Decomposition Theory Meets Reliability Analysis: Processing of Computation-Intensive Dependent Tasks Over Vehicular Clouds With Dynamic Resources

Payam Abdisarabshali®, Minghui Liwang®, Member, IEEE, Amir Rajabzadeh®, Mahmood Ahmadi®, and Seyyedali Hosseinalipour®, Member, IEEE

Abstract—Vehicular cloud (VC) is a promising technology for processing computation-intensive applications (CI-Apps) on smart vehicles. Implementing VCs over the network edge faces two key challenges: (C1) On-board computing resources of a single vehicle are often insufficient to process a CI-App; (C2) The dynamics of available resources, caused by vehicles’ mobility, hinder reliable CI-App processing. This work is among the first to jointly address (C1) and (C2), while considering two common CI-App graph representations, directed acyclic graph (DAG) and undirected graph (UG). To address (C1), we consider partitioning a CI-App with (DAG) and undirected graph (UG). To address (C1), we consider partitioning a CI-App with m dependent sub-tasks into k ≤ m groups, which are dispersed across vehicles. To address (C2), we introduce a generalized reliability metric called conditional mean time to failure (C-MTTF). Subsequently, we increase the C-MTTF of dependent sub-tasks processing via introducing a general framework of redundancy-based processing of dependent sub-tasks over semi-dynamic VCs (RP–VC). We demonstrate that RP–VC can be modeled as a non-trivial semi-Markov process (SMP). To analyze this SMP model and its reliability, we develop a novel mathematical framework, called event stochastic algebra ((e)-algebra). Based on (e)-algebra, we propose decomposition theorem (DT) to transform the presented SMP to a decomposed SMP (D-SMP). We subsequently calculate the C-MTTF of our methodology. We demonstrate that (e)-algebra and DT are general mathematical tools that can be used to analyze other cloud-based networks. Simulation results reveal the exactness of our analytical results and the efficiency of our methodology in terms of acceptance and success rates of CI-App processing.

Index Terms—Event stochastic algebra, decompositions theory, vehicular cloud, semi-Markov process, stochastic analysis, reliable service provisioning, directed acyclic graphs (DAG) tasks/applications, undirected graph (UG) tasks/applications.

I. INTRODUCTION

RECENT years have witnessed explosive growth in the number of smart vehicles equipped with powerful on-board processors. Researchers have subsequently promoted the utilization of on-board computing resources of vehicles for innovative Internet of Things (IoT) applications (e.g., self-driving, augmented reality, and crowd processing) [1], [2], [3], which are predominantly computation-intensive and energy-hungry. Nevertheless, the limited on-board computing and storage resources of a single vehicle may fail to meet the execution demands of computation-intensive applications (CI-App), e.g., augmented reality. One method to overcome this limitation is to offload application data from vehicles to centralized cloud servers [4], [5] or edge cloud servers, enabled via technologies such as fog computing [6], through vehicle-to-infrastructure (V2I) connections. Although this can potentially alleviate the computing resource shortage of vehicles, application offloading through V2I and data relaying across the core network may result in extra delay, network congestion, and excessive network resource (e.g., computing and spectrum) utilization [7].

To alleviate the aforementioned issues, vehicular cloud (VC) has been introduced, which orchestrates the distributed and dynamic resources (e.g., processors, storage, and sensors) of smart vehicles in a cooperative manner through both vehicle-to-vehicle (V2V) and V2I communications [8]. VC has attracted tremendous attentions in supporting various services such as traffic management and entertainment (e.g., VR gaming) [7], [9], [10]. Through exploiting V2V connections and cooperative application processing over the vehicles, VC reduces the service latency and overhead on backhaul links [10].

A. Motivations and Challenges

Implementing VC on the network edge faces two challenges.

(C1) Heterogeneity and limitation of computing resources across vehicles: Computing resources are often varying across the vehicles. Particularly, a vehicle may suffer from resource limitations to process a CI-Apps, such as augmented reality [3] and data analysis [11], which can degrade the users’ quality of experience (QoE).

(C2) Dynamics and volatility of available computing resources: Vehicles may enter and leave the VC due to their mobility, imposing new challenges in CI-App processing ranging from handling the highly dynamic resources [12], [13] to ensuring reliable resource provisioning [14].

Addressing (C1). One promising approach to tackle (C1) is partitioning a CI-App into smaller dependent sub-tasks that...
can be dispersed across vehicles [1]. Dependent sub-tasks structure can be used to represent a wide range of applications (also known as graph-structured applications) [1], [3], [11], [15], [16], and have been recognized and incorporated into modern computing architectures (e.g., micro-services architecture offered via IBM [16]). Execution of a graph-structured application requires processing its constituent dependent sub-tasks, where processing a sub-task is contingent on the reception of data from other sub-tasks. Hence, the failure of processing of one sub-task can lead to a chain of failures of other dependent sub-tasks [15]. Hence, although partitioning an CI-App into multiple dependent sub-tasks can improve the system performance in terms of CI-App execution time (e.g., due to possible parallel processing of a fraction of sub-tasks) and computing resource utilization [3], [11], reliability enhancement (e.g., reducing the chance of failure of CI-App processing) remains an open challenge.

Addressing (C2). Studying the reliability of application processing in the VC is still in its early development stage and only handful of research works [14], [17], [18], [19], [20] have been devoted to address (C2) to mitigate the effect of fluctuations in available resources. These works aim to increase the mean time to failure (MTTF) of application processing, which is one of the well-known fault tolerance metrics defined as the expected time it will take for the system to encounter a failure. The main assumption of these works is that a single vehicle has sufficient computing resources to process applications of any sizes, which is unrealistic in case of having CI-Apps.

Reliability Analysis under Application Partitioning. None of the existing works have so far jointly addressed (C1) and (C2). Motivated by this, we tackle the following research questions:

(Q1) How to enhance the processing reliability (e.g., increasing the MTTF) in a semi-dynamic VC (e.g., a VC over a parking lot) when a CI-App with m dependent sub-tasks is partitioned into k ≤ m groups and processed distributedly on different vehicles?

(Q2) How application partitioning and dispersing the corresponding sub-tasks can affect the reliability of CI-App processing over a semi-dynamic VC?

To answer (Q1), we propose a general framework for redundancy-based processing of dependent sub-tasks over semi-dynamic VCs⁷(RP–VC). Roughly speaking, RP–VC (i) disperses the processing of an application with m dependent sub-tasks, modeled as a directed acyclic graph (DAG) or undirected graph (UG), over a semi-dynamic VC through application partitioning and (ii) enhances the reliability of the CI-App processing via considering redundancy-based resource provisioning to mitigate the impact of vehicles’ unexpected departures from the VC.

To answer (Q2), we present mathematical modeling to analyze the reliability of dependent sub-tasks processing over a semi-dynamic VC, which has been an open problem and is the key contribution of our work. In particular, we introduce a reliability metric, called conditional mean time to failure (C-MTTF), to quantify the reliability of processing DAG- and UG-structured CI-App. We next propose a non-trivial semi-Markov process (SMP) to model RP–VC. We then introduce a novel mathematical framework, which we refer to as event stochastic algebra ((ε)-algebra), which makes the reliability analysis of dependent sub-tasks processing tractable. We then exploit (ε)-algebra and introduce decomposition theorem (DT) to transform the proposed non-trivial SMP into a decomposed SMP (D-SMP). We obtain the closed-from expression of C-MTTF through in-depth analysis of D-SMP. We will show that (ε)-algebra and DT are general mathematical tools that can be utilized for other problems in cloud-based systems.

It is worth mentioning that the term “decomposition” has been used in other math domains, e.g., in optimization [21], [22], where an optimization problem is decomposed into several simpler sub-problems. In this paper, we develop a new notion of “decomposition”, where our theory enables us to disentangle the states of β-SMP into several simpler sub-states.

B. Related Work

VC architectures considered in literature can be roughly divided into two categories: semi-dynamic (e.g., parked vehicles) and dynamic (e.g., moving vehicles). Service provisioning through semi-dynamic VCs [8] has been widely studied in literature. For example, research works [14], [17], [18], [19], [20], [23] have considered VCs in parking lots formed via parked vehicles. Also, dynamic VC architectures for moving vehicles have also been investigated [1], [13], [24], [25], [26], [27], where most of the works aim to study the impact of the mobility patterns of the vehicles on application processing. The above-mentioned works have tackled various problems in VCs, such as resource provisioning, application partitioning, and reliability, as discussed below.

1) Resource Provisioning: There exist several works dedicated to addressing resource provisioning problems in VCs. Works [26] and [27] proposed innovative resource provisioning strategies based on semi-Markov decision processes (SMDP). The authors in [13] have modeled the VC as a cluster of connected vehicles, where a head vehicle supervises and allocates resources in each cluster. However, these studies focused on resource provisioning without considering the reliability aspects and application partitioning.

2) Application Partitioning: Several studies have tackled effective partitioning of CI-Apps into smaller sub-tasks [1], [28]. Authors in [1] have studied the joint application partitioning and power control problem in a fog computing network to optimize the long-term system utility measured in terms of execution delay and energy consumption. In [28], the authors considered a multi-user application partitioning in industrial mobile edge computing (MEC) systems, where the workload of a vehicle is partitioned and offloaded to rented edge devices in a MEC platform. However, works in this literature, none of the conducted research has dealt with the effect of application partitioning on reliability metrics, e.g., MTTF.

3) Reliability: The dynamics of VCs has made their reliability analysis a vital research topic [14], [17]. Although reliability of application processing over VCs has not been investigated profoundly, some prior works have taken initial steps toward this direction via proposing a variety of methods to improve the reliability of VCs, such as migration of applications (i.e., migrating applications of the imminent leaving vehicle to the nearby vehicles) [25], [29], checkpointing (i.e., storing snapshots of the application’s state at multiple time-stamps, providing an opportunity to recover the application if failures occur) [30] and redundancy-based execution (i.e., multiple images/replicas of the application are executed among multiple vehicles to provide robustness against vehicles departure) [14], [17], [18], [19], [20].

¹The departure of a vehicle from the VC leads to a failure of the applications/tasks offloaded to the vehicle.
Among these methods, redundancy-based strategies are of particular interest\footnote{Designing efficient VM migration strategies faces significant challenges in the VCs with unpredictable vehicular sojourn times since predicting the optimal moment for conducting VM migration is non-trivial\cite{17}. Moreover, checkpointing adds considerable overhead to the VC\cite{30}.} to enhance the MTTF of application execution\cite{14,17,18,19,20}. One of the primary works has proposed two strategies called $J_2$ and $J_3$ to mitigate the impact of VC dynamics on application processing\cite{14}. These strategies increase the MTTF by allocating two/three vehicles to an application. Authors in\cite{14} also provided mathematical models based on a semi-Markov process (SMP) to calculate the MTTF. The proposed models in\cite{14} require complete information on the probability distributions of the vehicles’ sojourn times (i.e., the time during which a vehicle is parked in a parking lot) and recruitment duration (i.e., the duration of time required to recruit a new vehicle). To relax these requirements, in a follow-up work, authors in\cite{18} provided a methodology to estimate the MTTF of strategies $J_2$ and $J_3$. In addition, a mathematical model is presented to estimate the completion time of an application under the supervision of $J_2$ in\cite{19}. Further, the authors in\cite{20} extended the framework of\cite{19} to obtain a more accurate estimation of application completion time. Finally, a generalization of $J_2$ to $J_n$ is proposed in\cite{17}, in which $n$ vehicles simultaneously process the same application, where each vehicle’s sojourn and recruitment times are modeled via exponential random variables\cite{17}. Strategies $J_2$ and $J_n$ enhance the reliability of application processing as follows.

$J_2$: In this technique, two vehicles ($C_1$, $C_2$) are assigned to an application, which process it independently and simultaneously. When one of the two vehicles leaves the VC coverage area, the other one pauses processing its application and starts recruiting a new vehicle. The application processing will be resumed on both vehicles (i.e., the recruiter and recruited vehicles) if the recruitment operation completes successfully.

$J_n$: The procedure conducted to enhance the reliability in this strategy is almost equivalent to $J_2$, except that it assigns $n$ different vehicles to an application. If the first vehicle leaves the VC coverage area, strategy $J_n$ first considers one of the other $n-1$ vehicles as the recruiter (namely $C_r$) and defines a variable $h = 0$. $C_r$ starts recruiting a new vehicle and increases $h$ by one. If another vehicle leaves the VC coverage area during recruitment, $C_r$ increases $h$ by one and recruits two vehicles. Likewise, if the third vehicle leaves the VC during recruitment, $C_r$ recruits three vehicles. This process continues until the recruitment operation completes by recruiting $h$ new vehicles or all of the $n$ vehicles leave the VC, in which case the processing of the application encounters a failure.

The only application model adopted in the literature of redundancy-based application execution in VCs is $J_n$ class of strategies, the processing of which is different than applications consisting of multiple dependent sub-tasks dispersing across different vehicles. In particular, upon execution of a CI-App with multiple dependent sub-tasks, failure of a single sub-task can lead to the failure of the entire application. To the best of our knowledge, none of the conducted studies have investigated the reliability of processing dependent sub-tasks in a VC.

\section{Outline and Summary of Contributions}

Our major contributions can be summarized as follows:

- We model an application using a general graph representation ensembling DAG and UG (Sec. II-B). We propose $RP-VC$, which is the first unified framework for studying the MTTF of redundancy-based processing of an application with $m$ dependent sub-tasks, modeled as a DAG or UG, partitioned into $k \leq m$ groups, in semi-dynamic VCs (Sec. II-C).
- We show that computing the MTTF of processing of an application modeled as a DAG is non-trivial. We then introduce an extension of MTTF, called conditional MTTF ($C-MTTF$), which can quantify the reliability of processing an application modeled as both DAG and UG (Sec. II-D).
- To model the dynamics of our system, we introduce a general model for a class of stochastic systems, which we call stochastic event system (SeS). We then present a new concept, called $\beta$-inhomogeneous, to characterize SeS (Sec. III-C). We demonstrate that a semi-dynamic VC under $RP-VC$ is an SeS, which we refer to as SeS$_{VC}$.
- We show that the execution of dependent sub-tasks over a SeS$_{VC}$ through $RP-VC$ can be modeled as an SMP (Sec. III-C2). We demonstrate that because of $\beta$-inhomogeneous property of SeS$_{VC}$, analyzing the presented SMP, referred to as $\beta$-inhomogeneous SMP ($\beta$-SMP), is non-trivial (Sec. III-C4).
- We develop a unified mathematical framework, called event stochastic algebra ($\langle e \rangle$-algebra), enabling us to investigate the dynamics of an SeS. We demonstrate the generality of the proposed $\langle e \rangle$-algebra, making it suitable for studying a variety of similar problems in literature (Sec. IV-A).
- Building on the $\langle e \rangle$-algebra, we develop the foundations of decomposition theorem (DT), used to decompose each state of $\beta$-SMP (Sec. IV-C). By utilizing DT, we demonstrate that $\beta$-SMP can be transformed into a decomposed SMP ($D$-SMP) (Sec. IV-D).
- Relying on D-SMP, we derive the general closed-form expression describing $C-MTTF$ of our methodology under general dynamics of vehicles in a VC (Sec. V). Subsequently, we obtain a special closed-form formula to compute $C-MTTF$ of $RP-VC$ for a realistic scenario, in which the sojourn times and recruitment duration of the vehicles follow exponential distribution (Sec. V-B).
- We present extensive simulations to verify the correctness of our mathematical results and demonstrate the efficiency of our proposed methodology in terms of application acceptance rate and application success rate (Sec. VI).

\section{System Model and Preliminaries}

In this section, we first describe the network model of a semi-dynamic VC and state the problem regarding question (Q1) (Sec. II-A). Afterward, we introduce the application and partition models (Sec. II-B). We next introduce our methodology, i.e., $RP-VC$ (Sec. II-C). Finally, we present the reliability model and problem statement regarding question (Q2) (Sec. II-D).

\subsection{VC Network Model and First Problem Statement}

Fig. 1 illustrates the network architecture of a semi-dynamic VC with a set of vehicles equipped with onboard computing
In this architecture, we are interested in addressing the execution of the incoming application [14], [17], [18], [19], [20]. In this architecture, we are interested in addressing the following problem:

**Problem 1:** How to increase the MTTF of processing a CI-App with $m$ dependent sub-tasks into $k \leq m$ groups and processed distributedly on different vehicles in the semi-dynamic VC presented in Fig. 1?

In the following, we introduce RP-VC to address Problem 1.

We first present application and partition models.

### B. CI-Apps and Partition Models

We consider a scenario in which a CI-App is partitioned into several groups and distributed across vehicles. We assume that there exist enough vehicles in the VC converge area, where the controller can allocate at least two processing vehicles to each sub-task of the incoming application [14], [17], [18], [19], [20]. In this architecture, we are interested in addressing the following problem:

**Problem 1:** How to increase the MTTF of processing a CI-App with $m$ dependent sub-tasks partitioned into $k \leq m$ groups and processed distributedly on different vehicles in the semi-dynamic VC presented in Fig. 1?

In the following, we introduce RP-VC to address Problem 1.

We first present application and partition models.

#### B. CI-Apps and Partition Models

We consider a scenario in which a CI-App is partitioned into several groups [1], [3], [11], [24], [33]. We first define a CI-App, and then we describe the partitioning procedure.

**Definition 1 (Application):** An application, denoted by $\mathcal{A}$, is a set of $m$ dependent sub-tasks, modeled by a graph:

$$\mathcal{A} \triangleq (\mathcal{V}, \mathcal{E}),$$

where $\mathcal{V} \triangleq \{T_1, T_2, \ldots, T_m\}$ denotes the set of sub-tasks (i.e., vertices) and $\mathcal{E} \triangleq \{(x, x') | x, x' \in \{1, 2, \ldots, m\}, x \neq x'\}$ is the set of dependencies between the sub-tasks (i.e., edges).

To characterize internal structures of an application, we consider the following two types of graph representation:

- **Directed Acyclic Graph (DAG).** DAG describes the order of execution of sub-tasks, leading to processing the sub-tasks partially in parallel [3], [11].
- **Undirected Graph (UG).** There is no order between sub-tasks and they all can be processed in parallel [34], [35].

Due to the resource deficiency of vehicles, a large application with $m$ dependent sub-tasks can be partitioned into $k \leq m$ deployable groups and offloaded to different vehicles for processing. We define a partition of application $\mathcal{A}$ as follows.

**Definition 2 (Partition):** A partition of an application $\mathcal{A} = (\mathcal{V}, \mathcal{E})$ with $m$ dependent sub-tasks, referred to by $\mathcal{P}(\mathcal{A})$, is a dependency-preserving grouping of sub-tasks into $k \leq m$ mutually exclusive nonempty groups $\mathcal{G} = \{G_1, \ldots, G_k\}$. That is

$$\mathcal{P}(\mathcal{A}) \triangleq (\mathcal{G}, \mathcal{E}),$$

where $G_x, G_x' \subset \mathcal{V}, \forall G_x, G_x' \in \mathcal{G},$ and $\bigcup_{i=1}^{k} G_i = \mathcal{V}$.

To clarify Definitions 2 and 3, Fig. 2 provides an example of a DAG of an application $\mathcal{A} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{T_1, T_2, \ldots, T_6\}$ and $\mathcal{E} = \{(1, 2), (1, 3), (2, 4), (2, 5), (4, 6), (5, 6)\}$. $\mathcal{A}$ is grouped by partition $\mathcal{P}(\mathcal{A}) = (\mathcal{G}, \mathcal{E})$ into three groups $\mathcal{G} = \{G_1, G_2, G_3\}$, where $G_1 = \{T_1, T_2\}$, $G_2 = \{T_3, T_4\}$, and $G_3 = \{T_5, T_6\}$.

We next introduce our general reliable application execution methodology called RP-VC to tackle Problem 1, which is utilized for processing applications constituted of multiple dependent sub-tasks. RP-VC extends the prior art (e.g., J2 [14], concerned with reliable resource allocation for a scenario in which an application is offloaded to the vehicles) to a scenario in which reliable execution of application sub-tasks are of interest.

#### C. RP-VC: Redundancy-Based Processing of CI-Apps

RP-VC has three main units, (i) partition strategy (PS), (ii) offload manager (OM), and (iii) 2-redundant allocation strategy (2-RAS), which are discussed below.

1) **Partition Strategy (PS):** The PS breaks down application $\mathcal{A}$ into smaller groups by partition $\mathcal{P}(\mathcal{A})$ and sends it to the OM. A broad range of partition strategies can be utilized for this intent [1], [28]. In this paper, we consider application partitioning under an arbitrary strategy since we are not concerned with the efficiency of PS, which is left as future work.

2) **Offload Manager (OM):** Consider a partition $\mathcal{P}(\mathcal{A}) = (\mathcal{G}, \mathcal{E})$ of an application $\mathcal{A}$. If $\mathcal{A}$ is a UG-structured application, the groups in $\mathcal{G}$ can be offloaded to different vehicles and processed in parallel. However, if $\mathcal{A}$ is a DAG-structured application, there is a definite order between the execution of the sub-tasks, and different groups should wait until the dependencies of their sub-tasks are satisfied. OM supports applications modeled by both DAG and UG structures. Consider discrete time instances $\tau = \{t_0, t_1, \ldots, \\}$. At time $t_0 \in \tau$ (i.e., the time that the processing of $\mathcal{A}$ is started), OM specifies a subset $\mathcal{V}_{t_0} \subseteq \mathcal{V}$ (the equality of the subset notation is for UG-structured applications) based on a specific strategy (e.g., [33]) and sends it to 2-RAS for offloading. Likewise, at time $t_1 \in \tau$, OM specifies another subset $\mathcal{V}_{t_1} \subseteq \mathcal{V} \setminus \mathcal{V}_{t_0}$ and sends it to 2-RAS. This procedure takes place until all the groups are offloaded to the vehicles. Referring to Fig. 2 as an example, at time $t_0$, OM specifies $\mathcal{V}_{t_0} = \{G_1, G_2\}$ and sends it to 2-RAS for
deployed on vehicle. Further, at time $t_1$, OM specifies $\overline{V}_{t_1} = \{G_3\}$ to be offloaded to the vehicles. Same as the $PS$, different strategies can be exploited for $OM$ [33]. In this paper, we consider an arbitrary OM and dealing with different OM strategies is left for future works.

3) 2-Redundant Allocation Strategy (2-RAS): 2-RAS has the responsibility of allocating 2 different vehicles\(^4\) to each group $G_h \in \overline{V}_t$ at each time $t \in \tau$. Note that in our mathematical analysis, we assume that each vehicle processes only one group at a specific time, which is a common assumption [14], [17]. In Sec. VI, we will investigate a scenario in which more than one group is processed by a vehicle.

Let $C_t$ and $C'_{t'}$ be two vehicles assigned to group $G_h \in \overline{V}_t$. As the second responsibility of 2-RAS, if $C_t$ departs the VC, 2-RAS determines $C'_{t'}$ as the recruiter, which in turn starts recruiting a vehicle to process $G_h$. Recruitment operation for each group $G_h$ can thus be summarized as follows: the recruiter (i.e., $C'_{t'}$) (i) finds a new vehicle from the vehicles that were in the VC at the time that processing of application $A$ has been started, and (ii) transfers an image/replica of $G_h$ (i.e., the current status of processing $G_h$ and its required data) to the recruited vehicle through a migration strategy. Studying 2-RAS is our major focus in this paper.

We next present the reliability model and the key problem addressed in this paper.

\section*{D. Reliability Model and Second Problem Statement}

Reliability is a major notion of quality of service (QoS) in cloud-based computing systems. One of the most well-known metrics to quantify reliability is mean time to failure (MTTF). Calculating MTTF of processing a DAG-structured application in a semi-dynamic VC under RP-VC faces significant challenges since, as mentioned in Sec. II-C2, different OMs may offload the groups of $\mathcal{G}$ to the vehicles progressively (see Fig. 2). For instance, authors in [33] present a strategy in which offloading a group of sub-tasks is parallelized with processing other deployed sub-tasks. Consequently, the number of deployed groups of partition $P(A)$ on the vehicles (i.e., the groups offloaded to the vehicles for processing) varies at different time instances during processing application $A$. Accordingly, we extend the MTTF to introduce a new reliability metric, called conditional mean time to failure (C-MTTF), enabling us to calculate MTTF at different time instances.

Let $C = \{C_1, \ldots, C_r\}$ denote the set of all $r$ vehicles in the VC. Before formalizing C-MTTF, we first introduce the following two definitions to characterize deployed groups on the vehicles at an arbitrary time instant $t$.

\begin{definition}[Deployment Map] At time instant $t$, consider partition $P(A) = (\mathcal{G}, \mathcal{C})$ with $k$ groups. Deployment map of partition $P(A)$ at time $t$, referred to by $M(P(A), t) = M = [M_{h,t}]_{1 \leq h \leq k, 1 \leq t \leq r}$, is a function outputting a $k \times r$ binary matrix $M$ at time $t$, where $M_{h,t} = 1$ implies group $G_h \in \mathcal{G}$ is deployed on vehicle $C_t \in \mathcal{C}$.
\end{definition}

\begin{definition}[Deployment] Consider deployment map $M(P(A), t)$. Deployment at time $t$, shown by $D(M(P(A), t))$, is a function outputting the set of all groups deployed on the vehicles at time $t$. That is

$$D(M(P(A), t)) = \left\{G_h | G_h \in \mathcal{G}, 1 \leq r \sum_{t=1}^{r} M_{h,t} \right\}. \quad (3)$$

\end{definition}

We next define $\Theta(M(P(A), t))$ as a function outputting the set of all groups that one of their vehicles is departed from the VC. Mathematically,

$$\Theta(M(P(A), t)) = \left\{G_h | G_h \in \mathcal{G}, \sum_{t=1}^{r} M_{h,t} = 1 \right\}. \quad (4)$$

To define C-MTTF, we make an assumption on the applications, which holds in practical systems [3], [11], [15], [24], [28].

\begin{assumption} Consider $M(P(A), t)$ at time $t$. If vehicle $C_t$, allocated to $G_h \in \Theta(M(P(A), t))$ (i.e., the only remaining vehicle processing $G_h$), leaves the VC before completing its recruitment, the execution of $A$ will encounter a failure.
\end{assumption}

To clarify Assumption 1, consider the DAG presented in Fig. 2. Assume that vehicles $C_2$ and $C_4$ are departed the VC. In this situation, the sub-tasks belonging to $G_1$ and $G_2$ are only processed by vehicles $C_1$ and $C_3$, respectively. Therefore, the departure of $C_1$ from the VC, while it processes $T_1$, results in the failure of $T_1$, leading to a chain of failures of $\{T_2, T_3, \ldots, T_6\}$. Otherwise, if $C_1$ processes $T_1$ successfully, $T_2$ and $T_3$ will be processed by $C_1$ and $C_3$ in parallel. At this moment, if $C_1$ departs the VC, $T_2$ will be failed, leading to a chain of failures of $\{T_3, T_4, T_5\}$. Due to Assumption 1, although the dependency between $T_2$ and $T_3$ is from $T_1$ to $T_2$ (i.e., vehicle $C_3$, which processes $T_2$, should wait until $C_1$ processes $T_1$), the departure of $C_3$ leads to the failure of $A$, because it is costly (e.g., in terms of power consumption) to re-offload $G_2$. Hence, the failure of $G_h \in D(M(P(A), t))$ leads to the failure of $A$.

Considering Assumption 1, we next define C-MTTF below.

\begin{definition}[C-MTTF] Consider $M(P(A), t)$ at time $t$. Also, let $R(t) = \Theta(M(P(A), t))$. C-MTTF of application $A$ given $M(P(A), t)$, $R(t)$, and $t$, referred to by $C-MTTF(A|M(P(A), t), R(t), t)$, is the expected time until one of the groups $G_h \in D(M(P(A), t))$ encounters a failure.
\end{definition}

It can be construed that C-MTTF extends the definition of MTTF [14] to a scenario in which the application consists of multiple sub-tasks. We next define the following non-trivial problem that we carefully investigate in subsequent sections.

\begin{problem} Consider an application $A$ partitioned by $P(A)$. What is the impact of distributed processing of the sub-tasks of $A$ according to deployment map $M(P(A), t)$ on the C-MTTF of CI-App processing in a semi-dynamic VC? Precisely, what is $C-MTTF(A|M(P(A), t), R(t), t)$ at time $t$?
\end{problem}

The main contribution of this paper is to present a mathematical framework to study Problem 2. To this end, we conduct theoretical analysis to study the C-MTTF of RP-VC. Note that we focus on the reliable processing of one application given the dynamics of the VC [14], [17], [18]. Hereafter, we use notation RP-VC$_n$ to specify the number of groups in $D(M(P(A), t))$ (i.e., $n = |D(M(P(A), t))|$).
III. MODELING CI-APP PROCESSING AS AN SMP

In this section, we model the dynamics of RP-VC. To this end, we first discuss semi-Markov process (Sec. III-A), and then overview the SMP model of Jn (Sec. III-B). Finally, we introduce the SMP model of RP-VC (Sec. III-C).

A. Semi-Markov Process: Definitions and Preliminaries

A Markov process is used to describe the dynamics of stochastic systems with limited information about their past, defined as follows.

**Definition 6 (Markov Process):** A random process \( \{X_t : t \geq 0\} \) is a Markov process (MP) if, for any \( s, t \geq 0 \), the conditional probability of transition to the future state \( X_{s+t} \) given the present state \( X_s \) and history \( X_u \), for \( 0 \leq u < s \), depends only on the present state. Mathematically, for non-negative integer values \( i, j, \) and \( x(u) \), where \( x(u) \) is an arbitrary function, we have

\[
P(X_{s+t} = i | X_s = j, X_u = x(u), 0 \leq u < s) = P(X_{s+t} = i | X_s = j).
\]

Definition 6 is called Markov property, implying that sojourn time (i.e., the duration that SMP stays in a state before transitioning to other states) in each state has memoryless property and thus follows an exponential distribution [37]. A generalization of Definition 6 where the sojourn time in a state can follow an arbitrary distribution is called semi-Markov process (SMP) [37]. In short, in semi-Markov process, the transition to a future state depends not only on the present state but also on the amount of time spent in it; however, in Markov process, the transition to a future state depends only on the present state. Semi-Markov process captures the scenario of our interest in this paper, where the expected sojourn time in each state (e.g., the time that it takes to successfully recruit a new vehicle) is a function of time the system has spent in a state itself (e.g., the time spent on recruiting from when the system has entered the state).

B. SMP Model of Jn Overview

Fig. 3(a) illustrates the SMP model of the state-of-the-art redundancy-based application execution strategy Jn for a single application A [17]. In this figure, state \( (C_1 C_2 \ldots C_n) \) is the initial state and refers to a situation in which \( n \) vehicles process application \( A \) simultaneously and independently. Generally, in all states \( \{(C_1 C_2 \ldots C_{n-i}), (C_1 C_3 \ldots C_{n-i+1}), \ldots (C_2 C_3 \ldots C_{n-i+1})\} \), \( n - i \) vehicles process application \( A \) and \( i \) vehicles are departed from the VC. In this SMP model, the departure of different vehicles from the VC results in transitions to different states, captured via the links depicted in Fig. 3(a). This SMP model is large and hard to handle specially since it has similar states. For instance, although in both states \( (C_1 C_3 \ldots C_n) \) and \( (C_2 C_3 \ldots C_n) \) one vehicle is departed from the VC, they are considered distinguished. The SMP model presented in Fig. 3(a) can be represented more simply and concisely by considering the symmetry of states as depicted in Fig. 3(b). For example, all states \( \{(C_1 C_2 \ldots C_{n-1}), (C_1 C_3 \ldots C_{n-i+1}), \ldots (C_2 C_3 \ldots C_{n-i+1})\} \) are equivalent and can be wrapped into a more compact state \( O_{i} \), depicted in Fig. 3(b). To elaborate on the compact model, state \( O_n \) on top of the figure represents having \( n \) vehicles processing an application A. Further, state \( O_{n-1} \) represents a state in which one vehicle departs the VC and one of the other \( n - 1 \) vehicles recruits a new vehicle. Generally, there are \( n - i \) vehicles in state \( O_{n-i} \), and \( i \) vehicles are departed from the VC. In state \( O_{n-i} \), there is only one recruiter responsible for recruiting new vehicles to replace the departed vehicles, since all the \( n - i \) vehicles execute the same application. Furthermore, completing the recruitment operation causes a transition to the initial state (i.e., \( O_n \)) with a probability of \( p_i \). Conversely, departing a vehicle in state \( O_{n-i} \) results in a transition to state \( O_{n-i-1} \) with probability of \( 1 - p_i \) [17].

C. SMP Model of RP-VC

We present a theoretical model to characterize the behavior of RP-VC, exploited later to calculate C-MTTF \( A | M (P(A), t), R(t), t \). We first present a general class of stochastic systems that we refer to as stochastic event systems (SeS), utilized to introduce the SMP model of RP-VC.

**Definition 7 (Stochastic Event System (SeS)):** A SeS is a stochastic system modeled as a tuple \( \{Q, \mathcal{E}, t, \mathcal{F}\} \), where \( Q = \{Q_1, \ldots, Q_r\} \) is an event set and \( \mathcal{F} \) can lead to scheduling one event. \( \mathcal{F} \) is an event set and \( \mathcal{G} \) denotes a set of random variables. Each \( Q_a \in \mathcal{Q} \) describes the time until occurring event \( e_a \in Q \). In SeS, the following conditions hold:

1. At time \( t \), event \( e_a \in Q_a \) is scheduled to occur after \( Q_a \) random time units.
2. The occurrence of each event \( e_a \) can lead to scheduling a set of \( s \) events \( \{e_{a_1}, \ldots, e_{a_s}\} \subset \mathcal{Q} \).

In this paper, we consider \( s = 1 \) for analytical simplicity (i.e., the occurrence of each event leads to scheduling of one event). One of the main characteristics of SeS is that the occurrence of the events are dependent (i.e., an event is scheduled only at time \( t_0 \) or by the occurrence of another event). To characterize the events of an SeS, we present following property.

**Definition 8 (β-inhomogeneous):** Consider an SeS \( \Delta = (Q, \mathcal{Q}) \). An event \( e_a \in Q \) is β-inhomogeneous if after occurring event \( e_a \), the residual occurring time of event \( e_a \) can follow β different possible distributions.

We next model the VC system depicted in Fig. 1 as an SeS.
1) **Semi-Dynamic VC as a Stochastic Event System:** The semi-dynamic VC presented in Fig. 1 under the supervision of RP-VC is an SSe, referred to by SeSSVC, defined as follows.

**Definition 9 (SeSSVC):** Let \( C \) denote the set of all the vehicles utilized to process \( A \). Also, let the set of i.i.d random variables \( Z = \{ Z_1, \ldots, Z_{|C|} \} \) and the set of i.i.d random variables \( U = \{ U_1, \ldots, U_{|C|} \} \) denote the sojourn times and the recruitment duration of the vehicles in \( C \). The semi-dynamic VC presented in Fig. 1 under the supervision of RP-VC is an SSe, referred to by SeSSVC = (\( \zeta, Q \)), where \( \zeta = \{ e^D(C_1), e^R(C_1) \} \cup \ldots \cup \{ e^D(C_j), e^R(C_j) \} \) for all vehicles and \( Q = Z \cup U \). Here, \( e^D(C_i) \) and \( e^R(C_i) \) are departure and recruitment events, defined below:

- \( e^D(C_i) \): Vehicle \( C_i \) departs the VC.
- \( e^R(C_i) \): Vehicle \( C_i \) completes its recruitment operation.

To clarify SeSSVC, consider an application \( A = (V, E) \), where \( V = \{ T_1, T_2, \ldots, T_k \} \) and \( E = \{ (1, 2), (1, 3), (2, 4) \} \). Also, let \( A \) be grouped by partition \( \mathcal{P}(A) = (G, E) \) into two groups \( G = \{ G_1, G_2 \} \), where \( G_1 = \{ T_1, T_2 \} \) and \( G_2 = \{ T_3, T_4 \} \). Further, assume that \( G_1 \) is deployed on vehicles \( C_1 \) and \( C_2 \), and \( G_2 \) is deployed on vehicles \( C_3 \) and \( C_4 \). Let \( Z = \{ Z_1, Z_2, Z_3, Z_4 \} \) and \( U = \{ U_1, U_2, U_3, U_4 \} \) denote the sojourn times and the recruitment duration of vehicles in \( C = \{ C_1, C_2, C_3, C_4 \} \). We have SeSSVC = (\( \zeta, Q \)), where \( \zeta = \{ e^D(C_1), e^R(C_1), \ldots, e^D(C_j), e^R(C_j) \} \). Fig. 4 depicts SeSSVC = (\( \zeta, Q \)), in which the departure events \( \{ e^D(C_1), e^D(C_2), e^D(C_3), e^D(C_4) \} \) and \( \{ e^R(C_1), e^R(C_2), e^R(C_3), e^R(C_4) \} \) are scheduled at time 0 to occur at times \( Z_1, Z_2, Z_3, \) and \( Z_4 \), respectively. The occurrence of events \( e^D(C_i) \) and \( e^D(C_i) \) results in scheduling recruitment events \( e^R(C_2) \) and \( e^R(C_4) \) at times \( t_1 \) and \( t_2 \), occurred at times \( t_1 + U_2 \) and \( t_2 + U_4 \). As can be seen from Fig. 4, the residual times of random variables can change after occurring each event. For instance, recruitment event \( e^R(C_2) \) is \( \beta \)-inhomogeneous with \( \beta = 3 \) since it is scheduled at time \( t_1 \) with initial recruitment duration \( U_1 \). After occurring event \( e^D(C_3) \) at time \( t_2 \), the residual recruitment duration of \( e^R(C_2) \) is \( U_2 - (Z_2 - Z_1) \). Finally, after occurring \( e^R(C_4) \) at time \( t_3 \), the residual recruitment duration of \( e^R(C_2) \) is \( U_2 - (Z_2 - Z_1) - U_4 \).

We next present an SMP model to characterize the dynamics of RP-VC in an SeSSVC.

2) **SMP Modeling of RP-VC in an SeSSVC:** Consider deployment \( D(\mathcal{M}(\mathcal{P}(A), t)) \) at time \( t \), where \( D(\mathcal{M}(\mathcal{P}(A), t)) = n \). Further, consider state set \( S = \{ S_0, S_1, \ldots, S_n, F \} \). We formalize the system behavior of RP-VC in an SeSSVC as an SMP model \( \{ X_t \in S \} \), depicted in Fig. 3(c). In this model, each state \( S_i \in S \) is defined as a set of computing vehicles utilized for processing each group \( G_h \in D(\mathcal{M}(\mathcal{P}(A), t)) \). Let \( C_i \) denote the set of \( i \) vehicles processing \( i \) different groups, where one of the vehicles of each group has departed the VC (i.e., each group is being processed by only one vehicle). Therefore, each vehicle \( C_i \in \hat{C}_i \) is a recruiter vehicle. Also, let \( C_{n-2i} \) denote the set of \( 2n - 2i \) vehicles processing \( n-i \) different groups, where each group is being processed by two vehicles. State \( S_0 = \hat{C}_1 \cup \hat{C}_{2n-1} \) is the initial state where each group \( G_{h_i} \in D(\mathcal{M}(\mathcal{P}(A), t)) \) has two vehicles. In state \( S_1 = \hat{C}_1 \cup \hat{C}_{2n-2} \), there is a group \( G_{h_i} \in D(\mathcal{M}(\mathcal{P}(A), t)) \) with only one vehicle, and each group \( G_{h_i} \in D(\mathcal{M}(\mathcal{P}(A), t)) \) \( \forall \) \( h_i \) has two vehicles. Generally, state \( S_i = \hat{C}_i \cup \hat{C}_{2n-2i} \) represents a situation that \( i \) groups \( \{ G_{h_1}, G_{h_2}, \ldots, G_{h_i} \} \) have one vehicle, and \( n-i \) groups, i.e., each group \( G_{h_i} \in D(\mathcal{M}(\mathcal{P}(A), t)) \) \( \forall \) \( h_i \) have one vehicle. Further, state \( F \in S \) refers to the failure of \( A \).

3) **Dynamics of the SMP Model of RP-VC in an SeSSVC:** Initially, RP-VC assigns two vehicles to each group of \( D(\mathcal{M}(\mathcal{P}(A), t)) \). Let \( C_{t_1} \) and \( C_{t_2} \) be two vehicles assigned to a group \( G_{h_i} \in D(\mathcal{M}(\mathcal{P}(A), t)) \). In state \( S_0 = \hat{C}_0 \cup \hat{C}_{2n} \), event \( e^D(C_{t_1}) \) occurs with probability \( q_{01} \) starting recruitment operation by \( C_{t_1} \). In this situation, the SMP transits to state \( S_1 = \hat{C}_1 \cup \hat{C}_{2n-2} \) since \( C_{t_1} \) has departed the VC and should be removed from \( \hat{C}_0 \). Besides, \( C_{t_1} \) now starts recruitment, which should be completed from \( \hat{C}_0 \), and added to \( S_1 \). In state \( S_1 \), event \( e^D(C_{t_2}) \) occurs with probability \( q_{11} \) before completing recruitment operation of \( C_{t_1} \) and the departure of any other vehicles, leading to the failure of the processing of \( A \). Conversely, in state \( S_1 \), event \( e^R(C_{t_2}) \) occurs with probability \( q_{21} \) and \( C_{t_2} \) successfully recruits a new vehicle to allocate to \( G_{h_i} \), leading to a transition of the SMP to state \( S_0 \) since there is no recruiter and all the groups have two vehicles. Further, let \( C_{t_2} \) and \( C_{t_3} \) be two vehicles assigned to group \( G_{h_i} \in D(\mathcal{M}(\mathcal{P}(A), t)) \). In state \( S_1 \), event \( e^D(C_{t_2}) \) occurs with probability \( q_{11} \) before any other events. In this situation, the SMP transits to state \( S_1 \), and \( C_{t_3} \) starts recruiting a new vehicle. In state \( S_1 \), \( C_{t_2} \) and \( C_{t_3} \) recruit new vehicles for groups \( G_{h_i} \) and \( G_{h_j} \), independently. The above events can happen in the other states as well.

4) **Complexity of the SMP Model of RP-VC:** In addition to considering the execution of dependent sub-tasks in RP-VC, which is fundamentally different from atomic application modeling in \( J_n \), the main differences between the SMP of RP-VC and \( J_n \) can be outlined as follows:

- In RP-VC, each group \( G_h \) has its own unique image/replica and is processed by two vehicles, e.g., \( C_{t_1} \) and \( C_{t_2} \), independently. If \( C_{t_1} \) departs the VC, only \( C_{t_2} \) can recruit a new vehicle to process sub-tasks of \( G_h \). As a result, \( i \) simultaneous active recruiters coexist in state \( S_i \). Moreover, in state \( S_i \), if a recruiter finishes its recruitment operation, the system will transit to state \( S_{i-1} \). However, in \( J_n \), whenever the only existing recruiter completes its recruitment, the SMP will transit to the initial state \( O_n \).
- In RP-VC, processing application \( A \) fails with probability \( b_i \) from each state \( S_i \), where \( 0 < b_i < 1 \) (due to the dependent sub-tasks structure of \( A \)). While, in \( J_n \), no failure occurs until reaching the last state \( O_n \), since \( n \) replications of the application are processed on \( n \) different vehicles.
by $\beta$-inhomogeneous SMP ($\beta$-SMP). As shown in Fig. 4, at each time instant $t$, residual occurring time of all $e_a \in \zeta$ can follow different distributions.

To analyze $R_P - V_n$ in an SeS$_V = (\zeta, Q)$ and consequently obtaining $C$-$MITF$ $(A, M((P(A, t), R(t), t))$, it is necessary to answer the following questions regarding a general SeS:

(Q3) What are the residual occurring times of events after occurring each event $e_a \in \zeta$?

(Q4) What is the value of $\beta$ for an $\beta$-inhomogeneous event?

To deal with (Q3), we introduce $(e)$-algebra as a general mathematical environment to ease conducting algebraic operations on random variables in an SeS.$^5$ To answer question (Q4) regarding SeS$_V$, we propose DT to disentangle $\beta$-SMP to a decomposed SMP (D-SMP).

IV. RELIABILITY ANALYSIS VIA EVENT STOCHASTIC ALGEBRA AND DECOMPOSITION THEOREM

We develop two new mathematical frameworks, which we refer to as event stochastic algebra ($(e)$-algebra) (Sec. IV-A) and decomposition theorem (DT) (Sec. IV-C). These frameworks are highly effective in analyzing a variety of dynamic systems, including the semi-dynamic VC of our interest under $R_P - V_n$.

Remark 1: Note that all of the subsequent mathematical modeling given in this section is among the first in the literature.

A. Event Stochastic Algebra ($(e)$-Algebra)

The pillars of $(e)$-algebra are built on top of two mathematical concepts that we introduce, event dynamic variable (EDV) and event dynamic list (EDL), defined as follows.

1) Event Dynamic Variable (EDV): An EDV represents the difference of two random variables. EDV is utilized to characterize stochastic behavior of a $\beta$-inhomogeneous event in an SeS. The name EDV stems from the fact that in our SeS the dynamics of the occurrence of the events are represented via the difference of random variables as discussed later. In this paper, we use EDV to define EDL exploited to formulate different states of D-SMP in Sec. IV-C.

Definition 10 (Event Dynamic Variable): Consider random variables $M > 0$ and $N > 0$. If $M > N$, random variable $V = M - N$ is named EDV, and its distribution $\alpha$ is called $\alpha$-distribution.

In the above definition, $M$ and $N$ can themselves be EDVs. Note that in the definition of EDV, the internal components of $V$, i.e., $M$ and $N$, make $V$ an EDV; otherwise, we refer to $V$ by simple random variable (SRV).

Characterizing an EDV. Consider an SeS $\Delta = (\zeta, Q)$. Also, let EDV $V = M - N$ be the residual time until occurring event $e_a \in \zeta$. Further, let $\alpha$ and $\alpha'$ be two $\alpha$-distributions. We characterize $V$ through three notions of duration, reducer, and pivot, defined as follows.

Definition 11 (Duration): The duration of $V = M - N$, shown by function $\hat{\Gamma}(V)$, is defined as follows:

$$\hat{\Gamma}(V) = \begin{cases} \hat{\Gamma}(M), & \text{if } V \sim \alpha, \\ V, & \text{if } V \text{ is SRV}, \\ 0, & \text{otherwise}. \end{cases}$$  

Intuitively, $\hat{\Gamma}(V)$ refers to the total time until the occurrence of event $e_a$. Referring to Fig. 4, $\hat{\Gamma}(U'_2) = U_2$.

Definition 12 (Reducer): The reducer of $V = M - N$, referred to by function $\hat{\delta}(V)$, is defined as follows:

$$\hat{\delta}(V) = \begin{cases} \hat{\delta}(M) + (N + \hat{\delta}(N)), & \text{if } V \sim \alpha, \\ 0, & \text{if } V \text{ is SRV}. \end{cases}$$  

To clarify the above definition, assume that event $e_a' \in \zeta$ is the latest event that has occurred before $e_a$. The reducer of $V$ is the latest passed from time 0 until occurrence of $e_a'$. As can be seen from Fig. 4, after occurring event $e_R(C_4)$, we have $\hat{\delta}(U'_2) = Z_3 + U_4$.

Definition 13 (Pivot): The pivot of $V = M - N$, referred to by function $\hat{\xi}(V)$, is defined as follows:

$$\hat{\xi}(V) = V + \hat{\delta}(V) - \hat{\Gamma}(V).$$  

Intuitively, the pivot of $V$ refers to the scheduling time of event $e_a$. As can be seen from Fig. 4, we have $\hat{\xi}(U'_2) = Z_1$. Considering above definitions, if $\hat{\delta}(V) = \hat{\xi}(V) = 0$, we refer to $V$ as SRV and its distribution is called simple distribution. Next, we introduce an operator to compare different EDVs.

Definition 14 (Pivotal Greater ($\preceq$)): Consider EDVs $V_1 \sim \alpha$ and $V_2 \sim \alpha'$, where $\hat{\Gamma}(V_1) \neq \hat{\Gamma}(V_2)$. The relations $V_1 \preceq V_2$ or $V_2 \preceq V_1$, read $V_2$ is pivotally greater than or equal to $V_1$, are defined as

$$V_2 \preceq V_1 \iff V_1 \preceq V_2 \iff \{\hat{\delta}(V_2) = \hat{\delta}(V_1) \& \hat{\xi}(V_1) \leq \hat{\xi}(V_2)\}.$$  

By convention $V_1 \preceq V_2$, if $V_1$ is an EDV and $V_2$ is a SRV.

Properties of EDVs. We define the following two properties, i.e., conditional closure and subtraction inequality, which are used later to do algebraic operations on EDVs.

Lemma 1 (Conditional Closure): Consider EDVs $V_1 \sim \alpha$ and $V_2 \sim \alpha'$, where $V_1 \preceq V_2$. $S_1 = V_1 - V_2$ is an EDV, calculated as follows, if $S_2 > 0$:

$$S_1 = V_1 - V_2 = (\hat{\Gamma}(V_1) - (\hat{\xi}(V_2) - \hat{\xi}(V_1))) - \hat{\Gamma}(V_2),$$  

and $S_2 = V_2 - V_1$ is an EDV, calculated as follows, if $S_2 > 0$:

$$S_2 = V_2 - V_1 = (\hat{\Gamma}(V_1) - (\hat{\xi}(V_2) - \hat{\xi}(V_1))),$$  

where $\hat{\Gamma}(V_1) > (\hat{\xi}(V_2) - \hat{\xi}(V_1)) > 0$.

Proof: See Appendix A.

Lemma 2 (Subtraction Inequality): Consider EDVs $V_1 \sim \alpha$ and $V_2 \sim \alpha'$, where $V_1 \preceq V_2$. Also, consider $V'$, where $V'$ can be EDV or SRV, and $V' \preceq V_1, V_2$. The following inequality holds.

$$V_1 \preceq V' \preceq V_2' \iff \{V_1 \preceq V' \& V' \preceq V_2\}.$$  

provided that

$$\forall V_i, V_j \in \mathcal{L}(\alpha)\langle n \rangle,$$  

$\alpha$-distribution, referred to by $\mathcal{L}(\alpha)\langle n \rangle = \{V_1, V_2, \ldots, V_n\}$, is called an EDL if $V_i \preceq V_j$, where $1 \leq i < j \leq n$.  

$^5$Note that $(e)$-algebra is not used to calculate the subtraction of random variables, studied through convolution, and probability characteristic function [38].
We let \( L(\alpha)(n)(i) \) denote the \( i \)th member of the list.

**Operators To Manipulate EDLs.** We introduce three essential operators, namely, *pivotally greater, concatenation,* and *subtraction,* to manipulate EDLs. First, we extend the definition of operator \( \preceq \) to compare different EDLs as follows:

**Definition 16:** Consider two EDLs \( L_1(\alpha)(n) \) and \( L_2(\alpha)(m) \). \( L_1(\alpha)(n) \preceq L_2(\alpha)(m) \) if

\[
V_i \preceq V_j, \forall V_i \in L_1(\alpha)(n), \forall V_j \in L_2(\alpha)(m).
\]

We then define the concatenation operator as follows, enabling us to concatenate EDLs.

**Definition 17 (Concatenation (\( \odot \))):** Let \( L_1(\alpha)(n) = [V_1, V_2, \ldots, V_n] \) and \( L_2(\alpha)(m) = [V'_1, V'_2, \ldots, V'_m] \) be two EDLs. The concatenation of \( L_1(\alpha)(n) \) and \( L_2(\alpha)(m) \) is defined as follows:

\[
L_1(\alpha)(n) \odot L_2(\alpha)(m) = [V_1, V_2, \ldots, V_n, V'_1, V'_2, \ldots, V'_m],
\]

where \( L_1(\alpha)(n) \preceq L_2(\alpha)(m) \).

Finally, we introduce the following subtraction operator, utilized to subtract EDLs from an EDV.

**Definition 18 (Subtraction (\( \ominus \))):** Consider an EDV \( L(\alpha)(n) = [V_1, V_2, \ldots, V_n] \) and an EDV \( S(\alpha') \). The substraction of \( L(\alpha)(n) \) and \( S \), denoted by \( L(\alpha)(n) \ominus S \), is defined as follows:

\[
L(\alpha)(n) \ominus S = [V_1 - S, \ldots, V_n - S].
\]

By convention, if \( S = V_i \), \( V_i \) will be removed from \( L(\alpha)(n) \).

**Special Class of EDL.** Utilizing concatenation operators, we next aim to introduce a class of EDLs, whose EDVs follow two different distributions.

**Definition 19 (Heterogeneous Event Dynamic List):** Let \( L_1(\alpha)(n) = [V_1, V_2, \ldots, V_n] \) and \( L_2(\alpha')(m) = [V'_1, V'_2, \ldots, V'_m] \) be two EDLs, where \( \alpha \) and \( \alpha' \) are different. Heterogeneous EDL (H-EDL), shown by \( H(\alpha; \alpha')(\nu; m) \), is defined as follows:

\[
H(\alpha; \alpha')(\nu; m) = L_1(\alpha)(\nu) \odot L_2(\alpha')(m),
\]

and

\[
H(\alpha; \alpha')(\nu; m) = L_1(\alpha)(\nu) \odot L_2(\alpha')(m),
\]

where \( L_1(\alpha)(\nu + m) \) is an EDL with \( \nu + m \) EDVs.

We next aim to utilize the notation of \( (\varepsilon) \)-algebra to model \( \mathbb{R}^+ - \mathbb{V}_C \). Then we present decomposition theorem to transform \( \beta \)-SMP, presented in Sec. III-C, to D-SMP.

**B. Analyzing the Dynamics of \( \mathbb{R}^+ - \mathbb{V}_C \) Through \( (\varepsilon) \)-Algebra**

Let \( C \) denote the set of all the vehicles utilized to process an application \( A \). Also, consider \( \mathbb{S}_C \) as \( \{\zeta, \emptyset\} \), presented in Definition 9, where \( \zeta = \{e^D(C_i), e^R(C_i)\} \) for \( i \in \emptyset \), and \( \emptyset = \mathbb{Z} \). Let i.i.d random variable \( Z \in \mathbb{Z} \) follow general distribution \( \mathbb{D}_Z(z) \). Also, let i.i.d random variable \( U \in \mathbb{U} \) follow a general distribution \( \mathbb{S}_U(u) \). We next aim to answer question (Q3) in Sec. III-C4 through utilizing the framework of \( (\varepsilon) \)-algebra. We first introduce a set of special EDVs, utilized to characterize the residual sojourn time (i.e., the residual time until the departure of the respective vehicle from the VC) and residual recruitment duration (i.e., the residual time until completing recruitment operation by the respective vehicle) of each vehicle after the occurrence of event \( e \in \zeta \).

**Special EDVs and Their Relationships.** Let \( Z_1, Z_2, \) and \( U \) be three independent random variables, where \( Z_1 \) and \( Z_2 \) follow general distribution \( \mathbb{D}_Z(z) \), and \( U \) has a general distribution \( \mathbb{S}_U(u) \). We define the following special EDVs, utilized to obtain our main upshots in Sec. IV-C.

**Definition 20 (\( \mathbb{X} \) Distribution):** Let \( Z_1' = Z_1 - Z_2 \) and \( Z_2' = Z_2 - (Z_2 + U) \) be two EDVs. The distribution of \( Z_1' \) and \( Z_2' \) are called first and second orders \( \mathbb{X} \) shown by \( Z_1' \sim \mathbb{X} \) and \( Z_2' \sim \mathbb{X} \).

**Definition 21 (\( \varphi \) Distribution):** Let \( W = U - Z' \) be an EDV, where \( Z' \sim \mathbb{X} \). The distribution of \( W \) is called \( \varphi \).

**Definition 22 (\( \psi \) Distribution):** Let \( Y = U - W \) be an EDV, where \( W \sim \varphi \). The distribution of \( Y \) is called \( \psi \).

**Definition 23 (\( \gamma \) Distribution):** Let \( X = W - U \) be an EDV, where \( W \sim \varphi \). The distribution of \( X \) is called \( \gamma \).

We characterize the relations between above EDVs through decomposability and absorbency properties, utilized to introduce decomposition theorem later in Sec. IV-C.

**Proposition 1 (Decomposability):** Consider two EDVs \( V_{1} \sim \alpha \) and \( V_{2} \sim \alpha' \), where \( \alpha \) and \( \alpha' \) are (i) \( \alpha, \alpha' = \varphi \) or (ii) \( \alpha, \alpha' = \psi \) or (iii) \( \alpha, \alpha' = \gamma \) or (iv) \( \alpha = \psi, \alpha' = \gamma \) or (v) \( \alpha = \gamma, \alpha' = \psi \). EDV \( S_1 = V_1 - V_2 \) follows \( \gamma \) distribution if \( S_1 > 0 \) and \( EDV \) \( S_2 = V_2 - V_1 \) follows \( \psi \) distribution if \( S_2 > 0 \).

**Proof:** See Appendix C.

To clarify the main implication of decomposability, Fig. 5(a) illustrates an example of three \( \varphi \) EDVs \( V_1, V_2, \) and \( V_3 \), where \( V_1 \leq V_2 \leq V_3 \). Decomposability states that the distributions of different substractions of these three EDVs, i.e., \( \{V_1 - V_3, V_2 - V_3\}, \{V_1 - V_2, V_2 - V_3\}, \{V_2 - V_1, V_3 - V_1\} \) follow three different permutations of \( \gamma \) and \( \psi \), i.e., \( \{\gamma, \gamma\}, \{\gamma, \psi\}, \{\psi, \psi\} \).

**Proposition 2 (Absorbency):** Consider four EDVs \( Z_1' \sim \mathbb{X}, Z_2' \sim \mathbb{X}, W \sim \varphi \) and \( V \sim \alpha \), where \( \alpha \) can be \( \psi \) or \( \gamma \), \( Z_1' \sim \mathbb{W} \), and \( Z_2' \sim \mathbb{V} \). EDVs \( S_1 = W - Z_1' \) and \( S_2 = V - Z_2' \) follow \( \varphi \) distribution if \( S_1 > 0 \) and \( S_2 > 0 \). Conversely, \( S_3 = Z_1' - W \) and \( S_3 = Z_2' - V \) follow \( \mathbb{X} \) distribution if \( S_3 > 0 \) and \( S_4 > 0 \).

**Proof:** See Appendix D.

Referring to Fig. 5(b) as an example, consider EDVs \( Z_1' \sim \mathbb{X}, Z_2' \sim \mathbb{X}, W \sim \varphi, V \sim \psi \), and \( V \sim \gamma \). Absorbency implies that \( W - Z_1', V_1 - Z_2', \) and \( V_2 - Z_2' \) result in the same ED following \( \varphi \) distribution. Decomposability and absorbency are utilized to prove Proposition 3 and Proposition 4, introduced below, characterizing the residual occurring times of the departure and recruitment events in \( 
\)
Special EDLs and Their Relationships. In the following, we present an example to motivate Proposition 3 and Proposition 4. Let $C$ denote the set of vehicles utilized for processing application $A$. Also, let $Z$ and $U$ denote two sets of random variables, where $Zt \in Z$ follows a general distribution $D_Z(z)$ referring to the sojourn time of vehicle $C_t \in C$ and $Ut \in U$ follows general distribution $R_{U(t)}$ referring to the recruitment duration of vehicle $C_t$. Consider deployment $D \left( \mathcal{M}(\mathcal{P}(A), t) \right)$, where $\left| D \left( \mathcal{M}(\mathcal{P}(A), t) \right) \right| = n$. Further, consider time instances $\tau = \{t_0, t_1, t_2\}$, where $t_0$ is the time that the processing of application $A$ started. Assume that $D \left( \mathcal{M}(\mathcal{P}(A), t) \right)$ stays unchanged for $t \in \tau$ (i.e., the number of groups is fixed). At time $t_0$, $2n$ vehicles are allocated to the groups of $D \left( \mathcal{M}(\mathcal{P}(A), t) \right)$. Assume that group $G_{h_1}$ is deployed on vehicles $C_{t_1} \in C$ and $C'_{t_1} \in C$, and group $G_{h_2}$ is deployed on vehicles $C_{t_2} \in C$ and $C'_{t_2} \in C$. Further, assume that vehicle $C_{t_2}$ departs the VC at time $t_2$, $Z_{t_2} \in Z$ should be subtracted from the sojourn times of other vehicles because $Z_{t_2}$ time units have passed since the start of processing $A$ (i.e., $t_0$). Accordingly, we model the residual sojourn times of the vehicles (i.e., $Z_{t_1} - Z_{t_2}$), where $Z_{t_1} \in Z$ as an EDL $\mathcal{L}(\gamma)(2n-1)$. Further, at time $t_1$, $C_{t_1}'$ starts recruiting a new vehicle. Likewise, assume that vehicle $C_{t_1}'$ departs the VC at time $t_2$. Hence, $Z_{t_2} - Z_{t_1}$ should be subtracted from the residual sojourn times and residual recruitment duration of the other vehicles because $Z_{t_2} - Z_{t_1}$ time units have passed since $t_1$. Consequently, we have $\mathcal{L}(\gamma)(2n-2) = \mathcal{L}(\gamma)(2n-1) \ominus (Z_{t_2} - Z_{t_1})$ and $U_{t_1} - (Z_{t_2} - Z_{t_1})$. Also, at time $t_2$, vehicle $C'_{t_2}$ starts recruiting a new vehicle. It is straightforward to verify that the residual recruitment times of recruiters $C_{t_1}'$ and $C'_{t_2}$ (i.e., $U_{t_1} - (Z_{t_2} - Z_{t_1})$ and $U_{t_2}'$) can be modeled as an H-EDL $\mathcal{H}(\psi;\gamma)(1;1)$. In the following, we generalize the above-mentioned example. We first define the following two special H-EDLs.

1) $\mathcal{H}(\gamma;\psi)(n;m)$: An H-EDL with $n + m$ elements, in which the first $n$ elements follow $\gamma$ distribution and the last $m$ elements follow $\psi$ distribution.

2) $\mathcal{L}(\gamma;\psi)(n-1;1)$: An H-EDL with $n$ elements, in which the first $n-1$ elements follow $\gamma$ distribution and the last one follows a general distribution $R$.

Accordingly, we obtain the following important technical results, utilized to prove decomposition theorem later. It is worth mentioning that decomposition theorem will be utilized to define the states of D-SMP model of $\mathcal{R}P-\mathcal{V}C_n$.

Proposition 3: Consider two EDLs $\mathcal{H}(\psi;\gamma)(n-1;1)$ and $\mathcal{L}(\gamma;\psi)(n-1;1)$, where $\mathcal{L}(\gamma;\psi)(n-1;1) \supseteq \mathcal{H}(\psi;\gamma)(n-1;1)$. The following statements hold:

1) If $\mathcal{H}(\psi;\gamma)(n-1;1)(k) > \mathcal{H}(\psi;\gamma)(n-1;1)(i)$ for $i \neq k$ and $\mathcal{L}(\gamma;\psi)(n-1;1)(r) > \mathcal{H}(\psi;\gamma)(n-1;1)(i)$, then

$$\mathcal{H}(\psi;\gamma)(n-1;1)(k) \ominus \mathcal{H}(\psi;\gamma)(n-1;1)(i) = \mathcal{H}(\psi;\gamma)(n-1;1)(i),$$

and

$$\mathcal{L}(\gamma;\psi)(n-1;1)(r) \ominus \mathcal{H}(\psi;\gamma)(n-1;1)(i) = \mathcal{L}(\gamma;\psi)(n-1;1)(i),$$

(20)

where $\mathcal{H}(\psi;\gamma)(n-1;1)(k)$ refers to the $k$th element of EDL $\mathcal{H}(\psi;\gamma)(n-1;1)$.

2) If $\mathcal{H}(\psi;\gamma)(n-1;1)(k) > \mathcal{L}(\gamma;\psi)(n-1;1)(r)$, then

$$\mathcal{H}(\psi;\gamma)(n-1;1)(k) \ominus \mathcal{L}(\gamma;\psi)(n-1;1)(r) = \mathcal{L}(\gamma;\psi)(n-1;1)(i),$$

and

$$\mathcal{L}(\gamma;\psi)(n-1;1)(r) \ominus \mathcal{L}(\gamma;\psi)(n-1;1)(r) = \mathcal{L}(\gamma;\psi)(n-1;1)(r),$$

(22)

where $\mathcal{H}(\psi;\gamma)(n-1;1)(k) \ominus \mathcal{L}(\gamma;\psi)(n-1;1)(r)$ refers to the $k$th element of EDL $\mathcal{H}(\psi;\gamma)(n-1;1)$.

Proof: See Appendix E.

Proposition 4: Consider two EDLs $\mathcal{H}(\gamma;\psi)(n;1)$ and $\mathcal{L}(\gamma;\psi)(n;1)$, where $\mathcal{L}(\gamma;\psi)(n;1) \supseteq \mathcal{H}(\gamma;\psi)(n;1)$. The following statements hold:

1) If $\mathcal{H}(\gamma;\psi)(n;1)(k) > \mathcal{H}(\gamma;\psi)(n;1)(i)$ for $i \neq k$ and $\mathcal{L}(\gamma;\psi)(n;1)(r) > \mathcal{H}(\gamma;\psi)(n;1)(i)$, we have

$$\mathcal{H}(\gamma;\psi)(n;1)(k) \ominus \mathcal{H}(\gamma;\psi)(n;1)(i) = \mathcal{L}(\gamma;\psi)(n;1)(i),$$

and

$$\mathcal{L}(\gamma;\psi)(n;1)(r) \ominus \mathcal{H}(\gamma;\psi)(n;1)(i) = \mathcal{L}(\gamma;\psi)(n;1)(i),$$

(24)

2) If $\mathcal{H}(\gamma;\psi)(n;1)(k) > \mathcal{L}(\gamma;\psi)(n;1)(r)$, we have

$$\mathcal{H}(\gamma;\psi)(n;1)(k) \ominus \mathcal{L}(\gamma;\psi)(n;1)(r) = \mathcal{L}(\gamma;\psi)(n;1)(r) + m, $$

and

$$\mathcal{L}(\gamma;\psi)(n;1)(r) \ominus \mathcal{L}(\gamma;\psi)(n;1)(r) = \mathcal{L}(\gamma;\psi)(n;1)(r),$$

(26)

where $\mathcal{L}(\gamma;\psi)(n;1)(k) \ominus \mathcal{L}(\gamma;\psi)(n;1)(r)$ refers to the $k$th element of EDL $\mathcal{H}(\gamma;\psi)(n;1)$.

Proof: See Appendix F.

To understand the implications of the results of the above propositions, we consider an example of the result given by (20) in Proposition 3 (the same argument holds for the other parts of Proposition 3 and Proposition 4). Assume that there are $m$ computing vehicles in a semi-dynamic VC, from which $n < m$ vehicles are busy recruiting new vehicles. Further, assume that $\mathcal{H}(\psi;\gamma)(n-1;1)$ is the list of residual recruitment times of recruiters. The result of (20) states that if the $i$th recruiter completes its recruitment operation successfully, the residual recruitment times of other vehicles is $\mathcal{H}(\psi;\gamma)(i-1;1)$, which represents a direct result of decomposability (Proposition 1).

C. Decomposition Theorem

In this section, we propose the decomposition theorem (DT) based on the notations of EDV and EDL developed in Sec. IV. Specifically, DT enables us to disentangle $\beta$-SMP to a decomposed SMP model (D-SMP). The key advantage of D-SMP compared to $\beta$-SMP is that, in D-SMP, calculating transition probabilities and expected sojourn time of the process in each state are much more straightforward. We first present the following definition.

Definition 24 (Order of Recruiter): For $0 \leq i \leq n$, let $\mathcal{C}_i$ denote the list of all $i$ recruiter vehicles in state $S_i \in \mathcal{S}$, defined in Sec. III-C. Further, let EDV $V_j$ be residual recruitment time of vehicle $C_j \in \mathcal{C}_i$. The order of recruiter $C_j \in \mathcal{C}_i$, shown by $O(C_j)$, is defined as follows:

$$O(C_j) = \sum_{j=1, j \neq i}^{n} 1_{V_j \leq V_i}.$$  

(28)
We next present decomposition theorem (DT).

Theorem 1 (Decomposition): Consider $S_e S_{VC} = (\zeta, Q)$ presented in Sec. III-C1, and deployment $D(\mathcal{M}(\mathcal{P}(A), t))$ at time $t$, where $\left| D(\mathcal{M}(\mathcal{P}(A), t)) \right| = n$. Further, consider $\beta$-SMP presented in Fig. 3(c) with state space $S = \{S_0, S_1, \ldots, S_n, F\}$. Let $X \in S$ and $X' \in S$ denote the current state and next state of $\beta$-SMP, respectively. Further, let $\varepsilon_{X} \in \varepsilon$ denote the event that occurred at current state $X$. Each state $S_i$, except for the $S_n$, can be decomposed into $i+2$ states, referred to as $H$-state. Mathematically,

$$S_0 \equiv \{S_{0.0} \ \text{initial state}, \}$$

$$X' = S_0 \equiv \{S_{0.1} \ \text{if } X = S_1, e_X = e_C(C_{i}, \forall C_i \in C_1), \}$$

$$S_{i.0} \ \text{if } X = S_{i-1}, e_X = e_C(C_{i}), \forall C_i \in C_2$$

$$S_{i.1} \ \text{if } X = S_{i+1}, e_X = e_R(C_{i}), O(C_{i}) = j, \forall C_i \in C_{i+1},$$

and

$$X' = S_n \equiv \{S_{n.0} \ \text{if } X = S_{n-1}, e_X = e_D(C_{i}), \forall C_i \in C_2, \}$$

where $S_{0.0}$ is

$$S_{0.0} = \left\{(Z_1, Z_2, \ldots, Z_{2n}), [\dot{\varepsilon}] \right\},$$

$S_{0.1}$ is

$$S_{0.1} = \left\{L_2(X)(2n), [\dot{\varepsilon}] \right\},$$

$S_{i.0}$, for $1 \leq i \leq n$, is

$$S_{i.0} = \left\{L_1(X)(2n-i), H_1^{(e,R)}(i-1;1) \right\},$$

and $S_{i,j}$, for $1 \leq j \leq i+1$ and $1 \leq i \leq n-1$, is

$$S_{i,j} = \left\{L_2(X)(2n-i), H_2^{(e,R)}(j-1;1-(j-1)) \right\}.$$ 

Moreover, we have $L_1(X)(2n-i) \preceq H_1^{(e,R)}(i-1;1)$ and $L_2(X)(2n-i) \preceq H_2^{(e,R)}(j-1;1-(j-1))$.

In (33), $Z_1, Z_2, \ldots, Z_{2n}$ are i.i.d random variables following general distribution $\mathcal{D}_2(\varepsilon)$ referring to the residency time of vehicles, and $[\dot{\varepsilon}]$ refers to the empty list. In (35) and (36), $L_1(X)(2n-i)$ and $L_2(X)(2n-i)$ are two lists of EDVs referring to the residual sojourn times of $2n-i$ vehicles processing the groups of $\mathcal{P}(A)$ in states $S_{0.0}$ and $S_{0.1}$, respectively. Further, $H_1^{(e,R)}(i-1;1)$ and $H_2^{(e,R)}(j-1;1-(j-1))$ are lists of EDVs referring to the residual recruitment duration of i recruiters in states $S_{0.0}$ and $S_{0.1}$, respectively.

Proof: See Appendix G.

Decomposition theorem implies that, for all $H$-states $S_{i,j}$, where $0 \leq i \leq n$ and $0 \leq j \leq i+1$, the departure of a vehicle leads to the transition to $H$-state $S_{i+1,0}$, which is a direct result of absorbency property presented in Proposition 2. Intuitively, in this situation, we say that $S_{i+1,0}$ absorbs the process from $H$-states $S_{i,j}$. Further, due to decomposability in Proposition 1, if recruiter $C_k$, for $1 \leq k \leq i$, completes its recruitment, D-SMP will transit to $H$-state $S_{i+1,k}$. In other words, from each $H$-state $S_{i,j}$, considering which recruiter completes its recruitment before any other event, D-SMP will transit to $k$ different $H$-states. Further, in all of the $H$-states, the departure of a recruiter leads to a transition to the failure state.

In the reminder of this paper, we refer to the first element of $S_{0.0}$ and $S_{i,j}$ by $S_{0.0}^{e} = L_2(X)(2n-i)$ and $S_{i,j}^{e} = L_2(X)(2n-i)$, respectively. Likewise, we refer to the second element of $S_{i.0}$ and $S_{i,j}$ by $S_{0.0}^{r} = H_1^{(e,R)}(i-1;1)$ and $S_{i,j}^{r} = H_1^{(e,R)}(j-1;1-(j-1))$, respectively. Using DT, we next aim to introduce decomposed SMP model (D-SMP) of $\mathcal{R} \setminus \mathcal{V}_{C_n}$.

D. Decomposed Semi-Markov Process (D-SMP) of $\mathcal{R} \setminus \mathcal{V}_{C_n}$

According to DT, each state $S_i$ of $\beta$-SMP presented in Fig. 3(c) can be decomposed into $i+2$ $H$-states resulting in the D-SMP model of $\mathcal{R} \setminus \mathcal{V}_{C_n}$ depicted in Fig. 6. In D-SMP model, there are $(n+1)(n+2) + 1$ different $H$-states, each of which is denoted by $S_{i,j}$, as defined in Theorem 1. We group all the $H$-states indexed with $i$ into a state set denoted by $S_i$ (i.e., $S_i = \{S_{i,0}, S_{i,1}, \ldots, S_{i,j}, \ldots, S_{i,n+1}\}$ for $0 \leq i \leq n$, $0 \leq j \leq i+1$). Also, state set $S_n = \{S_{n,0}\}$ has only one $H$-state. We denote the overall set of all the states by $\Psi = \left\{\bigcup_{i=0}^{n} S_i \right\} \cup \{F\}$, where $F$ indicates the failure of processing application $A$. We next aim to model the dynamics of D-SMP below.

Transition Probabilities of D-SMP. To characterize the dynamics of D-SMP, we obtain transition probabilities below. Let $p_{i,j}^{r}, g_{i,j}$, and $b_{i,j}$ be the transition probabilities from state $S_{i,j}$ to $S_{i-1,k}$, from state $S_{i,j}$ to $S_{i+1,0}$, and from state $S_{i,j}$ to $F$, respectively. We first specify $p_{i,j}^{r}$, which is equal to the probability that the recruiter $k$ completes its recruitment operation before any other events. Considering $S_{i,j} \in S_i$, presented in (31), the general expression for $p_{i,j}^{r}$ is as follows:

$$p_{i,j}^{r} = \Pr [S_{i,j}^{r}(k) < \min_{1 \leq r \leq 2n-i} \left\{ \min_{1 \leq h \leq i, h \neq k} S_{i,j}^{r}(h) \right\}],$$

where $0 \leq i \leq n$, $0 \leq j \leq i+1$, and $1 \leq k \leq i$. Further, $S_{i,j}^{r}(k)$, $S_{i,j}^{r}(h)$, and $S_{i,j}^{r}(r)$ refer to the residual recruitment time of vehicle $k$, residual recruitment time of vehicle $h$, and residual residency time of vehicle $r$, respectively. Further, since there are $2n-i$ vehicles in state $S_{i,j}$, of which $i$ vehicles are recruiter, and also $q_{i,j} = b_{i,j} + \sum_{k=1}^{i} b_{i,j}$, we have

$$b_{i,j} = \frac{i}{2n-i} \times \left(1 - \sum_{k=1}^{i} p_{i,j}^{r,k} \right),$$

and

$$q_{i,j} = \frac{2(n-i)}{2n-i} \times \left(1 - \sum_{k=1}^{i} p_{i,j}^{r,k} \right),$$
where \( p_{0,0}^k = p_{0,1}^k = 0 \) because all of the groups have two vehicles in states \( S_{0,0} \) and \( S_{0,1} \). The final derivation of \( p_{i,j}^k \) is presented later upon assuming exponential distribution of the sojourn times and recruitment duration of the vehicles (Sec. V-B), which is a common assumption [14], [17]. Nevertheless, general expressions for \( p_{i,j}^k \) is obtained in Appendix N under arbitrary distributions. Let \( R(t) = \Theta(M(P(A), t)) \), (see (4)). Relying on D-SMP presented in Fig. 6, we calculate \( C-MTTF(A, M(P(A), t), R(t), t) \) in the next section.

V. CALCULATING \( C-MTTF(A, M(P(A), t), R(t), t) \)

In this section, we derive the closed-form expression of \( C-MTTF(A, M(P(A), t), R(t), t) \), when \( R(t) = 0 \) and \( t = 0 \) (i.e., all the groups have two vehicles), resembling the start of application execution (Sec. V-A) while \( t > 0 \) is left as future work. Finally, we study a special case of \( RP-VC_n \), where sojourn times and recruitment duration of the vehicles follow the exponential distribution (Sec. V-B).

A. Closed-Form of \( C-MTTF(A, M(P(A), t), R(t) = 0, t = 0) \)

We first present the following technical result to calculate \( \mathbb{E} [\hat{Q}_{i,j}] \), utilized to obtain \( C-MTTF(A, M(P(A), t), R(t) = 0, t = 0) \).

**Lemma 3:** Let D-SMP be in state \( S_{i,j} \). Also, let random variable \( \hat{W}_{i,j} \) denote the sojourn time in state \( S_{i,j} \). The expected value of the time until failure from state \( S_{i,j} \) is given as follows:

\[
\mathbb{E} [\hat{Q}_{i,j}] = \mathbb{E} [\hat{W}_{i,j}] + \mathbb{E} [\hat{Q}_{i+1,0}] q_{i,j} + \sum_{k=1}^{n} \mathbb{E} [\hat{Q}_{i-1,k}] p_{i,k,j}, \quad 0 \leq i \leq n, 0 \leq j \leq i + 1,
\]

where \( \mathbb{E} [\hat{Q}_{n+1,0}] \triangleq 0 \), \( p_{i,k,j} \) is given by (39), and \( q_{i,j} \) can be calculated by (37).

**Proof:** See Appendix H.

Using Lemma 3, we obtain the closed-form of \( C-MTTF(A, M(P(A), t), R(t) = 0, t = 0) \) in the following key theorem.

**Theorem 2 (C-MTTF \((A, M(P(A), t), R(t) = 0, t = 0)\))**

Under \( RP-VC_n \), \( C-MTTF(A, M(P(A), t), R(t) = 0, t = 0) = \mathbb{E} [\hat{Q}_{0,0}] \), where

\[
\mathbb{E} [\hat{Q}_{0,0}] = \mathbb{E} [\hat{W}_{0,0}] + \sum_{i=1}^{n} \beta(i) \mathbb{E} [\hat{W}_{i,0}] + \sum_{i=1}^{n} \sum_{j=1}^{i} \beta(i) \mathbb{E} [\hat{W}_{i-1,j}] A(i + 1, j).
\]

(41)

In (41), \( \beta(i) \) is given by

\[
\beta(i) = \prod_{k=1}^{i} \alpha(k),
\]

where

\[
\alpha(k) = \frac{q_{k-1,0}}{1 - \left( \sum_{j=1}^{k} q_{k-1,j} A(k + 1, j) \right)}.
\]

(43)

Also, function \( A(\cdot, \cdot) \) is a recursive function defined as

\[
A(i, k) = \alpha(i) \sum_{j=1}^{i} p_{i-1,j}^k A(i + 1, j) + p_{i-1,0}^k.
\]

(44)

**Proof:** See Appendix I.

As can be seen from Theorem 2, calculation of \( C-MTTF(A, M(P(A), t), R(t) = 0, t = 0) \) requires computing \( \mathbb{E} [\hat{W}_{i,j}] \), which is carried out as follows. The expected time that D-SMP resides in H-state \( S_{i,j} \in S_i \) is equivalent to the expected value of

\[
\min \left\{ \min_{1 \leq r \leq 2n-i} \left\{ \hat{S}_{i,j}^{S_{ij}}(r) \right\}, \min_{1 \leq h \leq i} \left\{ \hat{S}_{i,j}^{Rec}(h) \right\} \right\},
\]

capturing a situation where a vehicle departs the VC or a recruiter completes its recruitment. As a result, we have

\[
\mathbb{E} [\hat{W}_{i,j}] = \int_{0}^{\infty} \Pr \left[ \min \left\{ \min_{1 \leq r \leq 2n-i} \left\{ \hat{S}_{i,j}^{S_{ij}}(r) \right\}, \min_{1 \leq h \leq i} \left\{ \hat{S}_{i,j}^{Rec}(h) \right\} \right\} > t \} dt.
\]

(45)

In next section, we compute \( \mathbb{E} [\hat{W}_{i,j}] \) (see (50)) for a version of \( RP-VC_n \), where sojourn times and recruitment duration of vehicles follow exponential distribution, which is a common assumption [17]. We further generalize our results to the case, where sojourn times and recruitment duration of vehicles follow arbitrary distributions in Appendix N.

B. \( RP-VC_n \) With Exponentially Distributed Sojourn Time and Recruitment Duration

We present the final theoretical results of this paper for the specific methodology \( RP-VC_n \sim e \), which refers to \( RP-VC_n \) when the recruitment duration follows exponential distribution with parameter \( \lambda_u \). We obtain the following result, which demonstrates that under \( RP-VC_n \sim e \), the residual sojourn and recruitment times of the vehicles follow exponential distribution.

**Corollary 1:** Let two random variables \( Z_i^r \) and \( U_i^r \), for \( 1 \leq k \leq 2n - i \) and \( 1 \leq r \leq i \), denote the residual sojourn and recruitment times of the vehicles in H-state \( S_{i,j} \). Under \( RP-VC_n \sim e \), if D-SMP is in H-state \( S_{i,j} \), random variables \( Z_i^r \) and \( U_i^r \) follow exponential distribution with parameters \( \lambda_z \) and \( \lambda_u \), respectively.
where and Also, \( A(i) \) is calculated by

\[
A(i) = \frac{p_{i-1,0}^1}{1 - i \times q_{i-1,0} \times A(i + 1)},
\]

and \( A(n + 1) = p_{n,0}^1 \).

\[ E \left[ Q_{0,0} \right] = \mathbb{E} \left[ \hat{W}_{0,0} \right] (1 + \beta'(1)A'(2)) + \beta'(n) \mathbb{E} \left[ \hat{W}_{n,0} \right] + \sum_{i=1}^{n-1} \mathbb{E} \left[ \hat{W}_{i,0} \right] (i+1)\beta'(i+1)A'(i+2) + \beta'(i)) \text{.} \tag{46} \]

Proof: See Appendix K.

To obtain the final result of (46), we calculate \( \mathbb{E} [\hat{W}_{i,0}] \) below.

Corollary 2: Considering methodology \( \text{RP–VC}_n \sim \text{e} \), the expected sojourn time in state \( S_{i,j} \) is given by

\[
\mathbb{E} \left[ \hat{W}_{i,0} \right] = \frac{1}{(2n - i)\lambda_u + i\lambda_u} \text{.} \tag{50} \]

Proof: See Appendix L.

We next aim to compute \( p_{i,0}^1 \), utilized to obtain (49), (47), and (48). \( q_{i-1,0} \) in (49) and (48) is computed based on (39).

Corollary 3: Considering methodology \( \text{RP–VC}_n \sim \text{e} \), the probability of transition from \( S_{i,0} \) to \( S_{i-1,0} \) is given by

\[
p_{i,0}^1 = \frac{\lambda_u}{(2n - i)\lambda_z + i\lambda_u} \text{.} \tag{51} \]

Proof: See Appendix M.

Replacing (50) and (51) back in Theorem 3 provides the value of \( C\text{-MTTF}(A|M(P(A), t), R(t) = 0, t = 0) \) and concludes the mathematical derivations of this paper. For a comprehensive case study of \( \text{RP–VC}_n \) refer to Appendix O.

C. Generalizability of \( \langle e \rangle \)-Algebra and DT

Our mathematical derivations can be utilized in similar cloud-based computing paradigms for the purpose of reliability analysis and fault-tolerant design. For example, authors in [15] have presented a strategy, referred to by delay-sensitive and reliable (DSR) placement, aiming at improving application placement availability in cloud-based systems. DSR aims to find an optimal sub-tasks placement \( P \) (i.e., placing virtual machines that process the application sub-tasks on the cloud servers) to minimize the number of utilized servers for processing application, while satisfying the application availability (i.e., decreasing the probability of failure) and delay constraints. To satisfy availability, DSR places two/three replications of each sub-task on two/three different servers. However, DSR considers the availability of each server and utilizes the inclusion-exclusion principle to compute the availability of the application, which has been proved to have exponential computation complexity. A similar problem can be found in [39] and [40]. By exploiting \( \langle e \rangle \)-algebra and DT, one can model application placement in cloud-based systems as a D-SMP and obtain MTTF of application processing, which can be further used as an optimization metric to design a fault-tolerant system.

VI. NUMERICAL EVALUATION

In this section, we carry out an extensive numerical analysis to verify the exactness of our mathematical investigations (Sec. VI-A) and prove the efficiency of \( \text{RP–VC}_n \), as compared to the current art method (Sec. VI-B). The default parameters used for modeling a VC are borrowed from [14] and [17].

A. Exactness of the Mathematical Analysis

We verify the result of Theorem 3 in a scenario where the sojourn times (i.e., \( Z_x \)) of the vehicles are i.i.d exponential random variables with parameter \( \lambda_z \), the value of which is chosen between \( \{1, \frac{1}{2}, \frac{1}{3}\} \) (i.e., the average sojourn time of each vehicle is one, two, or three hours). The duration of the recruitment operations are also chosen to be i.i.d exponential random variables with parameter \( \lambda_u \in \{6, 4, 3\} \) (i.e., the average recruitment times are 10, 15, and 20 minutes). The figures are obtained via Monte-Carlo method, where each result is an average of \( 10^6 \) independent runs.

Fig. 7 illustrates how partitioning a large application into \( n \) smaller dependent groups (ranging from \( n = 1 \) to \( n = 6 \)) can affect \( C\text{-MTTF} \) of \( \text{RP–VC}_n \sim \text{e} \). The figure demonstrates a match between the simulation results, shown by simulated MTTF (SMTTF) and that of the mathematical model obtained via Theorem 3, reflected by predicted MTTF (PMTTF). Also, as can be seen from Fig. 7, for a fixed value of \( \lambda_u \), as the number of groups increases, \( C\text{-MTTF} \) declines because more vehicles are needed to process application \( A \), and thus the chance one of them departing the VC increases.

In real situations, due to computing resource deficiency of the vehicles, a single vehicle may not be able to meet the computing requirements of a CI-App. Consequently, if J2 could...
find two vehicles with enough computing resources to start processing a CI-App, when one of the vehicles allocated to the CI-App departs the VC, the other one may need to wait for a long time until a vehicle with enough computing resources arrives at the VC (demonstrated numerically in the Sec. VI-B). However, RP-VC_{n} breaks down a CI-App into smaller groups, through partitioning. Consequently, more vehicles can satisfy the requirements of the CI-App groups. Besides, the time of transferring images/replicas of the groups among vehicles will be decreased. Accordingly, the value of $\lambda_{n}$ is not fixed in real situations. Fig. 8 indicates that how changing the value of $\lambda_{n}$ can alter the value of $\mathcal{C}$-MTTF. For this experiment, we define a function $\lambda'_{n}(\alpha) = \lambda_{n} \times \alpha$, where $\alpha$ is a decline factor. We then calculate $\mathcal{C}$-MTTF for $\lambda'_{n}(\alpha)$ (i.e., recruitment duration of the vehicles follow an exponential distribution with parameter $\lambda'_{n}(\alpha)$). As can be seen from the figure, as the value of $\alpha$ decreases (i.e., the value of $\lambda'_{n}(\alpha)$ declines) $\mathcal{C}$-MTTF of RP-VC_{n} increases since a smaller $\alpha$ implies a faster vehicle recruitment. Below, we will compare the efficiency of J2 and RP-VC_{n} in processing CI-App.

B. Efficiency of RP-VC_{n}

In the following, we show how partitioning an application into smaller groups can improve the system performance. We first compare the performance of RP-VC_{n} with J2 [14], where both methods consider the execution of applications over two vehicles, while J2 does not consider application partitioning. Afterward, RP-VC_{n} and J2 are compared with their simple versions with no additional replicas referred to by No-R. These versions operate the same as the original strategies with the difference that one vehicle is allocated to each group for RP-VC_{n} and each application for J2.

1) Simulation Setup: To verify the efficiency of RP-VC_{n}, we consider the following simulation environment. Vehicles enter the VC based on a Poisson distribution with parameter $\lambda_{ve}$. Users send request for processing CI-App to the system based on a Poisson distribution with parameter $\lambda_{app}$. For simplicity, we use a number, drawn uniformly from interval $[l_{a}, h_{a}]$, to refer to the resource requirement (i.e., computing, memory, and network resources) of an application. Similarly, the resource capacity (i.e., computing, memory, and network resources) of each vehicle is represented through a number, drawn uniformly from interval $[l_{c}, h_{c}]$. The duration of each application follows exponential distribution with parameter $\lambda_{a}$, and the vehicle’s sojourn time follows exponential distribution with parameter $\lambda_{d}$. Minimum recruitment duration follows exponential distribution with parameter $\lambda_{t}$. Default values of the aforementioned parameters are presented in Table I.

We compare RP-VC_{n} with J2 and their No-R versions in terms of the following quality of service (QoS) metrics:

$$\text{Acceptance Rate (AR)} = \frac{\#\{\text{Accepted Applications}\}}{\#\{\text{Applications}\}}, \quad (52)$$

$$\text{Success Rate (SR)} = \frac{\#\{\text{Successful Applications}\}}{\#\{\text{Applications}\}}, \quad (53)$$

where $\#\{}$ reads “number of”. The acceptance rate (AR) in (52) captures the number of accepted applications divided by the total number of applications. Similarly, (53) refers to the success rate (SR), which is the number of successfully executed applications divided by the total number of applications.

2) Comparison of RP-VC_{n} With J2 for Different Application Sizes: Fig. 9 illustrates the behavior of J2 and RP-VC_{n}, for n \in \{2, 3, 4\}. It can be observed that for light applications, i.e., $l_{a} = 10$ and $l_{a} = 15$, J2 has a reasonable AR (Fig. 9(a)) with slightly lower value than that of RP-VC_{n}, which is 100%. However, there is a dramatic decrease in the AR of J2 starting from $l_{a} = 20$ until $l_{a} = 30$, where the AR reaches almost 0%. RP-VC_{n}, for n \in \{2, 3\}, maintains its high AR for $l_{a} = 10$ to $l_{a} = 40$ with a moderate decline from 100% to (on average) 85%, while RP-VC_{4} has an AR close to 100% in this range. From $l_{a} > 40$, the AR of RP-VC_{2} starts dropping and reaches 0% for $l_{a} > 55$. RP-VC_{3} and RP-VC_{4} have almost equal AR and (on average) they reach the AR of 85% for $l_{a} = 60$.

We next study the SR metric in Fig. 9(b). As can be observed from the figure, from $l_{a} = 10$ to $l_{a} = 15$, J2 has 80% SR, on average. However, SR for J2 drops sharply to reach the value of zero from $l_{a} = 15$ to $l_{a} = 30$ and stays zero for $l_{a} > 30$ since acceptance rate is zero in this range. Likewise, for RP-VC_{2}, SR experiences a moderate decline from the value of 65% at $l_{a} = 10$ to reach around 50% at $l_{a} = 40$ and drops quickly to zero from $l_{a} = 40$ to $l_{a} = 50$ and remains flat onward. Finally, for RP-VC_{3} and RP-VC_{4}, SR gradually declines from 65% at $l_{a} = 10$ to reside around 40% at $l_{a} = 60$.

3) Comparison of RP-VC_{n} With J2 for Different Values of $\lambda_{d}$: In this section, we study AR (Fig. 10(a)) and SR (Fig. 10(b)) for different application’s execution time ranging from two hours to 17 minutes (i.e., $\lambda_{d} \in \{0.5, 1, 1.5, 2, 2.5, 3, 3.5\}$). The comparison is conducted between J2, RP-VC_{2}, RP-VC_{3}, and RP-VC_{4}. Overall, the acceptance rate of all the versions of RP-VC_{n} is considerably

---

**Table I**

| Parameter | Value | Parameter | Value |
|-----------|-------|-----------|-------|
| $\lambda_{ve}$ | 20 | $\lambda_{d}$ | 1 |
| $\lambda_{app}$ | 2 | $\lambda_{a}$ | 1 |
| $[l_{a}, h_{a}]$ | [20, 30] | $\lambda_{t}$ | 1 |
| $[l_{c}, h_{c}]$ | [20, 20] | - | - |
systems and demonstrated that a semi-dynamic VC under RP-VC is an SeS. We then modeled RP-VC via a non-
trivial semi-Markov process (β-SMP) and characterized the dynamics of β-SMP through developing (ε)-algebra. Using (ε)-algebra, we relaxed the innate complexities of β-SMP via proposing decomposition theorem and transforming β-SMP into a decomposed semi-Markov process (i.e., D-SMP). Relying on D-SMP, we then calculated the C-MTTF of our methodology. In addition to multiple future work directions discussed in the paper, we intend to extend the analysis of RP-VC to scenarios with more replications of groups dispersed across vehicles. Also, efficient recruitment of new vehicles (e.g., considering outage probability of links) is an interesting future work.

REFERENCES

[1] Z. Cheng, M. Min, M. Liwang, L. Huang, and Z. Gao, “Multiagent
DDPG-based joint task partitioning and power control in fog comput-
ing networks,” IEEE Internet Things J., vol. 9, no. 1, pp. 104–116, Jan. 2022.
[2] M. LiWang, S. Dai, Z. Gao, X. Du, M. Guizani, and H. Dai, “A comput-
ingulation offloading incentive mechanism with delay and cost constraints
under 5G satellite-ground IoV architecture,” IEEE Wireless Commun.,
vol. 26, no. 4, pp. 124–132, Aug. 2019.
[3] X. Chen and G. Liu, “Energy-efficient task offloading and resource
allocation via deep reinforcement learning for augmented reality in mobile edge networks,” IEEE Internet Things J., vol. 8, no. 13,
pp. 10843–10856, Jul. 2021.
[4] P. Mach and Z. Beccar, “Mobile edge computing: A survey on architec-
ture and computation offloading,” IEEE Commun. Surveys Tuts., vol. 19,
no. 3, pp. 1628–1656, 3rd Quart., 2017.
[5] C. Jiang, X. Cheng, H. Gao, X. Zhou, and J. Wan, “Toward comput-
ingulation offloading in edge computing: A survey,” IEEE Access, vol. 7,
pp. 131543–131558, 2019.
[6] M. Chiang and T. Zhang, “Fog and IoT: An overview of research
opportunities,” IEEE Internet Things J., vol. 3, no. 6, pp. 854–864, Dec. 2016.
[7] E. Lee, E. Lee, M. Gerla, and S. Y. Oh, “Vehicular cloud networking:
Architecture and design principles,” IEEE Commun. Mag., vol. 52, no. 2,
pp. 148–155, Feb. 2014.
[8] M. Eloweissy, S. Olariu, and M. Younis, “Towards autonomous vehicu-
lar clouds,” in Proc. Int. Conf. Ad Hoc Netw., 2010, pp. 1–16.
[9] S. Bitam, A. Mellouk, and S. Zeada, “VANET-cloud: A generic cloud
computing model for vehicular ad hoc networks,” IEEE Wireless Commun.,
vol. 22, no. 1, pp. 96–102, Feb. 2015.
[10] A. Boukerche and R. E. D. Grande, “Vehicular cloud computing:
Architectures, applications, and mobility,” Compot. Netw., vol. 135,
pp. 171–189, Apr. 2018.
[11] H. Liao, X. Li, D. Guo, W. Kang, and J. Li, “Dependency-aware
application assigning and scheduling in edge computing,” IEEE Internet
Things J., vol. 9, no. 6, pp. 4451–4463, Mar. 2022.
