Low Latency Scalable Point Cloud Communication in VANETs using V2I Communication

Anique Akhtar*, Junchao Ma†§, Rubayet Shafin†, Jianan Bai†, Lianjun Li†, Zhu Li*‡, and Lingjia Liu†
*University of Missouri-KC, MO, USA †Virginia Tech., VA, USA
§Southwest Jiaotong University, China ‡Peng Cheng Lab (PCL), Shenzhen, China
aniqueakhtar@mail.umkc.edu*, {jcma, rshafin, jnbai, lianjunli, ljliu}@vt.edu†, lizhu@umkc.edu*‡

***Abstract*—Mobile edge and vehicle-based depth sending and real-time point cloud communication is an essential subtask enabling autonomous driving. In this paper, we propose a framework for point cloud multicast in VANETs using vehicle to infrastructure (V2I) communication. We employ a scalable Binary Tree embedded Quad Tree (BTQT) point cloud source encoder with bitrate elasticity to match with an adaptive random network coding (ARNC) to multicast different layers to the vehicles. The scalability of our BTQT encoded point cloud provides a trade-off in the received voxel size/quality vs channel condition whereas the ARNC helps maximize the throughput under a hard delay constraint. The solution is tested with the outdoor 3D point cloud dataset from MERL for autonomous driving. The users with good channel conditions receive a near lossless point cloud whereas users with bad channel conditions are still able to receive at least the base layer point cloud.**

***Index Terms*—Point cloud, Autonomous driving, Adaptive Random Network Coding (ARNC), V2I**

I. Introduction

A point cloud is a 3D data representation that is becoming increasingly popular due to the advent of various depth scanning sensors like LiDAR and mmWave radars. Point cloud represents the geometry and shape of any 3D regular structure. In the future, we can expect most autonomous driving cars to be able to "see" the world [1] by receiving real-time point cloud data and be able to make "intelligent" decision by perceiving the unstructured environment. The advances in cellular network and the proposal of 5G new radio technology [2] allows us to multicast large amount of point cloud data through vehicle-to-infrastructure (V2I) and vehicle-to-vehicle (V2V) communication in vehicular ad hoc networks (VANETs).

The performance of the point cloud multicast and the Quality of Experience (QoE) of the vehicles is affected by both the source-coding accuracy as well as network coding under hard latency deadline. To maximize the quality of the received point cloud, it is crucial to devise a clever method to encode point cloud in a scalable manner as well as have an optimal network coding scheme that would maximize each vehicle's QoE. We employ a variable-rate encoder at the transmitter called the Binary Tree embedded Quad Tree (BTQT) encoder to create a scalable layered-encoded point cloud. This scalability generates enhancement layers where the more layers a particular vehicle is able to receive the better the quality of received point cloud. Network coding is originally proposed in information theory and can be seen as a promising technique to improve the network's throughput, efficiency, and scalability [3]. Network coding can make the V2I communication efficient by letting the base station (BS) combine the packets prior to multicasting them to receivers, therefore, simplifying the BS scheduling process [4]. It has been proven that one can approach the multicast capacity by using the random network coding (RNC) technique [5], [6], where each user can decode the original data when it receives a full set of independent linear coded packets. However, our source encoded point cloud has enhancement layers, and if the enhancement layers are too many, some users might not be able to receive a full set of network coded packets due to hard latency constraint and the channel loss causing a considerable decrease in QoE. On the other hand, if the BS transmits too few enhancement layers apart from the base layer, most users would be able to decode the data before the hard deadline but the QoE of users would be compromised. Therefore, we employ Adaptive Random Network Coding (ARNC) [7] to ensure a receiver can decode part of the transmitted packets even if it cannot receive a full set of network coded packets. This provides protection to the base layer and each user is able to decode a different number of enhancement layers based on their channel conditions.

For source encoding, Octree is commonly used to compress the static point cloud or to intra-code the frame in a dynamic point cloud. [8] and [9] compress the point cloud using Octree and employ local surface estimation with predictive coding to estimate the child cell configurations. Dynamic point cloud compression techniques use the difference between two frames. [10] encodes the structural difference between the octree structure of frames. [11] creates a fixed size block in a voxelized point cloud and computes the motion associated with these blocks in the next frame. [12]–[14] transforms the point cloud into a mesh and use mesh compression techniques to reduce the number of edges and vertices using a surface approximation. In the network layer, network coding is widely studied in V2V and V2I communication. In [15], network coding for content dissemination in V2I and V2V communication is studied, but it does not consider scalable transmission. In [7], the scalable video transmission is studied by ARNC, however, the influence of vehicle's mobility is not

The work of Anique Akhtar and Zhu Li are partially supported by NSF I/UCRC award 1747751. The work of Rubayet Shafin, Jianan Bai, Lianjun Li, and Lingjia Liu are partially supported by NSF under Grants NSF/ECCS-1802710, NSF/ECCS-1811497, and NSF/CNS-1811720.

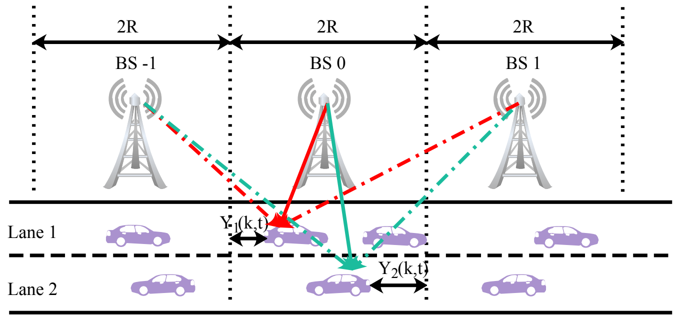


Fig. 1: System Model.

considered.

To the best of our knowledge, this is the first work to provide a scalable point cloud source encoding using ARNC to multi-cast real-time 3D point cloud, with desired tradeoffs with point cloud 3D representation accuracy and radio resources/channel conditions. The main results and contributions of this paper include:

- We provide a framework for point cloud multicast with hard latency requirements for scenarios like autonomous driving. We adopt a Binary Tree embedded Quad Tree (BTQT) based point cloud source encoder [16] to generate a scalable layered representation of point cloud with the base layer and enhancement layers. We can represent the geometry of the same point cloud at different quality level achieving different compression levels. If we receive more enhancement layers, we would obtain smaller voxel size and hence finer point cloud representation.
- Layered data is inherently more sensitive to transmission loss, as decoding has dependency across layers, and transmission loss in top layer (base layer) affects the decoding of deeper layers (enhancement layers). The use of ARNC mitigates such sensitivity by combining the deeper layers with upper layers enabling the decoding of upper layers if the deeper layer is received. This is done by adaptively and carefully designing the combination coefficients of the packets in during ARNC.
- We use the average throughout as the metric to model the QoE of the vehicles. We implement a Rayleigh fading channel to analytically model the probability of successful transmission for every vehicle. We optimize our solution to maximize the average throughput by solving the Markov Decision Process (MDP).
- We compare the performance of the ARNC with other benchmark algorithms in our simulations. Furthermore. we analyze the performance of our system in Rayleigh fading channel as well as spatial channel model (SCM) which is a more realistic channel model in the 3rd Generation Partnership Project (3GPP) [17].

II. Framework

We first encode the point cloud using the BTQT encoder. We convert the BTQT encoded data into layers with each layer

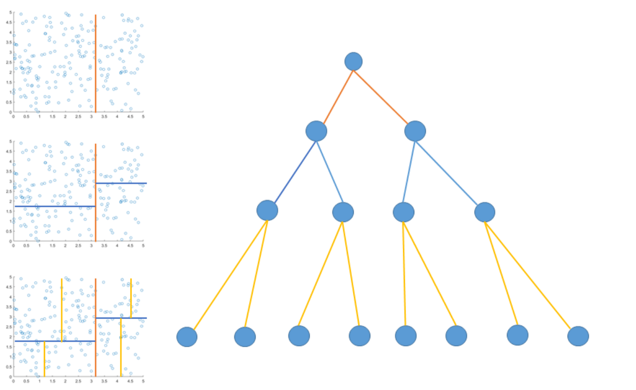


Fig. 2: Binary-tree depiction in 2D plane.

refining the point cloud by increasing its quality. Then we use ARNC to optimize the QoE of vehicles with different channel conditions and maximize the throughput. The system model is shown in Fig. 1.

A. Binary Tree embedded Quad Tree (BTQT)

Point cloud has a huge amount of data and therefore creates a bottleneck in communication. We employ a variable-rate encoder at the transmitter called the BTQT encoder to create layered-encoded point cloud data. The characteristics of binary tree and quad tree, and how they are used together to encode a point cloud are explained in [16]. A voxel is the volume element, defined in 3D space. With our encoding, we try to achieve as small voxel size as possible to get the better quality reconstruction of point cloud. The BTQT creates lowest to highest representation of the point cloud in layers with different quality. The BTQT generates a base layer and a lot of enhancement layers where the decoding of a layer has dependency on the previous layers. The BTQT encoder encodes both the geometry as well as the color of a point cloud. Since in this paper, we are concerned with the geometry transmission of the point cloud, we will ignore the colors in the point cloud and only encode the geometry and transmit it through the channel. We encode the point cloud using BTQT using a two-step approach. The first step of the encoding includes a lossy encoding using a Binary Tree (BT). In the second step, we refine the BT encoding by Quadtree (QT) if the points lie in an approximate 2D plane, or by Octree (OT) if the points cannot be approximated by a 2D plane.

In the first step, we encode the point cloud using the binary tree. Binary tree helps us create block formation in the point cloud frame. Binary tree divides our frame into subframes during each iteration. One example of Binary-tree in 2D plane is shown in Fig. 2. The division occurs at the median of the dimension that has the most variance. Binary tree iteratively divides the frame into 2^L subframes where L is the depth of the binary tree. Each of these subframes are called leafnodes and contain all the datapoints (x, y, z) from the original point cloud that were located in this subframe. Fig. 3 shows one frame of the MERL dataset divided into leafnodes

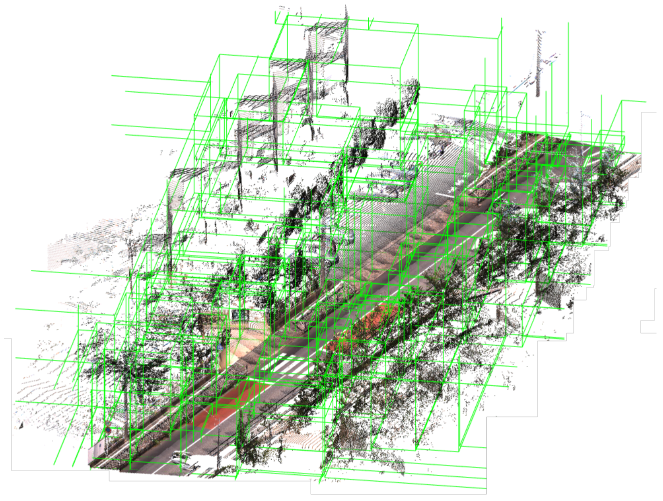
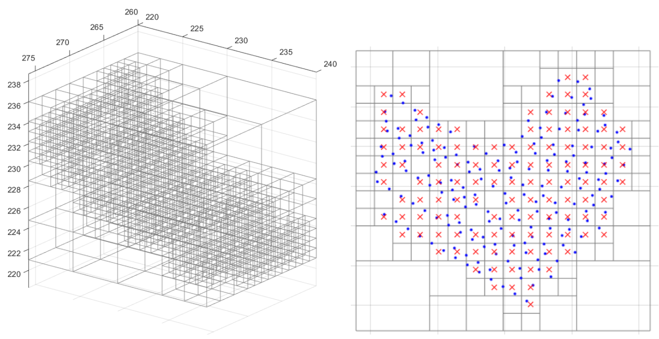


Fig. 3: Block formation using binary tree for MERL dataset.



(a) Octree (b) Quadtree

Fig. 4: Octree and Quadtree visualization.

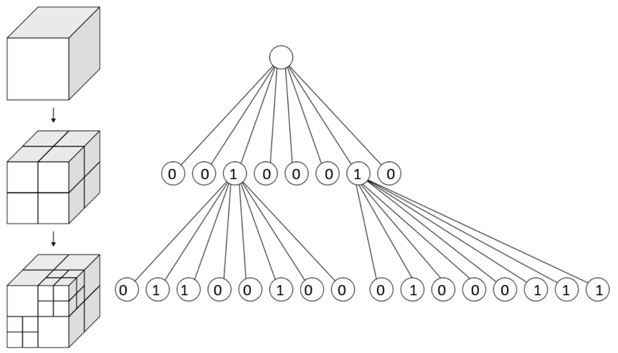


Fig. 5: Occupancy bit assignment in the octree.

using Binary-tree. Binary tree partitions the point cloud into leafnodes where each node contains almost the same amount of data points.

In the second step, we further encode each of the leafnode with either Quadtree or Octree based on the projection of the data points. If the data points within a BT leafnode exhibits flat characteristics they are projected onto a 2D plane and encoded with Quadtree. Otherwise, the BT leafnode is encoded in 3D

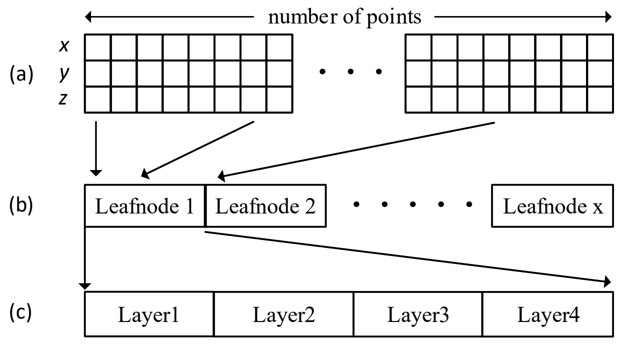


Fig. 6: Bitstream of the encoded BTQT point cloud.

using Octree. Fig. 4 shows how an Octree and a Quadtree encoded leafnode looks like. Octree recursively divides a 3D block into eight sub-blocks by splitting the block into two halves in all three dimensions. These sub-blocks are called child-cells or child-nodes. The sub-cells that are non-empty are further divided into eight sub-cells until some stopping criteria are met or maximum depth is reached. Encoding is done by storing the occupancy bit at each level of division. Each division splits the cell into eight sub-cells. Any sub-cell that is empty is assigned 0, whereas non-empty sub-cells are assigned 1, as observed in Fig. 5. Each layer of the Octree is assigned occupancy bits and then scanned in a breadth-first-search or top-to-bottom and left-to-right manner. The bitstream we get is the Octree encoded leafnode. A Quadtree is the 2D version of Octree and follows a similar procedure. In Fig. 4 we can compare both Quadtree and Octree. In Fig. 4b we can also observe that the point cloud points (blue dots) are approximated by the centroid of the box (red cross).

The coding process and the final bitstream is shown in Fig. 6. Fig. 6(a) shows the unordered and uncoded point cloud data. After the BTQT encoding, we convert it into different leafnodes through BT encoding and each leafnode has multiple layers due to OT and QT encoding. Each layer here is a different quality representation of point cloud where deeper layers are enhancement layers and are dependent on the decoding of the upper/previous layers.

B. System Model

As shown in Fig. 1, we consider a two lanes, straight road model [18], [19]. Vehicles in the first lane move from left to right, whereas the vehicles in the second lane move in the opposite direction. The locations of the BSs are equally spaced with an interval of $2R$ meters. The transmitter wishes to multicast L encoded point cloud layers to all users within a hard deadline of $T \geq L$ time slots, where it is assumed that each layer can be encapsulated into one packet. For our formulation, we focus on the transmission from a single BS and label it as BS 0. The neighboring BSs can cause interference and are respectively denoted by BS $(-i)$ and BS (i) for the left and right sided neighbors where $i = \cdots, -2, -1, 1, 2, \cdots$.

K_1 and K_2 define the number of vehicles in lane 1 and lane 2 respectively which are under the coverage of BS 0. The initial location of every vehicle k is randomly chosen within the coverage of BS 0. As shown in Fig. 1, the boundary for the coverage area of BS 0 is represented by the vertical dotted lines in left and right sides of BS 0. For a vehicle k_1 in lane 1 ($k_1 = 1, 2, \cdots, K_1$), its initial distance from the left-side boundary of BS 0 is denoted by $Y_1(k, 0)$, where the value of $Y_1(k, 0)$ is randomly selected between 0 and $2R$. Similarly, the initial distance of a vehicle k_2 in lane 2 ($k_2 = 1, 2, \cdots, K_2$) from the right-side boundary of BS 0, $Y_2(k, 0)$, is randomly chosen from $[0, 2R]$. Thus, for a vehicle k_1 and k_2, their locations at the beginning of a time slot t can be expressed by $Y_1(k_1, t) = Y_1(k_1, t-1) + v(k_1)\Delta t$ and $Y_2(k_2, t) = Y_2(k_2, t-1) + v(k_2)\Delta t$ respectively, where Δt is the duration of one time-slot. The velocity of an arbitrary vehicle k ($k = 1, 2, \cdots, K$ where $K = K_1 + K_2$) is denoted by $v(k)$, and is randomly chosen from uniform distribution $[V_{\min}, V_{\max}]$, where $V_{\min}$ and $V_{\max}$ are the minimal and maximal velocity of the vehicles respectively. In this work, we assume the serving BS is fixed to BS 0 during one transmission period. Let us denote the height of each BS by h; then the distance d_1 and d_2 between BS i and vehicle k in lane 1 and lane 2 respectively can be written as:

$$d_1(k,i,t) = \begin{cases} \sqrt{\left[(2|i|-1)\,R + Y_1\,(k,t)\right]^2 + h^2}, & i < 0 \\ \sqrt{\left[R - Y_1\,(k,t)\right]^2 + h^2}, & i = 0 \\ \sqrt{\left[(2i+1)\,R - Y_1\,(k,t)\right]^2 + h^2}, & i > 0 \end{cases}$$

and

$$d_2(k,i,t) = \begin{cases} \sqrt{\left[(2|i|+1)\,R - Y_2\,(k,t)\right]^2 + h^2}, & i < 0 \\ \sqrt{\left[R - Y_2\,(k,t)\right]^2 + h^2}, & i = 0 \\ \sqrt{\left[(2i-1)\,R + Y_2\,(k,t)\right]^2 + h^2}, & i > 0 \end{cases}$$

The signal from each BS i to the vehicle k at time t will suffer path-loss, small-scale fading, and noise. The received signal-to-interference-plus-noise ratio (SINR) can be expressed as:

$$\mathrm{SINR}\,(k,t) = \frac{P\;d\,(k,0,t)^{-\alpha}|h_0|^2}{\sum_{i:i\neq 0} P\;d\,(k,i,t)^{-\alpha}|h_i|^2 + N_0}, \quad (1)$$

where P is the transmission power, $\alpha > 2$ is the path-loss exponent, and N_0 is the noise power. $d(k,i,t)$ is valid for $d_1(k,i,t)$ and $d_2(k,i,t)$, depending on which lane the vehicle k is located in. Here, the Rayleigh block fading channel model is assumed, and thus $|h_i|^2 \sim \exp(1)$. The transmitted message from the serving BS at time t will be recovered successfully if the receive-SINR exceeds the decoding threshold, θ. The probability of this event can be expressed as

$$\begin{aligned} \mathrm{P}_c(k,t) &= P_r\left(\mathrm{SINR}\,(k,t) \geq \theta\right) \\ &= e^{\left(-\theta P^{-1} d(k,0,t)^{\alpha} N_0\right)} \prod_{i:i\neq 0} \frac{1}{1+d\,(k,0,t)^{\alpha}d\,(k,i,t)^{-\alpha}\theta}, \end{aligned} \quad (2)$$

which is highly impacted by the mobility pattern. The detailed derivation of this equation can be found in [20].

C. Adaptive Random Network Coding (ARNC)

During source coding, we encode the point cloud using scalable BTQT to generate L layers. The first layer is the base layer and each next layer is an enhancement layer which can only be decoded if the initial layers are all received. Every vehicle would recover the data with a different quality based on how many successive l layers ($l = 1, 2, \cdots, L$) it receives within the hard-time delay constraint of $T \geq L$ time slots. A larger l implies a higher quality video at the receiver side. We represent the data in layer l by α_l and assume it can be transmitted in a single attempt. Due to the hard deadline constraint, channel loss, and mobility choosing which layers to transmit is challenging for the BS. If the BS transmits too few enhancement layers apart from the base layer, although most vehicles would be very likely to recover all the layers, the QoE of the vehicles would be compromised. On the other hand, if the enhancement layers are too many, vehicles with good channel conditions would see a high QoE, however, the vehicle with low channel quality might not even receive the base layer (α_1).

To optimize the QoE of the vehicles and maximize the multicast throughput, we use adaptive random network coding (ARNC). In ARNC, the transmitted message is a combination of multiple layered data weighted by a coefficient for each layer. Generation of the message in ARNC is a two-step process: determine the *generation* $a_t \in \{1, 2, \cdots, L\}$; and generate the message which is a combination of the first a_t layers data. The transmitted message in time slot t can be represented by

$$c(t) = \sum_{j=1}^{a_t} \beta_{t,j}\alpha_j,$$

where $\beta_{t,j}$ is the encoding coefficient of α_j at time slot t, and it is randomly chosen from a large finite field $\mathbb{F}_q$ [7].

In the non-network coding, the BS transmits one layer whereas in ARNC the BS transmits a combination of different layers at a time. The receiver only cares about the number of messages it has already received and the rank of the receive coefficients [7]. In the traditional RNC, the *generation* in each transmission is always L whereas ARNC is more flexible with *generation* $a_t \in \{1, 2, \cdots, L\}$ and provides more protection for the base layer. For example, when $L = 3$, if a vehicle receives two messages, it would not be able to decode anything in traditional RNC but can still receive α_1 and α_2 in ARNC if the *generation* $a_t = 2$. An optimal BS scheduling policy to determine the *generation* dynamically in each transmission is crucial to maximizing the ARNC performance. The *generation* determination should consider the channel condition of the vehicles which is influenced by its mobility. For a vehicle with a good channel, it is desirable to transmit a message containing more layers while vehicles with bad channel condition would prefer a message only containing few layers as to maximize the chance of receiving the base quality of the point cloud data. Furthermore, we assume that each user may inform the transmitter on whether a packet has been successfully received or not via uplink feedback.

We define average throughput as the average number of successive layers a vehicle receives within T time slots. The average throughput of the network can be maximized by optimizing the BS scheduling policy at each transmission. This optimization problem can be perfectly cast into Markov Decision Process (MDP) framework [7]. In MDP, the BS makes a decision a_t at time t based on the current network state $\mathbf{S}_t$ which records the messages that have already been received by the vehicles. In specific,

$$\mathbf{S}_t = \begin{bmatrix} s_{1,1} & \cdots & s_{1,L} \\ \vdots & \ddots & \vdots \\ s_{K,1} & \cdots & s_{K,L} \end{bmatrix}, \tag{3}$$

where $s_{k,l}$ ($k = 1, 2, \cdots, K$ and $l = 1, 2, \cdots, L$) is the number of messages received by user k containing the first successive l layers. An arbitrary user k with state $\mathbf{s}_k = [s_{k,1}, \cdots, s_{k,L}]$, can decode the first l data if

$$\begin{cases} \sum_{i=1}^{l-1} \min(s_{k,i}, 1) + s_{k,l} = l, \\ \sum_{i=1}^{l} \min(s_{k,i}, 1) + s_{k,l+1} < (l+1). \end{cases}$$

Then the average throughput of the network at time slot t, denoted by τ_t can be expressed as:

$$\tau_t = \frac{1}{K} \sum_{k=1}^{K} l_k \qquad \text{layer/vehicle},$$

where l_k is the number of useful packets (from layer 1 to layer l_k) user k has received.

Suppose $a_t = m$ is determined as the action, then based on the channel condition of vehicle k, the element $s_{k,m}$ changes to $s_{k,m} + 1$ with probability $P_r(\text{SINR}\,(k,t) \geq \theta)$ as shown in (2) or $s_{k,m}$ otherwise. Then the current network state $\mathbf{S}_t$ is updated to $\mathbf{S}_{t+1}$. With the updated $\mathbf{S}_{t+1}$, we can also calculate the throughput performance τ_{t+1}. Then the benefit of the action a_t can be represented as $r(\mathbf{S}_t, a_t) = \tau_{t+1} - \tau_t$.

The action set in the T transmissions can be denoted as $\Omega = \{a_1, a_2, \cdots, a_T\}$, and the throughput of the network can be maximized by solving the optimization problem

$$\max_{\Omega} \sum_{t=0}^{T-1} r(\mathbf{S}_t, a_t). \tag{4}$$

The optimization problem can be solved by greedy method and is shown in [7], therefore, omitted in our current work.

III. Simulation results

In this section, we simulate the throughout performance of multicasting BTQT encoded point cloud data with hard deadline constraint. The values of the parameters used in the simulations are listed in Table I. For source encoding, we kept the Binary tree depth to 10 whereas the Octree/Quadtree depth determined the number of layers $L = 7$. As an attribute of BTQT encoding, it is to be noted that as a vehicle receives more layers, the voxel size of the decoded point cloud decreases exponentially giving us a finer and of a higher quality point cloud. Our simulation study the throughput performance of the network with a varying number of vehicles on the road, and different hard latency constraints. In Fig. 7 we visualize the received point cloud for three users each able to receive and decode a different number of layers.

TABLE I: Parameter setting in the simulation.

Parameter	**Value**	**Parameter**	**Value**
BS coverage R	100 m	delay constraint T	10 slots
threshold, θ	1 dB	maximal speed V_{max}	120 km/h
BS power	40dbm	minimal speed V_{min}	60 km/h
vehicles in lane 2 K_2	2	noise power N_0	-110 dbm
vehicles in lane 1 K_1	2	pathloss exponent α	3.85
number of layers L	7	Binary tree depth	10

In addition, as stated in [7], despite the proposed ARNC with full feedback (referred to hereinafter as AFF) the performance of some benchmark schemes like ARNC with limited feedback (referred to hereinafter as ALF), ARNC without feedback (referred to hereinafter as ANF), and RNC are also given. In ALF scheme, the BS determines the generation as l in the lth transmission without considering the feedback of the vehicles if $l \leq L$; while in ANF scheme, the generation in the lth transmission is l if $l \leq L$ or L otherwise. In the analytical part, we assumed, for simplicity, a Rayleigh fading channel without doppler effect. However, in the simulation process, we use both Rayleigh fading channel as well as 3GPP's spatial channel model (SCM) and compare the results. SCM channel is widely used for wireless system simulation in industrial labs. We use 'suburban macro' scenario for SCM channel generation as prescribed by 3GPP. The number of resolvable paths (channel taps) is assumed to be 6, while the path delays are drawn from the delay distribution specified in [17]. We assume 20 sub-paths for each path. Azimuth angles for both BS and mobile devices are drawn from $U(-180, 180)$.

A. *Varrying the number of vehicles K*

Fig. 8 shows how the number of vehicles, K, influences the throughput performance. The simulation is performed by fixing the number of vehicles in lane 2, $K_2 = 1$, and changing the number of vehicles in lane 1, K_1, from 0 to 9. We can observe from the figure that the average throughput decreases with K. When the traffic is sparse, it is easier for the BS to schedule so as to maximize the throughput with a limited compromise in QoE. However, in a dense network, the channel of the vehicles are diverse in each time slot and thus the BS has to consider more vehicles while scheduling as to maximize the throughput of all the vehicles. We can observe that the AFF performs the best, followed by ALF, ANF, and traditional RNC. The overhead involved in the feedback is limited compared to the throughput gain of AFF and ALF. When the traffic is sparse, the feedback is even smaller whereas the throughput improvement is significant. In dense traffic, the gap between AFF and ALF is negligible. Thus, in

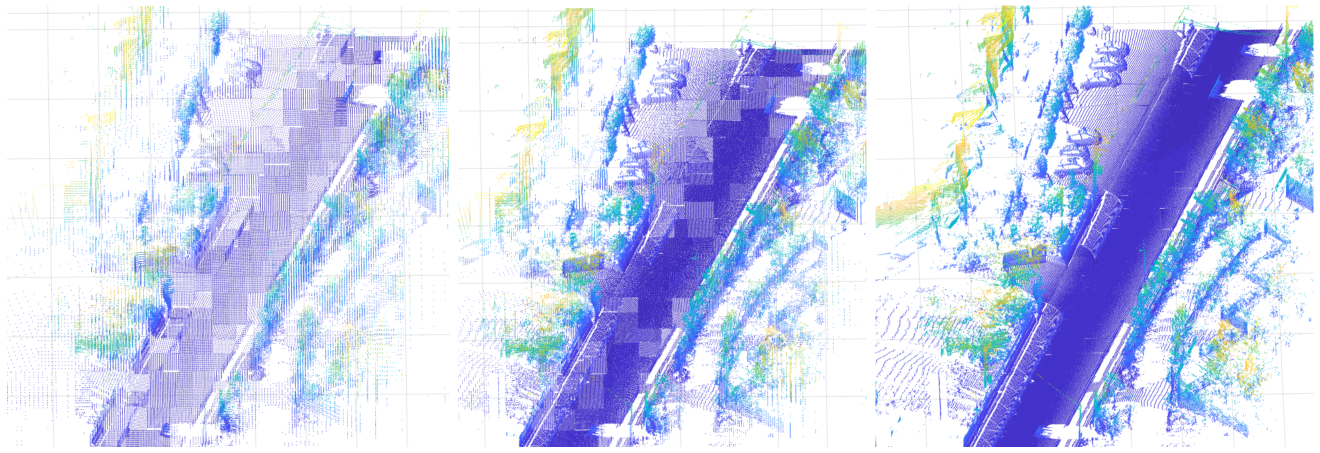

(a) 1 Layer decoded (b) 2 Layers decoded (c) 4 Layers decoded

Fig. 7: Received quality for three users.

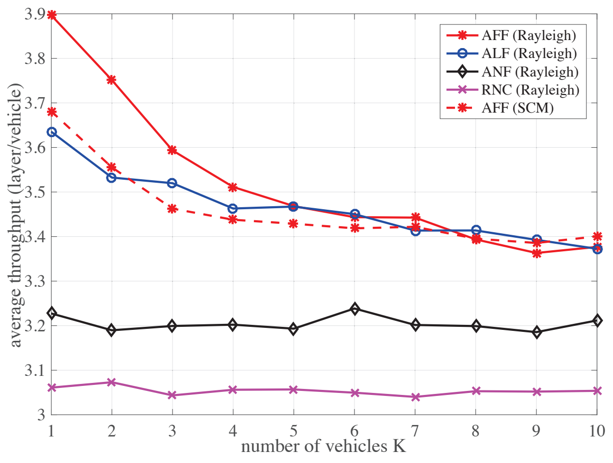


Fig. 8: Vehicle density vs average throughput performance.

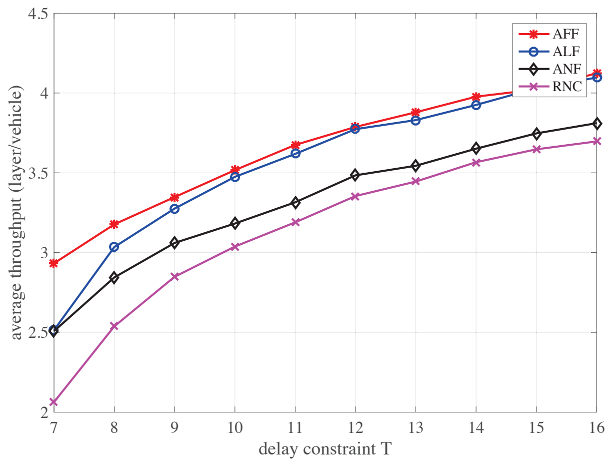


Fig. 9: Delay constraint T vs average throughput performance.

that case, the ALF is better than AFF since it decreases the feedback. The performance of ANF and RNC is very stable even with the increase in K. This is because the BS has a fixed scheduling policy in each transmission without considering the conditions of vehicles.

In Fig. 8 we also compare the performance of AFF in the Rayleigh channel and the SCM channel. The gap between the two schemes is large when K is small, however, the gap decreases with an increase in K. The performance of AFF highly depends on the BS scheduling decision in each transmission, and the scheduling decision further depends on the channel condition and the feedback of the vehicles. From this simulation, we can deduce that the channel condition contributes more for the BS scheduling decision when the traffic is sparse, whereas when the traffic is dense both AFF and ALF perform similarly and so we should choose the scheme that minimizes the feedback overhead.

B. *Influence of hard-deadline T*

For this experiment, We change the hard latency constraint, T, from 7 to 15 and measure the effect it has on the average throughput performance. The results are shown in Fig. 9. As could be predicted, the increase in the latency constraint improves the throughput of the whole network. This is because, with a higher T, the vehicles have a higher probability of receiving more messages, therefore, they will be able to decode more layers. Similar to Fig. 8, we can observe that ARNC with feedback outperforms other schemes, and the gap between different schemes keep stable with T. When T is close to L, the ALF performs like ANF because it does not have enough feedback. However, with an increase in T ALF performs close to AFF.

IV. Conclusion and Future Work

In this paper, we present a framework for scalable real-time 3D point cloud multicast via vehicle-to-infrastructure (V2I) communication with hard latency constraint. We offer cross-layer optimization by adopting Binary Tree embedded Quad Tree (BTQT) source encoder and adaptive random network coding (ARNC) in the multicast system which offers scalability with improved QoE for vehicles. Each vehicle may receive a different quality/voxel size of the point cloud depending on the mobility and channel conditions of that particular vehicle. We solve a Markov decision process (MDP) to maximize the throughput performance of our layered and ARNC coded point cloud. Simulation results show the ARNC with feedback outperforms other benchmark scenarios. As our future work, we would analyze the influence of handover on our network and how the incorporation of vehicle-to-vehicle (V2V) communication enhances the network performance. We would further study and optimize the source and network coding dependencies as well as choosing the right packet size for network coding. In addition to the point cloud geometry, the attributes from the point cloud, such as LiDAR reflectance, will also need to have a scalable source encoding compression scheme.

References

[1] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *Robotics and automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1–4.

[2] S. E. Elayoubi, M. Fallgren, P. Spapis, G. Zimmermann, D. Martín-Sacristán, C. Yang, S. Jeux, P. Agyapong, L. Campoy, Y. Qi *et al.*, "5G service requirements and operational use cases: Analysis and METIS II vision," in *Networks and Communications (EuCNC), 2016 European Conference on*. IEEE, 2016, pp. 158–162.

[3] Z. Chen, T. J. Lim, and M. Motani, "Digital network coding aided two-way relaying: Energy minimization and queue analysis," *IEEE Trans. Wireless Commun.*, vol. 12, no. 4, pp. 1947–1957, 2013.

[4] E. Bourtsoulatze, N. Thomos, J. Saltarin, and T. Braun, "Content-Aware Delivery of Scalable Video in Network Coding Enabled Named Data Networks," *IEEE Trans. Multimedia*, vol. 20, no. 6, pp. 1561–1575, 2018.

[5] C. Khirallah, D. Vukobratovic, and J. Thompson, "Performance analysis and energy efficiency of random network coding in lte-advanced," *IEEE Trans. Wireless Commun.*, vol. 11, no. 12, pp. 4275–4285, 2012.

[6] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Trans. Inf. Theory*, vol. 52, no. 10, pp. 4413–4430, 2006.

[7] B. Li, H. Li, and R. Zhang, "Adaptive Random Network Coding for Multicasting Hard-Deadline-Constrained Prioritized Data," *IEEE Trans. Veh. Technol.*, vol. 65, no. 10, pp. 8739–8744, Oct 2016.

[8] R. Mekuria, K. Blom, and P. Cesar, "Design, implementation, and evaluation of a point cloud codec for tele-immersive video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 4, pp. 828–842, 2017.

[9] R. Schnabel and R. Klein, "Octree-based point-cloud compression," *Proc. IEEE/Eurographics Symp. Point-Based Graphics (PBG 06),*, vol. 6, pp. 111–120, 2006.

[10] J. Kammerl, N. Blodow, R. B. Rusu, S. Gedikli, M. Beetz, and E. Steinbach, "Real-time compression of point cloud streams," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 778–785.

[11] R. L. de Queiroz and P. A. Chou, "Motion-compensated compression of dynamic voxelized point clouds," *IEEE Trans. Image Process.*, vol. 26, no. 8, pp. 3886–3895, 2017.

[12] J. Peng, C.-S. Kim, and C.-C. J. Kuo, "Technologies for 3d mesh compression: A survey," *J. VIS. COMMUN. IMAGE R.*, vol. 16, no. 6, pp. 688–733, 2005.

[13] P. Alliez and C. Gotsman, "Recent advances in compression of 3d meshes," in *Advances in multiresolution for geometric modelling*. Springer, 2005, pp. 3–26.

[14] S.-R. Han, T. Yamasaki, and K. Aizawa, "Time-varying mesh compression using an extended block matching algorithm," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 11, pp. 1506–1518, 2007.

[15] K. Liu, J. K.-Y. Ng, J. Wang, V. C. Lee, W. Wu, and S. H. Son, "Network-coding-assisted data dissemination via cooperative vehicle-to-vehicle/-infrastructure communications," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 6, pp. 1509–1520, 2016.

[16] B. Kathariya, L. Li, Z. Li, J. R. Alvarez, and J. Chen, "Scalable point cloud geometry coding with binary tree embedded quadtree," in *Multimedia and Expo (ICME), 2018 IEEE International Conference on*. IEEE, 2018.

[17] G. TR25, "3GPP SCM channel models, 3GPP TR25. 996," *Vol. v6*, vol. 1, 2003.

[18] Z. Zhou, C. Gao, C. Xu, Y. Zhang, S. Mumtaz, and J. Rodriguez, "Social big-data-based content dissemination in internet of vehicles," *IEEE Trans. Ind. Informat.*, vol. 14, no. 2, pp. 768–777, Feb 2018.

[19] Q. Wang, P. Fan, and K. B. Letaief, "On the joint v2i and v2v scheduling for cooperative vanets with network coding," *IEEE Trans. Veh. Technol.*, vol. 61, no. 1, pp. 62–73, Jan 2012.

[20] J. G. Andrews, A. K. Gupta, and H. S. Dhillon, "A primer on cellular network analysis using stochastic geometry," *arXiv preprint arXiv:1604.03183*, 2016.