From the Real Vehicle to the Virtual Vehicle

Jiangchuan Huang
Systems Engineering Group
Univ. of California, Berkeley
Email: jiangchuan@berkeley.edu

Christoph M. Kirsch
Dept. of Computer Sciences
Univ. of Salzburg, Austria
Email: ck@cs.uni-salzburg.at

Raja Sengupta
Systems Engineering Group
Univ. of California, Berkeley
Email: sengupta@ce.berkeley.edu

***Abstract*—We apply virtual machine abstractions to networked autonomous vehicles enabling what we call cloud computing in space. In analogy to traditional system virtualization and cloud computing, there are (customer-operated) virtual vehicles that essentially perform like real vehicles although they are hosted by possibly fewer, shared (provider-operated) real vehicles. Here the focus is, however, on motion rather than computation. In the service-level agreement, a virtual vehicle is a virtual machine plus a virtual speed. We define virtual deadline for each task based on virtual speed, and make the spatial cloud a soft real-time system [7], [1]. The performance isolation is measured by the average of tardiness and delivery probability. We use Voronoi tessellation to allocate the tasks to the real vehicles, and design scheduling policies such as Earliest Virtual Deadline First (EVDF), Earliest Dynamic Virtual Deadline First (EDVDF) and credit scheduling policy for the real vehicles. EVDF is shown to minimize the tardiness. Under EVDF, we identify a worst case arrival process maximizing tardiness. We show in simulation that abstracting real vehicles such as cars or planes to virtual vehicles enables virtual vehicles to move in space like real vehicles with guaranteed tardiness (e.g. $\leq 1\%$) while being hosted by significantly fewer, e.g. 1-for-7.5, shared real vehicles.**

I. Introduction

Consider the following systems with tasks arriving in time and space: (i) Google Street View, Google is running numerous data-collection vehicles with mounted cameras, lasers, a GPS and several computers to collect street views while minimizing the distance travelled [3]. (ii) Real-time traffic reporting, a radio station uses a helicopter to overfly accident scenes and other areas of high traffic volume for real-time traffic information. (iii) Unmanned aerial vehicle (UAV)-based sensing, a fleet of UAVs equipped with greenhouse gas sensors collect airborne measurements of greenhouse gases at several sites in California and Nevada, executing every task at its location before its deadline [19]. (iv) Enabling mobility in wireless sensor networks (WSN), mobile elements (vehicles) capable of short-range communications collect data from nearby sensor nodes as they approach on a schedule [36].

In these applications, a task is a triple (T^a, X, T^S) where T^a is the arrival time of the task, X its location in space, and T^S its execution time (task size). Servers, henceforth referred to as real vehicles, are networked vehicles each having all or some of the sensing, computation, communication, and locomotion capabilities. A real vehicle is considered to serve a task by visiting the location X and staying there for execution time T^S after the task has arrived, i.e., after time T^a. The performance metrics include system time (latency), throughput, delivery probability, and total distance traveled by the vehicles.

Traditionally, these applications were modeled by (i) the vehicle routing problem (VRP) [13], its variations such as (ii) the Dynamic Travelling Repairman Problem (DTRP) [5], (iii) the stochastic and dynamic vehicle routing problem with time windows (SDVRPTW) [29], and (iv) the mobile element scheduling (MES) problem [36]. Tasks are allocated and sequenced based on their arrival times T^a, e.g., first come first served (FCFS) [5], locations X, e.g., nearest neighbor (NN) [5], and sizes T^S, e.g., shortest job first [38], to minimize the distance traveled [13], the average time each task spent in the system [5], or the deadline violation ratio [29]. A richer set of scheduling policies include direct tree search, dynamic programming, and integer linear programming for VRP [24]; FCFS, stochastic queue median (SQM), travelling salesman policy (TSP), NN [5], divide and conquer [30] for DTRP and SDVRPTW [29]; earliest deadline first, earliest deadline first with k-lookahead, the minimum weight sum first [36] and partition based scheduling [14] for MES. While the current scheduling policies work for the single customer systems, they do not create performance isolation in multi-customer systems. In the VRP tasks from different customers are indistinguishable to the scheduling policies, meaning a customer submitting more tasks would use more resources, possibly to the detriment of other customers.

To resolve this problem, we formulate the multi-customer VRP by borrowing the concept of a virtual machine from cloud computing. In cloud computing, the virtual machine abstraction creates performance isolation so that resources consumed by one virtual machine do not necessarily harm the performance of other virtual machines [34]. To extend the idea of the virtual machine [34] used in cloud computing, we define an idea called the virtual vehicle. The virtual vehicle is an analog of the virtual machine for location specific computing tasks. Just as the cloud computing customer has a service-level agreement (SLA) for a virtual machine, our customer would have an SLA for a virtual vehicle. To a customer, a virtual vehicle is exactly like a real vehicle that travels at the virtual speed specified in the SLA. For example, suppose a radio station (the customer) uses a helicopter to overfly accident scenes (the task) at 100 mile per hour (mph) for real-time traffic reporting. Such a helicopter would arrive at an accident scene 10 miles away in 6 minutes. We now virtualize this helicopter. Instead of buying or renting, and operating a real helicopter, the radio station would buy a virtual helicopter that travels at 100 mph using an SLA. Our spatial cloud would then take responsibility for flying some real helicopter to the accident site in 6 minutes or some near approximation to this

Research supported in part by the National Science Foundation (CNS1136141) and by the National Research Network RiSE on Rigorous Systems Engineering (Austrian Science Fund S11404-N23).

time. The radio station can estimate the completion time of each task by dividing the consecutive distances between tasks by the virtual speed just as they did for their real helicopter before. The task completion time expected by the radio station (customer) is given by (4) in Section II. The radio station would enjoy the advantage of reserving the virtual helicopter only when the traffic reporting program is on, and not paying for other times. If the real helicopters executing the virtual helicopters could serve other customers at these other times, the enhanced vehicle utilization would reduce costs for all customers much like cloud computing. The analysis in this paper quantifies these gains.

Our scheduling algorithms are designed to have the real vehicles travel to and execute each task such that the real completion time is no later than the expected completion time. The system achieves high performance isolation if a statistically dominant subset [32], e.g., 98%, of the virtual vehicle's tasks are completed no later than their expected completion times. The fraction must be small enough for the customer to consider herself adequately compensated for this loss by the reduced costs delivered by the spatial cloud. We design the spatial cloud to treat the expected completion time as a "deadline" and call it the virtual deadline. The virtual deadlines make the spatial cloud a soft real-time system [7], [1], meaning performance metrics such as tardiness and delivery probability defined in (5) and (6) can be used to measure performance isolation. These performance metrics are the performance isolation measures [39] and Jain's fairness indices [18] defined separately in (12) and (13), and in (14) and (15) in Section II. The slack defined in (7) measures the earliness of a task completion. How the provider realizes the virtual deadlines of the tasks is of no concern to the customer. For example, the provider can use a real vehicle to travel to and execute a task as the virtual vehicle does, or migrate the information of the virtual vehicle through a network to another real vehicle closer to the task. In the latter case, the real distance traveled is smaller than the virtual distance resulting in the phenomenon we have called migration gain. The idea works when the communication costs of migrating a virtual vehicle over the network are small. The migration cost of a virtual vehicle is defined in (11).

We proposed cloud computing in space as a concept that makes sensing a service in [11] and described our work on the software engineering required to build a virtual vehicle. The protocols required for migration are described in [23]. The results of this paper show that the provider can support a given number of virtual vehicles with significantly fewer real vehicles that travel at the virtual speed while guaranteeing high performance isolation. We quantify the gain by the ratio of the number of virtual vehicles over the number of real vehicles. The gain arises from two phenomena. (i) A customer may not fully utilize her virtual vehicle, enabling the spatial cloud to multiplex several virtual vehicles onto one real vehicle. This type of gain is called multiplexing gain. It is known in communication networks [16] and cloud computing [33]. (ii) The real vehicles save travel distance by migrating the virtual vehicle hosting a task to another real vehicle closer to the task, creating migration gain. This paper focuses on migration gain since it is unique to this spatial cloud. We use a very simple model of the task size T^S itself since we focus on the migration gain arising from the spatial distribution of the tasks. Migration gain is the reason spatial cloud computing outperforms conventional cloud computing.

We analyze the system in Section III using results in queueing theory [10], [2], stochastic and dynamic vehicle routing [5], [30], [17], and soft real-time systems [7], [1]. Figure 1 depicts the $GI/GI/1$ queue [10] of each virtual vehicle (Theorem 1), and the $\Sigma GI/GI/1$ queue [2] of each real vehicle (Theorem 2) under the Voronoi tessellation [28] in Definition 1. We propose scheduling policies adapted from conventional cloud computing [8], named Earliest Virtual Deadline First (EVDF) when task size is known a priori (Definition 5), its variation Earliest Dynamic Virtual Deadline First (EDVDF) when task size is not known a priori (Definition 6), and the credit scheduling policy (Definition 7) in Section IV. Theorem 3 asserts that EVDF minimizes tardiness. This is based on the optimality of earliest deadline first scheduling for real-time queues [27]. Theorem 4 identifies a worst-case arrival process maximizing tardiness called the $\eta = 1$ arrival process in Definition 8. Theorem 5 asserts task size dependent economy of scale [26] in the sense that the largest achievable gain without compromising performance isolation increases as the square root of the number of real vehicles when the task size is zero, and increases but is upper bounded by a constant when the task size is greater than zero. Theorem 6 bounds the migration cost. Theorem 7 bounds the slack under the $\eta = 1$ arrival process.

We simulate the system with homogeneous real vehicles (Definition 3) and homogeneous virtual vehicles (Definition 4) under the EVDF, EDVDF and credit scheduling policies in Section V. Each virtual vehicle generates an $\eta = 1$ arrival process that maximizes tardiness, and excludes multiplexing gain by having each virtual vehicle fully utilized. Thus, this is a worst-case simulation and the gain is migration gain only. Figure 4 shows the performance isolation and fairness index based on tardiness and delivery probability, together with slack and migration costs under the EVDF for different numbers of real vehicles and gains and different number of virtual vehicles. We conclude that (i) the provider can support a given number of virtual vehicles with significantly fewer real vehicles that travel at the virtual speed while guaranteeing high performance isolation, e.g., the gain equals 7.5 while guaranteeing tardiness $\leq 1\%$ when the number of real vehicles is 100 as shown in Figure 5. (ii) The migration gain increases with the number of real vehicles while guaranteeing the same performance isolation as shown in Figure 5, and asserted in Theorem 5. (iii) The migration cost is bounded (Figure 4). (iv) The virtual vehicle concept works best when the task sizes are small and the vehicle spends more time traveling to tasks than it does loitering or standing still. As the virtual vehicles spend more time standing still, cloud computing in space converges to conventional cloud computing as shown in Figure 6. (v) The provider can easily determine the appropriate number of real vehicles for a given number of virtual vehicles and a guaranteed performance isolation because the transition between low and high performance isolation is sharp as shown in Figure 7. (vi) EVDF has better performance than EDVDF, and EDVDF has better performance than the credit scheduling policy as shown in Figure 8.

II. MODEL

The spatial cloud is defined as follows: A service provider controls $M \in \mathbb{N}$ real vehicles (RVs), $\{RV_m\}_{m=1}^{M}$, in a convex region $\mathbf{A}$ of area A to host $K \in \mathbb{N}$ virtual vehicles (VVs), $\{VV_k\}_{k=1}^{K}$. Each RV travels at constant speed v^R. Each VV has virtual speed v^V in the service-level agreement.

To a customer, a virtual vehicle with speed v^V is a replica of a real vehicle with speed v^V. It can be reserved to host a sequence of tasks $\langle Task_{ki}\rangle_{i=1}^{\infty}$, where k denotes the k-th VV and i the i-th task hosted by it. Each $Task_{ki}$ has arrival times T_{ki}^a, location X_{ki}, and size T_{ki}^S. $\langle Task_{ki}\rangle$ is ordered by the arrival time T_{ki}^a. The sequences of tasks hosted by different VVs are independent of each other, i.e., $Task_{ki}$ are independent in k.

Each task arrival process $\{T_{ki}^a\}_{i=1}^{\infty}$ is assumed to be a renewal process. Thus the interarrival time, $I_{ki}^a \equiv T_{ki}^a - T_{k(i-1)}^a$ is independent and identically distributed (i.i.d.) in i, where $T_{k0}^a \equiv 0$. The generic interarrival time I_k^a is assumed to be integrable. Thus the task arrival rate of each VV is

$$\lambda_k^V = \frac{1}{E\left[I_k^a\right]} \tag{1}$$

X_{ki} is i.i.d. in k and i, and uniformly distributed in $\mathbf{A}$ with probability density function (pdf) $f_X(x) = \frac{1}{A}$, $x \in \mathbf{A}$. The task size T_{ki}^S is i.i.d. in k and i with pdf $f_{T^S}(t)$, $t \in [0, \infty)$. We denote by T^S the generic term of T_{ki}^S. Then $E\left[T_{ki}^S\right] = E\left[T^S\right]$, which is assumed to be finite.

Let L_{ki} denote the distance between $Task_{k(i-1)}$ and $Task_{ki}$ hosted by VV_k. Then

$$L_{ki} = \parallel X_{ki} - X_{k(i-1)} \parallel \tag{2}$$

where $\parallel . \parallel$ is the Euclidean norm defined on region $\mathbf{A}$, $i = 1, 2, \ldots$. X_{k0} is the initial position of the first RV hosting VV_k given by the provider, which is assumed to be uniformly distributed in $\mathbf{A}$, and independent of X_{ki}. From (2) we know L_{ki} is i.i.d. in k and i. We thus denote by L the generic term of L_{ki}.

VV_k is assumed to serve the sequence of tasks $\langle Task_{ki}\rangle_{i=1}^{\infty}$ under the first come first served (FCFS) policy. It travels to the location X_{ki} of each task and executes it taking time T_{ki}^S. Then the virtual service time of $Task_{ki}$, denoted by T_{ki}^{Vserv}, includes the flying time and execution time as follows

$$T_{ki}^{Vserv} = \frac{L_{ki}}{v^V} + T_{ki}^S \tag{3}$$

Thus T_{ki}^{Vserv} is i.i.d. in k and i. We denote by T^{Vserv} the generic term of T_{ki}^{Vserv}.

Each VV_k is a queue. The virtual departure time of $Task_{ki}$ from this queue is

$$T_{ki}^{dead} = \max\left\{T_{ki}^a, T_{k(i-1)}^{dead}\right\} + T_{ki}^{Vserv} \tag{4}$$

where $T_{k0}^{dead} \equiv 0$. We use the superscript dead for deadline because this virtual departure time is used by our scheduling policies as a task deadline.

Each task, upon arrival, is passed to one of the M real vehicles. Thus we have M real vehicle queues. Each $Task_{ki}$ is served by some real vehicle RV_m as per an *allocation policy* given in Definition 1 in Section III-B. The order of service is determined by a scheduling policy. The scheduling policies, discussed in Section IV, are the main subject of this chapter.

Each $Task_{ki}$ will be completed by an RV at some time T_{ki}^{comp}. We assume customer k will be satisfied if $T_{ki}^{comp} \leq T_{ki}^{dead}$. Then the aim of the provider is to achieve $T_{ki}^{comp} \leq T_{ki}^{dead}$ for as many tasks as possible. Thus T_{ki}^{dead} is like a "deadline" for $Task_{ki}$. We call the T_{ki}^{dead} the *virtual deadline* for $Task_{ki}$. This makes the spatial cloud a soft real-time system [7], [1].

We define the relative expected *tardiness* of VV_k as

$$TD_k = \frac{1}{E\left[T^{Vserv}\right]} \lim_{n\to\infty} \frac{\sum_{i=1}^{n} \max\left\{T_{ki}^{comp} - T_{ki}^{dead}, 0\right\}}{n} \tag{5}$$

The *delivery probability* of VV_k is

$$DP_k = \lim_{n\to\infty} \frac{\sum_{i=1}^{n} \mathbb{1}\left\{T_{ki}^{comp} \leq T_{ki}^{dead}\right\}}{n} \tag{6}$$

where $\mathbb{1}\{\Omega\}$ is the indicator function defined as $\mathbb{1}\{\Omega\} = 1$ if Ω is true, and $= 0$ otherwise.

The relative expected *slack* of VV_k is

$$SL_k = \frac{1}{E\left[T^{Vserv}\right]} \lim_{n\to\infty} \frac{\sum_{i=1}^{n} \max\left\{T_{ki}^{dead} - T_{ki}^{comp}, 0\right\}}{n} \tag{7}$$

We define the virtual system time of $Task_{ki}$ as $T_{ki}^{Vsys} = T_{ki}^{dead} - T_{ki}^a$, and the real system time of $Task_{ki}$ as $T_{ki}^{Rsys} = T_{ki}^{comp} - T_{ki}^a$. Thus $T_{ki}^{comp} - T_{ki}^{dead} = T_{ki}^{Rsys} - T_{ki}^{Vsys}$, and $T_{ki}^{comp} \leq T_{ki}^{dead} \Leftrightarrow T_{ki}^{Rsys} \leq T_{ki}^{Vsys}$. When the queue at VV_k is stable, $T_{ki}^{Vsys} \to T_k^{Vsys}$ in distribution. When the M real vehicle queues are stable, $T_{ki}^{Vsys} \to T_k^{Vsys}$ in distribution.

Thus (5) is equivalent to

$$TD_k = \frac{E\left[\max\left\{T_k^{Rsys} - T_k^{Vsys}, 0\right\}\right]}{E\left[T^{Vserv}\right]} \tag{8}$$

(6) is equivalent to

$$DP_k = P\left(T_k^{Rsys} \leq T_k^{Vsys}\right) \tag{9}$$

(7) is equivalent to

$$SL_k = \frac{E\left[\max\left\{T_k^{Vsys} - T_k^{Rsys}, 0\right\}\right]}{E\left[T^{Vserv}\right]} \tag{10}$$

These three measures are determined by the virtual vehicle queues and the real vehicle queues as shown in Section III.

Two consecutive tasks of an VV might be executed by two different RVs, which involves migrating the VV from one virtual vehicle to another. Let Z_{ki} be an indicator set to 1 if there is a migration between $Task_{k(i-1)}$ and $Task_{ki}$, and 0 otherwise. When the M real vehicle queues are stable, $Z_{ki} \to Z_k$ in distribution. B_{Vk} is the number of bits to migrate VV_k

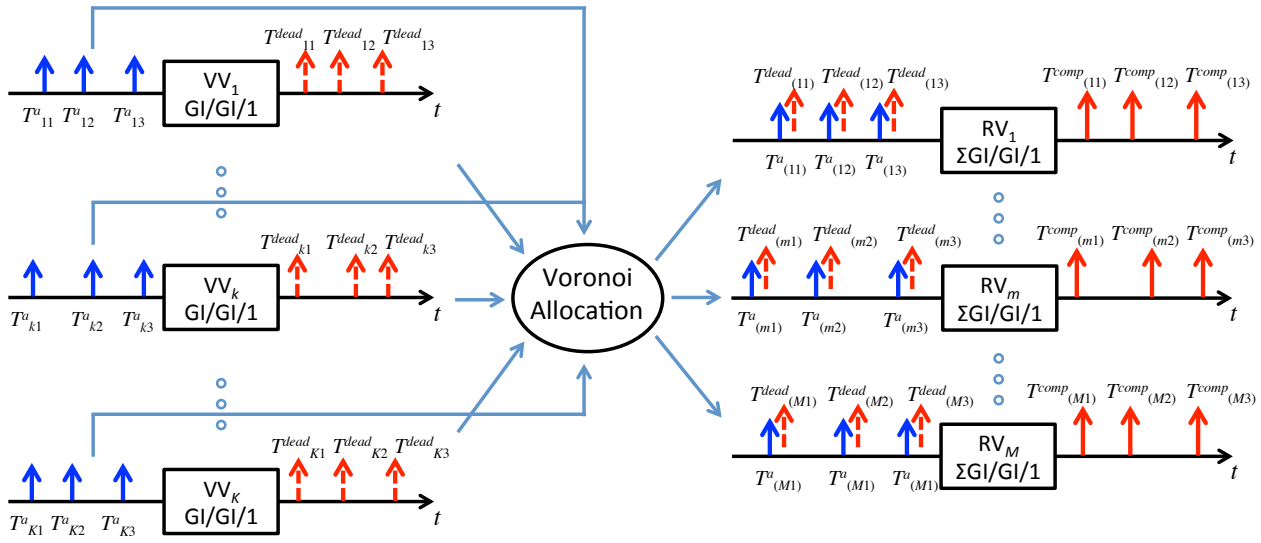


Fig. 1. The spatial cloud with virtual vehicle queues and real vehicle queues.

at the migration time. We assume $B_{Vk} = 1$ or equivalently a constant. L is the generic distance between two consecutive tasks. We define the inter-virtual deadline time as $I_{ki}^{dead} = T_{ki}^{dead} - T_{k(i-1)}^{dead}$. Since T_{ki}^{Vserv} is i.i.d. in k and i, then $I_{ki}^{dead} \to I_k^{dead}$ in distribution. The migration cost of VV_k is

$$MC_k = B_{Vk} \frac{E[Z_k L]}{E\left[I_k^{dead}\right]} \tag{11}$$

The migration cost has the same unit (bit-meters/second) as in [15].

A. Performance Isolation

We measure *performance isolation* by the average of the tardiness and delivery probability.

$$TD = \frac{1}{K} \sum_{k=1}^{K} TD_k \tag{12}$$

$TD = 0$ implies $TD_k = 0$ for all virtual vehicle k, meaning the system achieves perfect performance isolation. Conversely $TD \to \infty$ means the relative expected tardiness of some VVs is unbounded, the system has very poor performance isolation.

$$DP = \frac{1}{K} \sum_{k=1}^{K} DP_k \tag{13}$$

$DP = 1$ implies $DP_k = 1$ for all virtual vehicle k, meaning the system achieves perfect performance isolation. Conversely $DP = 0$ means the delivery probability of each virtual vehicle is zero, meaning the system has no performance isolation.

Fairness is the equality of work divided among different concurrent environments [6]. We use Jain's fairness index [18] to quantify the fairness between virtual vehicles. The fairness index based on tardiness TD_k is

$$FI(TD_k) = \frac{\left(\sum_{k=1}^{K} e^{-TD_k}\right)^2}{K \sum_{k=1}^{K} \left(e^{-TD_k}\right)^2} \tag{14}$$

where we use e^{-TD_k} to map TD_k from $[0, \infty)$ to $(0, 1]$, the lower TD_k, the higher e^{-TD_k}, indicating higher performance.

The fairness index based on delivery probability DP_k is

$$FI(DP_k) = \frac{\left(\sum_{k=1}^{K} DP_k\right)^2}{K \sum_{k=1}^{K} DP_k^2} \tag{15}$$

Jain's fairness index is the ratio of the square of the first moment over the second moment of the set of performance metrics of all the VVs. If $TD_k = TD_l > 0$ (resp. $DP_k = DP_l > 0$), $\forall k, l$, then $FI(TD_k) = 1$ (resp. $FI(DP_k) = 1$), indicating completely fair. If $TD_k > 0$ and $TD_l = 0$ (resp. $DP_k > 0$ and $DP_l = 0$), $\forall l \neq k$, then $FI(TD_k) = \frac{1}{K}$ (resp. $FI(DP_k) = \frac{1}{K}$), indicating completely unfair. Thus $FI(TD_k)$ (resp. $FI(DP_k)$) ranges between $\frac{1}{K}$ and 1. The greater the fairness index, the more fair the system. Jain's fairness index has been used to evaluate virtualization systems [40], [20], [4]. Other performance metrics in this literature include throughput, latency and response time.

B. Gain

When a provider supports K virtual vehicles with M real vehicles we define the gain κ to be

$$\kappa = \frac{K}{M} \tag{16}$$

The provider gains if $\kappa > 1$. There are two ways a provider can gain:

- *Multiplexing gain*: a customer may not utilize her virtual vehicle fully, enabling the provider to multiplex several virtual vehicles onto one real vehicle.
- *Migration gain*: the provider gains by migrating the VV hosting the task to another RV closer to the task location.

The multiplexing gain is observed in communication networks [16] and cloud computing [33]. Migration gain is unique to

cloud computing in space. When every virtual vehicles are fully utilized, there is no multiplexing gain, but there is still migration gain.

III. Systems

The system has queues at the virtual vehicles and queues at the real vehicles as depicted in Figure 1.

A. Virtual Vehicle Queues

The task arrival rate for VV_k is $\lambda_k^V = \frac{1}{E\left[I_k^a\right]}$ by (1), where I_k^a is the generic interarrival time of VV_k. The service time T_{ki}^{Vserv} is i.i.d. in k and i with generic term T_k^{Vserv}. We define the generic virtual vehicle service rate as

$$\mu^V = \frac{1}{E\left[T^{Vserv}\right]} = \frac{1}{\frac{E[L]}{v^V} + E\left[T^S\right]} \tag{17}$$

as usual, with T_{ki}^{Vserv} defined in (3).

The random process at the output of VV_k is the virtual deadline process $\left\{T_{ki}^{dead}\right\}_{i=1}^{\infty}$. The virtual deadline rate is defined as

$$\lambda_k^{Vdead} = \lim_{i\to\infty} \frac{1}{E\left[T_{ki}^{dead} - T_{k(i-1)}^{dead}\right]} \tag{18}$$

Since VV_k is a work-conserving server, it is busy if it has a queue and idle if not. Thus VV_k repeats cycles of busy and idle periods. We define Θ_{kl}^V as the l-th busy period and I_{kl}^V as the l-th idle period. We define virtual vehicle utilization as

$$u_k^V = \lim_{l\to\infty} \frac{E\left[\Theta_{kl}^V\right]}{E\left[\Theta_{kl}^V\right] + E\left[I_{kl}^V\right]} \tag{19}$$

A single server queuing system is $GI/GI/1$ if the interarrival times at the input and the service times are positive i.i.d. random variables, separately [10].

The tasks created by a customer are passed to the cloud at the rate chosen by the customer. The following theorem asserts that when the arrival rate is less than the service rate, the virtual deadline rate is equal to the arrival rate. However, if the customer exceeds the service rate determined by the contracted virtual speed, the virtual deadline rate is equal to the service rate, i.e., the contract throttles the customer's virtual deadline rate. A higher virtual deadline rate requires more tasks to be completed in a unit time. A customer cannot require more than her share of resources by simply generating tasks faster and faster because the virtual deadline rate is throttled by the VV service rate.

Theorem 1: Each virtual vehicle VV_k is a $GI/GI/1$ queue. Moreover,
If $\lambda_k^V < \mu^V$, then $u_k^V < 1$ and $\lambda_k^{Vdead} = \lambda_k^V$.
If $\lambda_k^V \geq \mu^V$, then $u_k^V = 1$ and $\lambda_k^{Vdead} = \mu^V$.

Proof: By assumption the task arrival process of VV_k is renewal. Thus the interarrival times are positive and i.i.d.. Also the service times T_{ki}^{Vserv} are positive and i.i.d.. Thus each virtual vehicle VV_k is a $GI/GI/1$ queue.

(i) When $\lambda_k^V < \mu^V$, the $GI/GI/1$ queue is stable by Theorem 1.1 in [10, p. 168]. The number of tasks waiting in the queue is finite almost surely, the mean interdeparture time $E\left[T_{ki}^{dead} - T_{k(i-1)}^{dead}\right]$ of the VV is $\frac{1}{\lambda_k^V}$ because no tasks are lost and no extra task is created. Thus $\lambda_k^{Vdead} = \lambda_k^V$. Stability also ensures that $\lim_{l\to\infty} E\left[\Theta_{kl}^V\right] = E\left[\Theta_k^V\right]$ and $\lim_{l\to\infty} E\left[I_{kl}^V\right] = E\left[I_k^V\right]$, where Θ_k^V and I_k^V are the generic busy period and idle period. Thus $u_k^V = \frac{E[\Theta_k^V]}{E[\Theta_k^V]+E[I_k^V]}$. According to [37, p. 21], $u_k^V = \frac{\lambda_k^V}{\mu^V}$ since no tasks are lost or created in the system. Thus $u_k^V = \frac{\lambda_k^V}{\mu^V} < 1$.

(ii) When $\lambda_k^V > \mu^V$, the $GI/GI/1$ queue is unstable by Theorem 1.1 in [10, p. 168], the number of tasks waiting in the queue goes to infinity as time goes to infinity. The busy period tends to infinity and the idle period tends to 0, Thus $u_k^V = \lim_{l\to\infty} \frac{E\left[\Theta_{kl}^V\right]}{E\left[\Theta_{kl}^V\right]+E\left[I_{kl}^V\right]} = 1$, and the interdeparture time equals the VV service time. Then $\lambda_k^{Vdead} = \mu^V$.

(iii) When $\lambda_k^V = \mu^V$, the number of tasks waiting in the queue can either be finite, or goes to infinity as time goes to infinity depending on the arrival process $\{T_{ki}^a\}$. So this case either goes to case (i) or (ii). In either case, we have $u_k^V = 1$ and $\lambda_k^{Vdead} = \mu^V$. ∎

When $\lambda_k^V > \mu^V$, the $GI/GI/1$ queue at VV_k is unstable, thus the virtual system time $T_{ki}^{Vsys} = T_{ki}^{dead} - T_{ki}^a \to \infty$ almost surely [10, p. 168]. Since the customer regards a VV a replica of an RV, we assume the customer will never run the VV under the unstable condition. Thus we assume $\lambda_k^V \leq \mu^V$ from now on. Also, when $\lambda_k^V = \mu^V$, we assume the customer only provide task arrival process that results in finite virtual system time, i.e., $T_{ki}^{Vsys} \to T_k^{Vsys}$ in distribution.

B. Real Vehicle Queues

Each task, upon arrival, is passed to one of the M real vehicles. Inside each RV subregion, the RV runs a scheduling policy to decide which task of which VV to execute when the RV becomes available. The scheduling policies are discussed in Section IV. We allocate the tasks hosted by each VV as follows.

Definition 1: The VV allocation has the following steps:

(i) Divide the region $\mathbf{A}$ into M subregions by computing an M-median of $\mathbf{A}$ that induces a Voronoi tessellation that is equitable with respect to $f_X(x)$ following [28]. An M-partition $\{\mathbf{A}_m\}_{m=1}^M$ is equitable with respect to $f_X(x)$ if $\int_{\mathbf{A}_m} f_X(x)\,dx = \frac{1}{M}$ for all $m \in \{1,\ldots,M\}$.

(ii) The real vehicles assign themselves to the subregions in a one-to-one manner.

(iii) Each RV serves the tasks that fall within its own subregion according to some scheduling policy. The VV hosting the task is migrated to the RV prior to task execution if the previous task was served by another RV. This migration incurs the cost quantified in (11).

We sequence the tasks contributed by all the virtual vehicles to subregion $\mathbf{A}_m$ by their arrival times. The sequence is denoted $\langle Task_{(mj)}\rangle_{j=1}^{\infty}$. Thus $Task_{(mj)}$ is the j-th task arrived at real vehicle m. Note each $Task_{(mj)}$ corresponds to some $Task_{ki}$ hosted by a virtual vehicle k at time T_{ki}^a.

$Task_{(mj)}$ has arrival time $T^a_{(mj)} = T^a_{ki}$, location $X_{(mj)} = X_{ki}$ and size $T^S_{(mj)} = T^S_{ki}$. Note $T^a_{(m(j-1))} \leq T^a_{(mj)}$ and $X_{(mj)} \in \mathbf{A}_m$ by construction. We write $Task_{(mj)}$ if the task is labeled according to the RV, and write $Task_{ki}$ if the task is labeled according to the VV. For the different labels of the same task, $Task_{ki}$ and $Task_{(mj)}$, since $T^a_{k(i-1)} \leq T^a_{ki}$ and $T^a_{(m(j-1))} \leq T^a_{(mj)}$, then

$$i \to \infty \Leftrightarrow T^a_{ki} \to \infty \Leftrightarrow T^a_{(mj)} \to \infty \Leftrightarrow j \to \infty \tag{20}$$

Since the task locations X_{ki} are i.i.d. in k and i, and uniformly distributed in region $\mathbf{A}$, then $X_{(mj)}$ are i.i.d. in j and uniformly distributed in each subregion $\mathbf{A}_m$. We denote by $D_{(m)}$ the distances between two random task locations in subregion $\mathbf{A}_m$. Thus,

$$D_{(m)} = \parallel X_{(mj)} - X_{(ml)} \parallel \tag{21}$$

where $X_{(mj)}$ and $X_{(ml)}$ are two random task locations in subregion $\mathbf{A}_m$.

Let $D_{(mj)}$ denote the distance between $Task_{(mj)}$ and the task executed before it under a scheduling policy ϕ in subregion $\mathbf{A}_m$. In general, $D_{(mj)}$ is policy dependent. We define a class of policies that produce i.i.d. $D_{(mj)}$ as follows.

Definition 2: A scheduling policy ϕ in a real vehicle subregion $\mathbf{A}_m$ is called *non-location based* if the distance between two consecutively executed tasks is i.i.d..

The common policies in queueing theory such as FCFS, last come first served (LCFS), random order of service (ROS), and shortest job first (SJF) [38] are non-location based policies in the sense of Definition 2. The scheduling policies we propose in Section IV such as Earliest Virtual Deadline First (EVDF), Earliest Dynamic Virtual Deadline First (EDVDF) and credit scheduling policy are shown to satisfy Definition 2 in Theorem 3.

The scheduling policies that utilize the location of the tasks such as nearest neighbor (NN) and traveling salesman policy (TSP) on a given set of tasks are not non-location based because the distances between two consecutively executed tasks are not independent.

Definition 3: The set of M real vehicle subregions $\{\mathbf{A}_m\}_{m=1}^M$ are said to be *homogeneous* if they all have the same scheduling policy, and $D_{(m)}$ is i.i.d. for all $m = 1, \ldots, M$.

In this section and next, we only consider the homogeneous RV subregions under non-location based scheduling policies. We denote by D the generic term of $D_{(m)}$. Then $D_{(mj)}$ is i.i.d. in m and j, and has the same distribution as D.

The service time of $Task_{(mj)}$ by RV_m, denoted by $T^{Rserv}_{(mj)}$, is

$$T^{Rserv}_{(mj)} = \frac{D_{(mj)}}{v^R} + T^S_{(mj)} \tag{22}$$

Since $D_{(mj)}$ and $T^S_{(mj)}$ are i.i.d. in m and j, separately, then $T^{Rserv}_{(mj)}$ is i.i.d. in m and j. We denote by T^{Rserv} the generic term of $T^{Rserv}_{(mj)}$, and define the generic real vehicle service rate μ^R as

$$\mu^R = \frac{1}{E\left[T^{Rserv}\right]} = \frac{1}{\frac{E[D]}{v^R} + E\left[T^S\right]} \tag{23}$$

We define

$$\kappa^c = \frac{\mu^R}{\mu^V} = \frac{\frac{E[L]}{v^V} + E\left[T^S\right]}{\frac{E[D]}{v^R} + E\left[T^S\right]} \tag{24}$$

Before analyzing the queues at the real vehicles, we list some of the basic results of the thinning and superposition of stochastic processes for convenience below.

(i) Example 4.3(a) in [12, pp. 75-76], thinning of renewal processes: Given a renewal process $\{S_n\}$ with rate λ, let each point S_n for $n = 1, 2, \ldots$ be omitted from the sequence with probability $1 - p$ and retained with probability p for some constant p in $0 < p < 1$, each such point S_n being treated independently. The sequence of retained points, denoted by $\{S^p_n\}$, is called the thinned process with retaining probability p. Then $\{S^p_n\}$ is also renewal with rate $p\lambda$.

(ii) From [21, Section 14], we know the following proposition for the superposition of independent, stationary processes. Let $N_k(t)$ be a stationary process with rate λ_k, then the superposition of K such independent processes $N(t) = \sum_{k=1}^K N_k(t)$ is also stationary with rate $\lambda = \sum_{k=1}^K \lambda_k$.

(iii) Two stationary stochastic processes are said to be probabilistic replicas of each other if their generic interarrival times are identically distributed [37, p. 21].

The following theorem asserts the queueing systems at each real vehicle is $\Sigma GI/GI/1$ [2] under non-location based policies. This means the arrival process at each real vehicle is the superposition of independent renewal processes and the service time process has positive i.i.d. interarrival times. It also establishes the critical role of κ^c in stability of these queues under the assumption that every customer keeps their VV stable, i.e., $\lambda^V_k \leq \mu^V$.

Theorem 2: Under non-location based scheduling policies, each real vehicle RV_m is a $\Sigma GI/GI/1$ queue with task arrival rate $\lambda^R = \frac{\sum_{k=1}^K \lambda^V_k}{M}$. Moreover, assume homogeneous real vehicle subregions as in Definition 3. Then

(i) all the real vehicle $\Sigma GI/GI/1$ queues are probabilistic replicas of each other, i.e., the interarrival time and service time of each queue are identically distributed, separately.

(ii) Let $\lambda^V_k \leq \mu^V$. When $\kappa < \kappa^c$, the $\Sigma GI/GI/1$ queue at each real vehicle is stable and TD_k exists. When $\kappa > \kappa^c$, the $\Sigma GI/GI/1$ queue at each real vehicle is unstable when $\lambda^V_k = \mu^V$, with κ and κ^c defined in (16) and (24).

Proof: (i) By Definition 1, our M-Voronoi tessellation creates equitable subregions and $\mathbf{A}_m$ is a Voronoi subregion. Then each task in the sequence $\langle Task_{ki}\rangle_{i=1}^\infty$ falls in subregion $\mathbf{A}_m$ with probability $\frac{1}{M}$. Hence the arrival time of tasks in the sequence $\langle Task_{ki}\rangle_{i=1}^\infty$ that fall in $\mathbf{A}_m$ is a thinned process of $\{T^a_{ki}\}_{i=1}^\infty$ with retaining probability $p = \frac{1}{M}$. We denote the thinned arrival process as $\{T^{ap}_{ki}\}_{i=1}^\infty$. Since $\{T^a_{ki}\}_{i=1}^\infty$ is renewal with rate λ^V_k, then $\{T^{ap}_{ki}\}_{i=1}^\infty$ is renewal with rate $\frac{\lambda^V_k}{M}$ by Example 4.3(a) in [12, pp. 75-76]. Thus the arrival

process in subregion $\mathbf{A}_m$, $\left\{T^a_{(mj)}\right\}_{j=1}^{\infty}$, is the superposition of $\{T^{ap}_{ki}\}_{i=1}^{\infty}$, $k = 1, \ldots, K$, or K independent renewal processes. This proves the ΣGI. By non-location based the distance between two consecutively executed tasks is i.i.d.. Since the task size is i.i.d., then the service times at RV_m, $T^{Rserv}_{(mj)}$, are i.i.d. in j. This proves the second GI. A renewal process is also stationary, so $\left\{T^a_{(mj)}\right\}_{j=1}^{\infty}$ is stationary, and the arrival rate at RV_m is $\lambda^R = \frac{\sum_{k=1}^{K}\lambda^V_k}{M}$ by [21, Section 14].

Moreover, the M thinned processes $\{T^{ap}_{ki}\}_{i=1}^{\infty}$ generated from the same VV_k arrival process are probabilistic replicas of each other because the retaining probabilities are all $\frac{1}{M}$. Since the arrival process of each RV subregion, $\left\{T^a_{(mj)}\right\}_{j=1}^{\infty}$, is the superposition of thinned replicas from each VV, then $\left\{T^a_{(mj)}\right\}_{j=1}^{\infty}$ are probabilistic replicas in m. Homogeneous RV subregions ensures that the service times at RV_m, $T^{Rserv}_{(mj)}$, are i.i.d. in m and j. Thus all the real vehicle $\Sigma GI/GI/1$ queues are probabilistic replicas of each other. (i) follows.

(ii) When $\lambda^V_k \leq \mu^V$, when $\kappa < \kappa^c$, $\lambda^R = \frac{\sum_{k=1}^{K}\lambda^V_k}{M} \leq \frac{\sum_{k=1}^{K}\mu^V}{M} = \kappa\mu^V < \kappa^c\mu^V = \mu^R$, thus each $\Sigma GI/GI/1$ queue at RV_m is stable by Loynes' stability condition [25]. $Task_{ki}$ corresponds to $Task_{(mj)}$, we denote by $Task_{ki(mj)}$ for the same task. By (20) we know

$$T^{Rsys}_{ki(mj)} \rightarrow_p T^{Rsys}_{k(m)} \tag{25}$$

Since the all the RV $\Sigma GI/GI/1$ queues are probabilistic replicas of each other, then $T^{Rsys}_{k(m)}$ is i.i.d. in m, we denote by T^{Rsys}_k the generic term of $T^{Rsys}_{k(m)}$. Since a task from VV_k can fall in any of the RV subregions with probability $\frac{1}{M}$, thus (25) becomes

$$T^{Rsys}_{ki(mj)} \rightarrow_p T^{Rsys}_k \tag{26}$$

Also, when $\lambda^V_k < \mu^V$, the $GI/GI/1$ queue at each VV is stable, $T^{Vsys}_{ki} \rightarrow T^{Vsys}_k$ in distribution. When $\lambda^V_k = \mu^V$, by our assumption that follows Theorem 1, the customer will provide arrival process that guarantees $T^{Vsys}_{ki} \rightarrow T^{Vsys}_k$ in distribution. Thus both T^{Rsys}_{ki} and T^{Vsys}_{ki} converges to T^{Rsys}_k and T^{Vsys}_k in distribution. The tardiness TD_k defined in (8) exists.

When $\kappa > \kappa^c$ and $\lambda^V_k = \mu^V$, $\lambda^R = \frac{\sum_{k=1}^{K}\lambda^V_k}{M} = \frac{\sum_{k=1}^{K}\mu^V}{M} = \kappa\mu^V > \kappa^c\mu^V = \mu^R$. Thus each $\Sigma GI/GI/1$ queue at RV_m is unstable by Loynes' stability condition [25]. ■

IV. Scheduling Policies

In this section, we design the scheduling policies inside each RV subregion. We assumed that the task arrival process $\{T^a_{ki}\}$ of VV_k is renewal with generic interarrival time I^a_k in Section II. In this section we further assume that I^a_k is i.i.d. in k, i.e., the interarrival times $I^a_{ki} = T^a_{ki} - T^a_{k(i-1)}$ are i.i.d. in k and i. We thus denote by I^a the generic term of I^a_k. Then the task arrival rate are the same for each VV, $\lambda^V_k = \lambda^V = \frac{1}{E[I^a]}$.

Definition 4: The set of K virtual vehicles are said to host *homogeneous* tasks if the task interarrival times I^a_{ki}, locations X_{ki} and sizes T^S_{ki} are all i.i.d. in k and i, separately, $k = 1, \ldots, K$ and $i = 1, 2, \ldots$.

In this section we assume that all the VVs host homogeneous tasks. Then the steady state real system time for VV_k, T^{Rsys}_k, is identically distributed in k. We denote by T^{Rsys} the generic term of T^{Rsys}_k. Then (25) and (26) become

$$T^{Rsys}_{ki(mj)} \rightarrow_p T^{Rsys} \tag{27}$$

Also the steady state virtual system time of VV_k, T^{Vsys}_k, is i.i.d. in k. We denote by T^{Vsys} the generic value of T^{Vsys}_k. Thus

$$T^{Vsys}_{ki(mj)} \rightarrow_p T^{Vsys} \tag{28}$$

Then the tardiness TD_k are the same for all the VVs, thus TD is not only the average value but also the generic term of TD_k. We have

$$TD_k = TD = \frac{E\left[\max\left\{T^{Rsys} - T^{Vsys}, 0\right\}\right]}{E\left[T^{Vserv}\right]} \tag{29}$$

From (27), (28) and (29) we know that when the virtual vehicles are homogeneous and the real vehicle subregions are homogeneous, it suffices to analyze only one RV subregion, and the results represent the generic results of the system.

Let Φ denote a class of non-location based scheduling policies that are non-preemptive and deadline smooth. A scheduling policy is said to be non-preemptive if under this policy the real vehicle always complete an initiated task even when a priority task enters the system in the meanwhile. A scheduling policy is said to be deadline smooth if under this policy the RV serves all the tasks including those whose deadline has passed [27]. The common policies in queueing theory such as FCFS, LCFS, ROS and SJF [38] are non-location based, non-preemptive and deadline smooth policies. Our scheduling policies introduced in this section such as EVDF, EDVDF and credit scheduling policy are shown to be non-location based, non-preemptive and deadline smooth policies in Theorem 3.

Let TD^{ϕ} denote the tardiness as defined by (29) under scheduling policy $\phi \in \Phi$. We consider the following optimization problem:

Find $\psi \in \Phi$ s.t.

$$TD^{\psi} = \min_{\phi \in \Phi} TD^{\phi} \tag{30}$$

for every $k = 1, \ldots, K$.

We propose the scheduling policies Earliest Virtual Deadline First (EVDF) when the task size is known a priori in Definition 5, its variation Earliest Dynamic Virtual Deadline First (EDVDF) when the task size is not known a priori in Definition 6, and the credit scheduling policy in Definition 7. These scheduling policies are motivated by the simple earliest deadline first and credit scheduler in the cloud computing literature [8]. The scheduling quantum for us is the task size. The size cannot be preempted. The Xen schedulers [8] have a constant scheduling quantum and can preempt tasks. In our kind of cloud computing preemption would waste the time spent traveling to the location. In this chapter we analyze

the value of cloud computing with moving servers without preemption.

Definition 5: Under the Earliest Virtual Deadline First (EVDF) scheduling policy, when a real vehicle becomes available, the real vehicle always hosts the virtual vehicle whose current task has the earliest virtual deadline as defined in (4) from the pool of virtual vehicles whose current task falls in the real vehicle subregion.

Definition 6: Under the Earliest Dynamic Virtual Deadline First (EDVDF) scheduling policy, when a real vehicle becomes available, the real vehicle always hosts the virtual vehicle whose current task has the earliest dynamic virtual deadline as defined in (36) from the pool of virtual vehicles whose current task falls in the real vehicle subregion.

Definition 7: Under the credit scheduling policy, when a real vehicle becomes available, the real vehicle always hosts the virtual vehicle with the maximum current credit as described in Section IV-C from the pool of virtual vehicles whose current task falls in the real vehicle subregion.

A. Earliest Virtual Deadline First

The optimality of our EVDF scheduling policy among all the non-location based non-preemtive and deadline smooth scheduling policies follows from Theorem 1 of [27]. We restate this result for convenience below.

Theorem 1 of [27]: For any convex function $g : \mathbb{R} \to \mathbb{R}$, $E\left[g\left(R^{\phi}\right)\right] \leq E\left[g\left(R^{\psi}\right)\right]$ whenever $\phi \ll \psi$.

Theorem 1 of [27] assumes a $G/GI/1$ queue served by non-preemtive and deadline smooth scheduling policies. The G in a $G/GI/1$ queue means the task arrival process is stationary and ergodic. ϕ and ψ denote two admissible non-preemptive and deadline smooth scheduling policies. $\phi \ll \psi$ when ϕ always chooses a customer having a deadline earlier than that of the customer chosen by ψ. In particular the earliest deadline first (EDF) scheduling policy always gives priority to the customer having the earliest deadline, and the latest deadline first (LDF) one gives priority to the customer having the latest deadline. Then, by definition $EDF \ll \phi \ll LDF$ for any admissible scheduling policy ϕ. R^{ϕ} is the steady-state value of $R_n = D_n - T_n - W_n$ under policy ϕ, where D_n, T_n and W_n are the deadline, arrival time and waiting time of the n-th task.

The following theorem shows that our EVDF, EDVDF and credit scheduling policy are in the class of non-location based, non-preemptive, and deadline smooth scheduling policies Φ. Moreover, EVDF optimizes tardiness within this class.

Theorem 3: (i) Let $\phi \in \{EVDF, EDVDF, Credit\}$, as in Definitions 5, 6 and 7, then $\phi \in \Phi$, i.e., ϕ is non-location based, non-preemptive, and deadline smooth.

(ii) Assume homogeneous virtual vehicles and homogeneous real vehicle subregions, let $\lambda^R < \mu^R$, with λ^R and μ^R defined in Theorem 2 and (23). Then $TD^{EVDF} = \min_{\phi\in\Phi} TD^{\phi}$.

Proof: (i) The EVDF, EDVDF and credit scheduling policies schedule only based on virtual deadlines, dynamic virtual deadlines of each task and credit of each VV, separately. They are independent of the distance between two consecutively executed tasks, $D_{(mj)}$. Thus $D_{(mj)}$ is the distance between two random task locations in subregion $\mathbf{A}_m$. Thus $D_{(mj)}$ is i.i.d. in j and has the same distribution as $D_{(m)}$. Thus the three policies are non-location based. The three policies always serve all tasks, even if deadlines have passed. Thus they are smooth with respect to the virtual deadlines as defined in (4). The three policies always completes an initiated task even when a priority task enters the system in the meanwhile. Thus the three policies are non-preemptive. This proves part (i).

(ii) Since $\phi \in \Phi$ is non-location based, the queue in an RV subregion is $\Sigma GI/GI/1$ by Theorem 2. Since ΣGI is a subset of G, then a $\Sigma GI/GI/1$ queue is also a $G/GI/1$ queue. Also, ϕ is non-preemptive and deadline smooth, Thus Theorem 1 of [27] holds in each RV subregion under $\phi \in \Phi$.

We define $R_{(mj)} = T^{dead}_{(mj)} - T^{a}_{(mj)} - W^{R}_{(mj)}$, where $W^{R}_{(mj)}$ is the waiting time of $Task_{(mj)}$, and is defined as the time difference between the arrival time $T^{a}_{(mj)}$ and when RV_m begins to travel to $Task_{(mj)}$. Thus $T^{Rsys}_{(mj)} = W^{R}_{(mj)} + T^{Rserv}_{(mj)}$.

Thus
$\max\left\{T^{Rsys}_{(mj)} - T^{Vsys}_{(mj)}, 0\right\} = -\min\left\{T^{Vsys}_{(mj)} - T^{Rsys}_{(mj)}, 0\right\}$
$= -\min\left\{T^{dead}_{(mj)} - T^{a}_{(mj)} - W^{R}_{(mj)} - T^{Rserv}_{(mj)}, 0\right\}$
$= -\min\left\{R_{(mj)} - T^{Rserv}_{(mj)}, 0\right\}$.

When $\lambda^R < \mu^R$, the $\Sigma GI/GI/1$ queue at RV_m is stable, we have $R_{(mj)} \to R_{(m)} = T^{dead}_{(m)} - T^{a}_{(m)} - W^{R}_{(m)}$ in distribution. Since all the RV subregions are homogeneous, we can write the generic term $R = T^{dead} - T^{a} - W^{R}$.

Thus $E\left[\max\left\{T^{Rsys} - T^{Vsys}, 0\right\}\right]$
$= E\left[-\min\left\{R - T^{Rserv}, 0\right\}\right]$
$= \int_{t=0}^{\infty} E\left[-\min\left\{R - t, 0\right\}\right] dF_{T^{Rserv}}(t)$,
where $F_{T^{Rserv}}(t)$ is the cumulative distribution function (cdf) of T^{Rserv}. Since function $g(x) = -\min\{x - t, 0\}$ is a convex function when t is a constant, and R has the same definition as R in Theorem 1 of [27], then $E\left[-\min\left\{R^{\phi} - t, 0\right\}\right] \leq E\left[-\min\left\{R^{\psi} - t, 0\right\}\right]$ when $\phi \ll \psi$ for any constant t by Theorem 1 of [27]. Thus $E\left[\max\left\{T^{Rsys} - T^{Vsys}, 0\right\}\right]^{\phi} \leq E\left[\max\left\{T^{Rsys} - T^{Vsys}, 0\right\}\right]^{\psi}$ when $\phi \ll \psi$, where the superscript $^{\phi}$ means the value is obtained when the scheduling policy is ϕ.

We know $TD = \frac{E\left[\max\left\{T^{Rsys} - T^{Vsys}, 0\right\}\right]}{E[T^{Vserv}]}$ by (29). Notice that $E\left[T^{Vserv}\right]$ is a constant, then $TD^{\phi} \leq TD^{\psi}$ when $\phi \ll \psi$. In particular, $TD^{EVDF} \leq TD^{\phi}$ since $EVDF \ll \phi$ for any $\phi \in \Phi$. Thus $TD^{EVDF} = \min_{\phi\in\Phi} TD^{\phi}$. ■

Different task arrival processes will generate different tardiness values. We identify a worst-case arrival process maximizing tardiness. We prove the special case $\eta = 1$ of Definition 8 generates the worst case. This is Theorem 4.

Definition 8: An arrival process $\{T^{a}_{ki}\}$ is called an η-arrival process if

$$T^{a}_{ki} = \begin{cases} 0, & i \leq \eta \\ T^{dead}_{k(i-\eta)}, & i > \eta \end{cases} \tag{31}$$

where $\eta \in \mathbb{N}$.

For $\eta = 1$ the definition implies the arrival of the current task is the virtual deadline of the previous task. Thus the service times are also the interarrival times and the process at the output of the VV is identical to the arrival process, i.e., it is also an $\eta = 1$ process. The following theorem establishes the special role of the $\eta = 1$ process.

Theorem 4: Assume homogeneous virtual vehicles and homogeneous real vehicle subregions under the EVDF scheduling policy, let $\kappa \leq \kappa^c$, then the η-arrival process with $\eta = 1$ for all the virtual vehicles achieves the maximum TD among all the renewal processes.

Proof: To prove the η-arrival process with $\eta = 1$ maximizes TD, we show for any given task locations $\{X_{ki}\}$ and task sizes $\left\{T^S_{ki}\right\}$, an arbitrary renewal arrival process $\{T^a_{ki}\}$ will have a TD less than that of the $\eta = 1$ arrival process. We denote by $\left\{T^{a(\eta=1)}_{ki}\right\}$ the $\eta = 1$ arrival process. Thus $T^{a(\eta=1)}_{ki} = T^{dead(\eta=1)}_{k(i-1)}$ by Definition 8. For an arbitrary renewal arrival process $\{T^a_{ki}\}$, we construct an arrival process $\left\{T^{a1}_{ki}\right\}$ such that $T^{a1}_{ki} = \max\left\{T^a_{ki}, T^{a(\eta=1)}_{ki}\right\}$ for all k and i. Thus $T^{a1}_{ki} = \max\left\{T^a_{ki}, T^{dead(\eta=1)}_{k(i-1)}\right\}$. We construct another arrival process $\left\{T^{a2}_{ki}\right\}$ such that $T^{a2}_{ki} = \max\left\{T^{a1}_{ki}, T^{dead1}_{k(i-1)}\right\}$. Note the three processes are constructed to have the same $\{X_{ki}\}$ and $\left\{T^S_{ki}\right\}$. With the same $\{X_{ki}\}$ and $\left\{T^S_{ki}\right\}$, we denote by $TD^{\phi}\left(\{T^a_{ki}\}\right)$ the tardiness under policy ϕ when the arrival processes of the VV_k are $\{T^a_{ki}\}$. We want to show $TD^{EVDF}\left(\{T^a_{ki}\}\right) \leq TD^{EVDF}\left(\left\{T^{a2}_{ki}\right\}\right)$ and $TD^{EVDF}\left(\left\{T^{a2}_{ki}\right\}\right) \leq TD^{EVDF}\left(\left\{T^{a(\eta=1)}_{ki}\right\}\right)$, separately.

Since $\{X_{ki}\}$ and $\left\{T^S_{ki}\right\}$ are the same for all three processes, then the service times T^{Vserv}_{ki} are the same for the processes. For an arbitrary arrival process $\{T^a_{ki}\}$, we have $T^{dead}_{ki} \geq T^{dead(\eta=1)}_{ki}$ by (4) and Definition 8.

Comparing $\{T^a_{ki}\}$ and $\left\{T^{a1}_{ki}\right\}$, we have $T^a_{ki} \leq T^{a1}_{ki}$, and $T^{dead}_{ki} = T^{dead1}_{ki}$ for all k and i. The latter can be seen by induction on (4). First, $T^{dead}_{k1} = T^{dead1}_{k1}$. Second, if $T^{dead}_{k(i-1)} = T^{dead1}_{k(i-1)}$, then $T^{dead}_{ki} = \max\left\{T^a_{ki}, T^{dead}_{k(i-1)}\right\} + T^{Vserv}_{ki}$, and $T^{dead1}_{ki} = \max\left\{\max\left\{T^a_{ki}, T^{dead(\eta=1)}_{k(i-1)}\right\}, T^{dead1}_{k(i-1)}\right\} + T^{Vserv}_{ki} = \max\left\{T^a_{ki}, T^{dead1}_{k(i-1)}\right\} + T^{Vserv}_{ki}$ since $T^{dead1}_{k(i-1)} \geq T^{dead(\eta=1)}_{k(i-1)}$. Thus $T^{dead}_{k(i-1)} = T^{dead1}_{k(i-1)}$ implies $T^{dead}_{ki} = T^{dead1}_{ki}$. This is true for every i by induction. Similarly, comparing $\left\{T^{a1}_{ki}\right\}$ and $\left\{T^{a2}_{ki}\right\}$, we have $T^{a1}_{ki} \leq T^{a2}_{ki}$, and $T^{dead1}_{ki} = T^{dead2}_{ki}$ by induction. Thus $T^a_{ki} \leq T^{a1}_{ki} \leq T^{a2}_{ki}$ and $T^{dead}_{ki} = T^{dead1}_{ki} = T^{dead2}_{ki}$ for all k and i.

In each RV_m subregion, we construct a scheduling policy $\phi \in \Phi$, possibly non-work-conserving, such that under ϕ and $\{T^a_{ki}\}$, $k = 1, \ldots, K$, RV_m will copy the behavior of RV_m under $EVDF$ and $\left\{T^{a2}_{ki}\right\}$, i.e., RV_m flies to, executes, and completes each task at the same time, separately, comparing the two cases. Such ϕ exists because the set of tasks available to RV_m under $\left\{T^{a2}_{ki}\right\}$ is always a subset of the set of tasks available to RV_m under $\{T^a_{ki}\}$ at any time since $T^a_{ki} \leq T^{a2}_{ki}$ for all k and i. Thus $TD^{\phi}\left(\{T^a_{ki}\}\right) = TD^{EVDF}\left(\left\{T^{a2}_{ki}\right\}\right)$. Also $TD^{EVDF}\left(\{T^a_{ki}\}\right) \leq TD^{\phi}\left(\{T^a_{ki}\}\right)$ by Theorem 3. So $TD^{EVDF}\left(\{T^a_{ki}\}\right) \leq TD^{EVDF}\left(\left\{T^{a2}_{ki}\right\}\right)$ for any given $\{X_{ki}\}$ and $\left\{T^S_{ki}\right\}$.

Under the arrival process $\left\{T^{a2}_{ki}\right\}$, we have $T^{a2}_{ki} = \max\left\{T^{a1}_{ki}, T^{dead1}_{k(i-1)}\right\}$ and $T^{dead1}_{k(i-1)} = T^{dead2}_{k(i-1)}$. By (4) $T^{dead2}_{ki} = \max\left\{\max\left\{T^{a1}_{ki}, T^{dead1}_{k(i-1)}\right\}, T^{dead2}_{k(i-1)}\right\} + T^{Vserv}_{ki} = \max\left\{T^{a1}_{ki}, T^{dead1}_{k(i-1)}\right\} + T^{Vserv}_{ki} = T^{a2}_{ki} + T^{Vserv}_{ki}$. Thus $T^{Vsys2}_{ki} = T^{dead2}_{ki} - T^{a2}_{ki} = T^{Vserv}_{ki}$. Thus the virtual system time is the same under both $\left\{T^{a2}_{ki}\right\}$ and $\left\{T^{a(\eta=1)}_{ki}\right\}$, and equals T^{Vserv}_{ki}. Also, $T^{a2}_{ki} - T^{dead2}_{k(i-1)} = \max\left\{T^{a1}_{ki}, T^{dead1}_{k(i-1)}\right\} - T^{dead2}_{k(i-1)} = \max\left\{T^{a1}_{ki}, T^{dead2}_{k(i-1)}\right\} - T^{dead2}_{k(i-1)} \geq 0$. Thus under $\left\{T^{a2}_{ki}\right\}$, VV_k idles for $T^{a2}_{ki} - T^{dead2}_{k(i-1)} \geq 0$ before generating $Task_{ki}$. But under $\left\{T^{a(\eta=1)}_{ki}\right\}$, VV_k never idles, i.e., $T^{a(\eta=1)}_{ki} - T^{dead(\eta=1)}_{k(i-1)} = 0$.

When every VV is under $\left\{T^{a(\eta=1)}_{ki}\right\}$, the interarrival time is T^{Vserv}_{ki}. The arrival process of the subregion of RV_m, $\left\{T^{a(\eta=1)}_{(mj)}\right\}$, is the superposition of K thinned renewal processes of $\left\{T^{a(\eta=1)}_{ki}\right\}$ by Theorem 2. When every VV is under $\left\{T^{a2}_{ki}\right\}$, the interarrival time is $T^{Vserv}_{ki} + Id_{ki}$, where $Id_{ki} \geq 0$ is the idle time between two consecutive tasks. Thus the arrival process of the subregion of RV_m, $\left\{T^{a2}_{(mj)}\right\}$, is the superposition of K thinned renewal processes of $\left\{T^{a2}_{ki}\right\}$. Thus the generic interarrival time of $\left\{T^{a2}_{(mj)}\right\}$, $I^{a2}_{(m)}$, is stochastically greater than that of $\left\{T^{a(\eta=1)}_{(mj)}\right\}$, $I^{a(\eta=1)}_{(m)}$, i.e., $I^{a2}_{(m)} \geq_{st} I^{a(\eta=1)}_{(m)}$. A random variable A is stochastically greater than B, denoted $A \geq_s tB$, if $P(A > t) \geq P(B > t)$ for all $-\infty < t < \infty$. Thus, we can construct a scheduling policy $\phi \in \Phi$ in the subregion of RV_m under $\left\{T^{a2}_{(mj)}\right\}$ such that under ϕ, RV_m executes tasks in the same order as EVDF but always idles for $I^{a2}_{(m)} - I^{a(\eta=1)}_{(m)}$ right after the completion of each task. Since the virtual system time of each task are the same for both cases. Thus $TD^{\phi}\left(\left\{T^{a2}_{ki}\right\}\right) = TD^{EVDF}\left(\left\{T^{a(\eta=1)}_{ki}\right\}\right)$. Also $TD^{EVDF}\left(\left\{T^{a2}_{ki}\right\}\right) \leq TD^{\phi}\left(\left\{T^{a2}_{ki}\right\}\right)$ by Theorem 3. Thus $TD^{EVDF}\left(\left\{T^{a2}_{ki}\right\}\right) \leq TD^{EVDF}\left(\left\{T^{a(\eta=1)}_{ki}\right\}\right)$ for any given $\{X_{ki}\}$ and $\left\{T^S_{ki}\right\}$.

So we have $TD^{EVDF}\left(\{T^a_{ki}\}\right) \leq TD^{EVDF}\left(\left\{T^{a(\eta=1)}_{ki}\right\}\right)$ for an arbitrary renewal arrival process $\{T^a_{ki}\}$ for any given $\{X_{ki}\}$ and $\left\{T^S_{ki}\right\}$. Thus this is also true when we take expectation on $\{X_{ki}\}$ and $\left\{T^S_{ki}\right\}$. So the η-arrival process with $\eta = 1$ achieves the maximum TD among all the renewal processes. ■

The provider wants to know the right gain κ for a given number of real vehicles M and a given task arrival process $\{T^a_{ki}\}$ for each virtual vehicle for a guaranteed level of tardiness. We assume homogeneous virtual vehicles and homogeneous real vehicle subregions, and consider the case when every customer is fully utilizing their VVs, i.e., $\lambda^V = \mu^V$. Under a scheduling policy ϕ, if one fixes the number of real

vehicles M and increases the number of virtual vehicles, one increases the gain κ, but increases the tardiness TD, thus reducing the performance isolation. Conversely at any level, say $TD = \alpha$ for the tardiness there is a largest value of κ in the sense that any increase in the number of virtual vehicles without increase in M will increase the tardiness TD above the level α. We denote this largest value of κ by $\kappa_\alpha^\phi(M)$ at each α. We denote by $TD^\phi(M,\kappa)$ the tardiness when the number of RVs is M and the gain is κ under scheduling policy ϕ. Thus $\kappa_\alpha^\phi(M)$ is defined as

$$\kappa_\alpha^\phi(M) = \max_{TD^\phi(M,\kappa)\leq\alpha, \lambda^V=\mu^V} \kappa \tag{32}$$

Recall that L is the generic term of L_{ki} as defined in (2), and D is the generic term of $D_{(m)}$ as defined in (21). We assume each RV subregion satisfies

$$E[D] = \frac{c_1 E[L]}{\sqrt{M}}, E\left[D^2\right] = \frac{c_2 E\left[L^2\right]}{M} \tag{33}$$

where c_1 and c_2 are positive constants. In particular, when region $\mathbf{A}$ and subregions $\mathbf{A}_m$ are squares, $c_1 = c_2 = 1$.

The following theorem asserts the largest achievable gain without compromising performance isolation actually increases with the number of real vehicles M. In other words, larger systems are able to support more customers per real vehicle without compromising performance isolation. We call this economy of scale [26].

Theorem 5: Assume homogeneous virtual vehicles and homogeneous real vehicle subregions satisfying (33) under the EVDF scheduling policy. $\kappa_\alpha^{EVDF}(M)$ increases with M. Moreover, when $T_{ki}^S = 0$, $\kappa_\alpha^{EVDF}(M)$ is $\Theta\left(\sqrt{M}\right)$. When $E\left[T^S\right] > 0$, $\kappa_\alpha^{EVDF}(M) < 1 + \frac{E[L]}{v^V E[T^S]}$.

Proof: Since this theorem is only about the EVDF scheduling policy, we omit the superscript ^{EVDF} in the notations.

From (29) we know it suffices to analyze only one RV subregion to obtain the average tardiness of the system $TD = \frac{E\left[\max\left\{T^{Rsys} - T^{Vsys}, 0\right\}\right]}{E[T^{Vserv}]}$.

From Theorem 2 we know that each RV_m is a $\Sigma GI/GI/1$ queue with arrival rate $\lambda^R = \frac{\sum_{k=1}^K \lambda_k^V}{M} = \frac{K\lambda^V}{M} = \kappa\lambda^V = \kappa\mu^V$. Let $T^{Rsys}(M,\kappa)$ denote the real system time when the number of RVs is M and the gain is κ. Let $E[D](M)$ and $\mu^R(M)$ denote the $E[D]$ and μ^R values when the number of RVs is M. If $M_1 < M_2$, then $E[D](M_1) > E[D](M_2)$ by (33). Thus $\mu^R(M_1) < \mu^R(M_2)$ by (23). Notice that the arrival rate of both the $\Sigma GI/GI/1$ queues under M_1 and M_2 are the same $\lambda^R = \kappa\mu^V$. Then $T^{Rsys}(M_1,\kappa) >_{st} T^{Rsys}(M_2,\kappa)$ because the arrival rate does not change but service rate increases. Utilizing the fact that $A \geq_{st} B$ if and only if for all non-decreasing functions g, $E[g(A)] \geq E[g(B)]$, and noticing that the virtual system time T^{Vsys} does not change with M or κ, and $g(x) = \max(x,0)$ is non-decreasing, we have $TD(M_1,\kappa) > TD(M_2,\kappa)$. Thus $\exists\kappa' > \kappa$, s.t. $TD(M_1,\kappa) = TD(M_2,\kappa')$. Thus $\kappa_\alpha(M_1) < \kappa_\alpha(M_2)$ by (32). Thus $\kappa_\alpha(M)$ increases with M.

$\kappa_\alpha(M)$ is defined based on $\lambda^V = \mu^V$. By Theorem 2 we know that the $\Sigma GI/GI/1$ queue at each real vehicle is unstable when $\kappa > \kappa^c$, then we should keep $\kappa \leq \kappa^c$. Thus $\kappa_\alpha(M) \leq \kappa^c = \frac{\frac{E[L]}{v^V} + E\left[T^S\right]}{\frac{E[D]}{v^R} + E[T^S]}$ by (24). When $T_{ki}^S = 0$, substituting (33) we have $\kappa_\alpha(M) \leq \kappa^c = \frac{v^R\sqrt{M}}{c_1 v^V}$. Then $\kappa_\alpha(M)$ is $O\left(\sqrt{M}\right)$.

To establish the lower bound of $\kappa_\alpha(M)$, we first analyze the tardiness under FCFS. Since $K = \kappa_\alpha(M)M$, and $\kappa_\alpha(M)$ increases with M, then as $M \to \infty$, $K \to \infty$. The superposition of independent renewal processes converges to a Poisson process as the number of component processes tend to infinity [9]. Then the $\Sigma GI/GI/1$ queue of each subregion becomes an $M/GI/1$ queue with arrival rate $\lambda^R = \kappa_\alpha(M)\mu^V$ as both M and K goes to infinity. Since the task size $T_{ki}^S = 0$, the service rate of the queue is $\frac{v^R}{E[D]}$ by (23). By Pollaczek-Khinchine formula [31], [22] we have the expected waiting time of a task in the $M/GI/1$ queue under FCFS, $E\left[W^{R(FCFS)}\right] = \frac{\lambda^R \frac{E\left[D^2\right]}{(v^R)^2}}{2\left(1-\lambda^R\frac{E[D]}{v^R}\right)}$.

$T^{Rsys(FCFS)} = W^{R(FCFS)} + \frac{D}{v^R}$, $T^{Vsys} = W^V + \frac{L}{v^V}$, where W^V is the steady state waiting time of a task in the $GI/GI/1$ queue at each virtual vehicle. T^{Vsys} and T^{Vserv} is only determined by the $GI/GI/1$ queue at each virtual vehicle, and do not depend on the scheduling policy of each RV subregion. When $M \to \infty$, $\frac{D}{v^R} \leq \frac{L}{v^V}$ almost surely (a.s.). Thus $T^{Rsys(FCFS)} - T^{Vsys} = W^{R(FCFS)} + \frac{D}{v^R} - W^V - \frac{L}{v^V} \leq W^{R(FCFS)}$ a.s..

Thus $TD^{FCFS} = \frac{E\left[\max\left\{T^{Rsys(FCFS)} - T^{Vsys}, 0\right\}\right]}{E[T^{Vserv}]}$
$\leq \frac{E\left[\max\left\{W^{R(FCFS)}, 0\right\}\right]}{E[T^{Vserv}]} = \frac{E[W^{R(FCFS)}]}{E[T^{Vserv}]}$
$= \frac{\lambda^R \frac{E\left[D^2\right]}{(v^R)^2}}{2E[T^{Vserv}]\left(1-\lambda^R\frac{E[D]}{v^R}\right)}$ a.s. when $M \to \infty$.

Since EVDF achieves minimum TD by Theorem 3, $TD^{EVDF} = \alpha$ implies $\alpha \leq \frac{\lambda^R \frac{E\left[D^2\right]}{(v^R)^2}}{2E[T^{Vserv}]\left(1-\lambda^R\frac{E[D]}{v^R}\right)}$.

Substituting (33) and $\lambda^R = \kappa_\alpha(M)\mu^V$ we have
$\kappa_\alpha(M) \geq \frac{2\alpha E\left[T^{Vserv}\right]}{2\alpha E[T^{Vserv}]\mu^V c_1 \frac{E[L]}{v^R\sqrt{M}} + \mu^V c_2 \frac{E\left[L^2\right]}{(v^R)^2 M}}$
$\geq \frac{2\alpha E\left[T^{Vserv}\right]}{2\alpha E[T^{Vserv}]\mu^V c_1' \frac{E[L]}{v^R\sqrt{M}}} = \frac{v^R\sqrt{M}}{\mu^V c_1' E[L]}$ for some $c_1' > c_1$ when $M \to \infty$. Thus $\kappa_\alpha(M)$ is $\Omega\left(\sqrt{M}\right)$.

Thus we have $\kappa(\alpha)$ is $\Theta\left(\sqrt{M}\right)$ when $T_{ki}^S = 0$.

When $E\left[T^S\right] > 0$, $\kappa_\alpha(M) \leq \kappa^c = \frac{\frac{E[L]}{v^V} + E\left[T^S\right]}{\frac{E[D]}{v^R} + E[T^S]} < \frac{\frac{E[L]}{v^V} + E\left[T^S\right]}{E[T^S]} = 1 + \frac{E[L]}{v^V E[T^S]}$. ∎

We define the travel ratio of VV_k as the expected travel time over the expected service time of a task.

$$r_{tr} = \frac{\frac{E[L]}{v^V}}{E\left[T^{Vserv}\right]} = \frac{\frac{E[L]}{v^V}}{\frac{E[L]}{v^V} + E\left[T^S\right]} \tag{34}$$

The following theorem asserts that the migration cost is bounded.

Theorem 6: Under the allocation policy in Definition 1, $MC_k \leq r_{tr} B_{Vk} v^V$. Moreover, when VV_k is fully utilized $MC_k \geq \left(1 - \frac{1}{M}\right) r_{tr} B_{Vk} v^V$.

Proof: By definition, $MC_k = B_{Vk} \frac{E[Z_k L]}{E\left[I_k^{dead}\right]}$. Z_k indicates migration between two consecutive tasks. Thus $E[Z_k L] \leq E[L]$. By Theorem 1 $\lambda_k^{Vdead} \leq \mu^V$. Since $\lambda_k^{Vdead} = \frac{1}{E\left[I_k^{dead}\right]}$ and $\mu^V = \frac{1}{E[T^{Vserv}]}$, then $E\left[I_k^{dead}\right] \geq E\left[T^{Vserv}\right]$. Thus $MC_k \leq B_{Vk} \frac{E[L]}{E[T^{Vserv}]} = B_{Vk} \frac{\frac{E[L]}{v^V}}{E[T^{Vserv}]} v^V = B_{Vk} r_{tr} v^V$ when substituting (34).

When VV_k is fully utilized, $\lambda_k^{Vdead} = \mu^V$ by Theorem 1, thus $E\left[I_k^{dead}\right] = E\left[T^{Vserv}\right]$. Greater L implies the distance between two consecutive tasks are larger, then it is more probable that the two tasks will fall in different RV subregions and cause a migration, i.e., $P(Z_k = 1 | L)$ increases with L. Thus L and Z_k are positively correlated. Then $Cov(Z_k, L) \geq 0$. Thus $E[Z_k L] = E[Z_k] E[L] + Cov(Z_k, L) \geq E[Z_k] E[L]$. Also, $P(Z_k = 0) = \frac{1}{M}$ and $P(Z_k = 1) = 1 - \frac{1}{M}$ since Z_k indicates whether two consecutive tasks fall in the same RV subregion. Thus $E[Z_k] = 1 - \frac{1}{M}$. Then $MC_k \geq B_{Vk} \frac{E[Z_k] E[L]}{E[T^{Vserv}]} = \left(1 - \frac{1}{M}\right) B_{Vk} \frac{E[L]}{E[T^{Vserv}]} = \left(1 - \frac{1}{M}\right) B_{Vk} r_{tr} v^V$. ■

The following theorem asserts that the slack under the $\eta = 1$ arrival process is bounded.

Theorem 7: Under the η-arrival process with $\eta = 1$, $SL_k \leq r_{tr} \leq 1$.

Proof: By Definition 8, $T_k^{Vsys} = \frac{L}{v^V} + T^S$. Also $T_k^{Rsys} \geq \frac{D}{v} + T^S$. Thus $T_k^{Vsys} - T_k^{Rsys} \leq \frac{L}{v^V} + T^S - \frac{D}{v} - T^S \leq \frac{L}{v^V}$. Thus $SL_k = \frac{E\left[\max\left\{T_k^{Vsys} - T_k^{Rsys}, 0\right\}\right]}{E[T^{Vserv}]} \leq \frac{E\left[\max\left\{\frac{L}{v^V}, 0\right\}\right]}{E[T^{Vserv}]} = \frac{E\left[\frac{L}{v^V}\right]}{E[T^{Vserv}]} = r_{tr} = \frac{\frac{E[L]}{v^V}}{\frac{E[L]}{v^V} + E[T^S]} \leq 1$. ■

B. Earliest Dynamic Virtual Deadline First

When the task sizes are not known a priori, the provider only knows the task size after execution, the RV hosts the VV whose current task has the earliest dynamic virtual deadline.

The task size may not be known a priori in practice as assumed by our EVDF scheduling policy. Therefore we define and evaluate another scheduling policy named Earliest Dynamic Virtual Deadline First (EDVDF) which assigns task virtual deadlines based on an estimated task size prior to task completion and updates size to the true value after completion. (35) and (36) specify the deadline computation. $T_{ki}^{S'}$ denotes the task size estimate. We define the estimated service time of VV_k on $Task_{ki}$ as

$$T_{ki}^{Vserv'} = \frac{L_{ki}}{v^V} + T_{ki}^{S'} \tag{35}$$

The dynamic virtual deadline $T_{ki}^{dead}(t)$ is calculated as follows.

$$T_{ki}^{dead}(t) = \begin{cases} \max\left\{T_{ki}^a, T_{k(i-1)}^{dead}(t)\right\} + T_{ki}^{Vserv'}, & t < T_{ki}^{comp} \\ \max\left\{T_{ki}^a, T_{k(i-1)}^{dead}(t)\right\} + T_{ki}^{Vserv}, & t \geq T_{ki}^{comp} \end{cases} \tag{36}$$

where $T_{k0}^{dead} \equiv 0$ and T_{ki}^{Vserv} is given in (3).

When $Task_{ki}$ is completed at $t = T_{ki}^{comp}$ by an RV, T_{ki}^S is known, $T_{ki}^{dead}(t)$ is updated, the scheduling policy updates all the tasks hosted by VV_k following $Task_{ki}$, i.e., updates $T_{k(i+1)}^{dead}(t), T_{k(i+2)}^{dead}(t), \ldots$ according to equation (36). The real vehicle then serves the next task with the earliest $T_{ki}^{dead}(t)$. In this way the scheduling policy utilizes the actual task sizes as they become known.

In our implementation of EDVDF in Section V, the task size estimate is set to be $T_{ki}^{S'} = E\left[T_k^{Sexe}\right]$, where $\left\{T_k^{Sexe}\right\}$ denotes the sizes of the set of previously executed tasks hosted by VV_k. $E\left[T_k^{Sexe}\right] = 0$ if $\left\{T_k^{Sexe}\right\}$ is empty. Time-series methods can also be used to estimate $T_{ki}^{S'}$.

C. Credit Scheduling Policy

We adopt the credit scheduler described in [41] with some changes for the our spatial case. Under the credit scheduling policy, each VV keeps a balance of credits which can be negative. Each credit has a value of 1 second of RV time while it emulates the VV perfectly. A token bucket algorithm [35] is implemented to manage the credits of each VV. Each VV has a bucket. Credits are added to the bucket at constant rate 1 per second, and are expended during service. The bucket can hold at most c credits. The inflow credits are discarded when the bucket is full.

As shown in Figure 2, VVs are divided into three states: UNDER, with a nonnegative credit balance, OVER, with a negative credit balance, and INACTIVE or halted. The VVs are listed in decreasing order of credit balance. Thus, those in UNDER state are ahead of those in OVER state. The VV at the head of the queue has the most credits and is selected for execution when an RV becomes available. In work-conserving (WC) mode, when no VVs are in the UNDER state, one in the OVER state will be chosen, allowing it to receive more than its share of RV time. In non-work-conserving (NWC) mode, the RV will go idle instead.

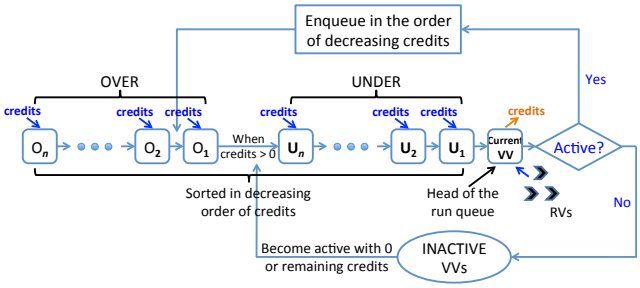


Fig. 2. The credit scheduling policy.

The VV with the most credits is called the current VV, say VV_k. The token bucket algorithm is illustrated in Figure 3. When an RV becomes available, the RV will travel to and execute the current task hosted by VV_k, say $Task_{ki}$. The scheduler debits $T_{ki}^{Vserv'}$ credits from the bucket of VV_k. $T_{ki}^{Vserv'}$ is calculated according to (35). The scheduler finds the VV with the maximum credit balance, and the VV with the most credits will become the new current VV. When $Task_{ki}$ is completed, the scheduler will know the size T_{ki}^S and compute the true service time T_{ki}^{Vserv}. The consumed credits of $Task_{ki}$, T_{ki}^{Vserv}, is calculated according to (3). This is the appropriate

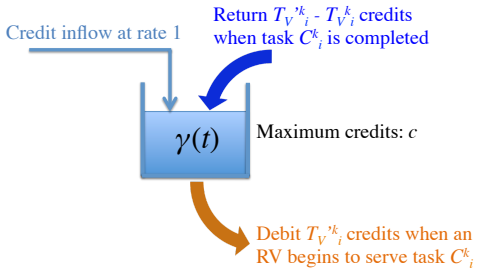


Fig. 3. The token bucket algorithm.

credits, or RV time, the scheduler should debit for executing $Task_{ki}$. The scheduler returns $T_{ki}^{Vserv'} - T_{ki}^{Vserv}$ credits to VV_k to adjust the debited credits to be exactly T_{ki}^{Vserv}. This method of debiting $T_{ki}^{Vserv'}$ before execution and adjusting to T_{ki}^{Vserv} after execution is because the scheduler does not know the size of $Task_{ki}$, and thus does not know T_{ki}^{Vserv}, before execution. If the task size T_{ki}^{S} is known a priori, then $T_{ki}^{Vserv'} = T_{ki}^{Vserv}$. The right amount of credits will be debited at beginning and the adjusted amount equals 0.

When $Task_{ki}$ is completed, there are two ways VV_k goes:

(i) Enqueue according to the credit balance such that the list of VVs is in decreasing order of credits.

(ii) Go INACTIVE if the next $Task_{k(i+1)}$ has not arrived yet, or if the customer's reservation of VV_k ends.

VVs in an INACTIVE state are divided into two categories:

(i) The VV is still under the reservation of a customer, all the arrived tasks hosted by the VV have been executed and the next task has not arrived yet. In this case, the credits continue flowing into the bucket up to a maximum of c. The VV becomes active again with the remaining credit balance and enqueues in decreasing order of credits upon arrival of the next task.

(ii) The VV is not under reservation. In this case, the credits stop flowing to the bucket. The VV becomes active again with 0 credit balance and enqueues the active VVs in decreasing order of credits when a customer begins to reserve it.

V. Experiments

We simulate the system under homogeneous real vehicles and homogeneous virtual vehicles under Earliest Virtual Deadline First (EVDF), Earliest Dynamic Virtual Deadline First (EDVDF) and credit scheduling policies in this section.

A. Simulation Setup

All the simulations are done in a square region $\mathbf{A}$ of size $a \times a$, where $a = 10\,\mathrm{m}$. The number of RVs is M, $\sqrt{M} \in \mathbb{N}$. The square region $\mathbf{A}$ is divided into M square subregions, each with edge length $\frac{a}{\sqrt{M}}$. The M RVs are assigned to the M subregions in a one-to-one manner. Each RV runs a scheduling policy in its subregion. The scheduling policies include the EVDF, EDVDF and the credit scheduling policy. The speed of the RVs equals the virtual speed $v^R = v^V = 1\,\mathrm{m/s}$. Each virtual vehicle hosts tasks with $\eta = 1$ arrival process. The $\eta = 1$ process maximizes tardiness, and excludes multiplexing gain by having each virtual vehicle fully utilized. So this is a worst-case simulation, and the gain observed is migration gain only. The performance measures include the performance isolation and fairness index based on tardiness and delivery probability, together with slack and migration cost. We simulate 1300 tasks per VV but calculate the metrics using only the 100-th to 600-th tasks to ensure the metrics are computed at steady-state. When the 600-th task of each VV is under execution, all the other VVs still have tasks. Each task is uniformly distributed in region $\mathbf{A}$. Table I summarizes the simulation setup.

TABLE I. Simulation Setup

Size of region $\mathbf{A}$	10 m × 10 m
Virtual speed, v^V	1 m/s
RV speed, v^R	1 m/s
Task size, T_{ki}^S	Uniformly distributed
Task location, X_{ki}	Uniformly distributed in region $\mathbf{A}$
Task arrival process, $\{T_{ki}^a\}$	$\eta = 1$ arrival process
# Tasks per VV	1300
Tasks used in metric calculation	100th - 600th task of each VV
Scheduling policies	EVDF, EDVDF and Credit scheduling policy

B. Simulation Results

Figure 4 shows the performance isolations and fairness indices based on tardiness and delivery probability, together with slacks and migration costs under EVDF for different numbers of real vehicles and gains, or different numbers of virtual vehicles, when the task size is zero.

Figure 5 verifies Theorem 5. Subfigures 5(a) and 5(b) are the contours of performance isolations based on tardiness and delivery probability shown in subfigures 4(a) and 4(b) when the task size is zero. We can see that the provider supports a given number of virtual vehicles with significantly fewer real vehicles that travel at the virtual speed while guaranteeing high performance isolation, for example, 750 VVs Vs. 100 RVs while guaranteeing the relative expected tardiness to be less than 1% in 5(a), and 560 VVs Vs. 100 RVs while guaranteeing the average delivery probability is greater than 98%. The migration gain increases in the order of the square root of the number of real vehicles while guaranteeing the same performance isolation based on tardiness and delivery probability, showing economy of scale. Subfigures 5(c) and 5(d) are the contours of performance isolations based on tardiness and delivery probability when the mean task size is 25% of the mean flying time, i.e., $E\left[T^S\right] = \frac{E[L]}{4v^V}$. We can see that the gain is upper bounded by a constant as asserted in Theorem 5. Also, for a given performance isolation level and number of real vehicles, the gain or the number of virtual vehicles hosted decreases as the task size increases comparing Subfigures 5(a) and 5(c), 5(b) and 5(d), separately, for example, 150 VVs Vs. 100 RVs while guaranteeing the relative expected tardiness to be less than 1% in 5(a), and 130 VVs Vs. 100 RVs while guaranteeing the average delivery probability is greater than 98%. The gain diminishes as the task size increases. The virtual vehicle generates gain when traveling, not when standing still.

Figure 6 also shows the diminishing gain as the relative task size $\frac{E\left[T^S\right]v^V}{E[L]}$ increases. For example, to guarantee the average relative tardiness to be less than 1% using 100 RVs, the number of VVs hosted decreases from 750 to 150 as the relative task size increases from 0 to 25%.

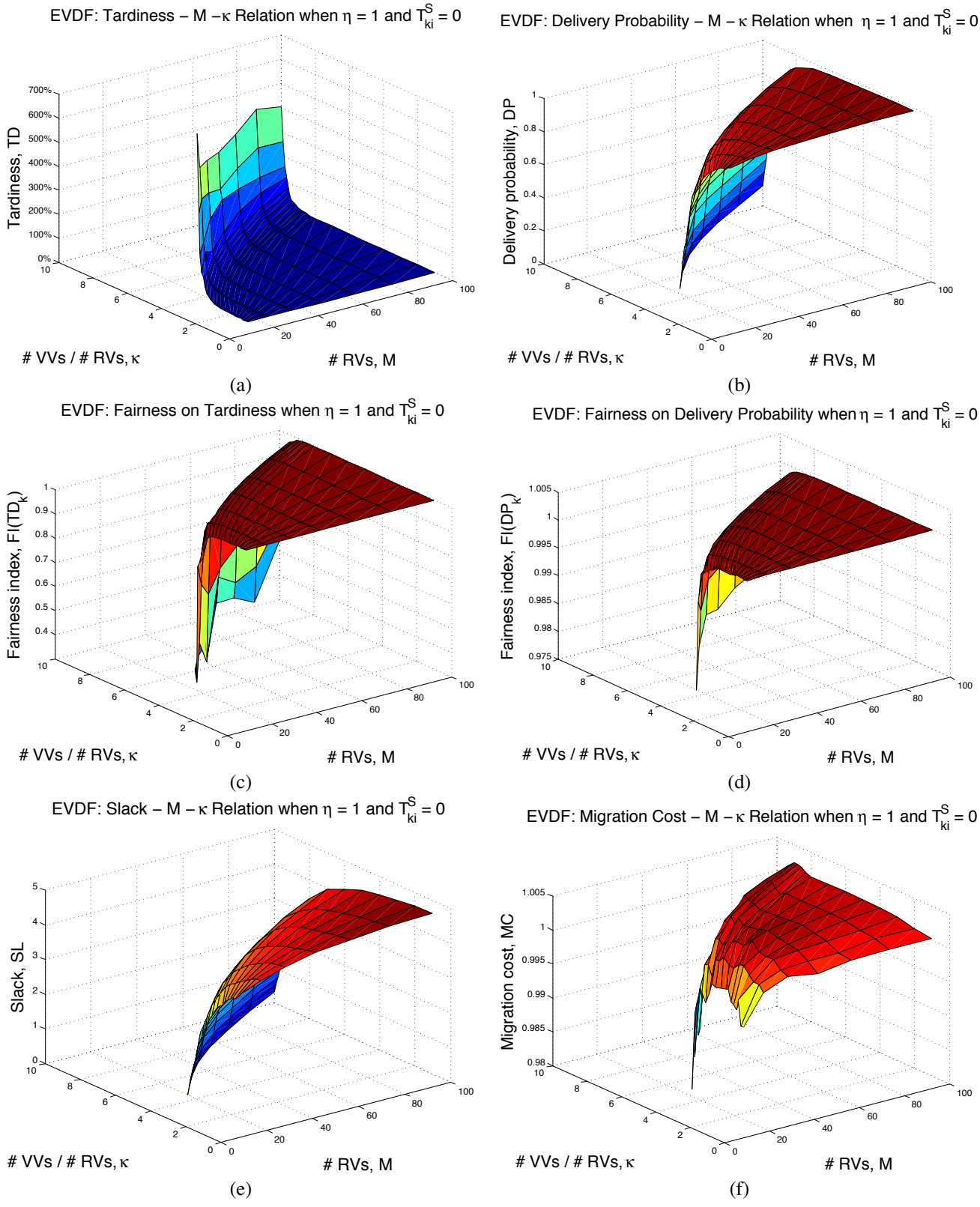


Fig. 4. Tardinesses, delivery probabilities, fairness indices, slacks, and migration costs with different numbers of RVs and gains when the task size is zero under the $\eta = 1$ process under the EVDF scheduling policy.

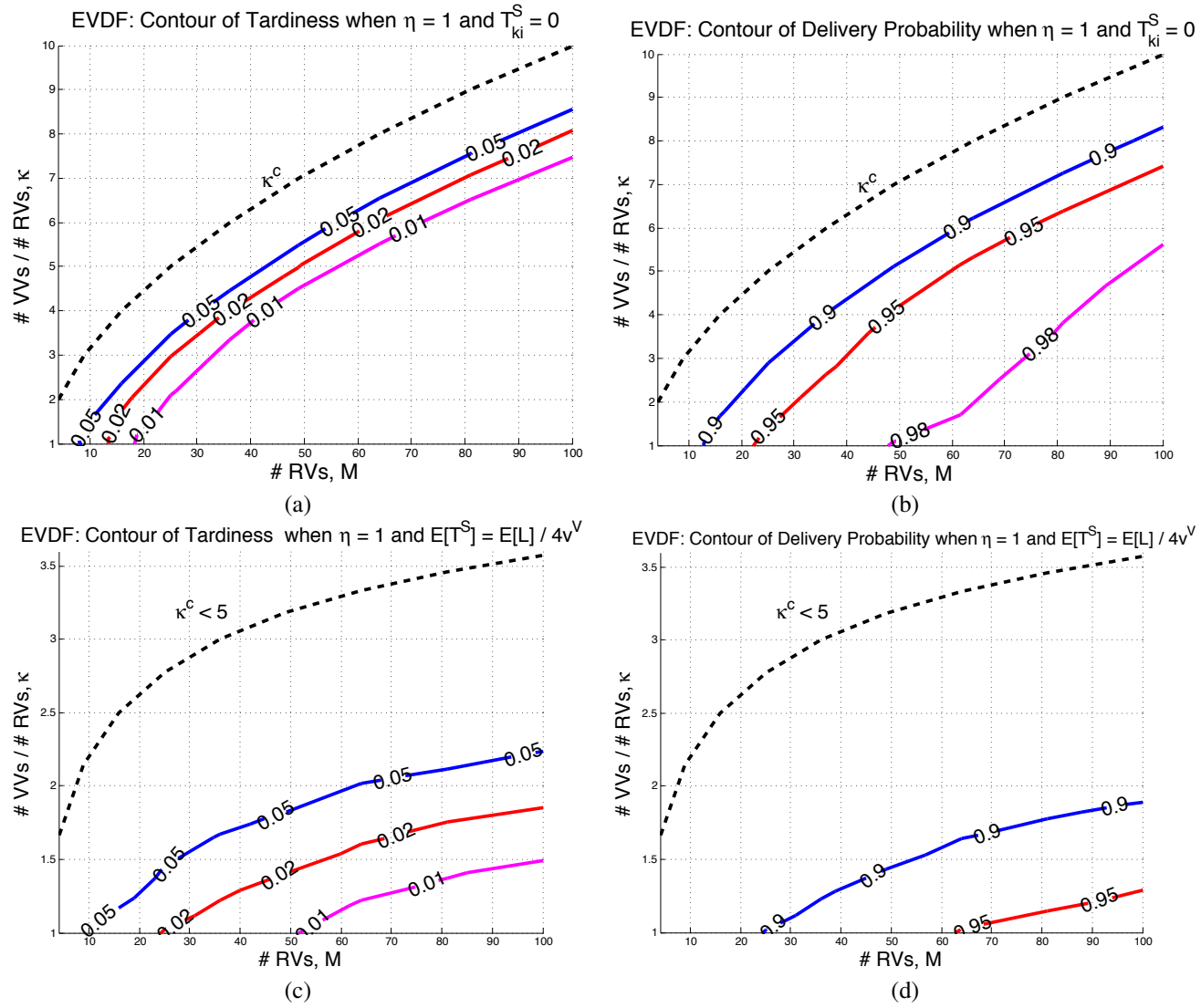


Fig. 5. Contours of tardiness and delivery probability with different numbers of RVs and gains when the task size is zero and $E\left[T^S\right] = \frac{E[L]}{4v^V}$ under the $\eta = 1$ process under the EVDF scheduling policy.

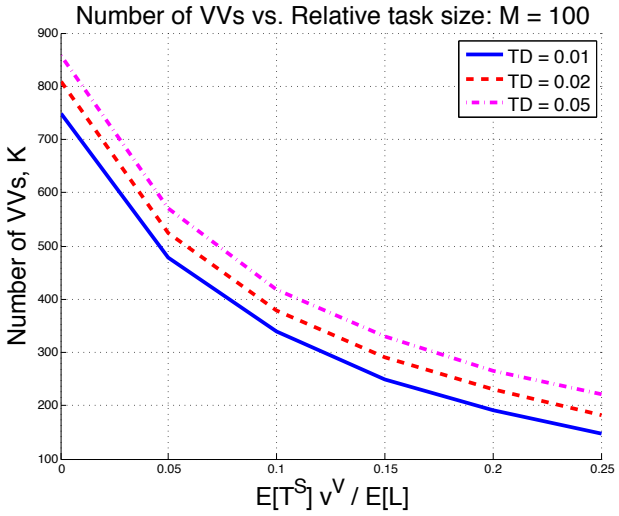
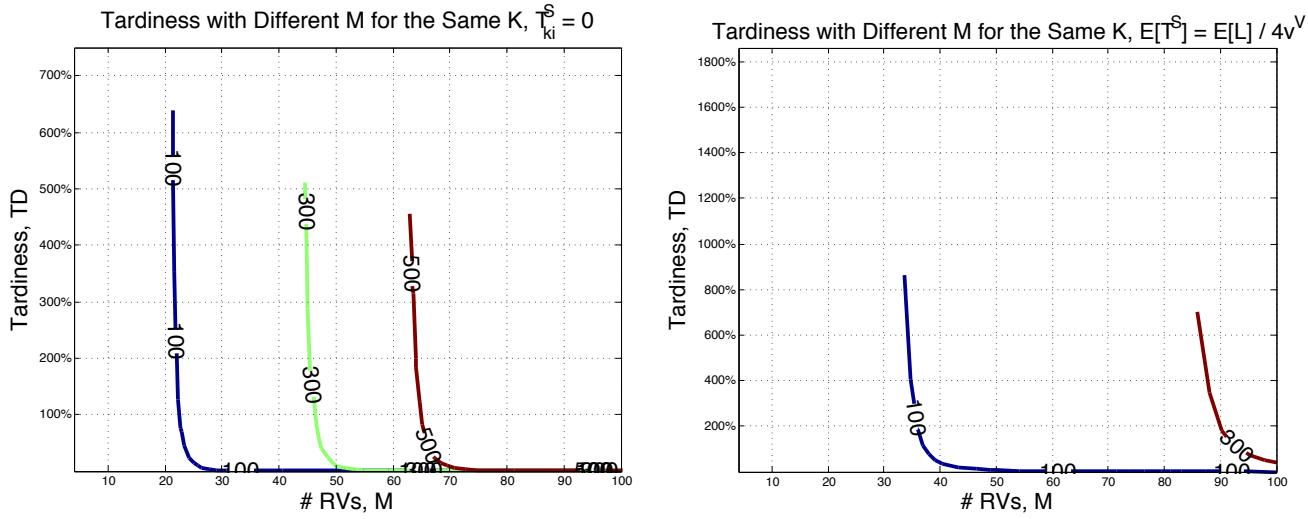


Fig. 6. The number of virtual vehicles hosted by 100 real vehicles to guarantee a certain level of tardiness under EVDF with different task sizes.

Figure 7 shows the performance isolations based on tardiness with different numbers of RVs M to host the same number of VVs K under EVDF with different task sizes. We can see in Subfigure 7(a) that when $T^S_{ki} = 0$ and the number of VVs K is fixed, average relative expected tardiness decreases as the number of RVs M increases. This change becomes very sharp when $M \approx 23$ for the case $K = 100$. The average tardiness is very small once M is greater than 28. This sharp change is also revealed in the case when $K = 300$ and $K = 500$, where the average tardiness becomes very small once $M \geq 52$ and $M \geq 74$, respectively. The transition between low and high performance isolations is sharp. This implies that the system is very easy to operate because the provider can easily determine the appropriate number of real vehicles to host a give number of virtual vehicle and a guaranteed performance isolation. For the case when the mean task size $E\left[T^S\right] = \frac{E[L]}{4v^V}$ as shown in Subfigure 7(b), similar phenomenon follows, but the number of RVs needed to guarantee the same performance isolation increases to host the same number of VVs. This also implies that the gain diminishes as the task size increases.



(a) (b)

Fig. 7. Tardiness with different numbers of RVs M for the same number of VVs under EVDF for different task sizes.

Figure 8 compares EVDF, EDVDF and credit scheduling policies on the same settings as Subfigures 5(c) and 5(d). We can see that EVDF achieves higher gain than EDVDF, and EDVDF achieves higher gain than the credit scheduling policy for a given number of RVs and a guaranteed performance isolation based on both tardiness and delivery probability.

VI. Conclusion

We proposed virtual vehicle in multi-customer systems with location specific tasks to create performance isolation [34], which enables cloud computing in space. In Section II, we illustrated virtual vehicle and its role in cloud computing in space. In the service-level agreement (SLA), each virtual vehicle has a virtual speed v^V, its performance is measured by the tardiness. We used the measure PI [39] to quantify performance isolation, and Jain's fairness indices FI [18] to measure fairness across virtual vehicles. In Section III, we design virtual vehicle allocation and scheduling policies. The allocation is done by dividing the service region into equal subregions. The real vehicles travel less by this division. The scheduling policies include earliest virtual deadline first, earliest dynamic virtual deadline first and credit scheduler, adapted from the CPU schedulers in conventional cloud computing [8]. We simulated the system under the three scheduling policies under the η-arrival process with $\eta = 1$. The $\eta = 1$ case is shown to be the worst case that achieves the largest tardiness among all the renewal processes. Simulation results show that (i) a virtual vehicle performs as well as a real vehicle with high performance isolation. (ii) The provider can support a given number of virtual vehicles with significantly fewer real vehicles that travels at the virtual speed while guaranteeing high performance isolation and bounded migration cost.

References

[1] L. Abeni and G. Buttazzo. Qos guarantee using probabilistic deadlines. In *Real-Time Systems, 1999. Proceedings of the 11th Euromicro Conference on*, pages 242–249, 1999.

[2] S. L. Albin. On poisson approximations for superposition arrival processes in queues. *Management Science*, 28(2):126–137, 1982.

[3] Dragomir Anguelov, Carole Dulong, Daniel Filip, Christian Frueh, Stphane Lafon, Richard Lyon, Abhijit Ogale, Luc Vincent, and Josh Weaver. Google street view: Capturing the world at street level. *Computer*, 43, 2010.

[4] Fabienne Anhalt and Pascale Vicat-Blanc Primet. Analysis and experimental evaluation of data plane virtualization with xen. *International conference on Networking and Services (ICNS'06)*, 0:198–203, 2009.

[5] Dimitris J Bertsimas and Garrett Van Ryzin. A stochastic and dynamic vehicle routing problem in the euclidean plane. *Operations Research*, 39(4):601–615, 1991.

[6] David Boutcher and Abhishek Chandra. Does virtualization make disk scheduling passé? *ACM SIGOPS Operating Systems Review*, 44(1):20–24, 2010.

[7] G.C. Buttazzo. *Soft real-time systems: predictability vs. efficiency*. Series in computer science. Springer, 2005.

[8] Ludmila Cherkasova, Diwaker Gupta, and Amin Vahdat. Comparison of the three CPU schedulers in Xen. *SIGMETRICS Performormance Evaluation Review*, 35(2):42–51, September 2007.

[9] E. Cinlar. Superposition of point processes. In *In Stochastic Point Processes: Statistical Analysis, Theory and Applications (P. A. W. Lewis, ed.)*, pages 549–606. New York: Wiley, 1972.

[10] J.W. Cohen. *The single server queue*. North-Holland series in applied mathematics and mechanics. North-Holland Pub. Co., 1982.

[11] S.S. Craciunas, A. Haas, C.M. Kirsch, H. Payer, H. Röck, A. Rottmann, A. Sokolova, R. Trummer, J. Love, and R. Sengupta. Information-Acquisition-as-a-Service for Cyber-Physical Cloud Computing. In *Proc. Workshop on Hot Topics in Cloud Computing (HotCloud)*, 2010.

[12] D.J. Daley and D. Vere-Jones. *An introduction to the theory of point processes*. Springer series in statistics. Springer-Verlag, 1988.

[13] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6(1):pp. 80–91, 1959.

[14] Yaoyao Gu, Doruk Bozdag, Eylem Ekici, Fsun zgner, and Chang-Gun Lee. Partitioning based mobile element scheduling in wireless sensor networks. In *In. Proc. Second Annual IEEE Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, pages 386–395, 2005.

[15] P. Gupta and P.R. Kumar. The capacity of wireless networks. *Information Theory, IEEE Transactions on*, 46(2):388–404, 2000.

[16] A. Host-Madsen and A. Nosratinia. The multiplexing gain of wireless networks. In *Information Theory, 2005. ISIT 2005. Proceedings. International Symposium on*, pages 2065–2069, 2005.

[17] Jiangchuan Huang and Raja Sengupta. Stability of dynamic traveling repairman problem under polling-sequencing policies. In *European Control Conference*, July 2013.

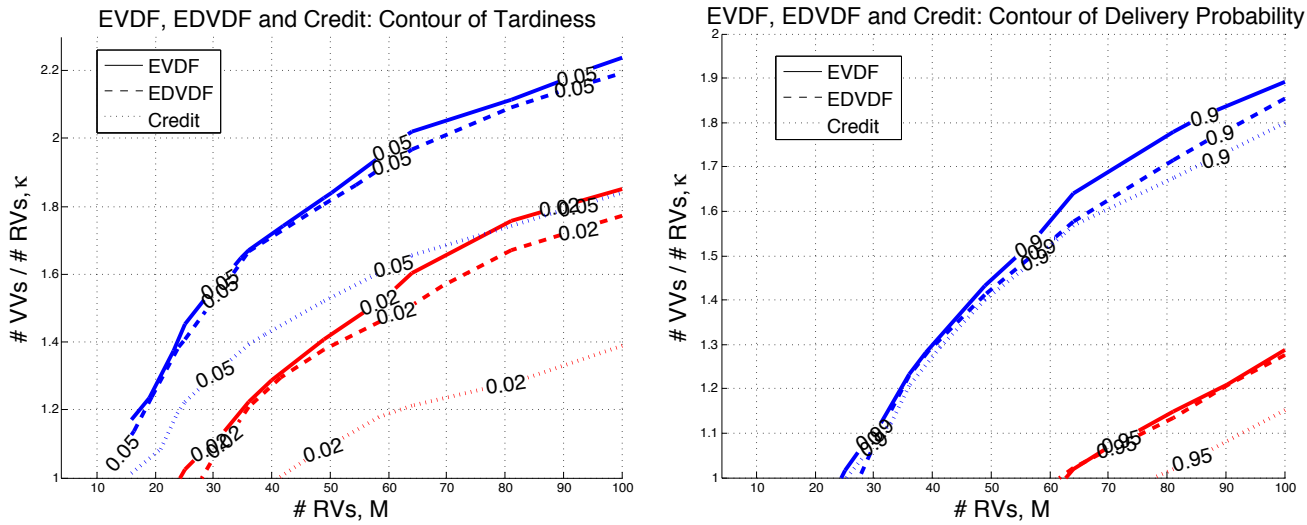


Fig. 8. Comparison of EVDF, EDVDF and Credit scheduling policies on tardiness and delivery probability when the mean task size $E\left[T^S\right] = \frac{E[L]}{4v^V}$ under the $\eta = 1$ process.

[18] Raj Jain, Dah-Ming Chiu, and William R Hawe. *A quantitative measure of fairness and discrimination for resource allocation in shared computer system*. Eastern Research Laboratory, Digital Equipment Corporation, 1984.

[19] Karen Jenvey. NASA Unmanned Aircraft Measures Low Altitude Greenhouse Gases, December 2011.

[20] Mukil Kesavan, Ada Gavrilovska, and Karsten Schwan. On disk i/o scheduling in virtual machines. In *Proceedings of the 2nd conference on I/O virtualization*, pages 6–6. USENIX Association, 2010.

[21] A.I.A. Khinchin. *Mathematical methods in the theory of queueing*. Griffin's statistical monographs & courses. Griffin, 1969.

[22] A.Y. Khinchin and RAND CORP SANTA MONICA CALIF. *The Mathematical Theory of a Stationary Queue*. Defense Technical Information Center, 1967.

[23] C. Kirsch, E. Pereira, R. Sengupta, H. Chen, R. Hansen, J. Huang, F. Landolt, M. Lippautz, A. Rottmann, R. Swick, R. Trummer, and D. Vizzini. Cyber-Physical Cloud Computing: The Binding and Migration Problem. In *Design, Automation and Test in Europe*, 2012.

[24] Gilbert Laporte and Yves Nobert. Exact algorithms for the vehicle routing problem. In Michel Minoux Silvano Martello, Gilbert Laporte and Celso Ribeiro, editors, *Surveys in Combinatorial Optimization*, volume 132 of *North-Holland Mathematics Studies*, pages 147 – 184. North-Holland, 1987.

[25] RM Loynes. The stability of a queue with non-independent inter-arrival and service times. In *Proc. Cambridge Philos. Soc*, volume 58, pages 497–520. Cambridge Univ Press, 1962.

[26] A. Mas-Colell, M.D. Whinston, and J.R. Green. *Microeconomic theory*. Oxford University Press, 1995.

[27] P. Moyal. Convex comparison of service disciplines in real time queues. *Operations Research Letters*, 36(4):496 – 499, 2008.

[28] M. Pavone, A. Arsie, E. Frazzoli, and F. Bullo. Equitable partitioning policies for robotic networks. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 2356 –2361, 2009.

[29] M. Pavone, N. Bisnik, E. Frazzoli, and V. Isler. A stochastic and dynamic vehicle routing problem with time windows and customer impatience. *Mob. Netw. Appl.*, 14(3):350–364, June 2009.

[30] M. Pavone, E. Frazzoli, and F. Bullo. Adaptive and distributed algorithms for vehicle routing in a stochastic and dynamic environment. *Automatic Control, IEEE Transactions on*, 56(6):1259–1274, June 2011.

[31] Felix Pollaczek. Über eine aufgabe der wahrscheinlichkeitstheorie. i. *Mathematische Zeitschrift*, 32(1):64–100, 1930.

[32] Gerald J. Popek and Robert P. Goldberg. Formal requirements for virtualizable third generation architectures. *Commun. ACM*, 17(7):412–421, July 1974.

[33] Benny Rochwerger, David Breitgand, Amir Epstein, David Hadas, Irit Loy, Kenneth Nagin, Johan Tordsson, Carmelo Ragusa, Massimo Villari, Stuart Clayman, et al. Reservoir-when one cloud is not enough. *Computer*, 44(3):44–51, 2011.

[34] Mendel Rosenblum. The reincarnation of virtual machines. *Queue*, 2(5):34–40, July 2004.

[35] Scott Shenker and John Wroclawski. General characterization parameters for integrated service network elements. *RFC 2215*, sept. 1997.

[36] Arun A. Somasundara, Aditya Ramamoorthy, and Mani B. Srivastava. Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines. In *Proceedings of the 25th IEEE International Real-Time Systems Symposium*, RTSS '04, pages 296–305, Washington, DC, USA, 2004.

[37] H. Takagi. *Queueing analysis: a foundation of performance evaluation, vol. 1 : vacation and priority systems*. Queueing Analysis. North-Holland, 1991.

[38] Adam Wierman. *Scheduling for today's computer systems: bridging theory and practice*. PhD thesis, Pittsburgh, PA, USA, 2007.

[39] Kejiang Ye, Xiaohong Jiang, Deshi Ye, and Dawei Huang. Two optimization mechanisms to improve the isolation property of server consolidation in virtualized multi-core server. In *Proceedings of the 2010 IEEE 12th International Conference on HPCC*, pages 281–288, Washington, DC, USA, 2010.

[40] Alin Zhong, Hai Jin, Song Wu, Xuanhua Shi, and Wei Gao. Performance implications of non-uniform vcpu-pcpu mapping in virtualization environment. *Cluster Computing*, pages 1–12, 2012.

[41] Fangfei Zhou, Manish Goel, Peter Desnoyers, and Ravi Sundaram. Scheduler vulnerabilities and attacks in cloud computing. *CoRR*, abs/1103.0759, 2011.