INL/CON-17-41438
PREPRINT

TASK ALLOCATION IN GEO-DISTRIBUTATED CYBER-PHYSICAL SYSTEMS

10th International Topical Meeting on Nuclear Plant Instrumentation, Control and Human Machine Interface Technologies (NPIC-HMIT)

Rachit Aggarwal and Carol Smidts

March 2017

The INL is a
U.S. Department of Energy
National Laboratory
operated by
Battelle Energy Alliance

TASK ALLOCATION IN GEO-DISTRIBUTED CYBER-PHYSICAL SYSTEMS

Rachit Aggarwal and Carol Smidts*
Department of Mechanical and Aerospace Engineering
The Ohio State University
201 W 19th Ave, Columbus, OH – 43210
aggarwal.112@osu.edu; smidts.1@osu.edu

ABSTRACT

This paper studies the task allocation algorithm for a distributed test facility (DTF), which aims to assemble geo-distributed cyber (software) and physical (hardware in the loop) components into a prototype cyber-physical system (CPS). This allows low cost testing on an early conceptual prototype (ECP) of the ultimate CPS (UCPS) to be developed. The DTF provides an instrumentation interface for carrying out reliability experiments remotely such as fault propagation analysis and in-situ testing of hardware and software components in a simulated environment. Unfortunately, the geo-distribution introduces an overhead that is not inherent to the UCPS, i.e. a significant time delay in communication that threatens the stability of the ECP and is not an appropriate representation of the behavior of the UCPS. This can be mitigated by implementing a task allocation algorithm to find a suitable configuration and assign the software components to appropriate computational locations, dynamically. This would allow the ECP to operate more efficiently with less probability of being unstable due to the delays introduced by geo-distribution. The task allocation algorithm proposed in this work uses a Monte Carlo approach along with Dynamic Programming to identify the optimal network configuration to keep the time delays to a minimum.

Key Words: task allocation, cyber-physical systems, optimization, dynamic programming

1 INTRODUCTION

A distributed test facility (DTF) is a facility which aims to assemble geo-distributed cyber (software) and physical (hardware in the loop) components into a prototype cyber-physical system (CPS). It provides an instrumentation interface for carrying out reliability experiments remotely such as fault propagation analysis and in-situ testing of hardware and software components in a simulated environment. Our research on DTF [1] arises from the need for such a facility towards the conversion of instrumentation and control systems in the nuclear power industry. Digital instrumentation and control systems are gradually replacing their analog counterparts in current nuclear power plants and are the new standard in future designs in both safety-critical and non-safety-critical systems.

The DTF is comprised of hardware (HW) and software (SW) components which are dispersed physically. The hardware components are immovable physical processes such as thermal hydraulic loops and associated sensors, filters, actuators, controllers, etc. and software components can be simulations of physical processes, digital controllers, etc. Unfortunately, the assembly of the geo-distributed software and hardware components in the DTF introduces artificial delays and poses a big challenge in keeping the

---

* **Rachit Aggarwal** is a Master of Science student at the Department of Mechanical and Aerospace Engineering at The Ohio State University. His research interests are controls and optimization of large-scale interconnected systems.

Carol. S. Smidts is the a Director of Reliability and Risk Laboratory and a Professor in Department of Mechanical and Aerospace Engineering at The Ohio State University.

system stable and running over a network. The geo-distribution in the assembly components and sub-systems introduces an overhead that is not inherent to the UCPS, i.e. it adds a significant time delay in communication that could compromise the stability of the ECP and result in an inappropriate representation of the behavior of the UCPS. This can be mitigated by implementing a task allocation algorithm to estimate an optimal configuration and assign the software components to appropriate computational locations in real-time. This would allow the ECP to operate as long as possible i.e. the probability of the ECP becoming unstable due to the delays (fictitiously introduced by the geo-distribution) is minimum.

Many of the task allocation problems [2]-[6] address the assignment problem with an objective of minimizing the net cost of resources used. The problem of task allocation in the distributed test facility is a network optimization problem with an objective to communicate (transfer data packets) in a geo-distributed network in the least possible time such that the system under test, *i.e.* ECP, remains stable. Most of the network optimization problems are formulated as minimum cost or maximum flow problems. References [7] and [8] discuss optimization algorithms where the cost of each arc in a multi-source, multi-sink network is a stochastic variable. These algorithms only address the problem of finding an optimal path through the set of connected nodes. However the DTF poses a challenge to the movement of the software components in the network as the route in the geo-distributed network (e.g. WAN) is already nearly optimal.

This paper presents a novel approach of solving the network optimization problem with uncertainties in arc/edge cost (time delay). This simple algorithm, coupled with its parallelization capability, allows it to be implemented in real-time.

2 TASK ALLOCATION

The task allocation algorithm described in this paper estimates an appropriate configuration of the ECP such that each software component is assigned to one of the available computational facilities while taking into account the amount of data being transferred in the network (flow variable), the time (cost) taken for transition of a software component, computational resources (cost) and the uncertainty in the delay time (cost) in the network. A software component can be installed at several locations depending on the computational requirements and additional legal restrictions. A computational facility can have more than one software component installed. A software component can be invoked at a particular location by sending a checkpoint file to that location.

The section is divided further into four subsections. Section 2.1 lists some definitions associated with the graph theory, which will be extensively used in the algorithm. Sections 2.2 and 2.3 describe the method in detail and then finally, in Section 2.4, two case studies are presented.

2.1 Definitions

Graph: A graph is a collection of *vertices* and *edges* which connect the vertices. It is represented as $G(V, E)$ where V is a set of vertices or nodes and E is a set of edges or arcs. In an *undirected graph*, an edge is an unordered pair of vertices whereas in a *directed graph*, an edge is an ordered pair of vertices. A graph can also be a weighted graph where $W(E_{G_N})$ denote the edge weights (also known as the cost of traversing the edge).

Subgraph: A subgraph of a graph G is another graph formed from a subset of the vertices and edges of G. The vertex subset must include all endpoints of the edge subset, but may also include additional vertices. A spanning subgraph is one that includes all vertices of the graph; an induced subgraph is one that includes all the edges whose endpoints belong to the vertex subset.

Graph Isomorphism: An isomorphism of graphs $G(V_G, E_G)$ and $H(V_H, E_H)$ is a *bijection* between the vertex sets of G and H, $f: V_G \rightarrow V_H$ such that any two vertices u and v of G are adjacent (*i.e.* connected by an edge) in G if and only if $f(u)$ and $f(v)$ are adjacent in H, *i.e.* $(u, v) \in E_G \Leftrightarrow \big(f(u), f(v)\big) \in E_H$.

Subgraph Isomorphism: For graphs $G(V_G, E_G)$ and $H(V_H, E_H)$, a subgraph isomorphism from G to H is a function $f: V_G \rightarrow V_H$ such that if $(u, v) \in E_G \Rightarrow \big(f(u), f(v)\big) \in E_H$.

2.2 Principle

The principle of task allocation is to find an optimal configuration of SW component assignment to the SW locations such that the time taken for the exchange of information between the hardware nodes (i.e. locations where physical processes exist) and the computational facilities (i.e. where the software components are executed) is minimal. This depends on the predicted values of the time delay in each network arc which is a stochastic variable modeled using time series analysis. The optimal configuration has an associated probability of having time delays below the critical time delay. The critical time delay is the time delay beyond which the system becomes unstable. This configuration is subject to the constraint that there are enough computational resources for all the software components assigned to a particular location. The next step is to check whether the software can be moved (transfer the checkpoint file) to a new location in a time lower than the critical time delay. This optimization approach facilitates estimation of the optimal software assignment, dynamically.

2.3 Method

Before the task allocation is performed, the *first stage* is to define the system and its constraints. The following is the list of steps in this stage.

- *Network layout*: A directed graph $G_N(V, E)$ of hardware locations and software locations as vertices and their network connectivity as edges is defined.
- *System layout*: The hardware (HW) and software (SW) components for the UCPS along with their communication link (the direction of the data between any two specified HW or SW components) is identified. A directed graph $G_S(V, E)$ of the UCPS with hardware and software components as vertices and their communication link as edges is defined.
- *Hardware map*: As each HW component is associated with only one HW location, a one-to-one map of component-location associations is defined *i.e.* $f_{HW}: V_{G_S,HW} \rightarrow V_{G_N,HW}$
- *Software constraints maps*: SW being a virtual component can be executed at any SW location which meets the requirements such as licensing, computational resources and memory. A one-to-many map of component-location associations is defined. $f_{SW}: V_{G_S,SW} \rightarrow V_{G_N,SW}$

For task allocation, the following are the steps in this *second stage*.

- *Feasible network configuration*: Based on f_{HW} and f_{SW}, a list of permutations is constructed. A set S_{FNC} of the permutations that are subgraph isomorphic to the network layout graph G_N are the *feasible network configurations* for implementing the UCPS.
- *Network reduction*: Some of the vertices and edges are never a part of any of the feasible network configuration subgraphs. Hence those edges can be dropped out to reduce the computational load in further steps.
- *Direct path definition*: Some vertices may have a limited direct connectivity for it to be qualified for inclusion in the network layout. This affects the performance during transitions between the feasible configurations. So, in this step, a set of additional network edges $E_{G_N,direct}$ is defined for the SW locations (vertices) which are not connected in the reduced network.

- *Transition time cost*: A list of all edges associated with each SW transition from $i\ to\ j$ where $i \neq j;\ \ i,j \in S_{FNC}$ feasible configuration is built. Transition time cost is defined as $C_{ii}(W(E_{G_N}))$, where $W(E_{G_N})$ is the set of edge weights.
- *Definition of one simulation iteration cost*: In the ECP deployment, for each time step of the simulation, the data needs to be moved from one vertex to another. Based on how the simulation iteration is defined, a cost function based on the time delays of the edges involved is defined. Let this cost be defined as $C_{ii}(W(E_{G_N}))$. For example, when the i^{th} feasible configuration is a single cyclic loop, the cost function can be defined as sum of the all the time delays due to network communication on each edge and computational time on each node. This can also be interpreted as the cost to stay in the i^{th} feasible configuration.
- *Monte Carlo Trial*: The following steps are performed for each random trial.
    - *Time delay sampling*: Based on the time series model (obtained a priori to the task allocation), randomized network time delay samples are generated. These samples form the set $W(E_{G_N})$.
    - *Cost matrix*: A cost matrix is generated for each time step, *k*. $C_{ii}^{k}(W(E_{G_N}))$ is the cost for staying in the same feasible configuration and $C_{ij}^{k}(W(E_{G_N}))$. is the cost for transition time to switch from configuration $i\ to\ j$.
    - *Cost function*: It is the actual cost for a switch to happen. It is defined as the sum of the cost to stay in the feasible configuration and the cost of the switch, *i.e.* $C_{ij}^{k} = C_{ii}^{k}(W(E_{G_N})) + C_{ij}^{k}(W(E_{G_N}))$.
    - *Dynamic Programming* (DP): In this step Bellman's principle [9] is used for dynamic programming. The equation below is a modified Bellman's optimality equation which accounts simulation time at each time step, $C_{jj}^{k}(\cdot)$, and the switching time, $C_{ij}^{k}(\cdot)$.

      $$V_i(k) = \min_j \begin{cases} C_{ij}^{k}(\cdot) + V_j(k+1) & \text{if } i = j \\ C_{ij}^{k}(\cdot) + C_{jj}^{k}(\cdot) + V_j(k+1) & \text{otherwise} \end{cases}$$, where $V(\cdot)$ is the value function.

    - *Final path*: The output of DP is the optimal policy consisting of feasible configuration *j* as a function on time *t* while accounting the transitions. This optimal policy will result in the least network time overhead for all the time steps in planning horizon.

The result of this task allocation problem is represented using a probability plot of the system being stable in a particular configuration.

2.4 Case Study

We present two case studies to illustrate the method described above. These are the assumptions for the following case studies.

- Only 1 SW component $(SW_i, i \in \{1, \dots, n(SW)\})$ can be executed at only 1 SW location $(SWL_j, j \in \{1, \dots, n(SWL)\})$ at a time.
- All SW locations support all SW components.
- Packet size is constant for simulation messages and checkpoint files.
- Additional software transition paths may be defined to facilitate faster software transitions.
- All software locations fulfill the computational requirements for the software components.

- The order of the computational time for executions at each node is considered negligible compared to the network time delay.

2.4.1 Case study 1 – two possible configurations

In this example, the network layout and the system layout are shown in Figure 1(a) and (b), respectively. Given the above assumptions for HW and SW mapping, a set of feasible configurations is found as shown in Table I and Figure 1(c). Figure 2(a) shows the reduced network graph which is a subgraph of the network graph as in Figure 1(a). There are two possible transitions between the feasible configurations – (a) E_1 to E_2 and (b) E_2 to E_1. E_1 to E_2 involves movement of SW2 from node 3 to node 4 and SW1 from node 4 to node 5, whereas E_2 to E_1 involves movement of SW2 from node 4 to node 3 and SW1 from node 5 to node 4. Since edge (4,3) and (5,4) does not exist in E_{G_N}, the transition via longer route could be costly. Hence, we attempt introduce a direct return path to determine the effect of such a modification. This case study compares two sub-cases – one with no direct return paths for transition (Figure 2(a)) and another with direct return paths for transitions (Figure 2(b)). Table II lists the various time delay distributions assumed for this case.

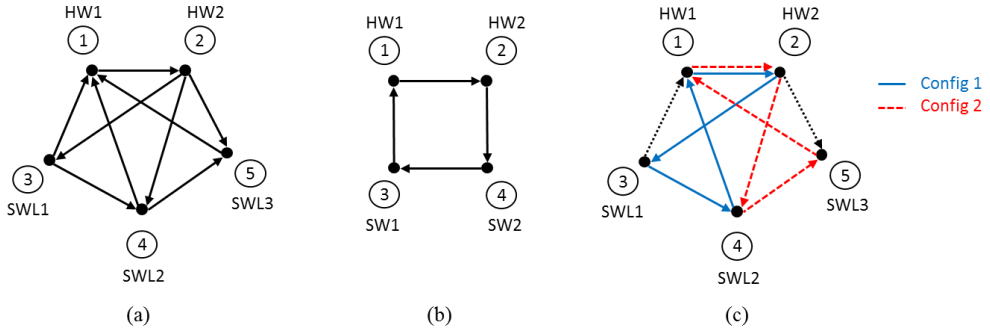

Figure 1. (a) Network Configuration: Network Layout for Early Conceptual Prototype (ECP) (b) Actual System Configuration (c) Feasible Configurations

Table I. Determination of Feasible Configurations

System Layout (Figure 1(b)) E_{G_S}	Feasible configurations (in Figure 1(a))	
	Config 1: $E_1 \subset E_{G_N}$	Config 2: $E_2 \subset E_{G_N}$
(HW1, HW2) or (1,2)	(1,2)	(1,2)
(HW2, SW2) or (2,4)	(2,3)	(2,4)
(SW2, SW1) or (4,3)	(3,4)	(4,5)
(SW1, HW1) or (3,1)	(4,1)	(5,1)

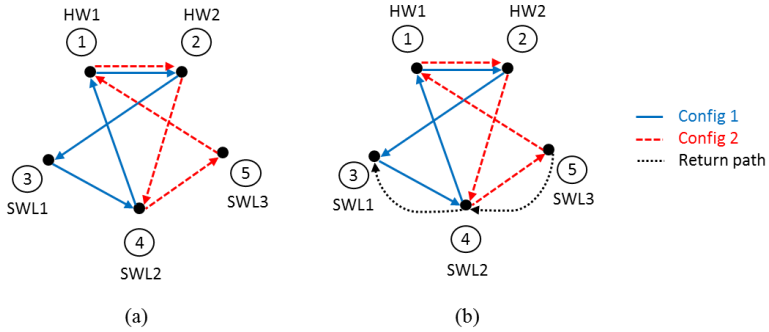

Figure 2. (a) Case without a dedicated return path (also a reduced network) (b) Case with dedicated return path

Table II. Time Delay Distributions

Edges	Time delay distribution [msec]	Edges	Time delay distribution [msec]
(1, 2)	$N(100,3)$	(4, 5)	$N(10,3)$
(2, 3)	$N(\mu_1(t),3)$	(5, 1)	$N(103,3)$
(2, 4)	$N(\mu_2(t),3)$	(4, 3)	$N(10,3)$
(3, 4)	$N(10,3)$	(5, 4)	$N(10,3)$
(4, 1)	$N(103,3)$		
where $N(a,b)$ is the Normal distribution with mean a and variance b; $\mu_1(t) = 0.5t + 110$; $\mu_2(t) = 100 + 4\lvert t - 10\rvert$; $t = k\Delta t$; $\Delta t = 0.04\ sec$			

Figure 3 shows a sample Monte Carlo trial. Figure 3(a) shows the simulation times for the ECP for the respective configurations. Based on the minimum of the simulation times at each time step a recommended configuration is shown in Figure 3(b). Figure 4 shows the optimal path determined using dynamic programming in each trial. Figure 5 shows the probability of the system (ECP) being optimal in the feasible configurations as a function of time. This process helps in making a decision for choosing the feasible configurations in planning horizon.

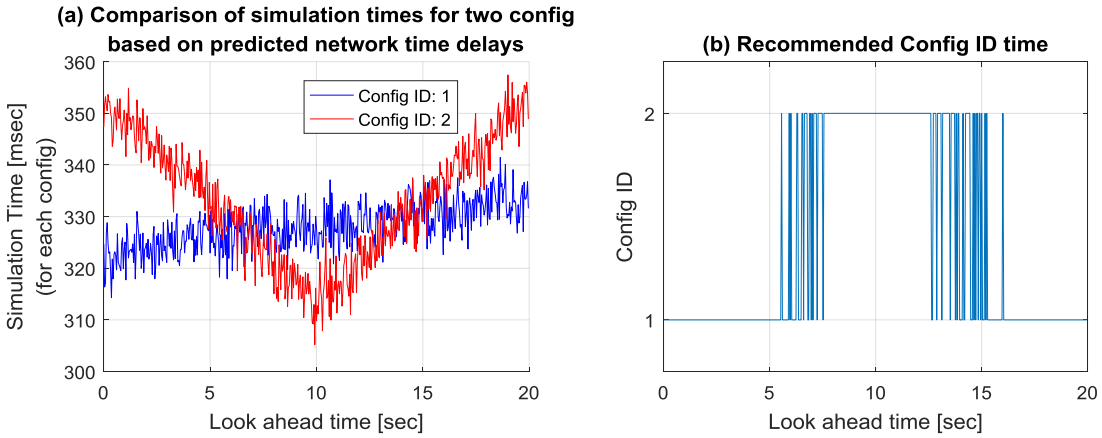

Figure 3. One sample Monte Carlo trial - (a) simulation times for ECP, (b) recommended configuration

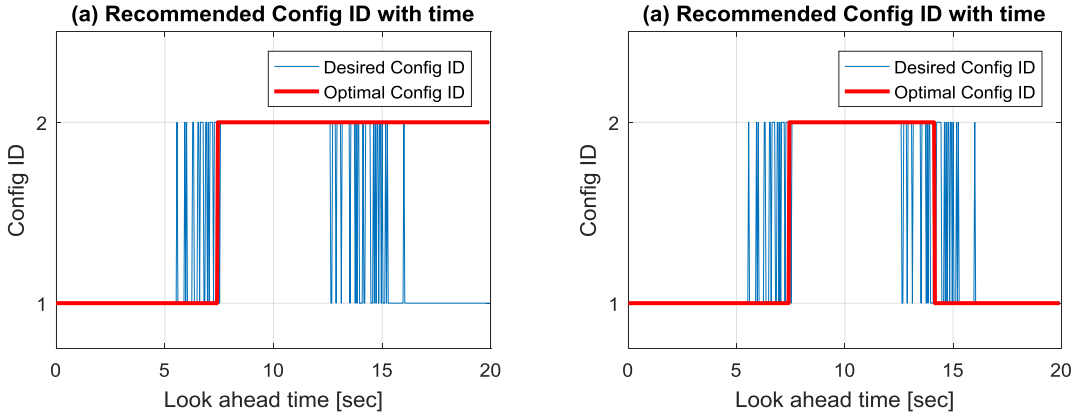

Figure 4. Determination of the optimal path in one sample Monte Carlo trial – (a) Case without a dedicated return path (b) Case with dedicated return path

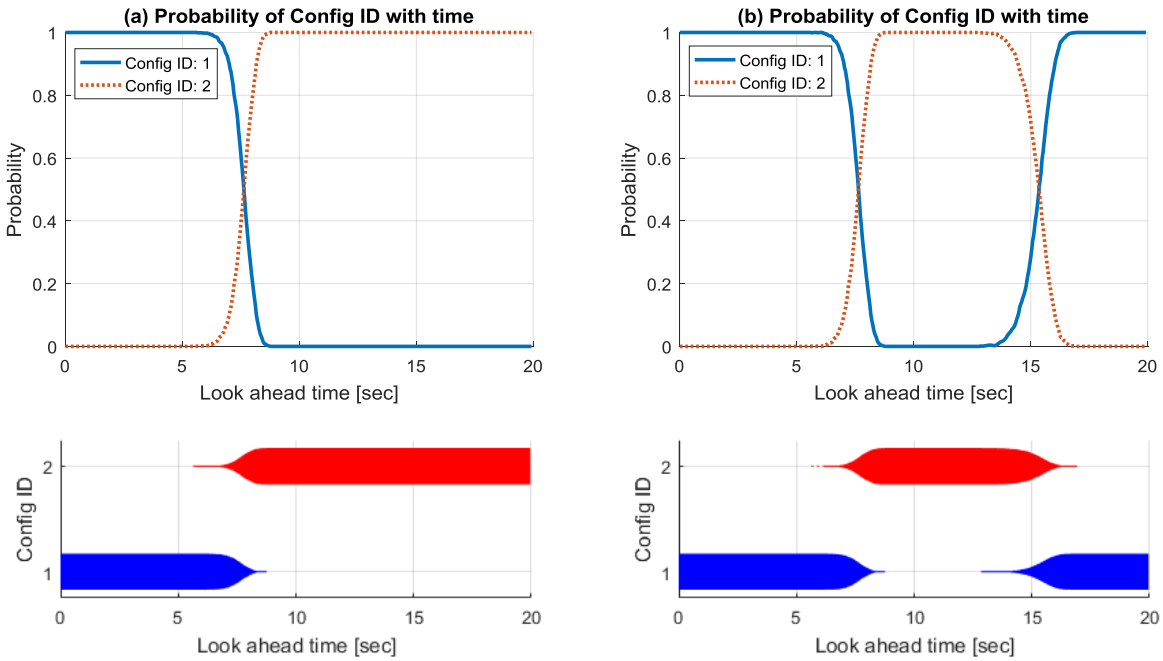

Figure 5. Probability of the system (ECP) being in the available configurations as a function of time; Lower plot shows the density of the Monte Carlo particles in a particular configuration - (a) Case without a dedicated return path (b) Case with dedicated return path

2.4.2 Case study 2 – three possible configurations

This case is similar to the previous example. One additional edge is introduced in the network layout as shown in Figure 6(a) with the same system layout as in Figure 1(b). Similar to the previous case, a set of feasible configurations is found as shown in Figure 6(c). This case study also compares two sub-cases – one with no direct return paths for transition (Figure 7(a)) and other with direct return paths for transitions (Figure 7(b)). Table III lists the various time delay distributions assumed for this case.

Figure 8 shows a sample Monte Carlo trial where a desired configuration is found based on the minimum of the simulation times at each time step. Figure 9 shows the optimal path determined using dynamic programming in each trial and Figure 10 shows the probability of the system (ECP) being optimal in the feasible configuration as a function of time. It can be noticed that in the case without the return path, it may not be optimal to switch at all because of the long switch times involved. Hence, it is recommended to have direct switching paths to have the best optimal configuration at all the times.

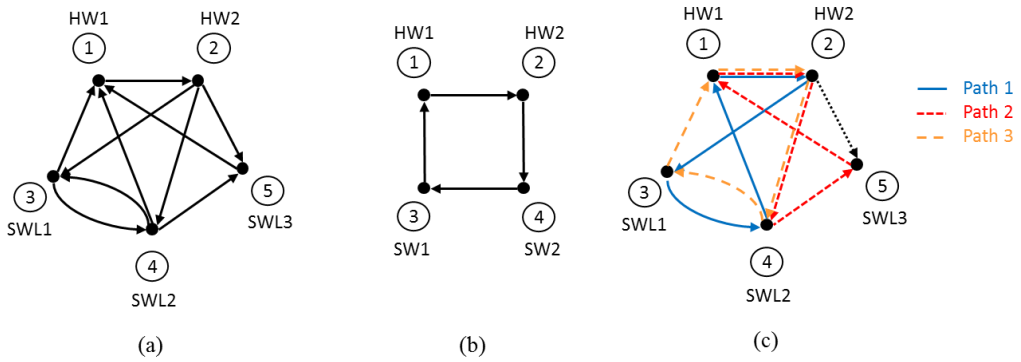

Figure 6. (a) Network Configuration: Network Layout for Early Conceptual Prototype (ECP) (b) Actual System Configuration (c) Feasible Configurations

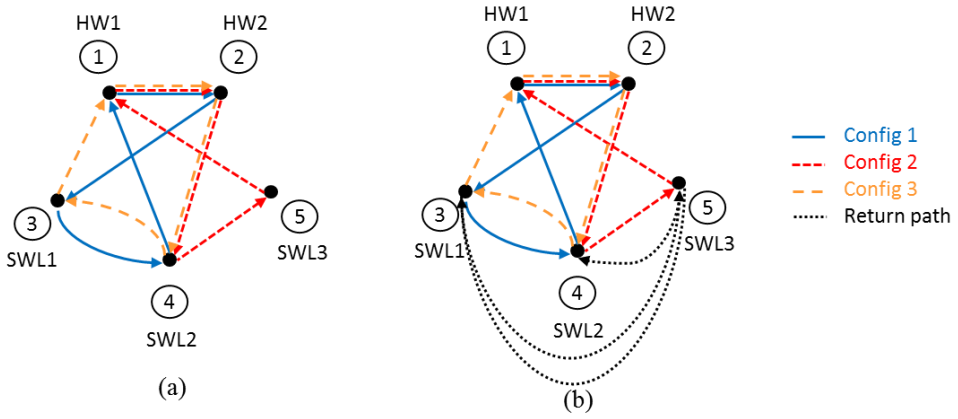

Figure 7. (a) Case without a dedicated return path (b) Case with dedicated return path

Table III. Time Delay Distributions

Edges	Time delay distribution [msec]	Edges	Time delay distribution [msec]
(1, 2)	$N(100,3)$	(4, 3)	$N(10,3)$
(2, 3)	$N(\mu_1(t),3)$	(4, 5)	$N(10,3)$
(2, 4)	$N(\mu_2(t),3)$	(5, 1)	$N(\mu_3(t),3)$
(3, 1)	$N(102,3)$	(5, 4)	$N(10,3)$
(3, 4)	$N(10,3)$	(3, 5)	$N(10,3)$
(4, 1)	$N(103,3)$	(5, 3)	$N(10,3)$
where $\mu_1(t) = 0.5t + 110$; $\mu_2(t) = 100 + 4\lvert t - 10\rvert$; $\mu_3(t) = 140 - 8\lvert t - 10\rvert$; $t = k\Delta t$; $\Delta t = 0.04\ sec$.			

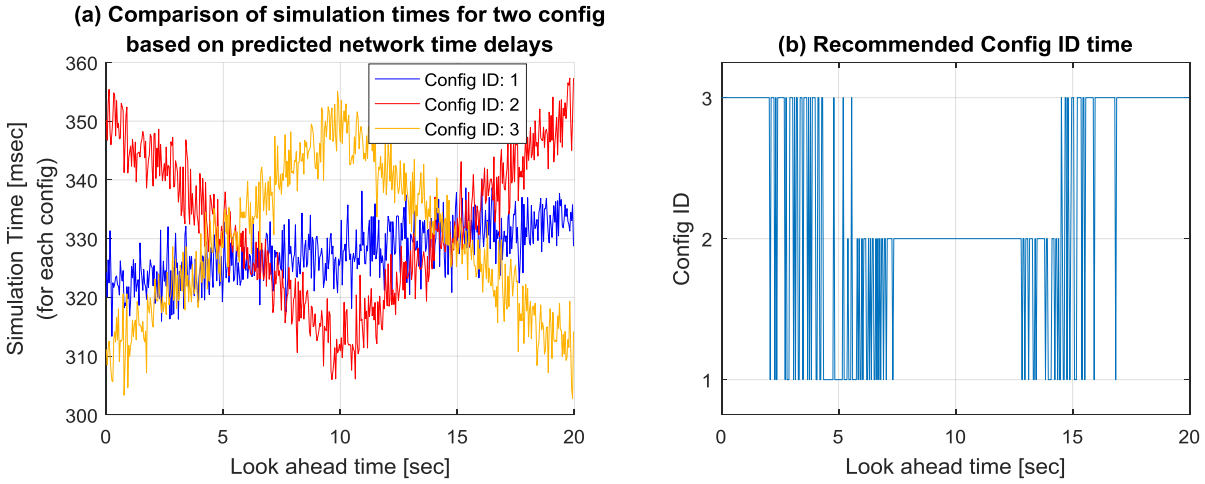

Figure 8. Sample Monte Carlo trial - (a) simulation times for ECP, (b) recommended configuration

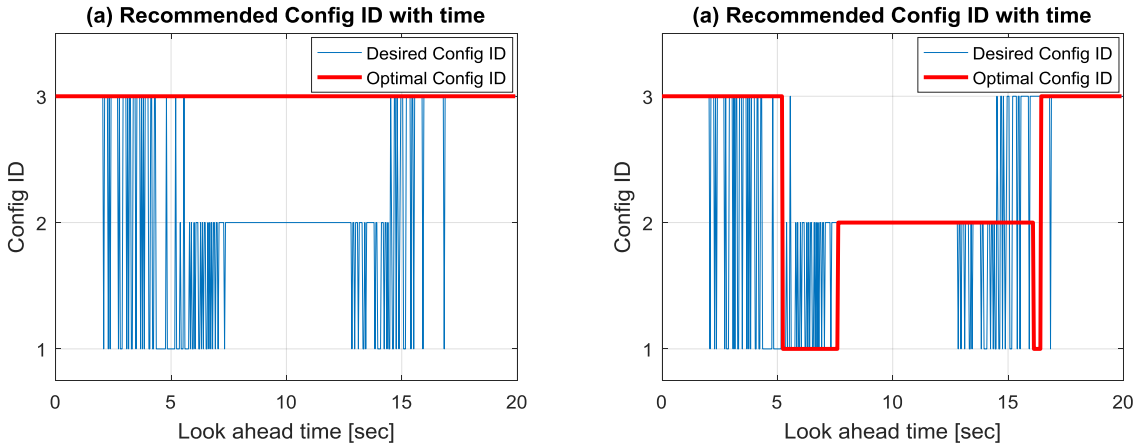

Figure 9. Determination of the optimal path in the sample Monte Carlo trial – (a) Case without a dedicated return path (b) Case with dedicated return path

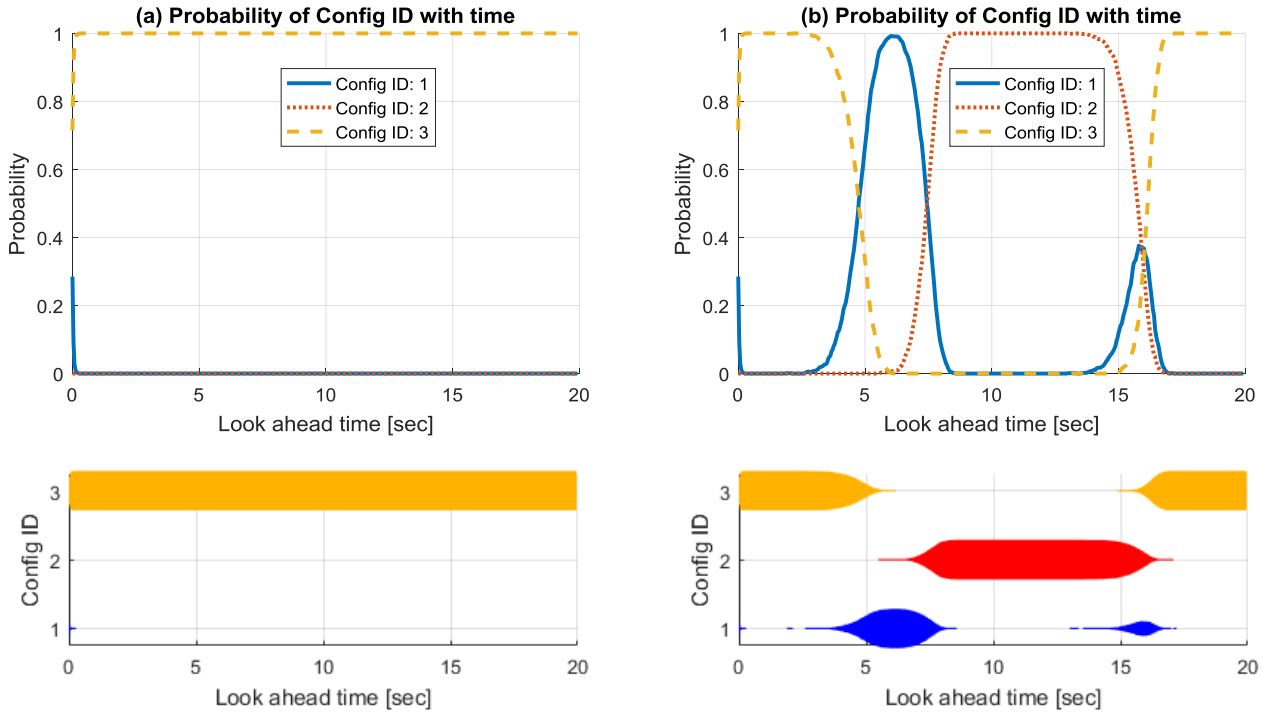

Figure 10. Probability of the system (ECP) being in the available configurations as a function of time; Lower plot shows the density of the Monte Carlo particles in a particular configuration - (a) Case without a dedicated return path (b) Case with dedicated return path

3 CONCLUSIONS

The algorithm described in this paper is aimed to mitigate the stability issues in a distributed test facility caused by significant time delays in a geo-distributed system. The algorithm uses graph theory techniques to find feasible configurations for deploying the ECP. It compares the feasible configurations and determines the optimal configuration within the planning horizon by using dynamic programming

while taking into account the switching costs. The Monte Carlo scheme allows for uncertainties in the time delays and yields the conditional probability of the system being stable in the respective configuration given that the simulation time is well under the critical time delay. This method also provides insight into the necessity of having direct paths for software transitions to improve the optimal solution.

Further research will delve more into the issue of computational complexity to determine the limitations on the network configurations especially with large and complex networks. The future models will also incorporate the software computational time for improved estimation. Efforts in optimizing the algorithm and parallelizing the calculations are ongoing.

4 ACKNOWLEDGMENTS

We would like to extend our gratitude towards our research group members Michael Pietrykowski and Xiaoxu Diao for their support in this work. We would also like to thank our INL collaborator Shannon Bragg-Sitton for her support. This work was supported through the INL Laboratory Directed Research & Development (LDRD) Program under DOE Idaho Operations Office Contract DE-AC07-05ID14517.

5 REFERENCES

1. Q. Guo and C. Smidts, "A facility framework for distributed application," *IEEE Workshops of Intl Conference on Advanced Information Networking and Applications*, pp.459-466. IEEE (2011).
2. K. Subramoniam, M. Maheswaran and M. Toulouse, "Towards a micro-economic model for resource allocation in grid computing systems," *Canadian Conference on Electrical and Computer Engineering*, vol. 2, pp.782-785. IEEE (2002)
3. A. Billionnet, M.-C. Costa and A. Sutter, "An efficient algorithm for a task allocation problem," *Journal of the ACM*, 39, no. 3, pp.502-518 (1992)
4. B. P. Gerkey and M. J. Mataric, "Multi-robot task allocation: Analyzing the complexity and optimality of key architectures," *IEEE International Conference on Robotics and Automation*, vol. 3, pp. 3862-3868. IEEE (2003).
5. M. J. Krieger, J. B. Billeter and L. Keller, "Ant-like task allocation and recruitment in cooperative robots," *Nature* 406, no. 6799, pp.992-995 (2000).
6. F. A. Kuipers, S. Yang, S. Trajanovski and A. Orda, "Constrained Maximum Flow in Stochastic Networks," *IEEE 22nd Intl Conference on Network Protocols*, Raleigh, NC, pp.397-408 (2014)
7. Ding, Sibo, "Uncertain minimum cost flow problem," *Soft Computing* 18.11. pp.2201-2207 (2014).
8. Boyles, Stephen D., and S. Travis Waller. "A mean-variance model for the minimum cost flow problem with stochastic arc costs." *Networks* 56.3. pp. 215-227 (2010).
9. Bellman, Richard, *The theory of dynamic programming*, No. RAND-P-550. RAND CORP SANTA MONICA CA (1954).