex that is used to control the timing error caused by a clock synchronization error, and to guarantee the causal relationship of events, the common application of such vertexes are multidata fusion/checking, events serialization, and reasoning.
3) The normal action vertexes.
4) The synchronization action vertex does not take actions until all inputs are received.

Hence, the problem of contract refinement can be formalized as follows. For a given deployment graph with redundant actors DWVCG $= (V(c), E, W_v, W_e)$, and a safety contract DAAN$_{\text{con}} = (V_\chi, E_\chi, R_{\text{con}})$, CPS should first prove that there exists at least one subgraph $g_{\text{DAAN}} = (V, E)$ of DWVCG that is isomorphic to the DAAN$_{\text{con}}$, where color$(v) =$ typeid(χ_{con}) and g_{DAAN} should meet the R_{con}. The CPS searches the (sub)optimal solution that tradeoffs multiple objectives under multiple constraints.

2) Formal Optimization Problem Statement: The optimal solution is the one with minimum processing time, highest reliability (lowest failure rate), and the lowest utilization ratio of resources (especially the global energy consumption). The objectives and constraints are shown in (1), where g_{DAAN} is the potential solution. $T(g_{\text{DAAN}})$, $T(g_{\text{DAAN}}^{\tau^{\text{bcet}}})$, and $T(g_{\text{DAAN}}^{\tau^{\text{wcet}}})$ are the mean execution time, BCET, and WCET. The calculation rules of $R(g_{\text{DAAN}})$ and $T(g_{\text{DAAN}})$ are listed in Table I. ft is the objective function of execution time. fr is the failure rate function. fe is an optimistic estimation of the global energy consumption, which includes two parts: 1) $w_e \times \sum_{e \in g_{\text{DAAN}}} \text{hop}(e)$ is the energy consumption of transmission, where w_e is the mean energy consumption of one hop transmission. 2) $\sum_{v \in g_{\text{DAAN}}} (c_v^p \times \tau_v^{\text{mean}})$ is about the energy consumption of activities, where c_v^p is the recent power of actor v (i.e., the recent energy consumption rate $\Delta e / \Delta t$), which aims to improve the fairness and to avoid the energy holes.

$$
\begin{aligned}
&\text{Min.} ft = T(g_{\text{DAAN}}) \\
&\quad fr = 1 - R(g_{\text{DAAN}}), \\
&\quad fe = w_e \times \sum_{e \in g_{\text{DAAN}}} \text{hop}(e) + \sum_{v \in g_{\text{DAAN}}} (c_v^p \times \tau_v^{\text{mean}}) \\
&\text{S.t.} \quad T\left(g_{\text{DAAN}}^{\tau^{\text{bcet}}}\right) \leq T_{\text{AC}}^r, \qquad (C1) \\
&\quad \forall (v_{\chi_p}, v_{\chi_s}), \tau^w < \tau^\theta_{(v_{\chi_p}, v_{\chi_s})}, \qquad (C2) \\
&\quad \forall \chi \in \text{DAAN}, 1 - \prod (1 - p_{v_\chi}^{\text{cp}}) \geq p_\chi^r, \quad (C3) \\
&\quad \forall \chi_{o1} \Rightarrow \chi_{o2} \in E_{\text{DAAN}}, \text{hops}(e) < \theta_1. \quad (C4)
\end{aligned} \tag{1}
$$

To avoid missing the possible solutions (because actors can adjust its process speed at runtime), a weak constraint of execution time bound $C1$ is defined. $C2$ is designed to guarantee the timeliness of events, which implies that the subsequent event χ_s should be processed in less than the time $\tau^\theta_{(v_{\chi_p}, v_{\chi_s})}$ after χ_p is observed. The constraint $C3$ is the reliability requirement of each activity. Considering the error of clock synchronization, the constraint $C4$ is defined to guarantee the timing reliability of $\chi_{o1} \Rightarrow \chi_{o2}$ by limiting the number of transmission hops. As mentioned in the Definition 2, we can choose a third actor v_t to check the timing order if $\text{hops}(v_{\chi_{o1}}, v_{\chi_{o2}}) \geq \theta_1$. Suppose that $v_{\chi_{o1}} || v_{\chi_{o2}} \Rightarrow v_t \Rightarrow v_{\chi_3}$ is the renewed solution of $\chi_{o1} \Rightarrow \chi_{o2} \Rightarrow \chi_3$ with a synchronization actor v_t, where $\text{hops}(v_{\chi_{o1}}, v_t) < \theta_1$ and $\text{hops}(v_{\chi_{o2}}, v_t) < \theta_1$. v_t can be a vertex with any color because it does not need any specified activity. To simplify, we refine v_t after the optimization of (1) and select a vertex with shortest total distance to all related vertexes (i.e., the discrete Fermat point problem, $\min(\text{hop}(v_t, v_{\chi_{\text{syn}}}) + \sum_{(\chi, \chi_{\text{syn}}) \in \text{DAAN}} \text{hop}(v_\chi, v_t))$). And we will not specifically discuss the refinement of v_t later.

Obviously, searching the optimal solution is a multiobjective combinatorial optimization (MOCO) problem, which is known as NP-hard problem. The common solutions are approximation metaheuristics algorithms [25]. However, current approximation algorithms are dedicated to the problem with static Pareto set. In the real-world CPS, the Pareto front is changeable, thus some solutions may become invalid because of various uncertainties,

such as failures, and resource competition. To deal with the uncertainties, we propose a gradual optimization solution to solve the (1). CPS can refine the process flow of contract according to the newest information at each period.

B. Checking the Rationality of Contract on the Global DSS

The global DSS should check if there exists a BCET solution $g^{\mathrm{bcet}}_{\mathrm{DAAN}}$, which meets $T(g^{\mathrm{bcet}}_{\mathrm{DAAN}}) < T^{r}_{\mathrm{AC}}$. $g^{\mathrm{bcet}}_{\mathrm{DAAN}}$ is a solution without redundancy composition and serial composition (all activities are processed in parallel at their fastest speed). The global DSS will send the contract and the $g^{\mathrm{bcet}}_{\mathrm{DAAN}}$ to the local DSS if the contract is feasible. Otherwise, the global DSS gives up the contract.

Step 1: Calculate all the shortest paths and build indexes of all redundant paths.

As the global DSS and local DSS frequently make and check various contracts through their whole lifecycle, to accelerate the rationality checking, we can first use Johnson's algorithm [26] to calculate all the shortest paths of DWVCG and build the index $\{\mathrm{color}(v_p v_s), \{(v_{p.e}, v_{s.s}), \mathrm{hops}(v_{p.e}, v_{s.s})\}\}$ in advance, where $\mathrm{color}(v_p v_s)$ is the connection operation $\mathrm{color}(v_p v_s) = \mathrm{tid}(v_p)\mathrm{tid}(v_s)$, and $\{(v_{p.e}, v_{s.s}), \mathrm{hops}(v_{p.e}, v_{s.s})\}$ is set of all redundant paths and their hops.

Step 2: Construct all isomorphic graphs of $\mathrm{DAAN}_{\mathrm{con}}$.

First, we transform all serial compositions to parallel compositions, then construct all available subgraphs $\{S_{\mathrm{DAAN}}\}$ in DWVCG which are isomorphic to $\mathrm{DAAN}_{\mathrm{con}}$. $\{S_{\mathrm{DAAN}}\}$ is built with all redundant vertexes and the corresponding edges between adjacent vertexes. $\{S_{\mathrm{DAAN}}\} = (V_{\{S_{\mathrm{DAAN}}\}}, E_{\{S_{\mathrm{DAAN}}\}})$ is a new DWVCG built with all S_{DAAN}, where $V_{\{S_{\mathrm{DAAN}}\}} = \cup\{v\}$, $\mathrm{color}(\{v\}) = \mathrm{typeid}(v_\chi)$ and $E_{\{S_{\mathrm{DAAN}}\}} = \cup\{(v_{p.e}, v_{s.s})^*\}_{\mathrm{rpath}}$, $\mathrm{color}(v_p v_s) = \mathrm{typeid}(v_{\chi.p} v_{\chi.s})$.

Step 3: Search the single-objective optimum solution.

In any path $\mathrm{path}_{i,j}$ of $\mathrm{DAAN}_{\mathrm{con}}$ (as shown in Fig. 12), we have a subgraph subg_k built with the redundant vertexes. By employing Dijkstra's algorithm, we search the minimum weighted path $\min(\mathrm{path}_{i,j})$ from the subgraph subg_k. As the synchronization vertexes cannot start until that all precursor branches are finished, we use the maximum execution time of all branches as the weight of synchronization vertexes. The best solution $g^{\mathrm{bcet}}_{\mathrm{DAAN}}$ is the graph with minimum weighted synchronization vertexes of all redundant vertexes. By the way, we can also get different g_{DAAN} by applying different weights of the vertexes (such as τ^{mean}_v and p^{cp}_v). For the reliability, we select the maximum weighted path. Moreover, we can store the sum weight of each redundant subgraph in the properties of synchronization vertexes to accelerate the next step of optimization. The detailed Algorithm 1 is shown in the Appendix.

C. Contract Evaluation and Refinement With Optimized NSGA-II on the Local DSS

To avoid meaningless optimization, the local DSS should check the fitness of contracts based on the recent information, predict the trend, reject the invalid contracts, and chose the correct optimization strategies.

1) Preprocessing and Situation Evaluation: The prophetic contracts $\mathrm{DAAN}_{\mathrm{con}}$ may be unsuitable to every local subsystem for the various reasons such as the uncertain hypothesis and the different environment. The local DSS first checks that there exists at least one candidate actor for each vertex in $\mathrm{DAAN}_{\mathrm{con}}$, [formally, $\forall v_{\mathrm{DAAN}} \in V_{\mathrm{DAAN}_{\mathrm{con}}}$, $\exists v_{\mathrm{DWVCG}} \in V_{\mathrm{DWVCG}}$ that $\mathrm{color}(v_{\mathrm{DWVCG}}) = \mathrm{color}(v_{\mathrm{DAAN}})$] and these candidates are connected. Meanwhile, the local DSS updates all atomic observations ε with the last information and sets the value with <true or false or unknown> (i.e., Fill $<$ Tid, comparison_op, threshold $>$, as seen in the example in Fig. 4), then calculates the Boolean value of the subexpression χ_o with short-circuit technology [27], and takes the action ε_a based on the value of its successor χ_o.

As optimization takes time, the local DSS should evaluate the risk of missing the trigger events, especially the physical events. The local DSS estimates the rates of changes based on recent information, and evaluates the remaining time with the domain knowledge or time series prediction methods. If there is no enough remaining time to run one round of nondominated sorting genetic algorithm II (NSGA-II) (in our approach, the threshold is 100 s), the local DSS can, respectively, use τ^{bcet}_v, τ^{mean}_v, $\tau^{\mathrm{mean}}_v / p^{\mathrm{cp}}_v$, and p^{cp}_v as the weight to generate the solutions and select the best one, where the selection rule is the same with the NSGA-II method.

2) Searching the Optimum Solutions With NSGA-II: For the time-rich contracts, we employ the NSGA-II [28] methods to search the optimized solution. NSGA-II is a well-known genetic algorithm for MOCO problem. It has been employed in architecture-based optimization problems and shows remarkable results [29], [30]. It uses a nondominated sorting procedure, ranks the solutions with several heuristic functions, and tries to iteratively evolve the populations on several dominant domains. The local DSS will run at most 20 rounds of NSGA-II and select the best solution as the final template solution.

A heuristic function $f(g_{\mathrm{DAAN}}(v)) = T(\mathrm{subg}_{[v_{\mathrm{start}} \to v)}) + \tau_v + T(\mathrm{subg}^{\mathrm{bcet}}_{[v_{\mathrm{end}} \to v)}) \le T^{r}_{\mathrm{AC}}$ is applied to find the Pareto set of solutions, where $T(\mathrm{subg}_{[v_{\mathrm{start}} \to v)})$ is the cost of searched subgraph considering the spatial and temporal redundancy; and $T(\mathrm{subg}^{\mathrm{bcet}}_{[v_{\mathrm{end}} \to v)})$ is the BCET of the subgraph from the end vertex to current vertex v. We can get $T(\mathrm{subg}^{\mathrm{bcet}}_{[v_{\mathrm{end}} \to v)})$ with Algorithm 1, [actually, we can accelerate the calculation with $T(\mathrm{subg}^{\mathrm{bcet}}_{[v_{\mathrm{end}} \to v)}) = T^{\mathrm{bcet}}_{[v_{\mathrm{end}} \to v_{\mathrm{syn}})} + T^{\mathrm{bcet}}_{[v_{\mathrm{syn}}, v)}$, where $T^{\mathrm{bcet}}_{[v_{\mathrm{end}} \to v_{\mathrm{syn}})}$ is the shortest path recorded in the properties of $v_{\mathrm{syn}} \in V_{\mathrm{syn}}$, (i.e., $T^{\mathrm{bcet}}_{[v_{\mathrm{end}} \to v_{\mathrm{syn}})}$ is the $bk.\mathrm{branchWeight}[][]$ of the Algorithm 1 searching from the end vertex). We can use $\mathrm{branchWeight}[][]$ and $bk.\mathrm{branchWeight}[][]$ of v_{syn} to accelerate checking the fitness of solutions in NSGA-II].

In our approach, the seed population is $\min(100, \prod_{\chi \in E_{\mathrm{DAAN}}} |\{v_\chi\}|)$. The size of 100 has been used in many research studies [28]–[30], $\prod_{\chi \in E_{\mathrm{DAAN}}} |\{v_\chi\}|$ is for the contract with few candidate set. The key operations of NSGA-II are as follows.

a) Operation 1. Fitness functions: The evaluation involves four dimensions, which are time (f_{time}), reliability $(f_{\mathrm{reliability}})$, the time cost if the critical path fails

($f_{\text{risk}}^{\text{path}_{\text{cp}}}$), and the energy budget (f_{energy}). The calculation rules of $T(g_{\text{DAAN}})$ and $R(g_{\text{DAAN}})$ have been shown in Table I.

$$f_{\text{time}} = \frac{1}{1+e^{T_{\text{AC}}^r - T(g_{\text{DAAN}})}} \qquad T(g_{\text{DAAN}}^{\tau^{\text{bcet}}}) < T_{\text{AC}}^r$$

$$f_{\text{reliability}} = \frac{1}{1+e^{\frac{2(R(g_{\text{DAAN}})-R_{\text{AC}}^r)}{(1-R_{\text{AC}}^r)\times R(g_{\text{DAAN}})}}}, \qquad T(g_{\text{DAAN}}^{\tau^{\text{bcet}}}) < T_{\text{AC}}^r$$

$$f_{\text{risk}}^{\text{path}_{\text{cp}}} = \sum_{v\in \text{path}_{cp}} \left(\tau_v^{\text{mean}} \times \frac{1-p_v^{\text{cp}}}{p_v^{cp}}\right) + \sum_{e\in \text{path}_{\text{cp}}} \left(\frac{\text{hops(e)}\times(1-\text{p}_{\text{hop}}^{\text{mean}})}{p_{\text{hop}}^{\text{mean}}} \times \tau_{\text{hop}}^{\text{mean}}\right) \times T(g_{\text{DAAN}}^{\tau^{\text{bcet}}}) < T_{\text{AC}}^r$$

$$f_{\text{energy}} = w_e \times \sum_{e\in g_{\text{DAAN}}} \text{hops}(e) + \sum_{v\in g_{\text{DAAN}}} (c_v \times \tau_v^{\text{mean}}) + \sum_{\chi_a \perp \chi_o} \alpha \times (w_e \times \text{hops}(v_{\chi_o}, v_{\chi_a}) + c_{v_{\chi_o}} \times \tau_{v_{\chi a}}^{\text{mean}}), \qquad T(g_{\text{DAAN}}^{\tau^{\text{bcet}}}) < T_{\text{AC}}^r$$

f_{time} is the execution time of solution without feedback loop. $f_{\text{reliability}}$ is the reliability of solution. $f_{\text{risk}}^{\text{path}_{\text{cp}}}$ is an estimation of the time cost of failures, we use value of the critical path as an estimation because it is more flexible for the actors in noncritical path to recover and $p_v^{\text{cp}} \approx 1$ in most cases. f_{energy} is the fitness function of energy budget. To avoid over-operation, observation vertex v_{χ_o} in feedback loop will continuously monitor the value and send the events to the action vertex v_{χ_a} when the action is almost finished. And α is the expected sampling number of v_{χ_o}. In our approach, we use the exponential decay function to control the sampling period, the number of sampling is $\lfloor \log_2^{\tau_{v\chi a}^{\text{mean}}/(\tau_{v\chi a}^{\text{mean}} + \text{hops}(v_{\chi a}, v_{\chi o})\times \tau_{\text{hop}}^{\text{mean}})} \rfloor$.

b) Operation 2. Crossover operation: The parents of crossover operation are randomly selected, and the crossover probability is 0.9, which is widely used in NSGA-II applications [28]–[30]. We apply uniform crossover operation to generate child solutions. As the valid parents may generate an invalid child, we should guarantee that all child solutions meet the constraint $T(g_{\text{DAAN}}^{\tau^{\text{bcet}}}) < T_{\text{AC}}^r$. To take full advantage of value of $\text{branchWeight}[][]$, we chose the branch or the subgraph between two adjacent synchronization vertexes as the unit of crossover operation.

In our approach, we also apply another crossover operation to generate the polyploid solution. These polyploid solutions have redundant branches inherited from parents (the synchronization vertexes are not included in this crossover). According to the bucket effect, we chose the weakest branch of the child solution and then strengthen it with the branch from another parent. In the polyploid solution, the redundant branches will be passed on to the next generation.

c) Operation 3. Mutation operation: We apply four types of mutation operations at the vertex level in group of branches. The synchronization vertexes are processed as a part of the critical path. The first type of mutation randomly replaces the selected vertexes with their redundant vertexes. Followed the common setting [28], we use the reciprocal of the number of decision variables as the mutation rate. To each mutation, the value is $p_m = 1/|\{v\}|$, where $|\{v\}|$ is the number of available redundant vertexes of the branch.

As a single actor generally cannot meet the requirement P_χ^r, we apply the second type of mutation at the beginning of each round of NSGA-II to quickly increase the reliability. For each son solution g_{DAAN}, if $\exists\chi$ cannot meet $(1-\prod_{v_\chi}(1-p_{v\chi}^{\text{cp}})) < P_\chi^r$, we randomly select 10% weakest χ and add a redundant vertex v_χ with $\min(\text{hops}(v_p, v_\chi) + \text{hops}(v_\chi, v_s))$ (the nearest one to the precursor and the successor).

To the vertexes of a polyploid solution, we apply the third type of mutation, which includes a set of selective operations that can be classified into two types: 1) adding a randomly selected redundant vertex for the weakest activity and 2) randomly removing a redundant vertex whose corresponding activity χ has the highest reliability. The detailed conditions of operation are shown in Table II. In noncritical path, we randomly select one of the redundant vertexes (note that the applied vertexes may be selected to build a temporal redundancy solution if the execution time of the new branch is less than the critical path). In critical path, we randomly select one from the set of unselected redundant vertexes.

In the solutions with parallel composition or serial composition, we apply the four type of mutation. If $T(g_{\text{DAAN}}) > T_{\text{AC}}^r$, we transform the serial composition to parallel composition based on the rule $\min(|\tau_{\text{sub\_path}i} - \tau_{\text{sub\_path}j}|)$ to balance the processing time of new subpaths, where sub_path_i and sub_path_j are the new subpaths in the new parallel composition. The number of new subpaths depends on the value of $\min(T_{\text{new}}(g_{\text{DAAN}}) - T_{\text{AC}}^r)$ and $T_{\text{new}}(g_{\text{DAAN}}) < T_{\text{AC}}^r$. If $T(g_{\text{DAAN}}) < T_{\text{AC}}^r$, we transform the parallel composition to serial composition if merging actors can reduce the number of transmission hops. Because the parallel solution has the same actors with the serial solution, but more edges (i.e., it generates more packages). The best solution of subcontract is the one with minimum number of global transmission hops.

The core idea of second mutation is that the failure rate λ of real system is generally less than 10^{-4}. As the reliability of n redundant actors is $R = 1-(1-p)^n$, we can effectively save the energy by removing some redundant vertexes without significant reduction in the reliability.

d) Operation 4. Selection operation: In our approach, we maintain 100 populations of nondominated solutions in each iteration.

e) Operation 5. Stopping condition: The NSGA-II is terminated after 250 generations. And the local DSS runs the NSGA-II for at most 20 rounds if time permits.

f) Operation 6. Solution selection: In every round, the NSGA-II algorithm generates a nondominated candidate solution set. The global candidate set is the union of the solutions of all rounds. All fitness values are firstly normalized with (2),

TABLE II
CONDITIONS AND THE SELECTION RULES OF POLYPOID MUTATION

Operation	Condition of mutation vertex	The preference rules of selection
Add a redundant vertex v_{χ_t} for the weakest activity χ_t; connect v_t with the closest precursor and successor	$\chi_t \in \chi_o$, $v_t \notin V_{syn}$ and $(1-\prod_{\chi_t}(1-p_v^{cp}))=\min_{\chi_t,\chi\in ebrach}(p_\chi)$	Select the redundant vertex with the $\min(hops(v_p,v_t)+hops(v_t,v_s))$ (the nearest one)
	$v_{\chi_t} \in V_{syn}$ and $(1-\prod_{\chi_t}(1-p_v^{cp}))=\min_{\chi_t,\chi\in ebrach}(p_\chi)$	Select the redundant vertex with $\max(p_v^{cp}/(c_v\times\tau_v^{mean}+w_e\times\Delta hops))$
	$\chi_t \in \chi_a$ and $(1-\prod_{\chi_t}(1-p_v^{cp}))=\min_{\chi_t,\chi\in ebrach}(p_\chi)$	Randomly select one from the top-10% highest reliable vertex
Remove a redundant vertex v_{χ_t} for the most reliable activity χ_t; connect the breakpoints with the closest precursor and successor.	$\lvert v_{\chi_t}\rvert>1$, $\chi_t \in \chi_o$, $v_{\chi_t} \notin V_{syn}$, $(1-\prod_{\chi_t}(1-p_v^{cp}))=\min_{\chi_t,\chi\in ebrach}(p_\chi)$	Remove the one with the $\min(p_v^{cp}/(\tau_v^{mean}+\Delta(hops*\tau_{hop}^{mean})))$
	$\lvert v_{\chi_t}\rvert>1$, ($v_{\chi_t}\in V_{syn}$ or $\chi_t \in \chi_a$), $(1-\prod_{\chi_t}(1-p_v^{cp}))=\min_{\chi_t,\chi\in ebrach}(p_\chi)$	Remove the one with the $\min(p_v^{cp}/(\tau_v^{mean}+\Delta(hops*\tau_{hop}^{mean})))$

Note: Δhops is the difference of hops between mutation and original solution.

where $f_{\max}$ and $f_{\min}$ are the maximum and minimum fitness value of all candidate solutions. Then we can use $\sum w_i \times \alpha_i^2$ to sort the solutions and select the top g^*_{DAAN} as the template, where α_i is the normalized value of fitness and w_i is the corresponding weight (currently, we set $w_i = 1$ to simplify).

$$\alpha = \frac{f - f_{\min}}{f_{\max} - f_{\min}}. \tag{2}$$

D. Refining the Runtime Specification and Decomposing the Solution

The synchronization vertexes have to wait for the events from their all precursors. The long waiting time may lead to violation of the timeliness constraint $C1$ [as seen in (1)], and long waiting time also increases the failure risk of synchronization vertexes. Hence, we refine the waiting time τ^w and the deadline τ^d to adjust the progress of activities and to improve the timing reliability and the predictability of solution.

The local DSS recursively processes the subgraph of g_{DAAN}, calculates the time difference $\Delta\tau = T^{\mathrm{new}}_{\mathrm{path_{cp}}} - T^{\mathrm{mean}}_{\mathrm{path_{ncp}}}$, where $T^{\mathrm{new}}_{\mathrm{path_{cp}}}$ is the renewed execution time (the critical path of an induced subgraph may be a noncritical path of its parent graph), $T^{\mathrm{mean}}_{\mathrm{path_{ncp}}}$ is the execution time of noncritical path of the processing subgraph. The local DSS will adjust the recommended deadline τ^d of activities with (3). Under the premise of finishing the activities during the time τ_v^d, the sensors and actuators could autonomously set τ^w and the process speed according to their task set.

$$\tau_v^d = \tau_v^{\mathrm{mean}} + \tau_v^w = \tau_v^{\mathrm{mean}} + \Delta\tau \times \tau_v^{\mathrm{mean}} \Big/ \sum_{u\in \mathrm{path_{ncp}}} \tau_u^{\mathrm{mean}}. \tag{3}$$

Since then, the local DSS has generated the optimum solution. Afterwards, it decomposes the decision and sends the subdecision to the candidates. The generic form of subdecision is $\Theta{<}\mathrm{Tid}_{\chi_o},\mathrm{hops}{>}{\Rightarrow}\mathrm{uuid}_{\chi_i}.{<}\tau^d,\tau^p,p^r{>}{\Rightarrow}\{{<}\mathrm{uuid}_{\chi_{i+1}},\tau^{\mathrm{mean}}_{v_{\chi_{i+1}}}{>}{\Rightarrow}\mathrm{uuid}_{\chi_{i+2}}\}$, where $\Theta < \mathrm{Tid}_{\chi_o}, \mathrm{hops} >$ is the precursor event, hops is the expected number of transmission hops (i.e., the shortest path) and $\Theta = \{\wedge, \vee, \neg\}$, $\chi_{\mathrm{op}} := \varepsilon$, or $\chi_{\mathrm{op}} := \varepsilon \perp < \chi_o >$; $< \tau^d, \tau^p, p_v^{\mathrm{cp}}, p^r >$ is the requirement specification of activity χ_{op}. $\{< \mathrm{uuid}_{\chi_{i+1}}, \tau^{\mathrm{mean}}_{v_{\chi_{i+1}}} > \Rightarrow \mathrm{uuid}_{\chi_{i+2}}\}$ is the subsequent to decision, where $\mathrm{uuid}_{\chi_{i+1}}$ is UUID of the recommended successor, $\tau^{\mathrm{mean}}_{v_{\chi_{i+1}}}$ is the expected execution time of χ_{i+1}, and $\mathrm{uuid}_{\chi_{i+2}}$ is the UUID of successor activity χ_{i+1}. By the way, to save energy the local DSS can send the subdecision to v_i at the time $\sum_{k=1}^{i-1} \tau^d_{\chi_k} - \tau_{(\mathrm{ldss},v_i)}$ or based on the feedback information from the previous actor.

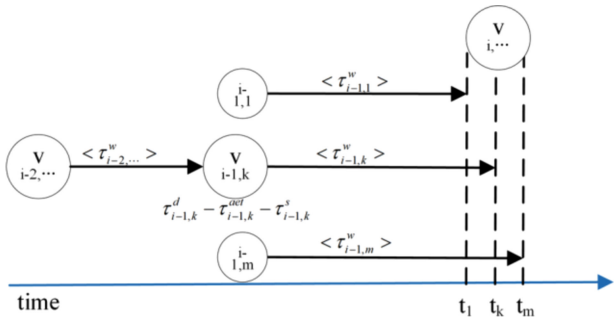


Fig. 7. Refining the τ^w for actor v_i.

E. Strategy of Runtime Refinement at Decision Processing Period

Without loss of generality, we assume that CPS has successfully finished $i-1$ steps of the decision, and v_i has $m \geq 1$ precursors (including the redundant actors) which are denoted as $v_{i-1,1}$ to $v_{i-1,m}$. As shown in Fig. 7, to the actor v_i, the kth output event from $v_{i-1,k}$ is $< \mathrm{Tid}_{i-1}, \mathrm{value} > . < \tau^w_{i-1,k} >$, where $\tau^w_{i-1,k} = \tau^w_{i-2,\ldots} + \tau^d_{i-1,k} - \tau^{\mathrm{aet}}_{i-1,k} - \tau^s_{i-1,k}$ is the accumulated remaining time budget of $v_{i-1,k}$ after processing, $\tau^w_{i-1,k}$ is the waiting time for $v_{i-1,k}$ before starting to process the activity,

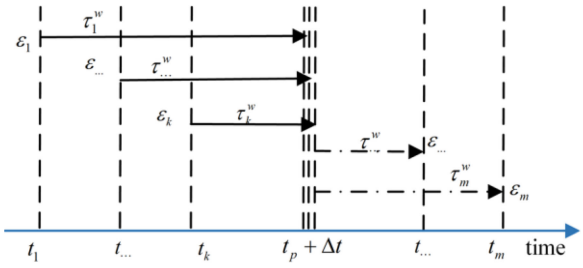


Fig. 8. Arrival timestamp and τ^w of events.

and $\tau^{\text{aet}}_{i-1,k}$ is the real processing time. $v_{i,j}$ waits for input $\Theta < \text{Tid}_{\chi_o}, \text{hops} >$ and updates its waiting time with $\tau^w_i = \tau^w_i + \sum_{k=1}^{m} (\tau^w_{i-1,k} - \tau^{\text{aet}}_{(v_{i-1,k},v_i)} + \text{hops} \times \tau^{\text{mean}}_{\text{hop}} + t_m - t_k)/m$, where t_m and t_k is the timestamp based on a local clock when the event is received. Since then, $v_{i,j}$ has the full specification $< \tau^w, \tau^d, \tau^p, p^r >$ of the activity.

The actor $v_{i,j}$ waits and composes the events from the precursors. When $\Theta_{\chi_p} < \text{Tid}_{\chi_o}, \text{value} >$ can be calculated with the short-circuit technology [27], it sends the feedback events to its precursors. And if the trigger condition is true, $v_{i,j}$ could adjust its speed (or sleep) and finish the activity within the time $\tau^w_{i-1,j} + \tau^d_{i,j}$.

As the future environment is full of uncertainties, the actors should cooperate closely and dynamically apply the strategies to recover from failures as well as to save energy. These strategies are as follows.

1) Saving Energy: If $\tau^w_{i,j} + \tau^d_{i,j} - \tau^{\text{mean}}_{i,j} \geq \tau^{\text{mean}}_{i+1,k}$ and $v_{i+1,k} \notin V_{\text{syn}}$, $v_{i,j}$ could start activity without waiting and reduce redundant successors by half to save global energy. If $v_{i+1,k}$ has no redundant actors, to avoid the violation of timeliness, $v_{i,j}$ could first sleep at most $\tau^s_{v_{i,j}} = (\theta_1 - \text{hops}) \times \tau^{\text{mean}}_{\text{hop}}$ and then start to process the activity normally.

2) Improving Reliability of Transmissions: If the network becomes terrible $(p^{\text{aet}}_{\text{hop}} \leq 0.9)$, $v_{i,j}$ should reserve time for retransmission if $\tau^w_{i,j} + \tau^d_{i,j} - \tau^{\text{mean}}_{i,j} > 0$. Otherwise, $v_{i,j}$ will apply both proactive temporal redundancy (multicopy transmission) and spatial redundancy (multipath transmission). The number of redundancy path is $\lfloor \log^{1-p^r_{\text{AC}}}_{1-(p^{\text{aet}}_{\text{hop}})^{\text{hops}}} \rfloor$.

3) Self-Remediation: According to the agreement, $v_{i,j}$ is considered failed if $v_{i-1,k}$ has not received the feedback event from any successor actors before time (note that $v_{i,j}$ may be still alive). Then, $v_{i-1,k}$ broadcasts the message $< \text{tid}_{\chi_i}, \tau^{'}_i > \Rightarrow \text{uuid}_{\chi_{i+1}}$ where $\tau^{'}_i = \tau^w_i - \tau^{\text{timeout}}_i$ and supposes $v_{i+1,h}$ as the actor whose type id is $\text{uuid}_{\chi_{i+1}}$. Any actor whose type id is tid_{χ_i} should send its observation to the actor $v_{i+1,h}$ as fast as possible. For the actor $v_{i+1,h}$, the waiting time (the delay time) is about $\tau^w_{i+1} \approx \tau^w_{i+1} - \tau^{\text{timeout}}_i - \tau^{\text{aet}}_i - \text{hops}(v_{i-i,k}, v_x)$ (ignore the case where all redundant actors v_i fail, and $\tau^{\text{aet}}_i \approx 0$ if several redundant actors are arranged to process χ_i). If $\tau^w_{i+1} < 0$, the actor $v_{i+1,h}$ (the one whose id is $\text{uuid}_{\chi_{i+1}}$) will start processing as soon as receiving the first events and will ignore all others.

4) Eliminating the Error of Time: In the real-world system, various issues will cause timing errors such as inaccurate clock, estimate errors of τ, and failures. We should remove outliers caused by these issues. As shown in Fig. 8, suppose the synchronization actor v_{syn} has *m* precursors, the arrival timestamps according to a local clock on v_{syn} are $\{t_1, \ldots, t_m\}$, t_p is the predetermined time, and $\{\tau^w_1, \ldots, \tau^w_m\}$ are the remaining time budgets of the precursor activities. ε_j is the *j*th arrived event; if $\tau^w_j > 0$, ε_j arrives earlier; otherwise it is late. As τ^d_v of all activities are aligned with the (3), ideally, $t_j + \tau^w_j \approx t_p$. We can estimate the predetermined time with $t_p = \sum_{k,k\leq m} (t_j + \tau^w_j)/k$.

5) Updating Statistic Value: All actors should periodically update failure counter of actors, the retransmission rate, and the remaining energy to the local DSS to estimate the value of p^{cp}_v, $p^{\text{mean}}_{\text{hop}}$, and c_v. Meanwhile, the actor should notify the local DSS when its link has changed.

V. Contract Optimization and Evaluation on Real-World System

A. Simulation Result of NSGA-II Solution Set and Waiting Time

According to $R(t) = e^{-\lambda t}$, the reliability of an actor depends on the failure rate and the performance (i.e., the processing time). We evaluate the refinement based on a random simulation. The failure rate of all actors is $\lambda = 0.0001$, each activity has 10 candidate actors and the related reliability requirement $p^r_\chi = 0.99$. The values of $\tau_v \in [\tau^{\text{bcet}}, \tau^{\text{mcet}}]$ follow a uniform distribution and are randomly generated with three groups, whose ranges are $\tau_1 \in [100, 300]$ (Curve1), $\tau_3 \in [100, 1000]$ (Curve3), and $\tau_4 \in [1000, 5000]$ (Curve4). For $\tau_2 \in [300, 500]$ (Curve2), we add 200 waiting times for fast actors based on the τ_1 in Curve1 (here, we assume that the sleep actor has the same λ as the active actor, if we ignore the failure during the waiting time, the result is a curve with the $\text{mean}(R(g_{\text{DAAN}}))$ of Curve1, the $\min(R(g_{\text{DAAN}}))$, and $\max(R(g_{\text{DAAN}}))$ of Curve2, which will be completely covered by Curve1. As the actors are compositional, we use a timing composition schema to simplify the contract. The number of activities increases from 1 to 40 with Step 1. We simulate 1000 times in each activity. The $\text{mean}(R(g_{\text{DAAN}}), \min(R(g_{\text{DAAN}}), \max(R(g_{\text{DAAN}}))$, and the related standard deviation of the reliability of the generated solution are shown in Fig. 9.

The result shows that the range of reliability $\max(R(g_{\text{DAAN}})) - \min(R(g_{\text{DAAN}}))$ and standard deviation of reliability increase with the increasing complexity of contract (the increasing number of activities), which implies that the behavior of decision process becomes more and more unpredictable with the increasing complexity of contract. Compared to curve1, the reliability of curve2 is more stable (the standard deviation is lower) though they have the same range of τ_v (similar conclusion can also be made according to the results of $\delta(R(g_{\text{DAAN}}))$ which is shown in Appendix B). Though the results of curve1 and curve4 have the same scale of $(\tau^{\text{wcet}}_v - \tau^{\text{bcet}}_v)/\tau^{\text{mean}}_v$, curve4 has much larger standard deviation. Similarly, $T^{\text{wcet}}_{g_{\text{DAAN}}} - T^{\text{bcet}}_{g_{\text{DAAN}}}$ increases with the complexity of contract (more detailed simulation result can be found in this article [18]). These conclusions

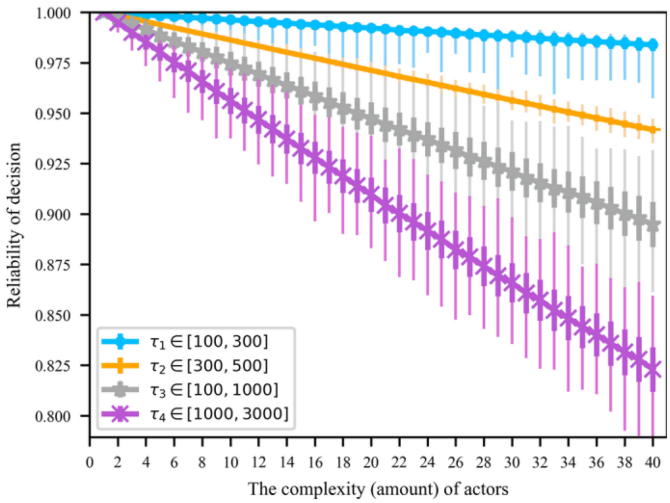


Fig. 9. Reliability range of solution against τ_v^{mean} and the number of activities.

show that the range of τ_v is the main factor of the stability of the reliability.

Hence, we can narrow the range of τ_v to reduce fluctuation of reliability (the comparison of Curve1 and Curve2). In our approach, we add additional waiting time to align the processing time of each path, so that the DSS can generate a more controllable and predictable solution. Moreover, the waiting time also reserves time for applying redundancy strategies when failures occur. (Proof. To simplify, suppose the failure of actors is an independent identical distribution, the execution time with k actors follows the Erlang distribution $X \sim \text{Erlang}(k, \lambda)$, and the mean execution time of the contract is $k \times \tau^{\text{mean}}$. The waiting time solution can tolerate at least $k \times (\text{WCET} - \tau^{\text{mean}})/\tau^{\text{mean}}$ times of failure].

However, adding additional waiting time increases the risk of deadline missing. We design the compositional contract to relieve the problem. Compared to the centralized solution, our approach does not need the round-trip communication to notify the local DSS about the result of activities and the local subsystems also do not have to wait the next command from local DSS. Moreover, our compositional contract can also be regarded as a pipeline solution, it can hide the communication time of sending the subdecision to actors. As a result, it can save at least $\sum_{\chi \in \text{path}_c/\{\chi_{\text{start}}\}} 2 \times \text{hops}(\text{ldss}, \chi) \times \tau_{\text{hop}}^{\text{mean}}$ time.

B. Testing on Real-World System

In this section, we implemented and tested our contract solution. Our platform uses one PC (Intel i5-75000 with 16G memory) as the local DSS and the Arduino Mega2560 boards as the sensors and actuators. According to the type of sensors, the Arduino boards can be classified into following three types:

1) Type 1 (top, Arduino 1) has one actor of light sensor (Keyes K853518).
2) Type 2 (middle) has one actor of soil moisture sensor (FC−28) and another actor of the humidifier, which is the actuator of this demonstration.

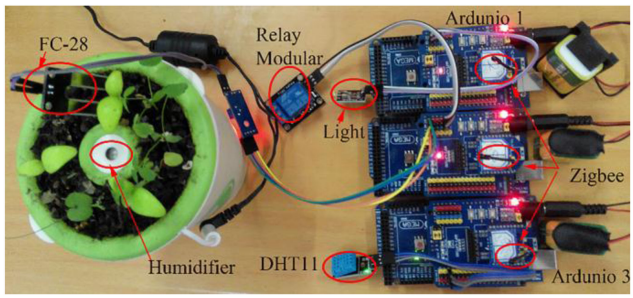


Fig. 10. Three types of Arduino boards applied in demonstration.

TABLE III
PARAMETERS OF ACTORS IN DEMONSTRATION

Actors (tid,uid)	$< \tau^{bcet}, \tau^{mean}, \tau^{wcet}, \lambda_c >$	p_v^{cp}	c_v (mw)
$\chi_l : v_l(01,01)$	<0.07,0.07,0.08, 0.015>	0.99895055	114.8
$\chi_l : v_l(01,02)$	<0.07,0.07,0.08, 0.015>	0.99895055	106.7
$\chi_l : v_l(01,03)$	#<0.08,0.09,0.11,0.016>	0.99856104	#98.2
$\chi_t : v_t(02,01)$	<0.02,0.05,30,0.009>[1]	0.99955010	262.4
$\chi_t : v_t(02,02)$	<0.02,0.05,30,0.023>	0.99885066	254.6
$\chi_h : v_h(03,01)$	<0.02,0.05,30,0.009>	0.99955010	262.4
$\chi_h : v_h(03,02)$	<0.02,0.05,30,0.023>	0.99885066	254.6
$\chi_m : v_m(04,01)$	<0.04,0.05,0.07,0.010>	0.99950012	303.4
$\chi_m : v_m(04,02)$	<0.04,0.05,0.07,0.011>	0.99945015	304.4
$\chi_w : v_w(05,01)$	<200,300,540,3*10^{-6}>	0.99980000	1500
Mainboard Xbee s2	NA	NA	603.5
	$\tau_{hop}^{mean} = 0.031$, $\lambda_{hop}^{mean} \approx 2 \times 10^{-4}$, $w_e \approx 606.8$		

[1]The values with # are modified to simulate the heterogeneous actors.
[2]It takes DHT11 30 s to be stable. Then it takes less than 50 ms to sense.

3) Type 3 (bottom) has two actors of temperature-humidity sensor (DHT11). All types of subsystmes use zigbee (xbee s2) to communicate with each other. The three type of boards are shown in Fig. 10.

To accelerate the evaluation, we use our light-weight fault injection tool [31] to increase the failure rate, where real failure rate $\lambda_r \in (-\ln(0.9995)/\tau, -\ln(1 - \sqrt[n]{0.001})/\tau)$. To simulate the delay of DC, we have collected the failure rate λ_c for DSS, where $\lambda_c = \sum_n \lambda_r + \text{random}(-0.001, 0.001)$. The τ^{bcet}, τ^{mean}, and τ^{wcet} are collected from the datasheet. Meanwhile, we changed some sensors with delay operation to simulate the heterogeneous actors (the value with # is adjusted), the c_v and w_e are testing value. The baud rate of xbee is 57 600, the mean transmission time is approximated with the half value of round trap time. The expected number of transmission hops is 1, except the transmission between $v_t : (02, 01)$ and$v_h : (03, 01)$, $v_t : (02, 02)$ and $v_h : (03, 02)$,$v_m : (04, 01)$ and $v_w : (05, 01)$. All parameters applied in demonstration are shown in Table III.

In this case, the involved boards include three type1 boards, two type2 boards, two type3 boards, and one humidifier. All boards can be accessed in one hop. Suppose the contract is watering the plants if the soil moisture is lower than

TABLE IV
REQUIREMENTS OF ACTIVITIES

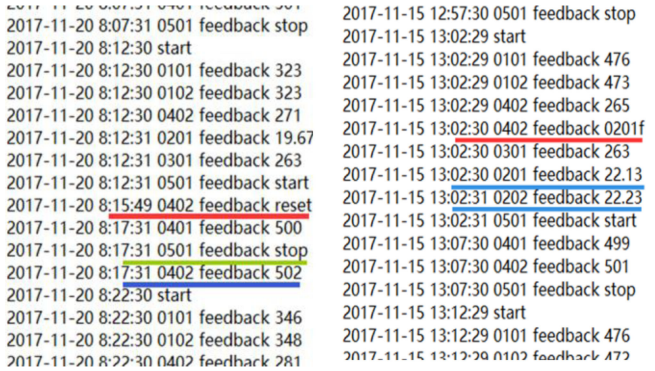
Activities	Operation and requirement
χ_l	$<01,(>,300)\wedge(\leq,800)>.<0.999>$
χ_t	$<02,(>,16)\wedge(<,34)>.<0.999>$
χ_h	$<03,(<,0.6)>.<0.999>$
χ_{m1}	$<04,<,0.3>.<0.999>$
χ_w	$<05>.<0.999>$
χ_{m2}	$<04,>,0.5>.<0.999>$

0.3, the light intensity is between 300 and 800, the temperature is between 16 and 34 °C, and air humidity is less than 60%; the humidifier stops watering if the soil moisture is higher than 0.6. The term of validity is 10 days, $T^r_{\mathrm{AC}}=300.5$ s and $R^r_{\mathrm{AC}}=0.99$. The contract is $c1:(\chi_{m1},\chi_l)\Rightarrow(\chi_t\wedge\chi_h)\Rightarrow(\chi_w\perp\chi_{m2}).<10d,300.5,600,99>$, and it can also be written as $c2:(\chi_{m1}||\chi_l)\Rightarrow(\chi_t\wedge\chi_h)\Rightarrow(\chi_w\perp\chi_{m2}).<10d,300.5,600,99>$. The detailed requirements of activities are shown in Table IV.

C. NSGA-II Solutions Set

It takes about 7.3 s to finish 20 rounds of the NSGA-II on our PC. Some solutions of the contract are shown Table V. Under given requirements, we prefer to apply the solution 1, though solution 1 is not the best one among the populations for any fitness value. To the given weight, solution 1 has the best weighted score. Comparing solution 1 and solution 2 (the detailed difference between solution 1 and solution 2 is shown in Fig. 14 in the Appendix), we can save the energy for battery-powered nodes by switching the order of activities if the precursor (the local DSS in this case) is powered by mains. Compared to solution 1 (a serial composition), solution 58 (a parallel composition, which is the fastest solution) saves 0.081 s but consumes 75.243 mJ more energy. Comparing solution 1 and solution 5, spatial redundancy can improve the reliability without too much increase in time budget. Hence, for time-critical contracts, we can give priority to apply the mutation with parallel composition and spatial redundancy to improve the timing reliability requirements. For time-rich contracts, we can give priority to apply the mutation with serial composition and temporal composition to save energy (especially, some contract may be terminated in advance when some trigger conditions are not met).

As shown in Table III, for the reliability of each activity $p^{\mathrm{cp}}_v>0.995$, redundancy strategies increase the value slightly (for two-model redundancy, $(p_{\mathrm{tmd}}-p)/p=1-p\propto 10^{-3}$). Hence, it is not necessary to calculate $f^{\mathrm{path_{cp}}}_{\mathrm{risk}}$ and $f_{\mathrm{reliability}}$ after each polyploid crossover and mutation. Hence, we can ignore the calculation of $f^{\mathrm{path_{cp}}}_{\mathrm{risk}}$ and $f_{\mathrm{reliability}}$ in some iterations to accelerate NSGA-II. If the weather is terrible (snow or heavy rain, $p^{\mathrm{cp}}_{\mathrm{hop}}<0.99$), we need to calculate $f_{\mathrm{reliability}}$ and $f^{\mathrm{path_{cp}}}_{\mathrm{risk}}$ after each polyploid crossover and mutation. To the f_{time}, χ_w takes more than 99.8% processing time. To effectively filtrate the refinement of topology of activities, and to make full use of the time budget, we can use the $T(g_{\mathrm{DAAN}})-T^r_{\mathrm{AC}}$ as the parameter of the sigmoid function.



(a) (b)

Fig. 11. Log of demonstration. (a) Failure recovery (0402). (b) False detection (0201 timeout).

D. Results of Real-World Testing

We have tested on the real-world board for seven rounds. For a comparative analysis, the local DSS repeatedly sends the contract with the same requirement in each round. The term of validity of contract is ten days, the period is 600 s. It records 1439×7 times of decision processing, and 48 actor-level failures have been recorded in seven rounds of testing (system has recovered from all failures) and the success rate is 0.995. Among 48 failures, χ_m failed 19 times, both χ_t and χ_h failed 10 times, and χ_l failed 9 times. Two failure records are shown in Fig. 11 (the format of log is "$time $type-id feedback $value").

In the Fig. 11(a), the actor of 0402 (χ_{m2}) failed and was reset at 199 s during this testing. Though the actor rejoined the team but an observable delay was introduced (feedback of 0402 is later than normal). As 0401 performed normally and fed back in time, the contract was finished successfully (0501 stopped normally). In another case, shown in Fig. 10(b), 0201 has not responded to actor 0402 in time (0201 was alive because it fed back to the DSS). According to runtime strategy of self-remediation, 0402 arranged another actor 0202 to process decision. The decision was finished successfully.

Theoretically, the failure number is $(1-0.996802)\times 1439\times 7=32.2$. The record is little higher than expected value. The reason could be 1) the statistical bias of small sample data. 2) The failure rate is higher than the setting (other code may increase the failure rate). 3) The failure injection rate changes because of the imprecise local clock.

As shown in Fig. 11, the decision can be finished in time, even if some actors fail (in our testing, all contracts finished in time). Take Fig. 11(a) as example, though 0402 failed and reset, its feedback information almost arrived at almost the same time with 0401, which implies that the additional waiting time can be used for hiding the operation overhead of recovery and temporal redundancy. By adding additional waiting time actors

TABLE V
PART OF THE NSGA-II SOLUTIONS SET

	Pattern	χ_{m1}	χ_l	χ_t	χ_h	χ_w	χ_{m2}	$T_{g_{DAAN}}$	f_{time}	$f_{reliability}$	$f_{risk}^{path_{cp}}$	f_{energy}	$\sum w_i \times \alpha_i^2$
1	C1.0	0401	0101‖0102	0201	0301	0501	0401‖0402	300.425	0.48126	0.203435	0.270299	451508.368	0.12303365
2	C1.1	0401	0101‖0102	0201	0301	0501	0401‖0402	300.425	0.48126	0.205050	0.270299	451545.990	0.15485995
3	C1.0	0402	0101‖0102	0201	0301	0501	0401‖0402	300.425	0.48126	0.203435	0.270299	451546.040	0.15490865
4	C1.1	0402	0101‖0102	0201	0301	0501	0401‖0402	300.425	0.48126	0.205050	0.270301	451546.040	0.16928043
5	C1.0/1.1	0401	$0102 \oplus 2$	0201	0301	0501	0401‖0402	300.495	0.49875	0.209933	0.270372	451476.958	0.31407393
7	C1.0/1.1	0401	$0101 \oplus 2$	0201	0301	0501	0401‖0402	300.495	0.49875	0.209933	0.270372	451477.525	0.31436143
9	C1.0/1.1	0402	$0102 \oplus 2$	0201	0301	0501	0401‖0402	300.495	0.49875	0.211585	0.270375	451477.008	0.35844357
…	…	…	…	…	…	…	…	…	…	…	…	…	…
17	C2	0401	$0102 \oplus 2$	0201	0301	0501	0401‖0402	300.414	0.47851	0.216600	0.270347	451458.139	0.42890362
…	…	…	…	…	…	…	…	…	…	…	…	…	…
58	C2	0401	0010‖0102	0201	0301	0501	0401‖0402	300.344	0.46108	0.216600	0.270225	451621.233	0.50165067
…	…	…	…	…	…	…	…	…	…	…	…	…	…

can leisurely apply self-remediation operations, which increases the possibility to finish the decision in time.

VI. DISCUSSION AND FURTHER WORK

As the future environment is full of uncertainties, it is necessary to apply systematic model@run.time solution to reduce the uncertainties and to improve the responsiveness, dependability, and efficiency of self-adaptation. In this article, we mainly focus on the contract and the runtime refinement solutions and introduced the formal specification of the compositional contract to help subsystems reach a consensus on requirements of decisions. Our systematic-contract-based solution clearly states the type id of coordinators, the targets, and the requirements for each step of the decision. By clearly stating the duties for every actor, it can balance the autonomy and controllability at runtime and can help the local subsystem adjust the progress without accessing the global information. It also is a solution to overcome selfishness and avoid making shortsighted adaptation decisions.

The contract also decouples the decision making, decision control, and decision processing. CPS can make decisions on remote resource-rich subsystems (i.e., Cloud system), and process the decision on resource-limited subsystems (i.e., sensors and actuators). Sensors and actuator can selectively process the decision according to their own situation. To balance the autonomy and controllability at runtime, we proposed a gradual optimization solution. CPS can refine the process flow of contract based on the newest information at each period. As well, the precursors could autonomously select the proper (number of) successors, and all subsystems could adjust the progress without accessing the global information. Our approach reduces the delay of observation by avoiding unnecessary round-trip communication of sharing messages. Moreover, reducing message can also reduce the failure risk of communication and improve the energy efficiency.

By the simulation, we first evaluated the relationship between the complexity of the contract and the predictability and reliability, and found that narrowing the range of the actors' execution time can improve the predictability and stability of decision processing. Then, we tested the refinement solution on our platform for 10×7 days. The result shows that our solution is highly reliable even under a situation with relatively high failure rate. With runtime refinement strategies, the local subsystems can take full advantage of the additional waiting time and apply temporal redundancy without significantly increasing the whole processing time.

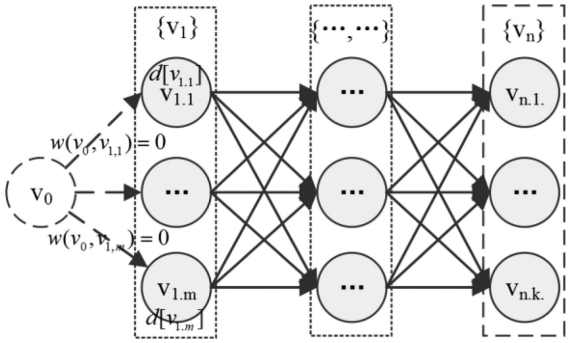


Fig. 12. Generic subgraph g of DWVCG built with redundant vertexes, which is isomorphic to the path $\mathrm{path} = (v1, \ldots, v_n)$ of $\mathrm{DAAN}_{\mathrm{con}}$.

Currently, the weights of selection rule are constant, we will research on the solution that can dynamically adjust the weights according to the requirements of contracts and the situation of environment. So that, the refinement solution can adapt to the environment and requirement dynamically. Another direction is improving the precision of timing, which includes precision timed system, which has been discussed in [5]. The third further work is increasing the scale of our platform and testing it in the wild with real-world application.

APPENDIX

A. Algorithm to Check the Feasibility of Contracts (in Section IV.B)

Suppose $g \in$ DWVCG is the corresponding subgraph of the $\mathrm{path}_k = [v_1, v^k_{\ldots}, \ldots, v_n]$, and path_k is one branch of the subgraph $\mathrm{sub}S_h \in \mathrm{DAAN}_{\mathrm{con}}$, which is shown in Fig. 12. g is built up with all available redundant vertexes $[\{v_1\}, \{v^k_{\ldots}\}, \{v_n\}]$, and all the

Algorithm 1: Search the Single-Objective Optimized Solution.

Input: $\{S_{\mathrm{DAAN}}\}$ is the isomorphic graphs set of $\mathrm{DAAN}_{\mathrm{con}}$ in DWVCG, V_{syn} is the set of synchronization vertexes of $\mathrm{DAAN}_{\mathrm{con}}$

Output: The target graph g_{DAAN}, the critical path $\mathrm{path}_{\mathrm{cp}}$, minimal weight $\min_{\mathrm{weight}}(g_{\mathrm{DAAN}})$, V_{syn} with properties of weight

/* $\{\mathrm{sub}S\}$ is the set of all subgraphs between two synchronization vertexes of $\mathrm{DAAN}_{\mathrm{con}}$, $\{\mathrm{sub}g\}_m = \Theta$ is the marked subgraph,$\{\mathrm{sub}S_{\mathrm{DAAN}}\}$ is the corresponding subgraph set of $\{S_{\mathrm{DAAN}}\}$, g is a corresponding subgraph of $\mathrm{path}_i \in \mathrm{sub}S$ in DWVCG*/

1. If $\{S_{\mathrm{DAAN}}\} = \phi$ then return NULL.
2. Process the $\mathrm{DAAN}_{\mathrm{con}}$ according to the partial order of V_{syn}
3. While $\{\mathrm{sub}S\} \neq \Theta$ do
4. Select a $\mathrm{sub}g_k$ where $\{\mathrm{subg}\}_{\mathrm{precursor}} \in \{\mathrm{subg}\}_m$ // all precursor subgraphs have been processed
5. branchWeight [amount_of_branch] $[|\{v_{\mathrm{syn}.s}\}|] = 0$
6. For $\mathrm{path}_i = (v_{\mathrm{syn}.p}, v_{\mathrm{syn}.s}) \in \mathrm{subg}$ do
7. Get the corresponding subgraph $g \in \{\mathrm{sub}S_{\mathrm{DAAN}}\}$ of path_i
8. Add a virtual vertex v_0 before $\{v_1\}_r$ and for $v_{1,i} \in \{v_1\}_r$ set $w(v_0, v_{1,i}) = 0$;//(as shown in Fig. 12)
9. For $v_n \in \{v_n\}_r$ do //$\{v_n\}_r$ is the set of all redundant vertex of v_n
10. For $\exists v_n \in \{v_n\}_r$, employ Dijkstra's algorithm [32] to calculate $\min(v_0, v_n)$, where the relaxation function is $d[v] > d[u] + w(u) + w(u, v)$. /*Notice that Dijkstra's algorithm is designed for edge-weighted only graph. $\{S_{\mathrm{DAAN}}\}$ is the doubly weighted graph. $d[v]$ is the shortest path from source to vertex v,$w(u)$ is the weight of vertex u. $w(u, v) = \mathrm{hops}(u, v)^* \tau_{\mathrm{hop}}^{\mathrm{mean}}$ is the weight of edge $e(u, v)$.*/
11. $\mathrm{branchWeight}[][i] += d[v_{\mathrm{syn}.i}] + w(v_{\mathrm{syn}.i})$// $w([v_{p.e}, v_{s.s}]) = \min_{\mathrm{path} \in \{(v_{p.e}, v_{s.s})^*\}_{\mathrm{rpath}}} (\sum_{e,v \in \mathrm{path}} w(e) + w(v)) + w(v_{s.s})$
12. End for
13. End for
14. Record the branchWeight[][] as a property of $\{v_n\}$ //this is for NSGA II
15. Set t $\{w(v_{n.s}, v_{n.e})\} = \min_{\mathrm{row}} \max_{\mathrm{column}} \mathrm{branchWeight}_{\mathrm{path}(n.s,n.e)}[][]$ /*$\min_{\mathrm{row}} \max_{\mathrm{column}}$ is the select the max number of each column, then select the min number among those max number */
16. Record the n potential solutions $\{g^*\}$
17. Set $\{\mathrm{subg}\}_m = \{\mathrm{subg}\}_m \cup \mathrm{subg}_k$, $\{\mathrm{sub}S\} = \{\mathrm{sub}S\}/\mathrm{sub}g_k$;
18. End while
19. Get the minimal weight $w_{\min}(g_{\mathrm{DAAN}}) = \min(\min_{\mathrm{row}} \max_{\mathrm{column}} \mathrm{branchWeight}_{\mathrm{path}(n.s,n.e)}[][]$
20. Backtrack from v_{end}^*, select the shortage path of each redundant branch that connected to v_{end}^*, and record the branch with maximal weight as the critical path path_{cp}. //If there exist several paths with best weight, record them all
21. The target graph g_{DAAN} is the graph built with all backtracking paths. //notice that, we may have several target graph. g_{DAAN}.

paths formed with these vertexes are the potential solutions of the path_k, where v_1 and v_n are the synchronization vertexes of $subS_h$ (note that, to the subgraph $\mathrm{sub}S_h$ with son subgraph, different paths of $\mathrm{sub}S_h$ share at least one synchronization vertex v_1 or v_n), and for $\forall v_i^k \in \mathrm{path}_k$, $\mathrm{colour}(\{v_i\}) = \mathrm{typeid}(v_i^k)$ and $\{v_{k.i}\} \in \mathrm{DWVCG}$. The problem of checking the feasibility of contracts is recursively searching the path with the minimal or maximal weight from DWVCG as the potential solutions.

B. Fluctuation Range and the Standard Deviation of Reliability

The fluctuation range of reliability and the standard deviation $\delta(R(g_{\mathrm{DAAN}}))$ are shown in Fig. 13. These curves are drawn with the same simulation data from Fig. 9. According to reliability equation in Table I, the stability of the reliability of a contract mainly depends on the range of $\tau_v^{\mathrm{wcet}} - \tau_v^{\mathrm{bcet}}$, the complexity of the contract, and failure rate λ. Moreover, the curves also depend on the reliability requirement and the distribution of τ_v. Obviously, we can improve the stability of reliability by increasing the number of redundant actors or reducing λ and τ_v. However, these methods generally are costly. Relatively speaking, adding waiting time is a low-cost but efficient solution to improve the stability of reliability.

C. Complexity of Topology and the Difference of Two Serial Solutions

To a serial composition, we can adjust the sequence of the activities without changing the function of contract. Hence, in a serial composition, there are A_n^n combinations, where *n* is the number of activity. To this case, without considering the temporal redundancy, the contract C1 has $567 \times 2 = 1134$ structure combinations and C2 has 567 structure combinations. In spatial redundancy, we ignore the serial version (χ_1, χ_1) as all cases of spatial redundancy are process in parallel (because $\chi_1 \oplus 2$ can save more energy than χ_1, χ_1). The topology of C1.0 and C1.1 in Table V is shown in Fig. 14. As the local DSS is mains powered, we ignore the energy budget of transmission from the local DSS to the sensors. Hence, C1.0 has less energy budget than C1.1 in this case. Moreover, we can arrange the best process order of a serial composition according to the rank of possibility that the activity is false.

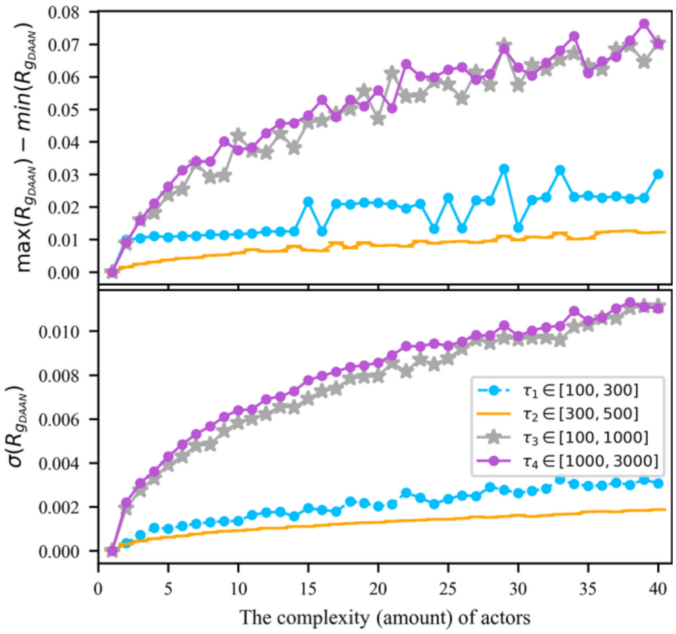


Fig. 13. Fluctuation range and standard deviation of reliability against the number of actors.

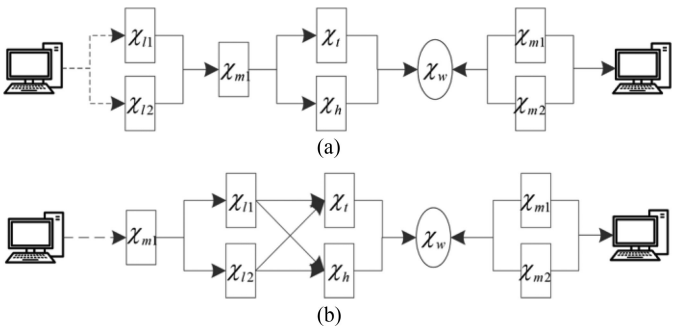


Fig. 14. Topologies of C1.0 and C1.1 in Table V. (a) Topology of solution 1 (C1.0 of contract c1, where χ_l is before χ_{m1}). (b) Topology solution 2 (C1.1 of contract c1, where χ_{m1} is before χ_l).

ACKNOWLEDGMENT

The authors would like to thank the editor and reviewers for their valuable comments, which help improve the quality of this article.

REFERENCES

[1] P. Leitao, S. Karnouskos, L. Ribeiro, J. Lee, T. Strasser, and A. W. Colombo, "Smart agents in industrial cyber–physical systems," *Proc. IEEE*, vol. 104, no. 5, pp. 1086–1101, May 2016.

[2] H. Liu, D. Sun, and W. Liu, "Lattice hydrodynamic model based traffic control: A transportation cyber-physical system approach," *Physica a-Statistical Mech. Its Appl.*, vol. 461, pp. 795–801, Nov. 2016.

[3] H. A. Nasir and A. Muhammad, "Control of very-large scale irrigation networks: A CPS approach in a developing-world setting," *IFAC Proc. Volumes*, vol. 44, no. 1, pp. 10739–10745, 2011.

[4] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Gener. Computer Syst.*, vol. 29, pp. 1645–1660, 2013.

[5] D. Broman *et al.*, "Precision timed infrastructure: Design challenges," in *Proc. Electron. Syst. Level Synthesis Conf.*, 2013, pp. 1–6.

[6] D. Broman, P. Derler, and J. Eidson, "Temporal issues in cyber-physical systems," *J. Indian Inst. Sci.*, vol. 93, pp. 389–402, 2013.

[7] P. Alho and J. Mattila, "Service-oriented approach to fault tolerance in CPSs," *J. Syst. Softw.*, vol. 105, pp. 1–17, 2015.

[8] J. Pfrommer, D. Stogl, K. Aleksandrov, S. E. Navarro, B. Hein, and J. Beyerer, "Plug & produce by modelling skills and service-oriented orchestration of reconfigurable manufacturing systems," *At-Automatisierungstechnik*, vol. 63, pp. 790–800, Oct. 2015.

[9] Z. Shu, J. Wan, D. Zhang, and D. Li, "Cloud-integrated cyber-physical systems for complex industrial applications," *Mobile Networks Appl.*, vol. 21, pp. 865–878, Oct. 2016.

[10] J. M. Bradley and E. M. Atkins, "Optimization and control of cyber-physical vehicle systems," *Sensors*, vol. 15, pp. 23020–23049, 2015.

[11] Y. Zhang, C. Qian, J. Lv, and Y. Liu, "Agent and cyber-physical system based self-organizing and self-adaptive intelligent shopfloor," *IEEE Trans. Ind. Inform.*, vol. 13, no. 2, pp. 737–747, Apr. 2016.

[12] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proc. Workshop Mobile Big Data*, 2015, pp. 37–42.

[13] N. M. Freris, S. R. Graham, and P. Kumar, "Fundamental limits on synchronizing clocks over networks," *IEEE Trans. Autom. Control*, vol. 56, no. 6, pp. 1352–1364, Jun. 2011.

[14] J. Ren, Y. Zhang, K. Zhang, A. Liu, J. Chen, and X. S. Shen, "Lifetime and energy hole evolution analysis in data-gathering wireless sensor networks," *IEEE Trans. Ind. Inform.*, vol. 12, no. 2, pp. 788–800, Apr. 2016.

[15] M. A. A. Al-Jaoufi, Y. Liu, Z.-J. Zhang, and L. Uden, "Study on selfish node incentive mechanism with a forward game node in wireless sensor networks," *Int. J. Antennas Propag.*, vol. 2017, 2017.

[16] H. Mo and G. Sansavini, "Dynamic defense resource allocation for minimizing unsupplied demand in cyber-physical systems against uncertain attacks," *IEEE Trans. Rel.*, vol. 66, no. 4, pp. 1253–1265, Dec. 2017.

[17] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in AI safety," 2016, *arXiv preprint arXiv:1606.06565*.

[18] P. Zhou, D. Zuo, K.-M. Hou, and Z. Zhang, "A decentralized compositional framework for dependable decision process in self-managed cyber physical systems," *Sensors (Basel, Switzerland)*, vol. 17, pp. 1–33, Nov. 2017.

[19] M. C. Huebscher and J. A. McCann, "A survey of autonomic computing—degrees, models, and applications," *ACM Comput. Surv. (CSUR)*, vol. 40, p. 7, 2008.

[20] J. Qin, Q. Ma, Y. Shi, and L. Wang, "Recent advances in consensus of multi-agent systems: A brief survey," *IEEE Trans. Ind. Electron.*, vol. 64, no. 6, pp. 4972–4983, Jun. 2017.

[21] X. Ma, S. Chisiu, R. Kacimi, and R. Dhaou, "Opportunistic communications in WSN Using UAV," in *Proc. 14th IEEE Annu. Consumer Commun. Netw. Conf.*, 2017, pp. 510–515.

[22] P. Derler, E. A. Lee, S. Tripakis, and M. Törngren, "Cyber-physical system design contracts," in *Proc. ACM/IEEE 4th Int. Conf. Cyber-Physical Syst.*, 2013, pp. 109–118.

[23] M. Toerngren, S. Tripakis, P. Derler, and E. A. Lee, "Design contracts for cyber-physical systems: Making timing assumptions explicit," *Electrical Engineering & Computer Sciences Dept.*, Univ. California, Berkeley, Berkley, CA, USA, Tech. Rep. UCB/EECS-2012-191, 2012.

[24] D. Xu, W. Xu, M. Tu, N. Shen, W. Chu, and C.-H. Chang, "Automated integration testing using logical contracts," *IEEE Trans. Rel.*, vol. 65, no. 3, pp. 1205–1222, Sep. 2016.

[25] A. M. Mora, P. Garcia-Sanchez, J. J. Merelo, and P. A. Castillo, "Pareto-based multi-colony multi-objective ant colony optimization algorithms: An island model proposal," *Soft Comput.*, vol. 17, pp. 1175–1207, Jul. 2013.

[26] D. B. Johnson, "Efficient algorithms for shortest paths in sparse networks," *J. ACM (JACM)*, vol. 24, pp. 1–13, 1977.

[27] G. Logothetis and P. Mishra, "Compiling short-circuit boolean expressions in one pass," *Software: Practice Experience*, vol. 11, pp. 1197–1214, 1981.

[28] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[29] J. Safari, "Multi-objective reliability optimization of series-parallel systems with a choice of redundancy strategies," *Rel. Eng. Syst. Saf.*, vol. 108, pp. 10–20, 2012.

[30] I. Meedeniya, B. Buhnova, A. Aleti, and L. Grunske, "Architecture-driven reliability and energy optimization for complex embedded systems," in *Proc. Int. Conf. Qual. Softw. Architectures*, 2010, pp. 52–67.

[31] P. Zhou, D.-C. Zuo, K.-M. Hou, Z. Zhang, and H.-L. Shi, "A Light-weight multilevel recoverable container for event-driven system: A self-healing CPS approach," in *Proc. Int. Conf. Wireless Commun. Sensor Netw.*, 2017, pp. 439–446.

[32] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

Peng Zhou was born in Zhejiang, China, in 1987. He received the B.S and M.S. degrees in computer science from the Harbin Institution of Technology (HIT), Harbin, China. He is currently working toward the joint Ph.D. degree with the Research Center of Fault-Tolerance and Mobile Computing, HIT and L'Institut Supérieur d'Informatique, de Modélisation et de leurs Applications, Université Clermont Auvergne, France.

His research area is the dependable self-managing cyber physical system design and evaluation.

Decheng Zuo received the Ph.D. degree in computer science and technology from the Harbin Institute of Technology (HIT), Harbin, China, in 2000.

Since 2007, he has been a Professor with the Research Center of Fault-Tolerance and Mobile Computing, School of Computer Science and Technology, HIT. His research interests include parallel computing and architecture, fault-tolerant computer, and computer system architecture evaluation theory and technology.

Prof. Zuo is a member of the Expert Committee for Information Field, National High-tech R&D Program of China (863 Program).

Kunmean Hou received the Ph.D. degree in system controls and the HDR degree in computer science from the University of Technology of Compiègne (UTC), Compiègne, France, in 1984 and 1996, respectively.

From 1984 to 1986, he was an Associate Professor with High-Performance Computing, UTC. In 1986, he joined IN2 as R&D engineer group leader to develop fault-tolerant super-minicomputer. From 1989 to 1996, he led a research group, which investigated parallel architecture dedicated to real-time image processing at laboratory HEUDIASYC UMR 6599 CNRS (UTC). Since 1997, he has been a Full Professor with the College of Engineering School, ISIMA, University Clermont Auvergne, Auvergne, France. His research interests include WSN, Internet of Things, and Web of Things (on robustness and energy efficiency).

Zhan Zhang received the Ph.D. degree in computer science and technology from the Harbin Institute of Technology (HIT), Heilongjiang, China, in 2008.

Since 2014, he has been an Associate Professor with the Research Center of Fault-Tolerance and Mobile Computing, School of Computer Science and Technology, HIT. His research interests include fault-tolerant computer, wearable computer, and system evaluation theory and technology.

Jian Dong received the Ph.D. degree in computer science and technology from the Harbin Institute of Technology, Heilongjiang, China, in 2008.

From 2012 to 2017, he was an Associate Professor with the Research Center of Fault-Tolerance and Mobile Computing, School of Computer Science and Technology, HIT. Since 2017, he has been a Professor with the School of Computer Science and Technology, HIT. His research interests include fault-tolerant computer, parallel computation, and cloud computing.