A Distributed System for Reducing Uploaded Data Redundancy in Vehicular Networks

Zezhi Wang*, Yajie Zhao*, Lewis Tseng
Boston College
{wanganh, zhaoyz, lewis.tseng}@bc.edu

Takamasa Higuchi, Onur Altintas
Toyota InfoTechnology Center, U.S.A.
{ta-higuchi, onur}@us.toyota-itc.com

***Abstract*—Uploading vehicle sensor data to support autonomous driving is necessary to understand the current situation to make the best decision. In this paper, we develop a system that relies on a peer-to-peer mechanism to obtain information in a vehicular network. Consider a scenario in which each vehicle equipped with a camera and communication capability is responsible to upload new snapshots to a datacenter, and the datacenter combines the snapshots to create a map. In a naïve sensor data upload scheme, each vehicle uploads its snapshot periodically and the datacenter will find out new information and integrate it to a map. This method might work if only a small number of vehicles are uploading at the same time. However, this naïve method is *not* scalable when dozens of vehicles need to upload, as the communication bandwidth will be a bottleneck. To address the challenge, we propose a novel distributed system to *reduce data redundancy*. As a result, the bandwidth consumption between vehicles and the datacenter is reduced as well. The key idea is to use location information (e.g., GPS coordinates) to simplify the design and coordination among peers (vehicles), and rely on computer vision algorithms to remove redundant data and identify important information to be uploaded. In this paper, we outline the design of our system and verify the efficacy of our system through a simulation study.**

Keywords – Distributed System, Upload Bandwidth, Peer-to-Peer, Data Redundancy, Bandwidth Consumption

I. Introduction

Uploading vehicle sensor data to support autonomous driving is necessary to understand the current situation to make the best decision. For example, a high precision map can be updated in a timely fashion by uploading sensor information, e.g., snapshot images, to the datacenter. In this paper, we focus on the problem of uploading images efficiently to the datacenter in a vehicular network. Concretely, we aim to solve the following problem: *How do we efficiently create a panorama image at a fixed location by collecting snapshot images from nearby vehicles?* The panorama application is useful in understanding current situation and constructing a highly accurate real-time map. In Section V, we also discuss how to extend our design to more general applications.

A naïve solution to the problem is for each vehicle to periodically upload all the images it captures to the datacenter, which then selects useful images to update the map. There are two major problems using this approach: (i) the shared wireless medium might be congested if the network bandwidth is constrained, and (iii) the computation workload at the datacenter might be too high.

Our system is motivated by the observation that using Vehicular Ad hoc NETwork (VANET), we can improve the performance by reducing bandwidth consumption and computation burden at the datacenter. The key improvement is twofold: (i) reducing the *data redundancy* by exchanging messages among vehicles, and (ii) distributing the upload bandwidth among vehicles.

To the best of our knowledge, two works have looked at similar problems but in a completely different context. Weinsberg et al. [1] designed a method to prioritize images to upload new and important information in challenged networks, i.e., the ones in disaster affected regions. Dao et al. [2] used an advanced computer vision algorithm to identify key information to be uploaded to a datacenter. The other related works are discussed in Section IV.

We share the same intuition – since there is abundant data redundancy, we need to suppress/defer transfers of redundant content to reduce the communication bandwidth. One novelty of our scheme is exploiting vehicles' GPS information to achieve efficient coordination. Then vehicles are able to jointly reduce redundant data and only upload important part of the images. Through a simulation study, we show that compared to the naïve solution, the bandwidth consumption is reduced by a scale factor between *1/2 and 5/6* using our system.

In Wireless Sensor Networks (WSN), in-network aggregation is extensively studied to reduce the energy and communication cost, e.g., [3], [4], or ensure security [5]. Effectively, our system can be viewed as the integration of tree-based and cluster-based approaches in WSN [3], [4], [5]. The main novelty of our design is to combine such protocols with image processing and use location information to improve efficiency.

II. Our System

In this section, we describe our system and two examples to illustrate the key algorithm used in the system.

A. Targeted Scenario

Recall that we want to create the panorama view from the snapshot images taken by vehicles near a fixed location. We envision that vehicles in that location will form a certain type of "vehicular clouds," e.g., [6], [7], [8], [9]. As a result, each vehicle can obtain a membership list of nearby vehicles, i.e.,

*Authors have same contribution.

978-1-5386-9151-9/19/$31.00 ©2019 IEEE

each vehicle knows the set of vehicles that will upload the images to the datacenter. Moreover, each vehicle maintains an accurate GPS information and camera orientation. The membership list allows each vehicle to know whom to coordinate with. The GPS information and camera orientation allow us to efficiently "stitch" images together to form a panorama view. Due to space limitation, please refer to prior work (e.g., [9]) on how a vehicular cloud is kept intact despite vehicle mobility.

B. Design

The whole uploading process starts with the datacenter broadcasting a *BEGIN* message to all vehicles within in the vehicular cloud. All vehicles process the snapshots they possess in a distributed fashion and upload resulting image pieces back to the datacenter. The process ends when the datacenter receives all image pieces and merges them into one. Due to space limitation, we only focus on the discussion of a simple design here, i.e, the process is initiated by the datacenter periodically. It is possible to further improve our system. For example, the whole process can be adaptive and/or initiated by vehicles in the vehicular cloud.

Hierarchical Stitching: A key element of our system is the *hierarchical stitching algorithm* presented in Algorithm 1, which uses a *tree structure* to assign computation and communication responsibilities. Before running the algorithm, the vehicular cloud assigns "logical IDs" (identifiers) to each vehicle with an image to upload. Vehicles with adjacent IDs should have the maximum overlaps between the snapshot images. This can be done efficiently using the GPS information, camera orientation, and membership list. Roughly speaking, vehicles with close GPS coordinates and similar camera orientations will be likely to have similar fields of view. Vehicular cloud can assign the IDs using this information

Afterwards vehicles know its position in the corresponding tree given the IDs and "Group Size" (which is a predetermined parameter that specifies how many images should be stitched together at the same time). Our hierarchical algorithm then requires each node in the tree to stitch the images from its children to remove the overlapping part among children's images, i.e., the redundant data.

When uploading images, we again adopt a simple design, which partitions the resulting panorama image into approximately even pieces and distribute to every vehicle within the group, and then every vehicle uploads its image piece to the datacenter who is responsible for merging the pieces into the panorama image. The reason is that we want to distribute the uploading workload evenly to each vehicle in the group to reduce latency. Obviously this design is *not* fault-tolerant, and in the future work, we aim to divide the image in a fashion that the datacenter can merge the image even if some pieces are missing. This can be improved by using more complicated upload schemes as will be discussed in Section V.

[1]This can be done in several different ways. One is simply choosing the vehicle with the largest index.

Algorithm 1 Hierarchical Stitching Algorithm

```
1:  procedure DATACENTER
2:      n ← number of members in the desired group
3:      broadcast(BEGIN) to n members of the group
4:      wait until receiving n image pieces
5:      merge received image pieces into one image
6:  procedure VEHICLE
7:      k ← pre-determined Group Size
8:      wait until receiving BEGIN message
9:      group themselves by ID and k
10:     select a leader in each group[1]
11:     while more than one leader do
12:         Leader:
13:             broadcast(REQUEST) to k − 1 members
14:             wait until receiving k − 1 images
15:             stitch k images into one image
16:             group leaders of last round by ID and k
17:         Group member:
18:             wait until receiving REQUEST message
19:             send its image to its leader
20:     Leader:
21:         divide the final image evenly into n pieces
22:         distribute image pieces among vehicles
23:     Group member:
24:         wait until receiving its image piece
25:         send the image piece to the datacenter
```

Message Complexity Analysis: For n vehicles, the message complexity among vehicles is $O(n)$. Suppose each level of the tree corresponds to one *round* of stitching. Observe that for Group Size $= k$, the tree depth (or height) is $O(\log_k n) = O(\log n)$, which means that the algorithm has $O(\log n)$ rounds of stitching. Let level 0 denote the leaf level and level $\log_k n$ denote the root level. Then for each level i, the total number of messages exchanged is $O(n/k^i)$. Hence, using the sum of a geometric series and the fact that k is a constant, the total message complexity is

$$O(\sum_{i=0}^{\log_k n} \frac{n}{k^i}) \leq O(n\frac{1}{1-1/k}) = O(n)$$

Since we require each vehicle to upload a small piece of the panorama image, the message complexity between vehicles and the datacenter is $O(n)$.

Salient Features: The first salient feature is that our algorithm reduces the amount of information sent from vehicles to the datacenter because we remove redundant data, i.e., the overlapped part among the images. The second one is that we distribute the stitching workload to several vehicles to reduce the computation load at the datacenter. Finally, the workload at each node can be made roughly equal by having an *imbalanced* tree where some nodes have smaller number of children.

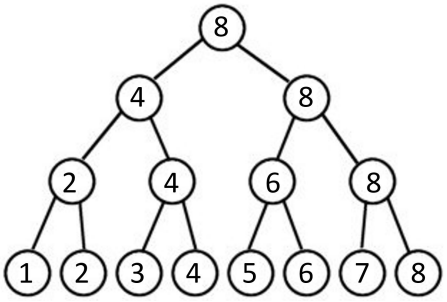

Fig. 1. A hierarchical structure for 8 vehicles

C. Example Scenarios

a) Example I: Hierarchical Structure: Referring to the example in Figure 1, we demonstrate how each leader is selected and which set of images it needs to stitch. In the beginning, 8 vehicles are asked to upload their snapshot images and they obtain IDs 1 to 8 from the vehicular cloud. These vehicles are represented as leaves in the tree. Here, we assume that a vehicle with the largest ID will be the leader. However, other leader election criteria might be used as well. Then vehicle 2 is responsible to stitch the images from vehicles 1 and 2 in the first round. Vehicle 4 is responsible to stitch the images from vehicles 2 and 4 in the second round. Vehicle 8 is responsible to stitch the images from vehicles 4 and 8 in the third round, so on and so forth.

Note that the balanced binary tree in the figure is only for illustration. The tree structure depends on the number of vehicles and Group Size.

b) Example II: Algorithm Flow: Figure 2 shows the algorithm flow and messages exchanged in a scenario of four vehicles and Group Size 2.

III. Evaluation

In this section, we describe the prototype implementation of our framework. Our prototype consists of a central server which acts as the role of the datacenter. A number of machines act as vehicles which have snapshot images to be uploaded. Each "vehicle" process has a front-end and back-end:

- The front-end is implemented in Python with the OpenCV package, and is responsible for (i) computer vision algorithms such as stitching, and (ii) algorithms to divide, distribute, and merge the resulting stitched images.
- The back-end is implemented in Golang, and is responsible for communication parts.

Test Image Sets: The images used in this experiment are from PAnorama Sparsely STructured Areas Datasets (PASSTA) by courtesy of Computer Vision Laboratory (CVL) at Linkoping Univeristy [10]. In the following experiments, we use three sets of images where each vehicle possesses one image. The images are from the Lunch Room image set from [10], which were acquired with a Canon DS70 and wide angle lenses Samyang 2.8/10mm (about 105 degree), with a resolution of 3548x5472 px (approximately 900 KB). A panorama head was used to approximate a fixed rotation of 5 degrees around the vertical axis about the optical center of the camera.

Experimental Setup and Results: The experiments in this paper are conducted on a single machine for a preliminary study. Each process is run on either the central server or a vehicle while these processes communicate through TCP channels that simulate communication in vehicular networks. A simulation on a single machine is appropriate for the purpose of understanding *bandwidth reduction* – the key benefit of our mechanism, since in our system, the amount of information and messaging overhead required is almost identical in both simulation and a practical scenario. One minor aspect not considered in our simulation is that in a practical setting, some control messages may need to be retransmitted due to message loss. However, compared to image size (around 900 KB), the size of control messages (a few bytes) is ignorable.

In the paper, we also report latency for completeness, while the latency observed in our prototype system may not necessarily be similar to a realistic vehicular network environment. However, the data shows that even in a not-so-benign case, the latency does *not* increase too much using our system. The reason that a simulation on a single machine favors naïve method is that transmission is perfect and no retransmission is ever required. Since our system reduces the amount of communication between the datacenter and each vehicle significantly, we envision that in a realistic setting, the latency of our system will be improved. In the future, we plan to perform experiments on a testbed consisting of mobile devices or connected vehicles, as well as simulations using a network simulator to thoroughly understand how our mechanism can help mitigate communication latency.

Before we present our simulation data, we define how we characterize the latency and bandwidth consumption.

- *Bandwidth consumption*: it measures the total amount of data transferred from vehicles to the server. Note that it does *not* include the data exchanged among vehicles.
- *Latency*: it measures the time between the event that the datacenter broadcasts the *BEGIN* message (line 3 in Algorithm 1) and the event that the datacenter merges all image pieces that it receives from vehicles (line 6 in Algorithm 1). Note that it includes both the communication time (to upload and exchange images) and computation time (to stitch images together).

Bandwidth consumption and latency results of the first, second, and third image sets are reported in Figures 3, 4, 5, 6, 7, 8, respectively. We test the scenarios of 4, 6, 8, 10 vehicles with Group Size 1, 2, 3. Note that under our design, Group Size 1 is exactly identical to the naïve solution in which each vehicle uploads its image to the datacenter and the datacenter performs all the stitching.

Discussion: Not surprisingly, our system greatly reduces the bandwidth consumption on the datacenter end as shown in Figures 3, 5, and 7. The best case of reduction is shown in Figure 7 (with 10 vehicles) which reduces bandwidth consumption by the scale of approximately 5/6. The worst

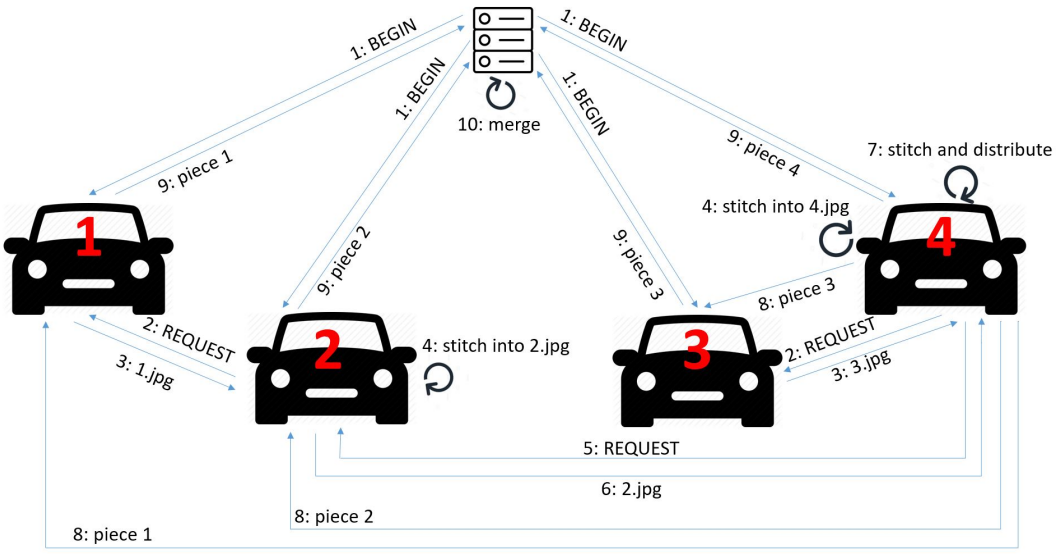

Fig. 2. Algorithm Flow for 4 vehicles and group size 2

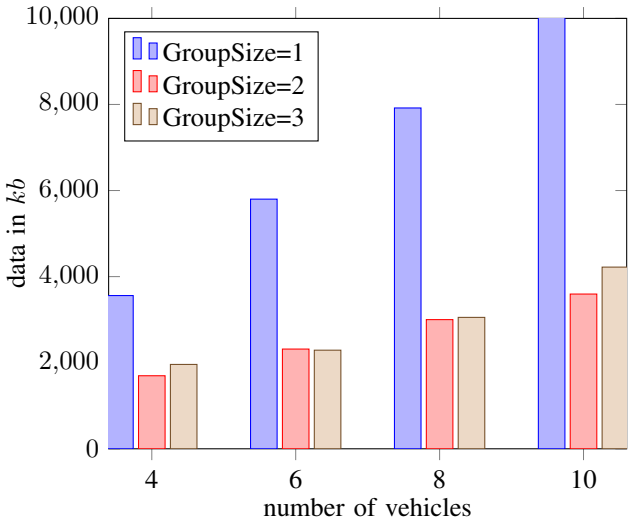

Fig. 3. First image set: Bandwidth Consumption

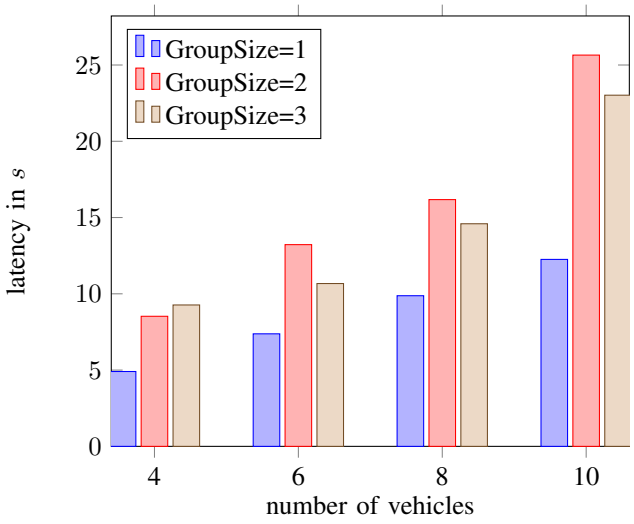

Fig. 4. First image set: Latency

case of reduction is shown in Figure 3 (with 4 vehicles) which reduces bandwidth consumption by the scale of approximately 1/2. This is because we truncate large amount of redundant information, i.e., the overlapped part between images.

As we can observe from Figures 4, 6, and 8, our system does *not* incur too much latency (less than 2x in the worst case). We envision that in a large-scale test where the datacenter is congested by both communication and computation load, our system is likely to have better latency.

Group Size does not seem to affect bandwidth consumption except for one outlier in Figure 5 (the case of 10 vehicles). This is reasonable, since the resulting stitched panorama images should be roughly the same. On the other hand, the latency analysis is more interesting. There are mainly two factors affecting latency. First, Group Size affects the depth (or height) of the tree structure, which affects the number of rounds to stitch images. Larger Group Size will result into smaller depth, since each node has more children. Second, Group Size affects the computation complexity of stitching algorithm. It appears that the stitching algorithm in OpenCV works more efficiently with fewer images at a time. This may be due to how the algorithm detects features from multiple images. These two factors are interleaving; hence, our results do not imply a conclusive choice on the best Group Size. Finding the best option is left as an interesting future work.

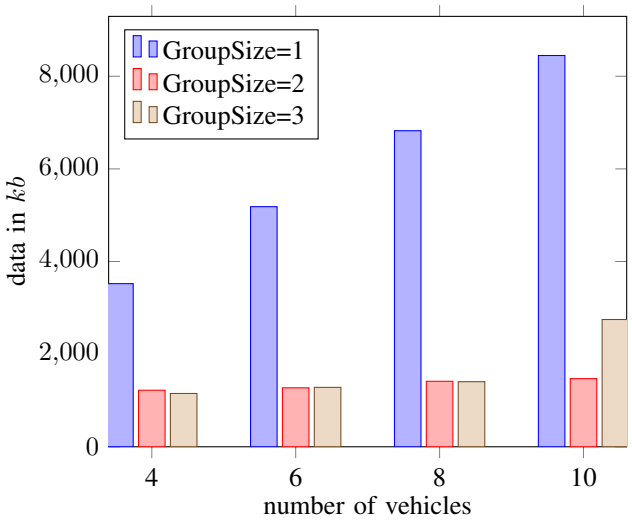

Fig. 5. Second image set: Bandwidth Consumption

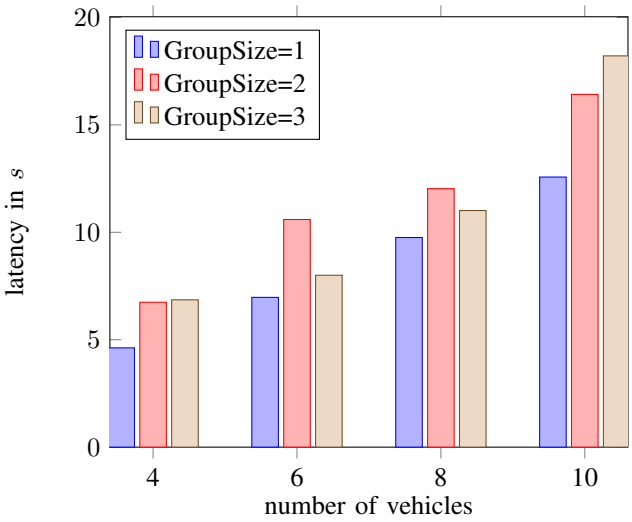

Fig. 6. Second image set: Latency

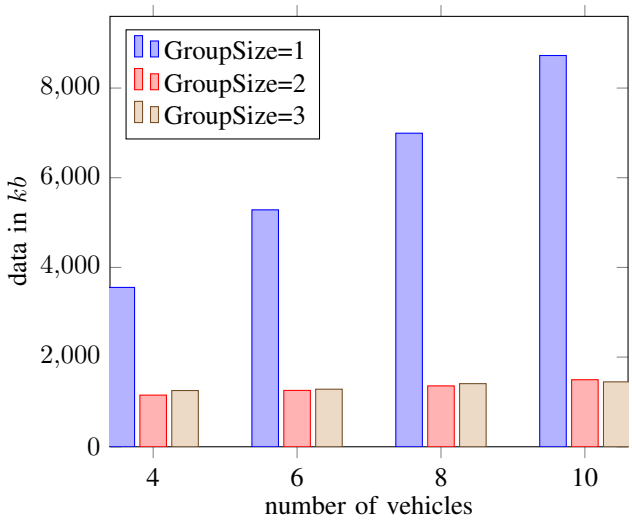

Fig. 7. Third image set: Bandwidth Consumption

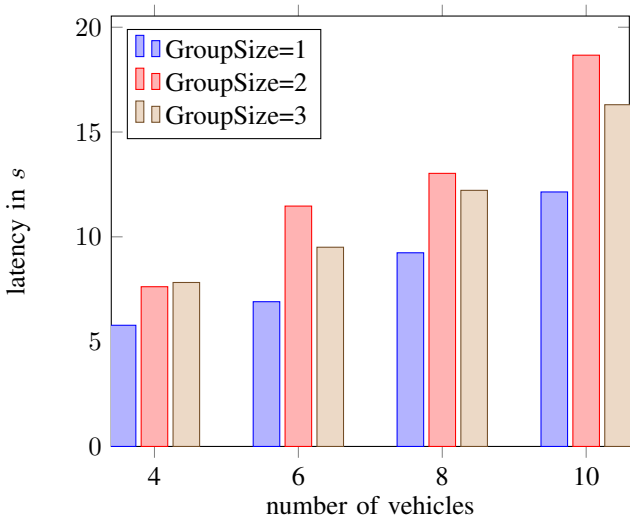

Fig. 8. Third image set: Latency

IV. Related Work

There is a long history of study on improving communication in challenged or bandwidth-constrained networks. We only discuss the most relevant systems here. Weinsberg et al. proposed a novel architecture called Content-Aware Redundancy Elimination (CARE) that maximizes informational values that a constrained network can offer its users [1]. The architecture functions by imposing redundancy elimination on top of the delay-tolerant networks protocol so that any message is evaluated for similarity before transmission. The experiments showed that CARE transmits more unique photos than other similar systems do. Similar to CARE, Uddin et al. focused on a disaster scenario, and proposed a picture delivery service that recognizes similarity [11]. PhotoNet assigns priorities to pictures in an effort to reduce semantic redundancy such as redundancy between pictures taken at the same location.

In an effort to eliminate redundant content among images, Dao et al. presented a framework comprised of 3-phase similarity comparison [2]. In the first phase, coarse-grained metadata of images are compared and if no matching is found, it enters the second phase, where fine-grained features are compared. The third phase is user manual comparison of thumbnails. If a matching is found, the upload of an image is suppressed; otherwise, it shall be uploaded.

Chum et al. proposed and compared two different approaches to detect near identical images and videos [12] in scenarios that require fast processing. The first approach is based on global hierarchical color histograms, while the second is local feature descriptors. The experiments showed that a weak approximation to histogram matching consumes

less storage and yields sufficiently good results. Chum et al. further extended the research with several novel image and video similarity measures based on methods such as color histogram, min-Hash and tf-idf weighting [13].

The focus of our system is completely different from any of the existing work discussed above. First, we target the scenario in a vehicular network; hence, we can intelligently use GPS information for coordination. Second, we focus on using P2P communication to reduce bandwidth consumption. Third, our contribution is on the system architecture whereas prior works focus on computer vision algorithms.

There is also a rich amount of research on vehicular clouds. The core idea of vehicular clouds is to provide cloud services by leveraging computation and communication resources of connected vehicles. Research topics include integration with MapReduce-like computation, and information-centric networking, e.g., [7], [8], [14], [15]. The architecture design of these works are different from ours, because they targeted a more general problem whereas our design and the hierarchical stitching algorithm are focused on a special family of problems.

The main research challenges in in-network aggregation in WSN [5], [3], [4] are on the robust routing protocols and the construction of forwarding tree or the identification of the clusters (of the sensor nodes). On the other hand, since our system is based on the membership protocol from vehicular clouds, routing and the hierarchical tree can be built easily by using location information and virtual IDs. Recall that we rely on ID to allocate the stitching responsibility. In other words, the focus of this work is mainly on the system design and integration and evaluation. Moreover, due to the bandwidth and computation constraints, aggregation in WSN is typically focused on simple functions, e.g., SUM, MAX, whereas we target a much more complicated operation – stitching images.

V. Summary and Future Work

We designed a new P2P scheme to upload images in a vehicular network that greatly reduces computation workload at the datacenter and bandwidth consumption between the datacenter and vehicles. The key of our system is a novel hierarchical stitching algorithm which removes redundant information. Compared to the naïve protocol, our system reduces the bandwidth consumption by a scale factor between 1/2 and 5/6 while incurring moderate latency from our simulation study. This paper presents the preliminary study, and we share some interesting future research directions below.

Our current system does not tolerate failure. For example, failure of any vehicle can lead to information loss and potentially ruin the stitching algorithm. There are two straightforward methods to provide some degree of fault-tolerance. First, instead of aggressively removing redundant information, we should allow some "slack" to recover lost information or computation. One simple mechanism is to have a backup leader to take over a failed computation. Second, instead of uploading pieces of the panorama image, we can divide the panorama with some overlaps between each pieces or even use coding for failure recovery. Consequently, even if a vehicle v fails to upload its image, the useful information it possesses will still be uploaded by other vehicles whose image overlaps with v's image or will be recovered from other coded data. Obviously, these two solutions increase bandwidth consumption, latency and computation load at each vehicle. An interesting future work is to find the balance between fault-tolerance and the metrics of interest.

While we only demonstrate the system that reduces bandwidth consumption for stitching images, we envision that it can be generalized to the applications which usually have sufficiently redundant data. Many data-intensive applications could potentially benefit from using our system. For example, data fusion algorithms can be used to identify and remove redundant data collected from multiple sensor devices, and we envision that a variation of our hierarchical algorithm can significantly reduce the bandwidth consumption as well.

References

[1] U. Weinsberg, Q. Li, N. Taft, A. Balachandran, V. Sekar, G. Iannaccone, and S. Seshan, "CARE: content aware redundancy elimination for challenged networks," in *HotNets*, pp. 127–132, ACM, 2012.

[2] T. Dao, A. K. Roy-Chowdhury, H. V. Madhyastha, S. V. Krishnamurthy, and T. L. Porta, "Managing redundant content in bandwidth constrained wireless networks," *IEEE/ACM Transactions on Networking*, vol. 25, pp. 988–1003, April 2017.

[3] K. Maraiya, K. Kant, and N. Gupta, "Wireless sensor network: A review on data aggregation," *Int. J. Sci. Eng. Res.*, vol. 2, 01 2011.

[4] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "In-network aggregation techniques for wireless sensor networks: a survey," *IEEE Wireless Commun.*, vol. 14, no. 2, pp. 70–87, 2007.

[5] S. Papadopoulos, A. Kiayias, and D. Papadias, "Exact in-network aggregation with integrity and confidentiality," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 10, pp. 1760–1773, 2012.

[6] L. Tseng, T. Higuchi, and O. Altintas, "When cars meet distributed computing: Data storage as an example," in *1st Workshop on Storage, Control, Networking in Dynamic Systems (SCNDS)*, 2017.

[7] M. Eltoweissy, S. Olariu, and M. Younis, *Towards Autonomous Vehicular Clouds*, pp. 1–16. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[8] F. Hagenauer, C. Sommer, T. Higuchi, O. Altintas, and F. Dressler, "Parked Cars as Virtual Network Infrastructure: Enabling Stable V2I Access for Long-Lasting Data Flows," in *23rd ACM International Conference on Mobile Computing and Networking (MobiCom 2017), 2nd ACM International Workshop on Smart, Autonomous, and Connected Vehicular Systems and Services (CarSys 2017)*, (Snowbird, UT), ACM, October 2017.

[9] T. Higuchi, F. Dressler, and O. Altintas, "How to keep a vehicular micro cloud intact," in *VTC Spring*, pp. 1–5, IEEE, 2018.

[10] G. Meneghetti, M. Danelljan, M. Felsber, and K. Nordberg, "Image alignment for panorama stitching in sparsely structured environments," in *SCIA*, 2015.

[11] M. Y. S. Uddin, H. Wang, F. Saremi, G. Qi, T. F. Abdelzaher, and T. S. Huang, "Photonet: A similarity-aware picture delivery service for situation awareness," in *RTSS*, pp. 317–326, IEEE Computer Society, 2011.

[12] O. Chum, J. Philbin, M. Isard, and A. Zisserman, "Scalable near identical image and shot detection," in *CIVR*, pp. 549–556, ACM, 2007.

[13] O. Chum, J. Philbin, and A. Zisserman, "Near duplicate image detection: min-hash and tf-idf weighting," in *BMVC*, pp. 1–10, British Machine Vision Association, 2008.

[14] R. Florin, P. Ghazizadeh, A. G. Zadeh, S. El-Tawab, and S. Olariu, "Reasoning about job completion time in vehicular clouds," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, pp. 1762–1771, July 2017.

[15] S. Arif, S. Olariu, J. Wang, G. Yan, W. Yang, and I. Khalil, "Datacenter at the airport: Reasoning about time-dependent parking lot occupancy," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, pp. 2067–2080, Nov 2012.