Fault Tolerance Analysis for Dependable Autonomous Agents using Colored Time Petri Nets

Lan Anh Trinh, Baran Cürüklü and Mikael Ekström
School of Innovation, Design and Technology, Mälardalen University, Västerås, Sweden

Keywords: Dependability, Fault Tolerance, Colored Time Petri Nets, Autonomous Agents.

Abstract: Fault tolerance has become more and more important in the development of autonomous systems with the aim to help the system to recover its normal activities even when some failures happen. Yet, one of the concerns is how to analyze the reliability of a fault tolerance mechanism with regards to the collaboration of multiple agents to complete a complicated task. To do so, an approach of fault tolerance analysis with the colored time Petri net framework is proposed in this work, where a task can be represented by a tree of different concurrent and dependent subtasks to assign to agents. Different subtasks and agents are modeled by color tokens in Petri network. The time values are added to evaluate the processing performance of the whole system with respect to its ability to solve a task with fault tolerance ability. The colored time Petri nets are then tested with simulation of centralized and distributed systems. Finally the experiments are performed to show the feasibility of the proposed approach. From the basics of this study, a generalized framework in the future can be developed to address the fault tolerance analysis for a set of agents working with a sophisticated plan to achieve a common target.

1 INTRODUCTION

Until recently, robots have played an important role mainly in controlled processes such as manufacturing. However, nowadays there is a shift which points in the direction of having various types of (semi-)autonomous agents, or robots, in our daily activities. This paradigm shift also means that these agents will most likely interact with each other, sometimes for collaboration, in most case without having human supervision. In short this is a shift to replace a conventional automatic system with its autonomous counterpart. Unlike an automatic robot that usually performs repetitive tasks within a well-controlled environment, an autonomous robot must perform its tasks with a very high level of automation and may collaborate with other robots and human to complete intended tasks. In this work, it is assumed that building a trustworthy system of collaboration raises a number of challenging questions, that could be addressed with the definition of dependability. Originally, dependability is devised from software development areas and can be stated by Avižienis et al. (Avižienis et al., 2004) as "*the ability to deliver service that can justifiably be trusted*". To realize this idea, the dependability is measured by attributes such as availability, reliability, safety, integrity, or maintainability. In general, the dependability of a system is assessed by one, several, or all above attributes. Within the scope of this paper, the dependability is implemented with the reliability which presents the continuity of a system to provide correct services. It is noted that the things affecting dependability consist of failures, errors, and faults. The link between the above factors is known as the fault-error-failure chain: The failure happens when the service provided by a system does not comply with its specification; The error affects the services and leads to the failure of the system; The hypothesized cause of an error is a fault. However, the failures are only detected at the system boundary. As a system contains a number of interconnected parts, the system boundary is defined to decide which elements are inside and which are outside the system. In some cases, the faults cause errors inside the system boundary and thus the errors may be not observable immediately but lead to a failure later. Therefore, the fault is the key that leads to a system failure and the approach to protect the system's dependability is to develop means of fault analysis and of fault removal to prevent failures from the system. For the sake of simplicity, the concept of agent is used in this paper to refer to software or hardware (robot) sys-

Trinh L., CÃijrÃijklÃij B. and EkstrÃűm M.
Fault Tolerance Analysis for Dependable Autonomous Agents using Colored Time Petri Nets.
DOI: 10.5220/0006196002280235
In *Proceedings of the 9th International Conference on Agents and Artificial Intelligence (ICAART 2017)*, pages 228-235
ISBN: 978-989-758-219-6

tems which perform actions to interact with an environment. Extensive literature reviews of conventional works to deal with the means to remain dependability of autonomous agents are presented by Guiochet (Guiochet, 2015). It is noted that not all the faults from the system always can be analyzed and removed. Fault tolerance therefore aims at continuing agent services even with the presence of faults during the operational stages of autonomous agents. Usually, fault tolerance is implemented by using redundant agents, i.e., once a failure is present, the backup agent is activated to replace the failed one. One limitation of most conventional works is that there is a lack of investigation on the analysis of fault tolerance mechanism within the scope of system dependability. Obviously, the evaluation on the fault tolerance ability of a system could provide valuable information to improve the system performance. This paper therefore proposes a fault tolerance analysis for autonomous agents within the context of agent-agent collaboration. By the means of fault analysis, different methods such as Petri net (PN), fault tree analysis (FTA), failure modes effects and criticality analysis (FMECA), and hazard operability (HAZOP) have been developed (Bernardi et al., 2013). Yet, the PN framework has received a lot of attention from research community due to its wide applications for fault prevention in both development and operational stages of an agent architecture. Which also includes support for mitigation of the implementation progress. With regards to those advantages, an extended PN with colored time PN (CTPN) for the analysis of fault tolerance mechanism of collaborative agents has been chosen.

The rest of the paper is organized as follows. Section 2 presents extensive literature reviews related to this work. The analyses of fault tolerance in both centralized and decentralized approaches together with PN background are described in Section 3. Experimental results are illustrated in Section 4. Finally, Section 5 concludes the paper with discussion of future works.

2 RELATED WORKS

As aforementioned, the assessment of system dependability is based on the basic attributes. Depending on the specific applications, different attributes are used to measure the dependability of a system. In the early developments of software platforms, a multi-level view of dependable computing was first developed by Parhami (Parhami, 1994), in which most dependability attributes were implemented. For robotics, with regards to the safety to assess the dependability, an intelligent home care robot to assist elderly people was introduced by (Graf and Gele, 2001). The proposed home care system was equipped with alternative levels of safety to prevent accidents caused by a person being hit by the robot. The safety navigation system consisted of user interface, path planning, and obstacle avoidance with extensive sensors for motion detection. In the work of (Mustapic et al., 2004), a safe platform for industrial robotics has been developed. The authors have initialized the architectural level of how to open a platform for quality constraints and how to implement fault prevention. Although the aspect of dependability was of major interest, these papers did not address issues related to collaborative robots.

For reliability analysis, PN has been applied as an effective technique to model dependability (Malhotra and Trivedi, 1995). Recently, reliability assessment was introduced with time PN and Markov chains (Kohlík, 2009). The analysis of fault tolerance in manufacturing systems by using PN was developed by (Miyagi and Riascos, 2004). In their study, the hierarchical and modular integration of PN was combined to analyze production process, fault detection process, and fault treatment process. Meanwhile, the application of the generalized stochastic PN used on the navigation of a single service agent was presented by (Kim and Chung, 2007). The coordination of multiple controllers for agent navigation was then introduced by (Moon and Chung, 2012). Similarly, PN was used for the control of a group of robotic agents (Joaquin et al., 2011).

In another aspect, the development of fault tolerance aims at increasing the reliability of a system. In the work of (Troubitsyna and Javed, 2014), adaptive fault tolerance was developed with regards to the system dependability. A research on fault tolerance for a group of agents in a cooperative environment was described by (Haddad and Haddad, 2004). In their research, the authors proposed a communication mechanism between the agents in a team to coordinate and allocate the resources. The PN was used to illustrate the model of the whole system. However, the research is limited to a scheduling protocol for an agent team. Close to our study, the fault tolerance analysis with PN for a coordination of multi agents was developed by (Acharya et al., 2014). However the approach proposed by Acharya et al. has just initialized a picture of how the system may look like. The study lacked experimental setup for validation. Moreover, the non-colored and non-hierarchy PN structure used in the approach made the design complicated and unclear.

In this paper, the colored time PN enhanced with a hierarchy structure is utilized to analyze the dependability of cooperative autonomous agents. The effectiveness of color tokens helps to distinguish multiple agents working together to address a complicated tasks of multiple subtasks. Unlike the original Petri nets, the hierarchy structure of colored PN combined with time value lacks for well-define mathematical tools for analysis. To deal with difficulty, the repeated experiments and recorded data at interested places and transitions of PN are applied for statistical analysis. Finally, the proposed approach are examined with centralized and distributed studies.

3 BACKGROUND AND PROPOSED APPROACH

3.1 Background of Petri Nets

Petri net (PN)(Yen, 2006) was described as a sequence of place-transition-place to move tokens within a PN network. Well defined mathematic models with a set of theory and linear algebra have been developed to analyze the state-transition of PN. Besides, a graphical presentation of PN helps to have a clear visualization of the modeled system, which may consist of synchronization, concurrency, and confusion stage within distributed manners.

In mathematical aspect, PN is a bipartite graph defined as a set of three tuples (Γ, Σ, Θ), where Γ and Σ are the set of finite places and transitions in such a way that there is no element belonging to both Γ and Σ,i.e, Γ and Σ are disjoint sets. Θ is the set of arcs so that an arc connects from a place to a transition and vice versa and the connections between places or between transitions are unacceptable. The arcs going out from a place to a transition are named input places of transitions, while the arcs going out from a place are defined as output places of transitions. An extension of PN adds the weight on the input and output flows of each transition. With regards to the set of output weights W^- and input weights W^+, PN is refined with a set of five tuples $(\Gamma, \Sigma, \Theta, W^-, W^+)$. A marking M of PN assigns a number of tokens to each place. Let the marking M be expressed by a vector $[M(p_1), M(p_2), ..., M(p_i), ..., M(p_n)]$, where p_i is a place, n the number of places in PN, and $M(p_i)$ the number of tokens at the place p_i. Let W^- be a two dimensional matrix of weights $W^-(p_i, t_j)$ from the place p_i to the transition t_j. W^+ is defined in a similar way with the weight $W^+(t_j, p_i)$ from the transition t_j to the place p_i. Note that $1 \leq j \leq m$, where m is the number of transition. With regards to the transition t, the change of the marking vector from M to M' is expressed by

$$M'(p) = M(p) + W^+(t, p) - W^-(p, t), \forall p. \qquad (1)$$

The tuple $(\Gamma, \Sigma, \Theta, W^-, W^+)$ of PN is extended to $(\Gamma, \Sigma, \Theta, W^-, W^+, M_0)$ with M_0 as the initial marking. Applying linear algebra based on equation (1), the reachability from the marking M to M' can be checked. Moreover, a full graph of all markings and possible transition from one marking to another are described by state-space analysis. As the number of vertices and edges of the state-space graph increase dramatically with regards to the number of places and transitions, state-space analysis is limited to a small PN network.

The colored PN (CPN) is an extension of PN, in the sense that CPN has different types of token marked with color (Jensen, 2003). The transition fires separately with respect to each kind of token. The arc expressions built from operators and functions are further used to decide the transition behavior of different colors. Unlike conventional PN, only backward-compatible CPN is able to be analyzed with available mathematical models. Other CPNs must rely on simulation with statical analysis to reveal the visiting frequencies of tokens at places and availability of a marking state.

Another extension is the stochastic PN which adds a time delay at each transition where the firing rate is determined by a random variable. The state-space analysis is performed by probabilistic inference in a Markov chain. Generalized stochastic PN extended the stochastic PN with the possibility of immediate transition to forward to token without any time delay. In this paper, the generalized framework with colored time PN (CTPN) is utilized to deal with the time delay of both non-deterministic and deterministic variables. CPNTools (Jensen, 2003) are used to create CTPN and perform the analysis.

3.2 Fault Tolerance with Cooperative Agents

In the proposed system, it is supposed that there is a pool of agents $A = \{a_1, a_2, ..., a_i, ..., a_N\}$, where N is the number of agents. A sequence of tasks $T = \{T_1, T_2, ..., T_i ..., T_t\}$ are assigned to the set of agents A to be processed one by one where t is the time index. For a complicated task, it is convenient to separate each task into a number of subtasks $T_i = \{t_{i1}, t_{i2}, ..., t_{iG}\}$, where iG is the number of subtasks of the task T_i. The subtasks are categorized into independent subtasks and dependent subtasks. The independent subtasks can be processed independently and concurrently by different agents. Meanwhile, the dependent subtasks, like $t_{ij} \rightarrow t_{ik}$, requires that the subtask t_{ij} must be completed before t_{ik}.

For fault tolerance, the definition of peer agent is introduced. The two agents a_i and a_j are peer agents if they are able to solve the common subtask. Thanks to the availability of multiple peer agents, once an agent fails to do a subtask, other peer agents with the similar function are used to continue the tasks. In this model, the agents $\{a_1, a_2, a_3\}$ and the subtasks $\{t_1, t_2, t_3\}$ are available. Agent a_1 is assigned with the subtasks $\{t_1, t_2\}$, agent a_2 with the subtasks $\{t_2, t_3\}$, and a_3 with the subtasks $\{t_1, t_3\}$ respectively.

3.3 CPN Models of Agents for Fault Tolerance Analysis of Centralized Systems

In this model, all subtasks are managed by a *supervision* module (SM) (Figure 1) with the tokens $t(1)$, $t(2)$, and $t(3)$. Once the SM receives a request to perform a set of subtasks from a *task management* module (TMM), it will send the subtasks to available agents according to the description provided in Section 3.2. If one of the agents fails to complete the subtask, the SM will assign the subtask to another peer agent. Meanwhile, the SM will collect all finished subtasks and send them to the TMM. For each subtask tree, the TMM is respectively designed. In this paper, the design of the TMM for a combined subtask tree is introduced as an example.

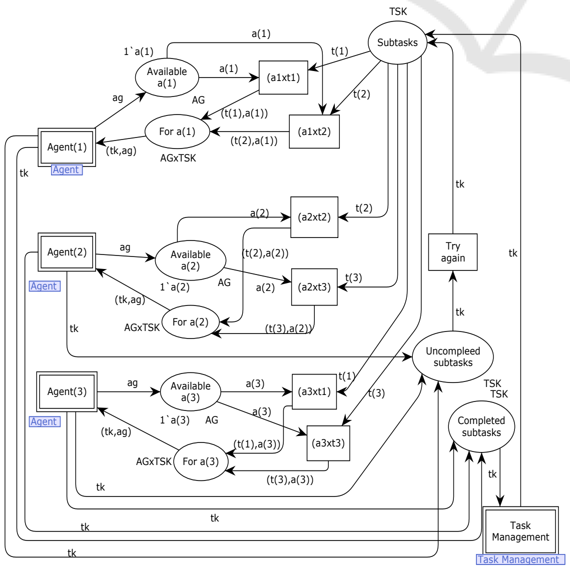

Figure 1: Design of a *supervision* module in centralized systems.

The agents a_1, a_2, and a_3 (Figure 2) share the same structure. It can be noted that this structure models the *agent* modules (AMs) in the SM. If there is an available agent, that agent will receive a subtask from the SM. The approximate processing time to finish the subtask is assumed to be τ_p (value *proctime* in *agent*). Meanwhile, the fail event of an agent is modeled by a random variable of the exponential distribution with the rate λ_e (value *failrate* in *agent*). Once the agent fails, the failed subtask will be returned to the SM and it requires a time τ_r (value *fixtime* in *agent*) to recover the normal activity of the agent. The tokens $a(1)$, $a(2)$, and $a(3)$ are used to reveal the availability of the agents. Obviously, the performance of fault tolerance mechanism will depend on the tuples $(\tau_p, \lambda_e, \tau_r)$ and the further analyses are presented in Section 4.

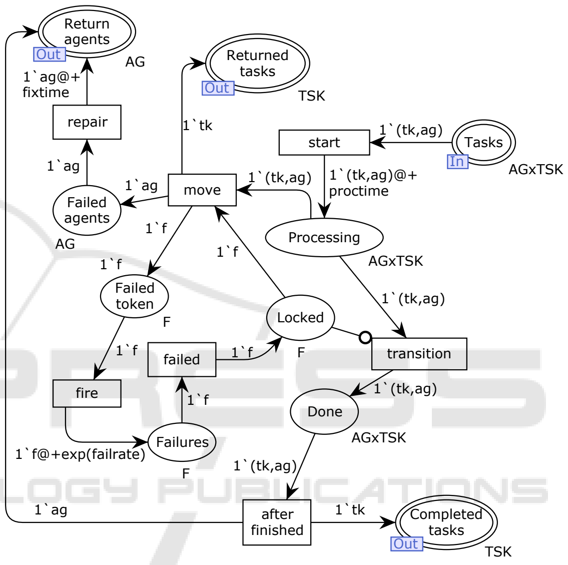

Figure 2: Design of an *agent* module in centralized systems.

3.4 CPN Models of Agents for Fault Tolerance Analysis of Distributed Systems

For a distributed system, in the SM (Figure 3), each agent will share the information of which subtasks have been completed and which subtasks are unfinished by broadcasting messages for any updates. Whenever an agent decides to do a subtask, it will send a broadcast message to the other agents. If there are no conflicts, the agent will start the subtask and remove the subtask from an *uncompleted subtasks* (USs) place to process. Once the subtask is accomplished, the agent broadcasts a message to inform the others to update and append the list in a *completed subtasks* (CSs) place with the new completed subtask. The agent, while doing a task, is being checked for its availability and will frequently send an *"alive"* message. All agents will be noticed of the failures of an agent and the information is updated in the USs place. In our system, it is assumed that the copy of the list USs and CSs are available in the memory of every agents. The design of an AM (Figure 4) is developed

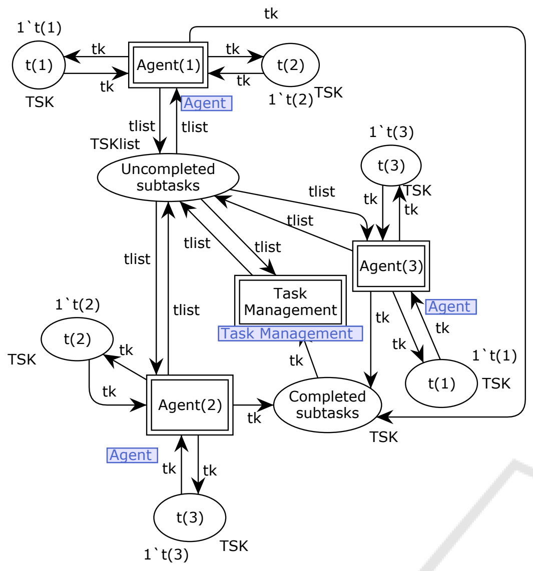

Figure 3: Design of a *supervision* module in distributed systems.

from the centralized systems. Yet, the module is advanced with the ability to choose and process the subtasks from the USs by itself. Besides, the time delay with a variable τ_b (value *broadcast* in *agent*) is introduced for the broadcast process.

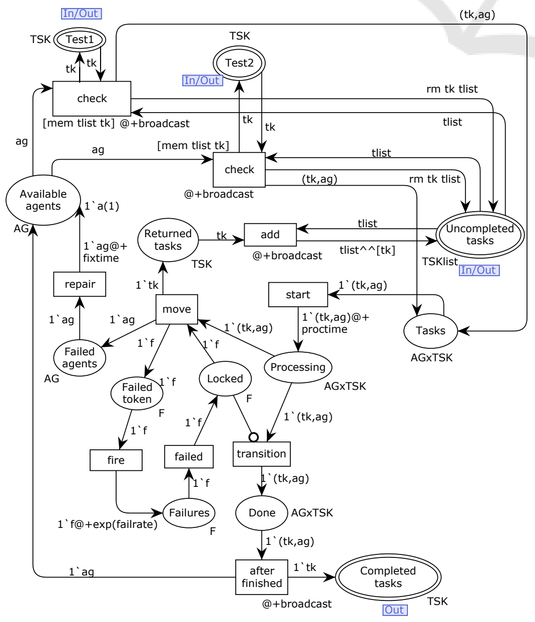

Figure 4: Design of an *agent* module in distributed systems.

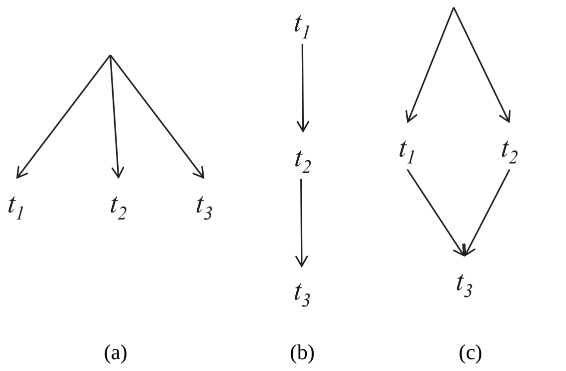

Figure 5: Subtask trees. (a) Three subtasks t_1, t_2, and t_3 are independent. (b) The order of processing subtasks is t_1, t_2, and then t_3. (c) Two subtasks t_1 and t_2 must be completed before the subtask t_3 is processed.

4 RESULTS

As it is depicted in Figure 5, the different subtask trees including independent subtasks, dependent subtasks, and the combination of independent and dependent subtasks are used for testing CPN models for both centralized and distributed systems.

4.1 Fault Tolerance Analysis of Centralized Systems

The TMM (Figure 6) is utilized to handle the complicated tasks requiring the subtask t_3 to follow the accomplishment of the subtask t_1 and t_2. The TMMs are designed similarly for the independent subtasks and dependent subtasks. One hundred tasks are generated to test the fault tolerance ability of the cooperative agents. Each task consists of three concurrent subtasks t_1, t_2, and t_3. Once all the subtasks are completed, the TMM checks the results to confirm the accomplishment of the task and requires a new task for the set of agents. The performance of fault tolerance is evaluated by the average processing time needed to process each task. The processing time of each agent for a subtask is assumed $\tau_p = 20$. The longer recovering time τ_r needed of each agent after a fail appears, the more time the whole system will need to finish a task in general. Therefore, three cases $\tau_r/\tau_p = 0.5$, $\tau_r/\tau_p = 1$, and $\tau_r/\tau_p = 1.5$ were evaluated respectively. The fail rate varies from 10 to 100, ($10 \leq \lambda_e \leq 100$).

The results (Figure 7(a)) reflect what is expected that the processing time increases when τ_r/τ_p increases. Meanwhile, the processing time decreases when the intervals between two fail events are pro-

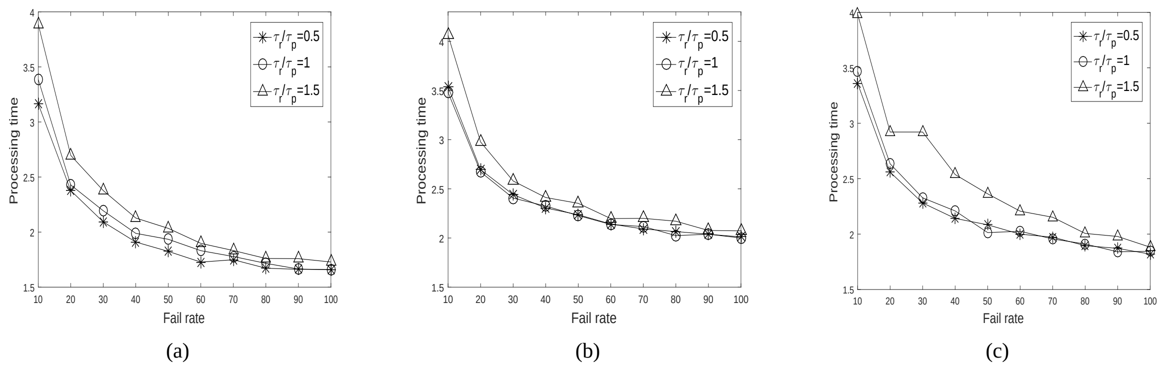

Figure 7: The average processing time to finish a task with respect to τ_r/τ_p in centralized systems. (a) Independent subtasks, (b) Dependent subtasks, and (c) Mixtures of independent and dependent subtasks.

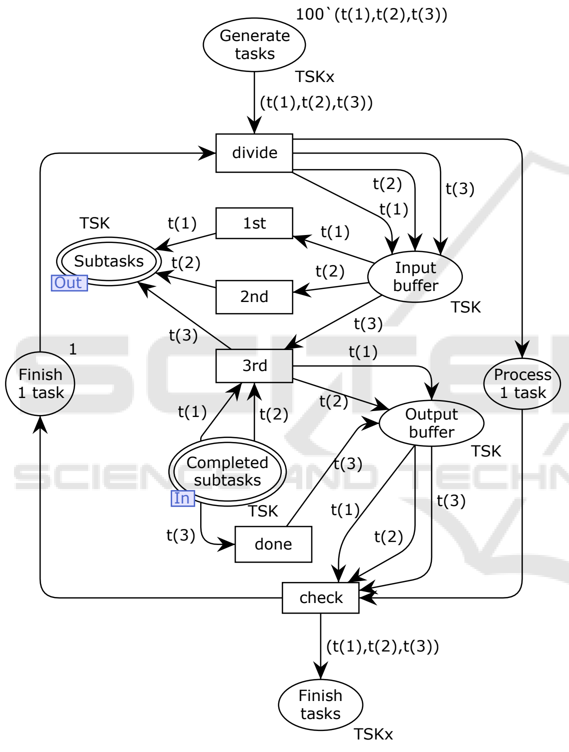

Figure 6: Design of *task management* module for mixtures of independent and dependent subtasks in centralized systems.

longed. Logarithm scale is utilized to present the processing time as it is very high with respect to the low fail rate λ_e. For dependent subtasks, the same configurations of parameters τ_p, τ_r, and λ_e are used to perform the simulation results. The similar results (Figure 7(b)) are obtained that the average processing time increases according to rising τ_r/τ_p and the decreasing interval between two fail events. However, there are not many differences between $\tau_r/\tau_p = 1$ and $\tau_r/\tau_p = 1.5$. This may be due to the similar probability of one of all three agents available to take care of a subtask each time in the two cases. In the case of combined subtask tree, the similar performance analysis of the processing time to complete all the tasks with regards the ratio τ_r/τ_p is acquired as given in Figure 7(c). In conclusion, the dependency of different subtask within a task will require more time to complete the task. Besides, the processing time increases with respect to the increasing ratio τ_r/τ_p in all the above studies.

4.2 Fault Tolerance Analysis of Distributed Systems

As there are not many differences in the design of the TMM between centralized and distributed systems, an example of the TMM to deal with the complicated tasks consisting of independent and dependent tasks is shown in Figure 8. The configuration parameters of τ_p, τ_r, and λ_e are similar to those used in Section 4.1, thus $\tau_b = 2$ is used in the following experiments.

The evaluation of fault tolerance performance of distributed agents to process the independent subtasks is given in Figure 9(a). Similarly, the average processing time to complete a task is proportional to the ratio τ_r/τ_p. However, due to the delay of broadcasting messages, in overall, the processing time in distributed system is higher than that in centralized system.

The results (Figure 9(b)) show the fault tolerance performance of the distributed system to deal with the tasks of dependent subtasks. However, the differences of the processing time with regards to the ratio τ_r/τ_p in the cases 0.5, 1.0, and 1.5 are not significant. Because all agents take time to deliver the broadcast messages, a failed agent may be recovered before a new subtask arrives. Therefore, the distributed system is less dependent on the recovering time rather than the centralized system. However, more experiments must be performed to validate this conclusion. The

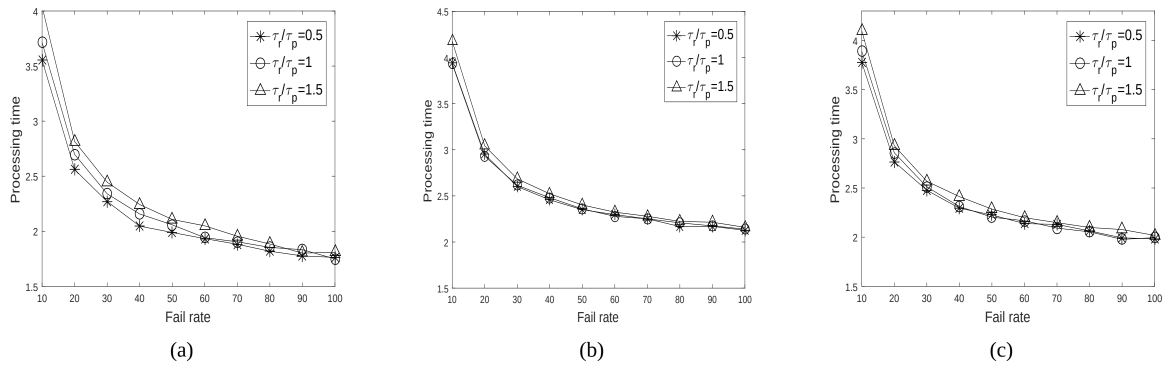

Figure 9: The average processing time to finish a task with respect to τ_r/τ_p in distributed systems systems. (a) Independent subtasks, (b) Dependent subtasks, and (c) Mixtures of independent and dependent subtasks.

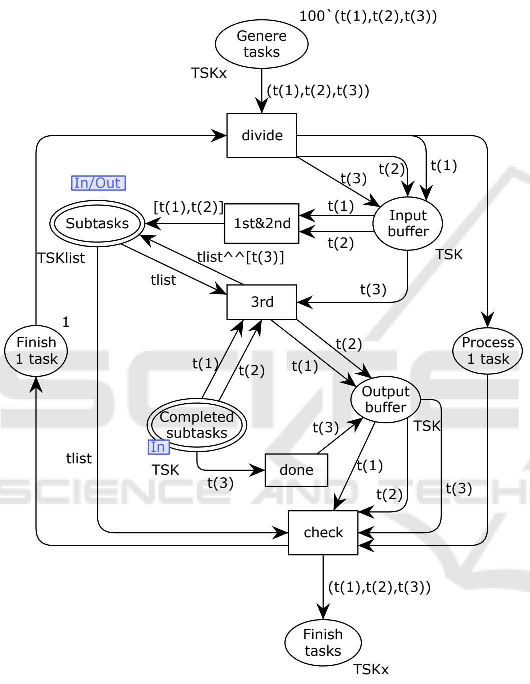

Figure 8: Design of *task management* module for mixtures of independent and dependent subtasks in distributed systems.

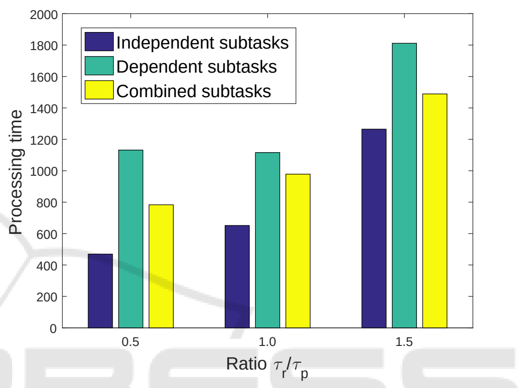

Figure 10: Overall evaluation of distributed systems.

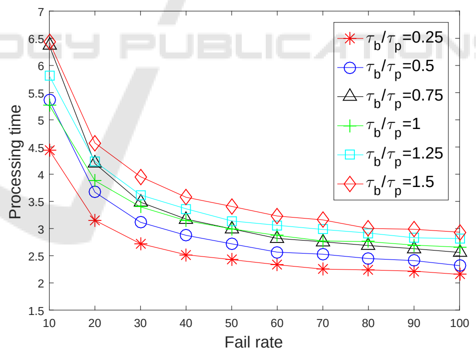

Figure 11: Correlation of time to deliver broadcast message with the fault tolerance performance.

last experimental results (Figure 9(c)) present how the distributed systems process the tasks of mixed independent and dependent subtasks. There are similar conclusions to those presented in previous simulations.

Finally, an overall evaluation (Figure 10) is used to assess the fault tolerance ability of cooperative distributed agents. It can be seen that the appearances of more dependent subtasks will prolong the whole process to perform the task. Besides, the time required by communication protocol used in distributed protocol also affected the fault tolerance performance. In order to investigate further on this concern, the processing time is evaluated with regards to the time needed to broadcast messages among agents. Similarly, the fixed ratio is in a range $\tau_r/\tau_p = 1$ and $0.25 \leq \tau_b/\tau_p \leq 1.5$. The results in Figure 11 clearly show the effects of communication time with fault tolerance ability of the system.

5 CONCLUSIONS

In this paper, the formulation of CTPN has been introduced for the fault tolerance analysis for a group of agents cooperating to solve complicated tasks. The fault tolerance is performed by replacing a failed agent to continue the unfinished tasks. The CTPN models have been designed to validate this method for both centralized and distributed approaches. In simulations, different trees of independent and dependent subtasks were evaluated. From the experimental results, the analysis has shown the correlation of the processing time to finish a complicated task with the failure rates of an agent. Besides, the experiments revealed that for distributed agents, the communication protocol also played an important role on the fault tolerance success of the whole system.

In future, the proposed approach will be extended to deal with a more complicated tree of subtasks. Other fault tolerance mechanisms for the group of agents are also concerned. Furthermore, experiments with real robots will be planned to compare the fault tolerance analysis provided by PN with that acquired from realistic setup.

ACKNOWLEDGEMENTS

The research leading to the presented results has been undertaken within the research profile DPAC - Dependable Platforms for Autonomous Systems and Control project, funded by the Swedish Knowledge Foundation.

REFERENCES

Acharya, S., Upadhyay, P. D., and Dutta, A. (2014). Fault tolerance multi agent co-ordination: A petri net based approach. In *Proceedings on International Conference on Recent Advances and future Trends in Information Technology*.

Avižienis, A., Laprie, J., Randell, B., and Landwehr, C. (2004). Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1).

Bernardi, S., Merseguer, J., and Petriu, D. C. (2013). *Model-Driven Dependability Assessment of Software Systems*. SPRINGER.

Graf, B. and Gele, M. (2001). Dependable interation with an intelligent home care robot. In *In proceedings of ICRA - Workshop on Technical Challenge for Dependable Robots in Human Environments*. IEEE.

Guiochet, J. (2015). Trusting robots : Contributions to dependable autonomous collaborative robotic systems. Technical report, LAAS - Laboratoire d'analyse et d'architecture des systmes [Toulouse].

Haddad, J. and Haddad, S. (2004). A fault-tolerant communication mechanism for cooperative robots. *International Journal of Production Research*, 42(14):2793–2808.

Jensen, K. (2003). *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*. Springer Verlag.

Joaquin, L., Diego, P., and Eduardo, Z. (2011). A framework for building mobile single and multi-robot applications. *Robotics and Autonomous Systems*, 59(3-4):151–162.

Kim, G. and Chung, W. (2007). Navigation behavior selection using generalized stochastic petri nets for a service robot. *IEEE Transactions on Systems, Man, and Cybernetics part C: Applicactions and Reviews*, 37(4).

Kohlík, M. (2009). Dependability models based on petri nets and markov chains. In *Information Science and Computer Engineering, 1st Class, Full-time study*.

Malhotra, M. and Trivedi, K. (1995). Dependability modeling using Petri-nets. *IEEE Transactions on Reliability*, 44(3).

Miyagi, P. and Riascos, L. (2004). Modeling and analysis of fault-tolerant systems for machining operations based on petri nets. *Control Engineering Practice*, 14(4):397–408.

Moon, C. and Chung, W. (2012). Coordination of multiple control schemes for mobile robot navigation on the basis of the generalized stochastic petri-nets. *Advanced Robotics*, 26(5-6):581–603.

Mustapic, G., Anderson, J., Norstrom, C., and Wall, A. (2004). *A Dependable Open Platform for Industrial Robotics - A Case Study*. SPRINGER.

Parhami, B. (1994). A multi-level view of dependable computing. *Computers and Electrical Engineering*, 20(4):347–368.

Troubitsyna, E. and Javed, K. (2014). Towards systematic design of adaptive fault tolerant systems. In *ADAPTIVE 2014, The Sixth International Conference on Adaptive and Self-Adaptive Systems and Applications*. IARIA.

Yen, H. (2006). Introduction to petri net theory, in recent advances in formal languages and applications. In *Studies in Computational Intelligence*, volume 25, pages 343–373.