2024 IEEE Intelligent Vehicles Symposium (IV)
June 2-5, 2024. Jeju Island, Korea

FMS: Enhancing Fleet Management Scheme with Long Term Low-Latency V2X Services and Edge-based Video Stream Analytics

Kashish Mahajan[1], Oshin Rawlley[1], Shashank Gupta[1], Shikhar Singh[1]

***Abstract*— V2X (Vehicle-to-everything) communication has garnered much attention in propelling the Internet of Vehicles (IoV) to seek shelter for many mission-critical edge-based applications in Intelligent Transportation Systems (ITS). The goal of achieving end-to-end latency (E2E) for on-road video analytics has become an essential critique to ensure the timely realization of computation-intensive tasks. By adopting the edge services (ES) along with the deployment of better application configuration, the co-optimization of video analytics accuracy and E2E latency can be achieved. However, there are certain challenges to this, such as poor application configuration, variable network conditions, erratic movement of the vehicles, which compromise the E2E latency, and passive strategies of congestion control that fail to avoid the oversubscription of the available bandwidth. To address the key challenges discussed, we propose a Fleet management scheme *(FMS)*, a traffic video stream orchestrator in this work, which introduces Synergetic Service placement and Cost Minimization algorithm (SSPCM) to provision accurate streaming analytics. SSPCM is solved based on Lyapunov optimization and operates online without needing future information, and attains a verifiable performance bound on the Long-term low-latency (LTLL) constraint violation. Extensive evaluations using realistic data reveal the superior performance of the proposed scheme in balancing accuracy with E2E latency.**

I. INTRODUCTION

Wide geographical area service streaming is becoming pervasive with the deployment of Internet of Vehicles (IoV) applications. Smart cities have installed many surveillance cameras across road intersections, offices, and market stores to record the traffic streams [1]. IoV applications such as self-assisted maneuvering, perception tasks, cognitive assistance, infotainment services, etc., require quick analysis on these video streams. Such applications also demand heavy computational resources and high bandwidths to ensure speedy implementation of computation-intensive perception tasks. Concurrent to this requirement, there is a need for adequate V2X services that mitigate the long-term latency incurred by the transmission of data through the uplink channels. Edge computing (EC) alleviates this problem and brings the computational resources closer to the end-users, thereby establishing a cost-effective and scalable data processing. In the current state-of-the-art [2] [3], the time-sensitive applications demand swift video analytics to infer nearly in real-time. For example, the object detection task requires consistent accuracy to be delivered in long-term low latency (LTLL). Consequently, the V2X services [4] have to be coupled to achieve the trade-off between the analytics accuracy and persistent latency. Moreover, LTLL also demands bounded performance gains under the budgetary constraints during the migration of V2X service provisions. From the perspective of optimizing latency, undoubtedly EC in V2X communications has also played an important role [5] . In one of the cited example in the US department of Transportation and the European Telecommunications Standards Institute (ETSI) [6]–[8], a vehicle has to receive a warning before a probable crash has to happen, and it has a latency requirement of 20-25 ms. For this purpose, EC is leveraged. Therefore, to deliver the overall performance of fast streaming video analytics, it is important to schedule the computational resources through configuration adaptation and networking resources through bandwidth management [6]. The main objective is to allocate an optimal share of the bandwidth, decide the frame rate, and resolution to obtain maximum accuracy in minimum LTLL and energy consumption.

However, the above-stated objective struggles with various challenges as discussed follows: One of such challenge is (a) *network bandwidth* that is often unpredictable, and there has been a staggering increase in the traffic demands nowadays which is difficult to cater with the given scarce and fluctuating V2X networks [7]. It so happens when the available network configuration is insufficient, different offloading schemes are adopted that have high frame resolution which results in high transmission latency. Another reason that can be attributed to this behavior is that multiple video streams share the same uplink transmission channel, which causes a delay in the latency. (b) Another challenge is *variable offloading configurations* over the time, which leads to diverse values of accuracies and energy consumption. In order to select the most expensive configuration for the sake of achieving higher accuracy, it results in the consumption of more energy and computational resources. It is required that the best configuration should achieve a trade-off between the energy and accuracy as well. Therefore, it is important to ensure coordinated V2X services for boosting edge-assisted real-time video analytics in less time. The above-mentioned challenges inspire us to propose an adaptive **Fleet management scheme (FMS)** that co-optimizes the response latency and traffic stream analytics accuracy in the domain of autonomous vehicles. The co-optimization is achieved by improving the application configuration under some constraints. To the best of our knowledge, this is the first scheme that co-optimizes the accuracy of traffic video stream analytics and LTLL under

[1]Kashish Mahajan [1]Oshin Rawlley, [1]Shashank Gupta, and [1]Shikhar Singh Department of Computer Science and Information Systems, Birla Institute of Technology And Science, Pilani, India f20200995@pilani.bits-pilani.ac.in, p20200063@pilani.bits-pilani.ac.in shashank.gupta@pilani.bits-pilani.ac.in

energy, latency, and accuracy constraints. We evaluate the proposed *FMS* in an edge-based testbed using real hardware Jetson series and also examine the proposed *FMS* in the simulation settings. We have enlisted the contributions of the manuscript as follows:

- To maintain a trade-off between the application configuration's detection accuracy and LTLL in V2X communication, our proposed *FMS* has introduced SSPCM algorithm that converts the original problem into a set of time single interval optimization problems using Lyapunov optimization. All the sub-problems are solved using Markov approximation and the Karush-Kuhn-Tucker (KKT) condition while, achieving the potential bound on LTLL constraint violation.
- Extensive experiments are carried out to prove the effectiveness of *FMS*. Moreover, we prove the superiority of our proposed *FMS* by comparing it with four benchmark schemes. *FMS* performs better in LTLL than the traditionally opted schemes.

II. System Model

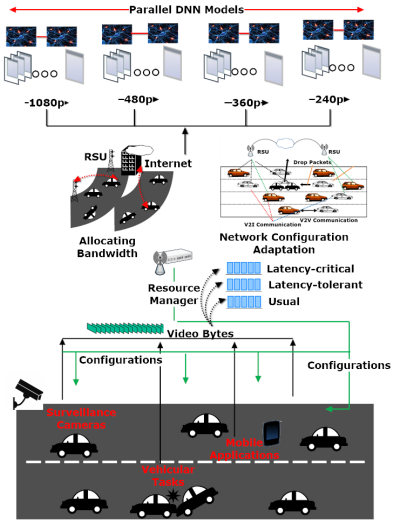

Fig. 1: Priority-based video stream management based on edge computing

Consider a single edge server shared by a group of S traffic streams denoted by $V = v_1, v_2, \ldots, v_S$. The S traffic streams share a common narrow uplink channel to offload the video frames onto the server. N parallel DNN models $P = p_1, p_2, p_3, \ldots, p_N$ which take inputs of different sizes, are deployed on the ES. The proposed system model architecture can be seen in *Figure 1*. res_i refers to the resolution of the input fed into the ith DNN. Let p_0 denote the lightweight DNN model local to each end device. For a model p_i, the time taken for processing the input and the cost on a per-frame basis is represented as inf_i and $cost_i$, respectively. Further, to deal with high-latency V2X services, our proposed *FMS* primarily implements a multi-queue management system that caters the issue of potential delays in the traffic streams. According to the study presented in [8], reducing the size of a DNN using methods like removing computation-heavy layers and reducing the input image resolution comes at the expense of a decrease in the accuracy. Hence, a DNN with low input image resolution takes less time for inference(lower inf_i) and is less resource intensive (lower $cost_i$). We divide the time into equally spaced intervals. The duration of each interval is adjusted according to the time scale at which updates can be performed to the offloading configuration. *Figure 2* describes the workflow block diagram of FMS architecture which shows the edge-driven video analytics and V2X service provisioning. Our major goal is to detect vehicular classes of objects like cars and pedestrians in LTLL, facilitated by V2X services. Videos are continuously recorded by a camera on the vehicle's side. The task of object detection is performed locally using lightweight DNNs. The algorithm runs in the Network Configuration Adaptation component in *Figure 1*. After taking the updated accuracy functions, state of the network, and delay constraint as input, they respond back to the client with the configuration and the bandwidth distribution for the set of given users. Analytics results are used to extract information about video streams, such as the speed and the size of the objects. This information is utilized while updating the accuracy profiles. Analytical models on the accuracy, energy consumption and latency are described subsequently. Sections II-A and II-B describe energy consumption and latency respectively. Subsequently, the problem formulation is presented in Section III.

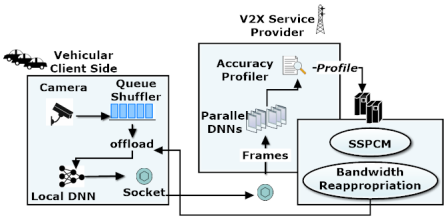

Fig. 2: FMS architecture

Fig. 3: Jetson Nano

Models for Analytics Accuracy: Calculating the accuracy of a configuration is difficult because a query configuration has multiple dimensions, and the various decision variables have different effects on the accuracy of the analytics. Our first task is to understand the interdependence between analytics accuracy and image resolution. To do so, we deploy YOLOv7, an object detection algorithm on NVIDIA Jetson Nano in *Figure 3*, to detect pedestrians in a video clip recorded by a surveillance camera. Video shots are modified to create various resolutions. *F1 score* is used as a measure of accuracy and is used for the accuracy computation of the compressed frame. We use $d(s,i,t)$, which is a binary variable indicating whether the traffic stream v_s has selected model p_i in time interval t. The frame resolution for a traffic stream v_s in the time interval t is given by $\sum_{i=0}^{N} d(s,i,t) res_i$. The accuracy of unsampled frames is calculated by taking the detections from the last sampled frame. Interval x has fast-moving cars, while interval y has slowed down cars because of heavy traffic. We approximate the accuracy as a function of the frame rate using a concave function $A^f_{(s,t)}(fps)$, which is modified at the beginning of

a time interval according to the target speed in the video. $fps_s(t)$ is the frame sampling rate for user v_s in time interval t. The accuracy corresponding to the configuration for traffic stream v_s in time interval t is given by :

$$A^r_{(s,t)} \left(\sum_{i=0}^{N} d(s,i,t) res_i \right) A^f_{s,t}(fps_s(t)) \tag{1}$$

The formula for calculating average accuracy for the group of S traffic streams in time interval t is as follows:

$$A_{avg}(t) = \frac{1}{S} \sum_{s=1}^{S} A^r_{(s,t)} \left(\sum_{i=0}^{N} d(s,i,t) res_i \right) A^f_{(s,t)}(fps_s(t)) \tag{2}$$

A. Model for Energy Consumption

Since recharging mobile devices is not feasible, energy consumption should be taken into consideration while formulating the cost function. There are two ways energy is consumed on mobile devices : (a) Energy involved in transmitting data to the ES, (b) Energy used in the processing of the video frames by the local model. The amount of energy expended during data transmission is proportional to the amount of data transmitted. The data contained within a single frame is given by $kres^2$ bits where k is a constant of proportionality. If e^s_{trans} is the transmission energy used for one bit for traffic stream v_s, the average energy consumption for transmission in time interval t is given by :

$$E_{trans}(t) = \frac{1}{S} \sum_{s=1}^{S} \sum_{i=1}^{N} e^s_{trans} k (d(s,i,t) res_i)^2 fps_s(t) \tag{3}$$

The average energy consumption during the local processing for all traffic streams combined during time interval t can be calculated as :

$$E_{loc}(t) = \frac{1}{S} \sum_{s=1}^{S} d(s,0,t) e^s_{loc} fps_s(t) \tag{4}$$

where e^s_{loc} denotes the energy cost for processing a single frame locally for user v_s. Combining *equations* (2) and (3) gives $E(t)$, i.e., the total average energy consumption for all users combined in time interval t to be $E_{trans}(t) + E_{loc}(t)$.

B. Model for Service Latency

Two factors mainly contribute to the global latency calculation for a single frame *w.r.t* a specific traffic stream in time interval t. These are : data transmission latency and DNN processing latency. Data transmission latency refers to the time involved in transmitting all the frames with different resolutions according to a certain configuration from the client to the ES. This time depends on the size of the input frame as well as the bandwidth share of the link given to the traffic stream. Further, we also consider differential latency requirements for various demanding applications, and also employ a traffic stream management system, that optimally allocates bandwidth to all the traffic streams. The bandwidth reappropriation algorithm is described in the subsequent sections for this purpose. We further define DNN processing latency as the time taken by all the DNN models to perform inference on a single frame. Let $bw_s(t)$ depict the bandwidth share allotted to the traffic stream v_s as per their latency requirements. The global latency per frame experienced by the user u_s in time interval t is given by the formula:

$$Gl_s(t) = \frac{\sum_{i=1}^{N} k (d(s,i,t) res_i)^2}{bw_s(t)} + \sum_{i=0}^{N} d(s,i,t) inf_i \tag{5}$$

, where k ensures that the frame size is in bits. The global average latency for the group of S traffic streams in time interval t is calculated as :

$$Gl_{Avg}(t) = \frac{1}{S} \sum_{s=1}^{S} Gl_s(t) \tag{6}$$

III. Problem Formulation

Performing real-time detections on traffic video streams usually requires high accuracy, energy comnsumption and less delays. The high latency is also attributed to the inappropriate allocation of bandwidth provisioned by V2X services for moving vehicles. Therefore, to cater the declining accuracy while ensuring minimum latency and keeping the energy usage minimal, we have designed our adaptive algorithm for achieving high accuracy under LTLL constraint. We design our utility function as the analytics accuracy minus the energy cost to penalize high energy usage and low accuracy. We also present a bandwidth reappropriation algorithm (*Algorithm 1*) to ensure LTLL constraint is addressed. The algorithm deals with traffic streams with differential priorities (*LC, LT, UA*) so as to efficiently distribute the bandwidth provisioned by V2X services. Now, the goal remains to maximise the utility function averaged over time for all traffic video streams. It can be formulated as follows:

$$O : max \{d, bw, fps\} \lim_{T \to +\infty} \frac{1}{T} \sum_{t=0}^{T} (A_{avg}(t) - \Omega E(t))$$

where constraints are as follows:

$C1 : \sum_{i=0}^{N} d(s,i,t) = 1, u_s \in V, t \in \tau = \{1, \ldots, T\}$

$C2 : d(s,i,t) = \{0,1\}, v_s \in V, t \in \tau$

$C3 : fps_s(t) < \sum_{i=0}^{N} d(s,i,t) \frac{1}{inf_i} v_s \in V, t \in \tau$

$C4 : \sum_{s=1}^{S} \sum_{i=1}^{N} d(s,i,t) cost_i < COST v_s \in V, t \in \tau$

$C5 : \sum_{s=1}^{S} bw_s(t) = BW(t), v_s \in V, t \in \tau$

$C6 : \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T} Gl_{Avg}(t) < GL_{max}$

The parameter Ω decides the relative importance of reducing energy consumption *w.r.t* to higher accuracy. Hence, the optimal solution to the optimization problem O will balance the trade-off between energy consumption and accuracy. The implication of the constraints C_1 and C_2 is that every traffic stream v_s has to select exactly one DNN model in a time interval t. Constraint C_3 ensures that the frame sampling rate is less than equal to the frame processing capacity of the

DNN (remote or local) model used by a traffic stream v_s in time interval t. This prevents traffic video frames from accumulating and increasing queueing delay. C_4 says that the memory resources being used collectively by all the users in a time interval should be within the capacity of the ES denoted by $COST(t)$. Constraint C_5 imposes a constraint that the sum of allocated bandwidths to the group of S users should be equal to the total uplink bandwidth available in the time interval t, denoted by $BW(t)$. Lastly, C_6 ensures that the average LTLL should be lower than the maximum latency threshold.

IV. Online Algorithm

In order to isolate the long-term delay constraint, we convert the time-averaged utility maximization problem to a set of on-the-fly minimization problems using the Lyapunov framework. We propose an online lightweight algorithm that forms the adaption strategy based on real-time bandwidth information and characteristics of the video frames without requiring this information beforehand.

A. Using Lyapunov Optimisation for Problem Transformation

Directly finding an optimal solution for O is challenging. This is because the DNN model allotment, frame sampling rate assignment, and the bandwidth distribution in the group of S traffic streams across all the time intervals are intertwined together because of the long-term delay constraint C_6. To solve this, we introduce a virtual queue i.e., a ***Queue shuffler*** in ***Figure 2***, whose length is a measure of the exceeded latency till any point in time. The initial backlog $(VirQ(0))$ is assumed to be zero. The length of the queue as a function of time is described by the following recursive relation:

$$VirQ(t+1) = max(VirQ(t)+\sum_{s=1}^{S} Gl_s(t)-GL_{max}, 0) \quad (7)$$

If we always go for the configuration that is the most expensive in order to achieve high accuracy, it will lead to a huge buildup in the virtual queue, causing a lot of delays and low user satisfaction. Hence, it is essential to ensure that the latency queue is stable. We can prove the stability of the latency queue, which directly translates to satisfying C_6. We use a quadratic Lyapunov function : $F(VirQ(t)) = \frac{1}{2}(VirQ(t))^2$ as a measure of congestion in the virtual queue, as the value of the function is directly proportional to the queue backlog. We use single-interval Lyapunov drift to maintain the queue stability by constantly moving toward a state where the queue is less congested.

$$\Delta(VirQ(t)) = E[F(VirQ(t+1)) - F(VirQ(t))|VirQ(t)] \quad (8)$$

$\Delta(VirQ(t))$ represents the expected change in the value of the Lyapunov function over an interval of the one-time interval, given a queue state at the starting of time interval t. $\Delta(VirQ(t))$ is inversely proportional to the stability of the virtual queue. We aim to formulate an optimal adaption scheme for users, taking into account the network state and the video content. By including this additional term $\Delta(VirQ(t))$ for stability of latency queue into the trade-off between energy and accuracy, we use another term known as the Lyapunov drift-plus-penalty term:

$$\Delta(VirQ(t)) - \kappa.E[A_{avg}(t) - \Omega E(t)|VirQ(t)] \quad (9)$$

where the parameter κ decides the trade-off between stabilizing the latency queue and maximizing the utility function. We do not minimize the drift-plus-penalty term for every time interval. Rather, we use the min-drift-plus-penalty algorithm [9] in Lyapunov optimization to minimize its upper bound which is stated by the following lemma.
The new problem O_1 can be formulated as follows:

$$O_1 : min\,\{d, bw, fps\}\, VirQ(t).Gl_{Avg}(t) - \kappa.E[A_{avg}(t) - \Omega E(t)$$

under the constraints C_1,C_2,C_3,C_4,C_5 O_1 can be solved on the basis of the current information, whereas the previous optimization problem O required future information to be solved. The additional term introduced in the problem $VirQ(t).Gl_{Avg}(t)$ takes into consideration the average global latency due to the transmission of data and model execution for a given time interval. As a result of this, whenever $VirQ(t)$ becomes large, minimizing the average global latency per frame becomes crucial. Thus, the core principle at work in this algorithm is that – "Whenever bandwidth is not sufficient, the configuration should be degraded to offset the increase in latency and keep it within the allowed limits." The latency queue works without requiring any information from the future and helps satisfy the LTLL constraint. This also guides the process of configuration adaption (frame sampling rate selection and model selection) and bandwidth distribution, making it possible to make decisions in real time. We are now left with the task of solving the problem O_1, which is discussed in the next subsection.

B. Online algorithm based on Lyapunov optimization

The problem O_1 is combinatorial and hence is NP-hard [10] because of which it is not possible to solve it in polynomial time. Hence, we form the ***SSPCM*** algorithm shown in *Algorithm 2* based on the Markov approximation method to solve this problem. We use $d(t)$ to denote the set $\{d(s,i,t)|\forall p_i \in P, \forall v_s \in V\}$, which has the model selection variables, $fps(t)$ to denote the set $fps_s(t)|\forall v_s \in V$ which holds the frame rate selection for all traffic streams, and $bw(t)$ to represent the set $\{bw_k(t)|\forall v_s \in V\}$ Containing the bandwidth distribution for the S traffic streams in time interval t. If we fix $d(t)$, then we have to focus on solving two problems, which are *optimizing bandwidth distribution* to reduce global average latency and *assigning frame rates to different users for maximizing the configuration utility*. Bandwidth distribution has no role to play in the configuration utility function. Hence, the only purpose of bandwidth distribution is to minimize the global average latency $Gl_{Avg}(t)$ in the time interval t. We can formulate it as a new problem : $O_2 : min\,\{bw(t)\}\, VirQ(t).Gl_{Avg}(t)$
The optimal bandwidth distribution is given by the following due to the *Karush-Kuhn-Tucker condition (KKT)* condition

satisfied by the solution of this problem :

$$Opt - bw_s(t) = \frac{\sqrt{\sum_{i=0}^{N} kd(s,i,t)res_i}}{\sum_{s=1}^{S}\sqrt{\sum_{i=0}^{N} kd(s,i,t)res_i}} \quad (10)$$

Given the bandwidth distribution $bw(t)$,we just solved for the incoming traffic streams of different priority tasks and model selection $d(t)$, which was presumed to be fixed, makes the average latency constant. Since the frame sampling rate allocation rate impacts only the configuration accuracy, the second problem can be solved by maximizing the utility function.

$$O_3 : max\left\{fps(t)\right\} A_{avg}(t) - \Omega E(t) \quad (11)$$

We take the accuracy function as a function of the frame rate for user v_s in time interval t as $k_1 - k_2 e^{\frac{-fps_s(t)}{k_3}}$ with k_1,k_2 and k_3 being coefficients of the constants. This is justified since we mentioned earlier that the dependency between frame rate and accuracy can be modeled as a concave function. The objective function of problem O_3 has a global maximum, and the optimal frame sampling rate that achieves it is given as :

$$Opt - fps_s(t) = \left\{ -k_3 log\left(\frac{\Omega e^s_{loc} k_3}{k_2 A^r_{s,t}(\sum_{i=0}^{N} kd(s,i,t)res_i}\right)\right.$$
$$\left.\left((-1-d(s,0,t))k_3 log\frac{\Omega e^s_{trans} k_3 k \sum_{i=1}^{N}(d(s,i,t)res_i)^2}{k_2.A^r_{(s,t)}\sum_{i=0}^{N} kd(s,i,t)res_i)}\right)\right. \quad (12)$$

We can infer that O_2 and O_3 are solvable once the optimal model selection $d(t)$ is found. However, the problem of optimal model selection can not be solved in polynomial time because it is a mix-integer nonlinear problem due to the model selection variables taking binary values. Hence, we will instead solve the problem by finding a good approximation of the optimal solution for model selection using Markov approximation in *Algorithm 1* . Problem O_1 is solved using the single-interval optimization for *SSPCM* as described in *Algorithm 1*. $d(s,t) = \{d(s,1,t), d(s,2,t), \ldots, d(s,N,t)\}$ is the model selection vector for the user v_s in time interval t. *SSPCM* has two optimization objectives – maximizing the utility in *C3* and finding optimal bandwidth distribution to all the priority traffic streams connected using a common uplink channel to an ES for minimizing the latency. Firstly, a user v_s is randomly selected to select a new DNN model p' for it, while for other users, the model selection vector remains the same. Like this, we obtain a new model selection vector for all users named $\dot{d}(ts)$. After this is done, now $fps'(t)$ and $bw'(t)$ can be obtained by solving the problems O_2 and O_3. The new value of the objective function is then calculated as O'_{val} while O_{val} is the value of the object function before the update using parameters $\{d(t), fps(t), bw(t)\}$. In the current time interval, the model selection vector for the traffic stream v_s is updated to p' with a probability ψ which depends on the difference $O'_{val} - O_{val}$, such that the value of ψ is higher when the new configuration $\{d'(t), fps'(t), bw'(t)\}$ results in a lower value of the objective function. This process continues till the maximum number of iterations T_{max} have been reached or $|O'_{val} - O_{val}| < 0.01$ for more than ten iterations. The parameter h is the smoothness parameter used to establish a trade-off between exploration and exploitation or the amount of randomness in this process. If the value of h is small, the new decision is retained with a larger probability. If it reduces the objective value, it promotes exploitation. Consequently, it will take a large number of iterations for the algorithm to find the global optimum because it might get stuck around a local optimum. On the other hand, when h is large, it promotes exploration and the new decision is kept with a lower probability if it is better compared to the current decision. This will reduce the number of iterations it takes for the algorithm to converge. An appropriate value of the parameter h ensures that *Algorithm 1* converges at a super-linear rate.

Algorithm 1 Single interval optimization for *SSPCM*

Data: $A^r_{s,t}(res), A^f_{s,t}(fps), e^s_{loc}, e^s_{trans}, GL_{max}$,$COST$, initial model selection vector $d(t)$

while T_{max} *iterations have been completed or* $|O'_{val} - O_{val}| < 0.01$ *for more than 10 iterations* **do**

end

Pick a user v_s at random and update its model selection vector from $d(s,t)$ to $d'(s,t)$ after selecting a new model p';

if $d^{'}(s,t)$ *is feasible* **then**

$d^{'}(t) \leftarrow \left\{d(1,t), d(2,t), \ldots, d^{'}(s,t), \ldots d(S,t)\right\}$;

Get the value of $bw'(t)$ by solving problem O_2;

Get the value of $fps'(t)$ by solving problem O_3;

Update $\psi \leftarrow \frac{1}{1+e^{\frac{(O'_{val}-O_{val})}{h}}}$

With probability ψ:

$d(t) \leftarrow d'(t)$,

$bw(t) \leftarrow bw'(t)$,

$fps(t) \leftarrow \dot{fps}(t)$;

With probability 1- ψ:

No updates performed

end

return $\{d(t), fps(t), bw(t)\}$

where μ_{min} is the objective value corresponding to the worst possible solution of problem O, μ_{opt} is the optimal utility that can be obtained by removing the delay constraints. Θ is a positive constant representing the LTLL excess obtained by using some stationary control technique. The proof for the same has been given in the Appendix. The utility delay trade-off is characterized by *equation 16 and 17* within $[O\frac{1}{\kappa}, O(\kappa)]]$. When κ gets arbitrarily large, the time-averaged utility will reach arbitrarily near μ_{opt} at the cost of latency. According to *equation 16*, the time-averaged latency is directly proportional to the parameter κ. This type of trade-off allows *SSPCM* to make configuration adaptation a flexible process. The impact of the parameter κ on the performance of the algorithm has been discussed in the simulation section.

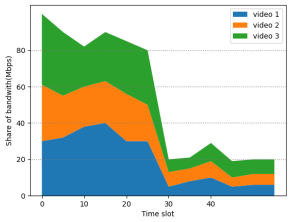

Fig. 4: Bandwidth Runtime

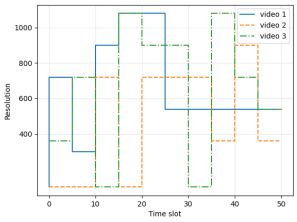

Fig. 5: Resolution Runtime

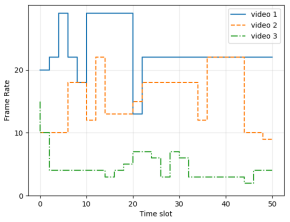

Fig. 6: Frame Rate Runtime

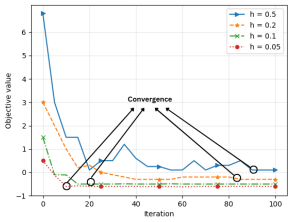

Fig. 7: SSPCM Convergence

V. Simulation

This section evaluates the performance of SSPCM and the impact of different parameters has been discussed. Our simulation uses an ES with five DNN models having resolutions *320p,400p,540p,700p, and 1100p* respectively. For simplicity, we take $e^s_{loc} = 5$ J/frame and $e^s_{trans} = 0.5 \times 10^{(-5)}$ (J) for all traffic streams.

A. A snapshot of the Algorithm in execution

Figure 4- 6 depicts how bandwidth share, resolution, and frame rate vary with the changing bandwidth and changing characteristics of the traffic video content. There are video streams connected to the ES. In the 20^{th} interval, objects in *video 1* slow down while the objects in *video 2* speed up. The expected change in the optimal frame rate for these streams can be seen in *Figure 6*. This adaption is based on the logic that "If the adjacent frames have a negligible difference between them, more frames can be skipped without compromising accuracy much". In the 15^{th} time interval, objects in *video 3* move up closer, resulting in the degradation of resolution for the third traffic stream for saving energy while maintaining accuracy. As can be seen from *Figure 4*, the network bandwidth decreases suddenly in the 30^{th} time interval, which causes all the users to lower their resolutions. Video stream 3 switches to local video processing for some time because its video stream has the largest objects, and hence accuracy is not compromised much.

B. Analyzing the effect of various parameters

Convergence: The convergence of *Algorithm 3* can be seen in *Figure 7*, which shows the objective value as a function of a number of iterations of *Algorithm 3*. At h =0.05 , *Algorithm 1* converges within 10 iterations. In our experiment, $h = 0.1$ is clearly the best value as it needs the lowest number of iterations and also maintains the lowest object value.

VI. Conclusions

We have presented "*FMS*", which is a traffic orchestrator having twin capabilities that are driven by the novel SSPCM algorithm along with the bandwidth reappropriation algorithm. We also formulate the LTLL minimization problem by maximizing the analytics accuracy in a hardware-in-loop setup. The experimental results showed a provable performance bound, which makes this algorithm easier to deploy. We also acknowledge that there are a lot of redundant video frames, which affects the analytics accuracy. To make our system more resilient, we envision devising a key frame selection mechanism that reduces the number of frames and makes our proposed algorithms more content-aware and adaptable in the supporting edge environment.

Acknowledgment

This work is supported by CHANAKYA Fellowships of IITI DRISHTI CPS Foundation under the National Mission on Interdisciplinary Cyber-Physical System (NM-ICPS) of the Department of Science and Technology, Government of India. Grant no: CF-2022-PhD-000002

References

[1] Y. Inagaki and A. Nakao, "Multi-layer edge computing for cooperative driving control optimization in smart cities," in *2023 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2023, pp. 1–8.

[2] H. Ke, S. Mozaffari, S. Alirezaee, and M. Saif, "Cooperative adaptive cruise control using vehicle-to-vehicle communication and deep learning," in *2022 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2022, pp. 435–440.

[3] S. Wan, S. Ding, and C. Chen, "Edge computing enabled video segmentation for real-time traffic monitoring in internet of vehicles," *Pattern Recognition*, vol. 121, p. 108146, 2022.

[4] S. Han, F.-Y. Wang, G. Luo, L. Li, and F. Qu, "Parallel surface: Service-oriented v2x communications for autonomous vehicles," *IEEE Transactions on Intelligent Vehicles*, 2023.

[5] J. Shao, X. Zhang, and J. Zhang, "Task-oriented communication for edge video analytics," *IEEE Transactions on Wireless Communications*, 2023.

[6] M. Hanyao, Y. Jin, Z. Qian, S. Zhang, and S. Lu, "Edge-assisted online on-device object detection for real-time video analytics," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–10.

[7] M. Hu, Z. Luo, A. Pasdar, Y. C. Lee, Y. Zhou, and D. Wu, "Edge-based video analytics: A survey," *arXiv preprint arXiv:2303.14329*, 2023.

[8] X. Lv, Q. Wang, C. Yu, and H. Jin, "A feedback-driven dnn inference acceleration system for edge-assisted video analytics," *IEEE Transactions on Computers*, 2023.

[9] F. Liu, P. Shu, and J. C. Lui, "Appatp: An energy conserving adaptive mobile-cloud transmission protocol," *IEEE transactions on computers*, vol. 64, no. 11, pp. 3051–3063, 2015.

[10] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 207–215.