Robust Resource Allocation Using Edge Computing for Delay Sensitive Tasks in Vehicle to Infrastructure (V2I) Networks

Razin Farhan Hussain*, Mohsen Amini Salehi†, Anna Kovalenko*
*The Center for Advanced Computer Studies
{razinfarhan.hussain1,2}@louisiana.edu
†HPCC Laboratory, Computer Science Department
University of Louisiana at Lafayette, Louisiana, USA
{amini}@louisiana.edu

Omid Semiari*
*Georgia Southern University
{osemiari}@georgiasouthern.edu

***Abstract*—Vehicular networks require quick and reliable services due to the hazards of the road environment. The majority of services provided in these networks (*e.g.,* Traffic Information, Weather Updates and Blind-spot Warning) are delay intolerant and there is no value to process them, once their individual deadline is passed. Vehicle to Infrastructure (V2I) systems must provide consistent real-time response to be useful for their users (*e.g.,* drivers or autonomous vehicle). Maintaining real-time response can be particularly difficult when there is a surge of request arrivals (*e.g.,* during peak hours or at a disaster time) to Road Side Units (RSU). RSUs' are edge devices that is defined as Base Station in this work. Accordingly, our goal in this research is to make the V2I system robust and reliable to it's users. That is, we enable the system to maintain its performance even in the presence of uncertain tasks' arrival rate. To achieve robustness, in this paper, we propose a resource allocation model that can dynamically utilize resources from neighboring edge devices when the system is oversubscribed. The model includes a Load Balancer component that allocates the arriving tasks among neighboring Base Stations (BS) prior to the task execution. We propose a method to calculate the probability of completing a task on different base stations. Then, the allocation model assigns the arriving task to the base station that maximizes the probability of completing the task before its deadline. We evaluate our proposed model under different workload conditions and when different scheduling policies are used within the base stations. Simulation results demonstrate that our proposed allocation model provides 40% decrease in task deadline miss rate compare to the conventional V2I architecture.**

***Index Terms*—Base Station, Vehicular Networks, End-to-End delay, Edge Computing, Vehicle to Infrastructure (V2I).**

I. Introduction

Recent advancements in communications and networking stimulated a rapid development of vehicular networks technology. The field of Vehicle to Everything (V2X) communications got the attention of the Federal Communications Commission which already reserved the 5.850 to 5.925 GHz frequency band specifically for V2X. Two types of vehicular communications can be distinguished. The first one is V2V or Vehicle to Vehicle communication. The second type is defined as Vehicle to Infrastructure communication (V2I). V2V and V2I are supposed to complement each other to provide reliable vehicular communication. In reality, considering the nature of the vehicular movement, V2V is pretty hard to implement, and V2I draws the majority of work to itself. The United States Department of Transportation (DoT) names V2I communications the next generation of Intelligent Transportation Systems [1].

Infrastructure in V2I is represented by a special Roadside Unit or, as we are going to call it in this article, a Base Station. A Base Station is a stationary edge device that is installed on the side of the road and possesses some computational power as well as communication capability [2]. Therefore, every vehicle in the network is able to access additional recourses provided by the Base Stations. Wrong Way Driver warning, Cooperative Forward Collision Warning, Lane Change warning, Weather reports, and other services now can be provided to moving vehicles in a real-time fashion. According to such real-time nature, the majority of the services in V2I systems are delay sensitive and require a very short response time. Moreover, after a certain period of time, the execution of a delay sensitive task loses its value.

Deadline is the main execution constraint, and it is directly dependent on the End-to-End Delay. End-to-End Delay in V2I communication is an accumulation of three components: the uplink delay, processing delay, and the downlink delay. Unfortunately, conventional cloud architecture increases the latency problem in V2I services. Communication with distant servers only contributes to the overall delay time of service. One approach currently used is edge computing concept. Being close to end users, the edge eliminates the delay produced by communication with the cloud. However, network delay is not the only source of problem for imposing delay within a vehicular network. In V2I communications Base Stations can have different computational power and communication medium (e. g., wireless, fiber, optic).Such heterogeneity also notably contributes to the overall delay. Another complication can occur during the emergency situations for instance natural disaster or road accidents as well as in the rush hours. A rapid increase of service requests can affect the response time significantly. Therefore, heterogeneity and over subscription

can notably decrease Quality of Service (QoS) in V2I systems.

The major challenge for the vehicular Ad-Hoc networks is an efficient resource allocation which will decrease the deadline missing rate and increase the robustness of the network that leads to the reliability of the system implicitly. There are several approaches that have been proposed. [3] proposes to remove unpromising service requests from the waiting queue before scheduling as well as balancing upload and download requests. In [7] suggests transferring delay tolerant requests to the neighboring Base Stations when receiving base station is overloaded. It also proposes a scheduling algorithm which prioritizes delay-sensitive, small, popular tasks over the delay tolerant, large, unpopular tasks. [5] suggests special models of local resource reservation and reallocation based on the priority of services.

The specific question our research addresses in this article is how to allocate the tasks arriving at an oversubscribed Base Station in a V2I network so that the number of tasks missing their deadline is minimized. We propose a probabilistic resource allocation model that assures robustness of a V2I service to satisfy QoS for the end-users in presence of heterogeneity and fluctuating request arrival in the system. Previous works mostly concentrated on prioritizing the service requests to define allocation algorithm. Our model uses historical data to calculate probabilities of certain task types to meet the deadline in respective Base Stations and allocates according to the calculated probabilities. In summary, the main contributions of this article are:

- Defining robustness for the proposed model with a presence of heterogeneity along the Base Stations and stochastic task arrival rate.
- Defining robustness of a task type as its probability to meet a required deadline in respective Base Station.
- Proposing an efficient task allocation algorithm that considers robustness of the task type and the level of the Base Station subscription.
- Providing heuristics to define task type robustness across the Base Stations.

The rest of the paper is organized as follows. Section 2 introduces the scenario and assumptions made to describe the model. Section 3 includes the problem statement and the proposed system model with formulation. Section 4 discusses probabilistic allocation approach. Sections 5 and 6 present heuristics and simulation. Experimental stage and performance evaluations are particularly described in section 7. Finally, section 8 wraps up the article with a conclusion.

II. Scenario

In V2I communication an ideal scenario of serving moving vehicles from road side Base Stations would be as follows: while moving through the roads, vehicles would be requesting tasks to the road-side Base Stations. As the request is generated, it would travel from the vehicle to the Base Station and after processing the task the result would be sent to the requesting vehicle. Majority of these tasks requested would be delay-sensitive which means the response of each request needs to reach the vehicle within a time frame of certain delay time. This time frame is defined in our research as individual deadline for each specific task.

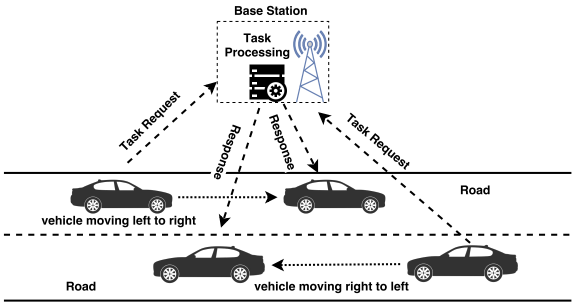


Fig. 1: A Vehicle to infrastructure communication scenario where vehicles send tasks to base station and receive the response from base station. A base station is a road side unit with processing power and communication unit.

III. Problem Statement and System Model

A. Problem Statement

The main goal of this research is to offer a robust Quality of Service in a V2I system in the presence of heterogeneity across base stations and over subscription. In order to strengthen the support of real time task processing in base stations, task allocation process needs to be efficient to meet individual task's deadline.

Therefore the problem can be defined as: *how to allocate arriving tasks to a base station in a V2I network so that the number of tasks missing their deadline is minimized?*

B. System Model

1) ***Formulation****:* In V2I scenario, a set of tasks will be generated by vehicles and send to the base station for processing. Every task has its own deadline within which it has to be completed. So our system will allocate the tasks to Base Stations considering individual deadlines so that number of task missing their deadline can be minimized or number of task meeting their deadline can be maximized.

According to problem definition, the set of arriving tasks can be defined as "T", where T = $\{t_1, t_2, t_3, t_4 \ldots, t_n\}$ and a set of base stations "BS", where BS = $\{bs_1, bs_2, bs_3, bs_4 \ldots, bs_m\}$. The set of tasks that meet their deadline can be denoted as, T_s, which is the subset of T ($T_s \subseteq$ T).

It is assumed that a task t_i is allocated to base station bs_j when the task t_i can meet it's deadline δ_i in that specific base station bs_j. Then the problem can be formulated as :

$$\underset{tbs}{\text{maximize}} \quad \sum_{i=1,j=1}^{n,m} t_i bs_j, where\ t_i \in_i \&bs_j \in BS$$

$$\text{subject to} \quad t_i \leq \delta_i$$

*2) **Assumptions**:* We assume that V2I services are provided to the moving vehicles in the road from base stations. In our system model base stations are stationary edge devices with memory storage, computational capacity and short wireless range transmission system [7]. It is also assumed that a base station is connected to its one hop distant base stations and can transfer tasks if needed. Here, different services provided by a vehicular network are offered through processing vehicular tasks. As such, we can categorize arriving tasks to a base station under different *task types*. Task types represent different type of requests users can submit (e.g., hazard around an area, nearby gas stations and weather forecast). According to task type delay tolerant tasks are bigger in data size (needs more time to execute) than delay intolerant tasks. Due to heterogeneity across Base Stations, one task type has different expected completion times in different Base Stations.

Vehicles in the vicinity of base station within it's transmission range can generate different types of task which arrives to the base station for being processed. Upon arrival of a task to a base station, it is assigned an individual deadline based on its arrival time and the end to end delay it can tolerate. As we are considering delay sensitive tasks, communication delays(uplink and downlink delay) can be significant and are considered for deadline calculation. For arriving task t_i, deadline δ_i can be defined as :
$\delta_i = arr_i + E_i + \varepsilon + \beta$, where arr_i is the arrival time of task, E_i is average task completion time, ε is a constant value defined by the base station(slack time) and β is the communication delay.

We assume in our system model, tasks arrive to the base station randomly and arrival rate of tasks is not known in advance by the system. For receiving base station, it is considered as over subscribed which means a base station receives too many task beyond its capacity and can not handle all tasks before their deadline. Therefore some tasks miss their deadlines and if those tasks are delay sensitive then those are dropped. Dropping task is also an usual practice in oversubscribed real time system []. In real time system there is no value of executing a task that is going to miss its deadline. For instance if a vehicle wants to get updated road condition ahead of it and doesn't get response back before going to that specific road than there is no value of that information when the vehicle is already on that road.

In this research work, we define the task robustness as the probability of meeting it's deadline in a particular base station. For example if a arriving task has greater probability of meeting it's deadline in base station "A" among a cluster of base stations, then base station "A" provides greater robustness for that arriving task. As we are considering to provide robust system from user perspective, it is assumed that a cloud server is connected to a cluster of base stations for fail over or compute intensive tasks that can not be processed in base stations.

*3) **Delay Estimation**:* In a V2I system, three distinct factors contribute to the definition of end-to-end delay (D_{V2I}). They are d_U, d_{BS} and d_D. Therefore, V2I end-to-end delay can be defined as :

$$D_{V2I} = d_U + d_{BS} + d_D \tag{1}$$

Where d_U = average uplink delay, d_{BS} = average delay in base station and d_D = average downlink delay. From equation (1) d_{UP} and d_D can be defined as,

For a task t_i requested from vehicle i to Base Station m, uplink delay from i to m is

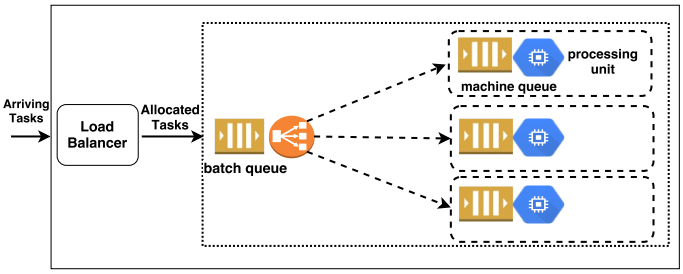
$$d_U = \frac{L_i}{g(i,m)} \tag{2}$$

and for t_i travelling back from m to i downlink delay is

$$d_D = \frac{L_i}{g(m,i)} \tag{3}$$

where L_i is the task data size, g(i,m) and g(m,i) is the effective transmission data rate for the link i to m(uplink bandwidth) and from m to i(downlink bandwidth) respectively.

4) System Model Scenario: Vehicle generated arriving tasks at first go through a load balancer of receiving base station. The load balancer works in immediate mode to allocate arriving tasks to base stations. It can allocate task to receiving base station or its one hop distant neighboring base station. Therefore arriving task immediately gets the decision of allocation from load balancer. When a task is allocated to a Base



Station, it enters into the batch queue of that Base Station for processing. From arrival to the end of task processing a delay is imposed which can be defined as "computational delay"(d_c).

Therefore d_{BS} can be defined as : $d_{BS} = d_c$, where d_c = average computational delay.

IV. Approach

Vehicle-generated task arrives at base station through uplink channel and enters the load balancer of receiving base station. Load balancer is the component for allocating arriving tasks to appropriate Base Stations. It takes the decision of task allocation with consideration of maximizing task robustness. Subsequently the load balancer allocates arriving task to Base Station that offers highest task robustness.

Every Base Station has two matrices. One is Estimated Task Completion(ETC) time matrix and other one is Estimated Task Transfer (ETT) time matrix. These two matrices help the load balancer to work efficiently.

ETC Matrix contains estimated task completion time distribution($X \sim \mathcal{N}(\mu, \sigma^2)$) for different task type in different Base Stations. Within a task type there are different data size tasks which have different completion time in a Base Station.

The value of ETC matrix cell is μ(mean) and σ(standard deviation) which represent normal distribution of different task type in different Base Stations. This distribution is found from the historical execution of different task types. In ETC matrix,every column defines the Base Stations and every row defines task type.

To capture the differences in the estimated completion time of a task type, we consider the worst- case analysis of completion time estimation, that is the sum of mean historic completion time plus it's standard deviation.

The Estimated Task Transfer(ETT) time matrix in every Base Station represent the historic task transfer time from received Base Station to other neighboring Base Stations for different task types. While a batch of tasks is transferred from one base station to another one, the historical task transfer time is stored as ETT matrix entry. Using that data a normal distribution is generated ($Y \sim \mathcal{N}(\mu, \sigma^2)$) with respect to each task type and we save the distribution as average (μ) and standard deviation (σ) in ETT matrix.

As in real time situations tasks will be entering base station with or without intervals, the two matrices need to be updated periodically. Thus the cloud server send periodic pulse which updates the ETC and ETT matrix values according to Base Stations current status.

When load balancer get the task "t_i" of task type "i", it calculates the probability(P_i^j) of meeting the task's deadline δ_i across the Base Stations.

For received Base Station "j", the probability can be defined as $P_i^j(\gamma_i^j < \delta_i) = P_i^j(Z < z)$ where "z" is $(\delta_i - \mu_i^j) / \sigma_i^j$. This is the process for standardize the distribution with $\mu = 0$ and $\sigma = 1$. After calculating the z value, the probability can be found from z-score table.

For neighboring Base Station before calculating the probability we convolve ETC matrix normal distribution with ETT matrix normal distribution. This convolution is important as transfer time impose some delay in completion of that particular task in a transferred base station.The convolved distribution is also a normal distribution which can be defined as:

$$W \sim \mathcal{N}(\mu, \sigma^2) = X \sim \mathcal{N}(\mu, \sigma^2) \circledast Y \sim \mathcal{N}(\mu, \sigma^2)$$

The resulting distribution ($W \sim \mathcal{N}(\mu, \sigma^2)$) is used to calculate the probability of the task in that specific Base Station. If neighboring Base Station is "k" and task type "i" then the probability can be defined as "P_i^k" where $z = (\delta_i - \mu_i^k) / \sigma_i^k$.

After calculating the probability of received task in all local Base Stations (received and neighboring), it is transferred to highest probability base station. When task's probability of meeting deadline is zero(0), then the task is dropped. As dropped task does not get allocated to any base stations. Therefor it does not enter into the batch queue of any Base Station which does not increase the historic mean of executing task. Therefore dropping task in over subscription situation increase the probability of ther tasks meeting their deadline implicitly.

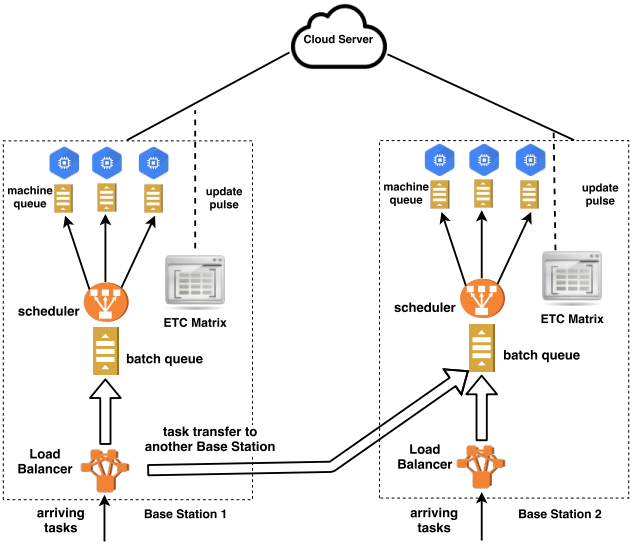


Fig. 2: Proposed method where load balancer efficiently allocate arriving tasks to meet their deadline

V. Heuristics

A. *Load Balancer Heuristics*

Load balancer within a base station is responsible to make an efficient decision of task allocation so that the task can be processed in allocated base station within its individual deadline. The load balancer works in immediate mode. Therefore every task is being served immediately upon its arrival to load balancer.

A buffer queue is assumed to be present in every load balancer in case of situations where many task are requested at the same time. Using this buffer load balancer can store some of the tasks for serving immediately in future. There is no re-allocation considered after allocation of a task by load balancer. This is because of the overhead that occurs from task transfer latency.

1) Maximum Robustness(MR): In this heuristic load balancer allocates an arriving task according to it's robustness across base stations. The robustness of a task is defined as the probability of that specific task meeting it's deadline in a particular base station. For instance if probabilities of a task meeting its deadline among 3 base stations "A", "B" and "C" are 0.8, 0.6, 0.5 respectively, then certainly that task has highest robustness in base station "A".

According to this heuristic, the load balancer at first calculates the probabilities of arriving task to receiving base station as well as neighboring ones. These calculated probabilities are the set of robustness for that arriving task across local base stations. From this set of robustness, the heuristic choose the base station, that provides maximum robustness or highest probability and allocate the task to that base station. In this heuristic the highest robustness base station's probability need to be greater than zero. If there is no such base station than the task is not allocated to any base stations.

As this heuristic only select the highest probability base station, a situation can occurs when two base stations have same highest probability. In this kind of situation the tie can be broken by looking at 2 things. One is considering distribution's sparsity which can be found from standard deviation of that distribution. Another one could be the vehicles direction of movement.

2) No Redirection(NR): This heuristic does not consider transfer of task to any base stations. Therefore whenever an arriving task enters load balancer of a specific base station, it has to allocate only in that receiving base station.

In this case allocation decision also based on task robustness. For the arriving task, the receiving base station needs to provides robustness (probability) greater than a certain value for that task to be allocated in the base station. If the condition is not satisfy by the receiving base station than the task is dropped.

3) Minimum Expected Completion Time(MECT): This heuristic is one of the base line approach in literature. The core concept of this heuristic is to minimizing the expected completion time of each arriving task for making allocation decision. At first this heuristic calculates the average expected completion (μ) time for arriving task in every local base station from ETC matrix. After that the heuristic select the minimum expected completion time base station for task allocation.

B. *Scheduler Heuristics*

1) FCFS: The scheduler use First Come First Serve policy for scheduling the tasks in batch queue. So the task which has arrived first, stays in queue head and late arriving task stays in queue tail. Scheduler schedule task from queue head to VM's local queue whenever free slot appears.

2) SDF: In this heuristic, the scheduler leverage the task's with soonest deadline. It means that tasks in the batch queue is arranged according to their deadline and soonest deadline task gets the highest priority. So task with soonest deadline stays in the queue head and whenever there is a free slot appears in VM's local queue, the task with soonest deadline assigned to the queue to be processed.

VI. Simulation

For simulation of our system model CloudSim [6] is used. Cloudsim is a java library for simulating cloud computing models. In our implementation we used "Data Centers" as Base Stations. Within a Base Station there is one machine with multiple virtual machines(VM's). We used 5 VMs in each Base Station with same computational capacity which represents homogeneous nature within a Base Station. VM's computational power is represented in MIPS(Million Instructions Per Second). For creating heterogeneity across Base Stations different Base Station is defined with different VM MIPS. For simulating the scenario 3 Base Stations are used with different computational power.

In our simulation vehicular tasks are represented as "cloudlets" with different configurations. Different tasks have different length which is represented by MI in CloudSim.

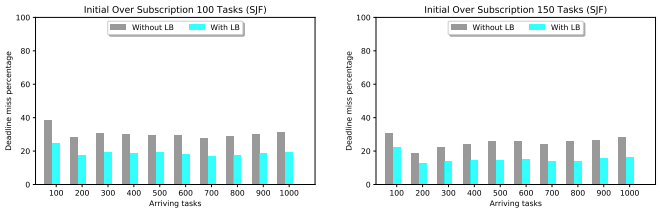


Fig. 3: Initial over subscription with 100 and 150 Tasks (SJF).

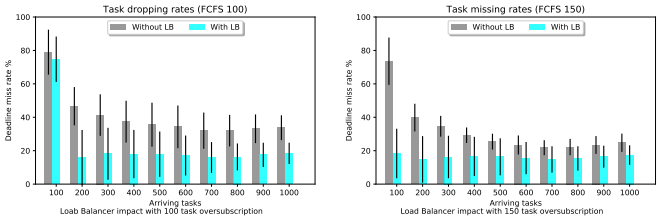


Fig. 4: Initial over subscription with 100 and 150 Tasks (FCFS).

The execution time of a task depends on its length(MI) in CloudSim. For example base station with greater MIPS has less execution time for a particular task. The requested tasks have different parameters (e.g., taskType, taskID, length, deadline, arrival time and fileSize).

VII. Experiments

1) Impact of Load Balancer:

a) Increasing over subscription level: In this experiment the impact of load balancer is tested with two different settings. In first setting we use 50 tasks to create over subscription in each Base Station. And in second setting over subscription is created with 100 tasks.

The simulation is run with starting workload of 100 tasks and incrementing 100 tasks in every trial. So simulation is run for 30 trial with different seeds using Load balancer and without Load Balancer. The average of every 30 trial is used for plotting barchart.

For this experiment the load balancer assigned the tasks to Base Station with highest probability. Here the probability condition of task dropping is zero which means that the highest probability value needs to be greater than zero. If there is no such Base Station then task is dropped or not allocated. This experiment is the implementation of MR heuristic which is mentioned in heuristic section. For this experiment two different scheduling heuristics are used. They are FCFS and SJF.The result plotting is given below :

From the above result it is visible that using load balancer leads to better performance. For smaller number of tasks difference between them is negligible. With the increased number of tasks load balancer works better than conventional scenario. From the two scheduling policy FCFS works better than SJF.

b) Impact of increased over subscription on Heuristics: This experiment is the comparison among 3 heuristics with

increased number of initial over subscriptions. We have initialized over subscription with 250 tasks in every Base Stations and gradually increased by 100 tasks in every trials. Two different heuristics, Maximum Robustness(MR) and Minimum Expected Completion Time(MECT), are used with load balancer in the experiment. Along with mentioned two heuristics, no load balancer option is used and compared with respect to deadline missing rate.

c) Impact of Deadline looseness on Heuristics: In this experiment we investigate the impact of deadline looseness on implemented heuristics. For this purpose initial over subscription is created with 3000 task workload in every base station. In every trial on top of initial over subscription, a batch of 1000 task with different arrival time is provided to the system. As slack is one of the parameter for calculation of deadline, it is used for loosen the deadline tightness. In every trial we run the simulation for 10 times using different seeds with MR load balancer, MECT load balancer and without load balancer. Then we take the average for plotting the result. We have increased the slack by adding integer value 10 in every trial starting from 0.12 to 90.12.The result is given below :

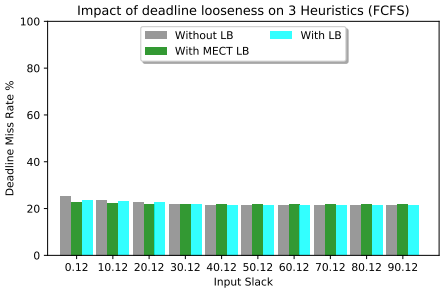


Fig. 5: Impact of deadline looseness on heuristics

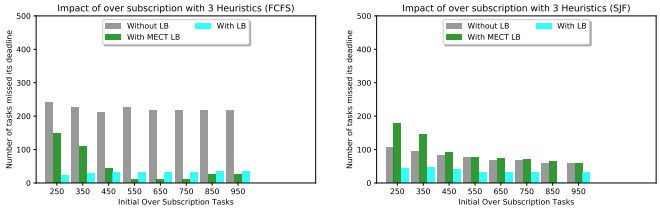


Fig. 6: Performance of heuristics on increasing over subscription for FCFS and SJF

d) Task Dropping with given probability: In earlier experiment we used highest probability Base Station with task dropping for zero probability. But in this experiment we used task dropping condition with given probability. So we used a certain probability value below which we will not allocate tasks to Base Stations.

VIII. Conclusions

In this paper, we have proposed a robust resource allocation technique for delay sensitive applications or tasks in vehicular network under low latency constraints. In the proposed model we have a load balancer module which take efficient decision for selecting best possible Base Station considering task's robustness and improvement of reliability of whole system.

References

[1] GG Md Nawaz Ali and Edward Chan. Co-operative data access in multiple road side units (rsus)-based vehicular ad hoc networks (vanets). In *Telecommunication Networks and Applications Conference (ATNAC), 2011 Australasian*, pages 1–6. IEEE, 2011.

[2] Kyoungsoo Bok, Seungwan Hong, Jongtae Lim, and Jaesoo Yoo. A multiple rsu scheduling for v2i-based data services. In *Big Data and Smart Computing (BigComp), 2016 International Conference on*, pages 163–168. IEEE, 2016.

[3] Jun Li, Carlos Natalino, Dung Pham Van, Lena Wosinska, and Jiajia Chen. Resource management in fog-enhanced radio access network to support real-time vehicular services. In *Fog and Edge Computing (ICFEC), 2017 IEEE 1st International Conference on*, pages 68–74. IEEE, 2017.

[4] Petri Luoto, Mehdi Bennis, Pekka Pirinen, Sumudu Samarakoon, Kari Horneman, and Matti Latva-aho. System level performance evaluation of lte-v2x network. In *European Wireless 2016; 22th European Wireless Conference; Proceedings of*, pages 1–5. VDE, 2016.

[5] Yoshiaki Adachi, Hirozumi Yamaguchi, Teruo Higashino, and Takaaki Umedu. Cloud-assisted dynamic content sharing among vehicles. In *Computer and Information Technology (CIT), 2016 IEEE International Conference on*, pages 516–523. IEEE, 2016.

[6] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose, and Rajkumar Buyya. Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exper.*, 41(1):23–50, jan 2011.

[7] Kai Liu and Victor CS Lee. Rsu-based real-time data access in dynamic vehicular networks. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 1051–1056. IEEE, 2010.