Decentralised Data Piggybacking and Link Scheduling for Reliable Broadcast in VANETs

Guangbing Xiao, Haibo Zhang, Zhiyi Huang and Yawen Chen
Department of Computer Science
University of Otago, Dunedin, New Zealand
Email: {$gxiao, haibo, hzy, yawen$}@cs.otago.ac.nz

***Abstract*—Unstable channel links in Vehicular Ad hoc Networks (VANETs) make the design of reliable broadcast schemes challenging. Existing solutions fail to balance the requirements in Packet Delivery Ratio (PDR), latency and communication overhead, rendering the vehicular broadcast either severe packet losses, long time delay or excessive duplication. In this paper, we propose a Decentralized Cooperative Broadcast (DCB) scheme to provide reliable broadcast in VANETs with short latency and lightweight overhead, where all vehicles jointly piggyback some received data to help other neighbors recover the lost packet, and the broadcast links are also flexibly scheduled to enhance the broadcast efficiency. DCB requires no global information but has comparable performance with those solutions holding hypothetical global network information. We prove that DCB is an optimal solution to enhance the PDR when the broadcast is interference-free, and such potential interference can also be solved in DCB with little cost of a bounded time delay. Simulation results show that DCB can achieve an average PDR of 99.6% in less than 100 ms within a 50-vehicle network, which is far more efficient than existing solutions, even very close to the hypothetical centralized broadcast scheme.**

Keywords-VANET; reliability; piggybacking; scheduling

I. Introduction

Buildings, vegetations and even vehicles can frequently block the traffic sight and cause unintentional accidents and injuries on the road [5]. To address this problem, vehicular communications have been adopted in smart vehicles, e.g., Toyota G-BOOK, Volvo On-call, etc., with the expectation of preventing the road hazards by exchanging traffic information among the vehicles. As suggested in the Intelligent Transport Systems (ITS) standards and the Dedicated short-range communications (DSRC) standard [8], each vehicle should broadcast Cooperative Awareness Messages (CAMs) periodically to inform the adjacent neighbors about its driving status, including the geolocation, instant speed, safety alerts, etc.

Reliable CAM delivery is a critical issue in the vehicular broadcast since each CAM contains the safety-related information. Any data loss in the broadcast may cause hidden risk of road crashes and human injuries. However, the empirical study in [1] claims that there exist frequent packet losses in the Vehicle-to-Vehicle (V2V) communications due to the radio jamming and other background noises in the traffic, which can easily deteriorate the PDR of the broadcast up to 50%. Hence, how to provide high PDR of the CAM broadcast in VANETs has become a major challenge to the safety on wheels.

Much work has been done on the reliable broadcast, but few can be adopted to VANETs directly due to the harsh environments and strict requirements, where the data packets should be delivered reliably with low communication overhead and short time delay over unstable channel links. Existing solutions on the reliable broadcast can be divided into three groups: (1) *flooding* [10][11], where the nodes forward each received data packet through all outgoing channel links. This is not suitable for VANETs as the excessive packet expanse can easily choke the channels and cause severe broadcast storms; (2) *cooperative broadcast* [9][7][16], where the data packet from the source node is re-broadcast by a limited number of cooperating nodes until it is received by the sink node. However, such cooperative broadcast schemes require the global network information to carry out the cooperation in the broadcast, which is infeasible in VANETs due to the high mobility of vehicles and lossy channels; (3) *hardware improvement* [3], where the broadcast signals are strengthened to reduce the packet loss by increasing the transmit power or using high performance antennas. Considering the signal's reflection and refraction on the propagation path, the packet loss is still unavoidable even with a high cost of transmission power and hardware investment.

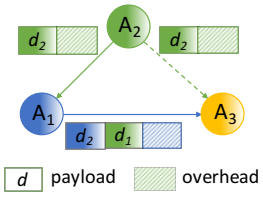

Fig. 1. Data Piggybacking

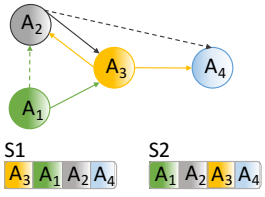

Fig. 2. Link scheduling

The motivations of this paper are twofold. First, we realize that data piggybacking can achieve a high PDR of the CAM braodcast without introducing too much communication overhead [15]. This is particularly true when the security verification is empolyed in the vehicular communications. For instance, the IEEE 1609.2 standard uses the Elliptic Curve Digital Signature Algorithm for authentication, where the signature takes up 209 bytes and the data payload containing the state information is only 53 bytes. As shown in Fig. 1, consider a vehicle A_1 piggybacks the data d_2 in its broadcast. The overall size of A_1's packet is 209+53$\times$(1+1)=315 bytes,

whereas the goodput has increased by 67% comparing to that without piggybacking. Second, we also notice that a flexible schedule can enhance the piggybacking efficiency in the CAM broadcast [13]. In Fig. 2, suppose vehicle A_2 lost the data transmitted by A_1, and vehicle A_4 lost the data transmitted by A_2. Assume that A_3 is selected as the forwarder, and both lost data can be recovered in one round if A_3 is scheduled after A_1 and A_2. Specifically, the scheduler *S2* in Fig. 2 can achieve a higher PDR and shorter latency than *S1*.

Driven by the motivations above, we propose a decentralized cooperative broadcast scheme to enhance the PDR of the CAM broadcast in VANETs by exploring the advantages of data piggybacking and link scheduling. All vehicles jointly select some received CAMs and piggyback them in the routine broadcast to help other vehicles recover the lost data; the broadcast links are also flexibly scheduled to enhance the piggybacking efficiency, expecting that all CAMs would be received with short latency and low communication overhead. The challenges are how to cooperate in the data piggybacking and how to schedule the broadcast links in decentralized VANETs, i.e., developing an optimal piggybacking strategy together with a link scheduler to maximize the PDR of the CAM broadcast with low communication overhead and short latency, only using each vehicle's local network information. The contributions of this work are summarized as:

- We propose a decentralized cooperative broadcast scheme with data piggybacking and link scheduling in VANETs to recover lost CAMs in the broadcast. Each vehicle develops the piggybacking strategy and the link scheduler only based on incomplete local network information;
- We prove that the proposed scheme can maximize the PDR of the CAM broadcast with short latency and lightweight overhead, and it can also solve the broadcast interference with little cost of a bounded time delay;
- We evaluate the proposed scheme with real experimental data, and the simulation results show that it can achieve significant improvement on broadcast performance in comparison with other related solutions.

The remainder of this paper is organized as below. Several existing broadcast schemes are reviewed in Section II, and the network model and the problem formulation are presented in Section III. In Section IV, a decentralized cooperative broadcast scheme, DCB, is proposed and the corresponding algorithm is discussed. Some theorems are given in Section V, and we prove that DCB is an optimal solution to maximize the PDR if the broadcast is interference-free, and we also demonstrate that such interference can be solved in DCB with a bounded time delay. Some simulations together with comparisons are presented in Section VI, and Section VII concludes the whole paper.

II. Related Work

Flooding is a frequently used method to enhance the PDR of the broadcast due to its simple structure and easy implementation. In [11], a probabilistic flooding is developed, where each vehicle forwards the received packets with a probability p. The probability p is pre-identified from the experimental curves, which demonstrate the relationship of PDR, vehicle density and driving speed. In [10], Lou *et al* proposed a double-covered broadcast scheme where each node is assigned with two forwarders. Once the node fails to receive the data, one forwarder will retransmit the data until the other one has eavesdropped this retransmission. These flooding-based schemes can reduce excessive duplication to some extend, but the inherent redundant forwarding still exists, putting all nodes at risk of broadcast storms [14].

Recently, cooperative broadcast has become a popular approach to enhance the PDR in VANETs. In [9], he vehicle's driving status is selectively piggybacked in the beacon packet to increase the driving safety. Considering the limited beacon size, only a small number of vehicles' data can be piggybacked in the broadcast, which is hard to provide global traffic information to the vehicles. This is extremely risky as any missing data may cause hidden collisions and accidents. Cooperative piggybacking is also employed in [7] to speed up the data propagation and minimize the delivery latency among the vehicles. However, it assumes that the network topology is fully-connected and all vehicles can obtain the global network information, which are quite strong for VANETs due to the high mobility of vehicles. A cooperative code-based broadcast is studied in [16], where a group of vehicles forward the message segments cooperatively after positive orthogonal coding, and the received segments are decoded and recovered at the sink. However, such coding and decoding can reduce the broadcast efficiency and cause long latency in the broadcast.

Some hardware-based methods are also studied in VANETs. For example, the vehicle experiments in [3] infer that increasing the broadcast rate doesn't help improve the PDR. Instead, higher transmit power and moderate transmission rate can achieve a better performance in the broadcast. It also indicates that the antennas mounted on the top of tall vehicles experience a better communicate channel due to the less obstruction in the traffic. However, we focus on the broadcast scheme in this paper, and the hardware improvement is beyond our scope.

III. System Model and Problem Formulation

A. System Model

Consider a group of vehicles driving on the road for cooperative broadacst. Each vehicle has a Region of Interest (RoI) to monitor the adjacent vehicles, which is defined as a disk area with a radius of r centred at the vehicle's weight point. Each vehicle is equipped with a radio transceiver, and the transmission range is the same as r. We assume that the channel links are symmetric and model the network topology as an undirected graph $\mathcal{G}$=(V, E). The vertices in V=$\{v_i | i \leq n, n \in \mathbb{Z}^+\}$ represent n vehicles in the network, and the edges in E=$\{l_{ij} | v_i, v_j \in V, i \neq j\}$ denote the channel links between vehicles. Time is synchronized with the Global Positioning System (GPS) and slotted with equal length of δ ms. Slotted CSMA/CA is employed to access the channel, which can achieve higher channel utilization than TDMA, and

also has a better scalability than CSMA in terms of delay predictability and collision probability [6].

Each vehicle broadcasts CAMs periodically and shares its driving status with other vehicles in its RoI, such as the geographical position, instant speed and safety alerts. The CAM has a Time-To-Live (TTL) of τ ms, and each vehicle competes for at least one broadcast before the TTL's deadline. Each vehicle v_i also has a receive buffer B_i to cache the received CAMs, and the CAM will be discarded once its TTL has expired.

Due to the multipath fading, radio jamming and other background noises in the traffic, many links in the V2V communications are unstable, and not all vehicles can receive the CAMs from other vehicles. To achieve high PDR of the CAM broadcast, cooperative piggybacking and link scheduling are adopted at each vehicle to recover the lost CAMs. Whenever vehicle v_i broadcasts a data packet, it can select at most w CAMs from its receive buffer B_i and piggyback them to help other vehicles recover the lost data. The broadcast links are also flexibly scheduled to enhance the piggybacking efficiency with short latency and lightweight overhead.

B. Problem Formulation

As shown in Fig. 2, the link scheduling and data piggybacking are tightly coupled since the CAMs available for piggybacking in the receive buffer highly depends on the broadcast sequence. We focus on developing a cooperative broadcast scheme by utilizing data piggybacking and link scheduling, and aim to maximize the PDR of CAM broadcast by recovering the lost CAMs with low latency and lightweight overhead. The two challenging questions we will address are: (1) how to cooperatively make the piggybacking strategy only based on each vehicle's local network information? (2) how to schedule the broadcast sequence for all vehicles in a decentralised way?

For any lost CAM m_k^i requested by v_j, we only consider the vehicles in v_j's RoI for data piggybacking and link scheduling, as these vehicles have a higher success probability to help each other recover the lost CAMs comparing to the vehicles outside the RoI. For convenience, some key notations are summarised in Table I.

Definition 1. *For a lost CAM m_k^i requested by v_j, suppose $d_t(i,j)$=$[v_i \dots v_x, v_{x+1} \dots v_j]$ is a path from v_i to v_j. $d_t(i,j)$ is called a Reliable Piggybacking Path (RPP) for m_k^i iff:*

(1) *$\forall v_x \in d_t(i,j)$, l_t(x,x+1)=1, i.e., each vehicle on the path has a reliable link to the next vehicle until it reaches v_j;*
(2) *$\exists v_x \in d_t(i,j)$, $a_t^x(m_k^i)$=1, i.e., there exists at least one vehicle on the path that has a copy of m_k^i.*

Let $\Theta(i,j)$ represent the set of all paths from v_i to v_j in v_j's RoI, and we use $p_t(i,j)$ in Eq. 1 to represent whether there exists a RPP from v_i to v_j to recover m_k^i.

$$p_t(i,j) = \begin{cases} 1, \sum\limits_{d_t(i,j)\in\Theta(i,j)} \prod\limits_{x\in d_t(i,j)} (a_t^x(m_k^i)\cdot l_t(x,x+1)) \geq 1; \\ 0, \ otherwise. \end{cases} \tag{1}$$

TABLE I
NOTATIONS AND DESCRIPTIONS

Notation	*Description*
m_k^j	the CAM generated by v_j at time k;
N_t^i	the set of vehicles that have v_i in their RoIs at time t;
$a_t^i(m_k^j)$	v_i's piggybacking decision on m_k^j at time t; 1: v_i has received m_k^j and will piggyback it at time t; 0: otherwise.
A_t^i	v_i's piggybacking strategy at time t, which is a set of v_i's piggybacking decisions on each CAM, i.e., A_t^i=$\{a_t^i(m_k^j) \mid t-\tau \leq k \leq t, j \leq n, j \in \mathbb{Z}^+\}$;
$\boldsymbol{A}_t$	the joint piggyback strategy of all vehicles at time t, i.e., $\boldsymbol{A}_t$=$\{A_t^i \mid i \leq n, i \in \mathbb{Z}^+\}$;
x_t^i	v_i's link scheduling at time t; 1: v_i is scheduled at time t; 0: otherwise.
$\boldsymbol{X}_t$	the link scheduler for all vehicles at time t, i.e., $\boldsymbol{X}_t$=$[x_t^1, x_t^2, \cdots x_t^n]$;
$r_t^j(m_k^i)$	whether v_j is requesting for m_i^k at time t; 1: v_j is requesting for m_i^k at time t; 0: otherwise.
$l_t(i,j)$	whether the link between v_i and v_j is reliable at time t; 1: the link between v_i and v_j is reliable at time t; 0: otherwise.
$\Delta(n)$	the time duration for n vehicles make one broadcast;
$L(t)$	the set of all lost CAMs at time t;

Definition 2. *Two RPPs $d_t(i,j)$ and $d_t(p,q)$ are called non-conflict RPPs iff:*

(1) *$d_t(i,j)$ and $d_t(p,q)$ have no shared link, or*
(2) *for any shared link l_{xy}, the relative position of v_x and v_y in both paths are the same, i.e. if v_x precedes v_y in $d_t(i,j)$, v_x also precedes v_y in $d_t(p,q)$, and vice versa.*

We use $f_t(ij,pq)$=0 to indicate $d_t(i,j)$ and $d_t(p,q)$ are non-conflict RPPs. Based on the definitions of RPP and non-conflict RPPs, we formulate the data piggybacking and link scheduling in the cooperative broadcast as an optimisation problem in Eq. 2, by firstly constructing a non-conflict RPP for each lost CAM, and then scheduling the broadcast links to coincide with the path sequence of the non-conflict RPPs.

$$\max \lim_{t\to\infty} \Big(\sum_t \frac{\sum_{m_k^i\in L(t)} \sum_{v_j\in N_t^i} r_t^j(m_k^i)\cdot p_t(i,j)}{\sum_{m_k^i\in L(t)} \sum_{v_j\in N_t^i} r_t^j(m_k^i)}\Big)/t \tag{2a}$$

$$\text{s.t.} \quad f_t(ij,pq)=0, \quad \forall\ r_t^j(m_k^i)=1\ \&\ r_t^q(m_s^p)=1, \tag{2b}$$

$$\sum\nolimits_{m_k^j\in B_i} a_t^i(m_k^j) \leq w, \quad \forall v_i \in V, \tag{2c}$$

$$\sum\nolimits_{v_i\in N_t^j} x_t^i \leq 1, \quad \forall v_j \in V, \tag{2d}$$

The objective function in (2a) is the long-run average ratio of the recovered CAMs to the total lost CAMs. Maximising the objective function will maximise the number of successfully recovered lost CAMs in the broadcast. Constraint (2b) restricts that the RPPs for any two lost CAMs are non-conflict. Constraint (2c) enforces that each vehicle can piggyback at most w CAMs in one broadcast. Constraint (2d) limits that only one vehicle can broadcast in a specific time slot in each RoI to eliminate the potential interference.

IV. Decentralized Solution

To maximise the number of recovered lost CAMs, a non-conflict RPP should be firstly constructed for each lost CAM.

Then, the joint piggybacking strategy $\boldsymbol{A}_t$ can be developed according to the data flow on the RPP, i.e., the vehicles on the RPP should piggyback the requested CAMs sequentially, and the link scheduler $\boldsymbol{X}_t$ should also be generated to coincide with the path sequence on the RPP.

Exploring such non-conflict RPPs requires global information on network connectivity, which is difficult in VANETs due to the dynamic topology and lossy channels. To address this problem, we propose a Decentralized Cooperative Broadcast (DCB) scheme with data piggybacking and link scheduling, which can enhance the PDR of CAM broadcast with short latency and light network load in decentralized VANETs. The basic ideas are:

(1) each vehicle makes broadcast periodically, including its driving status, the request vector, w piggybacked CAMs (if any) and etc., where the request vector is a index of the vehicle's lost CAMs to inform other vehicles which CAMs it has lost.

(2) each vehicle overhears other's broadcast and infers the link connectivity based on the received request vectors;

(3) each vehicle explores non-conflict RPPs for all lost CAMs. The vehicles on the RPP piggyback the requested CAMs which forms the piggybacking strategy; the broadcast links are scheduled to match the data flow (path sequence) on the RPPs which forms the link scheduler.

A. Link Inferring

Each vehicle v_i should infer the link connectivity before exploring RRPs for lost CAMs. As some vehicles may be involved into multiple RoIs, only inferring the link connectivity in v_i's own RoI is not sufficient. As illustrated in Fig. 3, v_1 is a vehicle in both v_i's RoI and v_2's RoI. Only considering v_1's piggybacking strategy and link scheduling in v_i's RoI may conflict with that in v_2's RoI. Hence, the link connectivity in any RoI that overlaps with v_i's RoI (v_2 and v_5's RoI in Fig. 3), should be inferred.

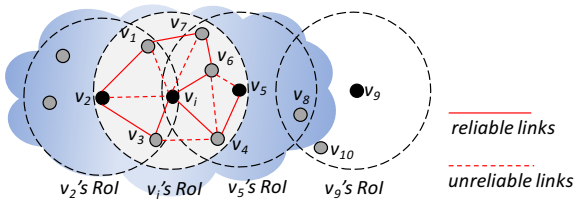

Fig. 3. Different types of links for inferring

For each vehicle v_i, the links in its own RoI and the overlapped RoIs can be divided into the following two groups:

(1) *Direct Links:* the links that directly connect v_i to its one-hop neighbors. In Fig. 3, $\{l_t(x,i)|1 \leq x \leq 7\}$ are all v_i's direct links. Any vehicle that has a reliable link to v_i is called a reliable neighbor; otherwise, it is an unreliable neighbor. For instance, $\{v_3, v_4, v_6\}$ are v_i's reliable neighbors and $\{v_1, v_2, v_5, v_7\}$ are unreliable neighbors. The direct links can be easily inferred based on each v_i's own CAM lost. In other words, we can have $l_t(a,i)$=0 if v_i lost the CAM broadcast by v_a; otherwise $l_t(a,i)$=1.

(2) *Indirect Links:* the links that connect v_i's two neighbors. If both vehicles are unreliable neighbors of v_i, it is an unreachable indirect link; otherwise it is a reachable indirect link. $\{l_t(2,3), l_t(3,4), l_t(4,5), l_t(4,6), l_t(6,7)\}$ in Fig. 3 are reachable indirect links, whereas $\{l_t(1,2), l_t(1,7)\}$ are unreachable indirect links.

For the reachable indirect links, we can use the overheard request to infer the connectivity. Suppose v_i in Fig. 3 overhears that v_6 is requesting for $\{m_k^4, m_s^5\}$. It can infer that the links from $\{v_4, v_5\}$ to v_6 are not reliable, and the links from other vehicles to v_6 are reliable, i.e., $l_t(4,6)$=$l_t(5,6)$=0, and $l_t(x,6)$=1 if $x \notin \{v_4, v_5\}$. For an unreachable indirect link, it is difficult for v_i to infer the connectivity directly as no information is heard over unreliable links. However, if a potential forwarder can piggyback the request of any vehicle that connected to this link, then v_i can infer its connectivity. As shown in Fig. 3, v_i cannot hear from $\{v_1, v_2\}$. If v_3 can piggyback v_2's requests to v_i, then the unreachable indirect link $l_t(1,2)$ can be inferred. The inferring policy is similar to that of reachable indirect links.

B. Decentralized Broadcast Scheduling and Piggybacking

As can be seen from Fig. 3, for each lost CAM requested by v_i, only the vehicles in v_i's RoI have the chance to piggyback the lost CAM as the vehicles outside of v_i's RoI will not have the CAM cached in its buffer even if it can receive it. It can also be seen that, based on the policies given in Section IV-A, each vehicle in v_i's RoI can infer all links in v_i's RoI. Hence, all vehicles in v_i's RoI will have the same knowledge on link connectivity for the sub-graph covered by v_i's RoI. Based on this common knowledge, the joint broadcast scheduling and data piggybacking can be decentralized, and each vehicle in v_i's RoI can compute the RPP for the lost CAM, and perform scheduling and piggybacking independently. The pseudocode of DCB is given in Algorithm 1. The key idea is to let each vehicle explore an non-conflict RPP for each lost CAM in its RoI via the Floyd-Warshall algorithm [2], and schedule the vehicles on the RPPs to piggyback the requested CAM sequentially. All vehicles' piggybacking decisions form the joint piggybacking strategy, and the schedulers are developed to coincide with the piggybacking sequences along the RPPs.

Upon broadcasting a CAM, each v_i should:

(1) check whether there is any lost CAM in its receiving buffer B_i, and generate a request vector rq_i to contain the indexes of all lost CAMs.

(2) check its transmitting buffer T_i, which is used to cache the selected CAMs for piggybacking. If it is not empty, choose all of cached CAMs for the coming broadcast.

(3) pack all data (the routine CAM, the request vector, piggybacked CAMs and etc.) into one packet, and make the broadcast via slotted CSMA/CA.

Upon receiving a data packet from v_j, each v_i should:

(1) for each piggybacked CAM, if there is no such a CAM in B_i, it will be cached; otherwise the one with older timestamp is discarded.

Algorithm 1: Piggybacking and Scheduling at v_i

```
Upon broadcasting a data packet:
  for each lost CAM in B_i do
      Add the index of the lost CAM into rq_i;
  Choose all CAMs in T_i for the coming broadcast;
  Make the broadcast, containing the routine CAM m_t^i, the
    request vector rq_i, w piggybacked CAMs, etc.;
Upon receiving a data packet from v_j:
  for each piggybacked CAM in the received packet do
      if there is no such CAM in B_i then
          Cache it in B_i.
      else
          Keep the CAM with the latest time stamp.
  for all requested CAMs in rq_j do
      Infer and update the link connectivity;
  if the number of CAMs in T_i is larger than w then
      Rank all CAMs in ascending order by TTL;
      Select w top-ranked CAMs for piggybacking;
  Explore non-conflict RPPs for all requested CAMs via
    Floyd-Warshall algorithm.
  if fail to find non-conflict RPPs then
      Dual scheduling the conflict vehicles and construct
        new RPPs;
  for each RPP to recover m_k^p do
      Develop the piggybacking strategy based on RPPs;
      Develop the slot scheduler to match the RPP;
      Merge the scheduler with existing ones via TTL;
      if v_i is involved in the piggybacking then
          if a vehicle is scheduled in the same slot then
              Compare the TTLs and reschedule the one
                with tight deadline;
          Copy the CAM m_k^p into T_i for piggybacking;
```

(2) infer and update the link connectivity based on the requests in rq_j.

(3) check the number of CAMs in B_i that have been requested for piggybacking. If it is larger than w, sort them in ascending order based on the remaining time before TTL expiration, and copy the w top-ranked CAMs to the transmit buffer for piggybacking.

(4) explore the non-conflict RPPs for the w requested CAMs via the Floyd-Warshall algorithm. Each requested CAM will be routed from the current vehicle to the one that requested this CAM along its selected RPP.

(5) if it fails to find non-conflict RPPs for all CAMs, the vehicles on the conflict path should be dual scheduled to construct new RPPs. Suppose v_5 in Fig. 3 is requesting for m_k^7, and v_3 is requesting for m_s^5. p=$[v_7 - v_6 - v_i - v_4 - v_5]$ is explored for recovering m_k^7, and p'=$[v_5 - v_4 - v_i - v_3]$ is explored as the RPP for recovering m_s^5, which conflicts at $\{v_i, v_4\}$. To solve this conflict, a new RPP is constructed with dual scheduling of v_i or v_4, e.g., p^*=$[v_7 - v_6 - v_5 - v_i - v_4 - v_i - v_5 - v_3]$.

(6) develop the piggybacking strategy and schedules based on the RPPs. Suppose v_i in Fig. 3 is requesting for m_k^1 and p=$[v_1 - v_2 - v_3 - v_i]$ is explored as the non-conflict RPP. The vehicles on p, starting from the one nearest to v_i with a copy of the requested m_k^1, ending with the one previous to v_i, make the decisions to piggybacking m_k^1 and copy it into their transmit buffers. Hence, the piggybacking strategy and scheduler are developed as $\boldsymbol{A}_t(p)$=$\{a_t^2(m_k^1) = 1, a_t^3(m_k^1) = 1\}$, $\boldsymbol{X}_t(p)$=$\{v_2, v_3\}$.

(7) merge all strategies and schedulers based on the CAM's TTL: the shorter remaining time a CAM has before TTL's deadline, the earlier the corresponding vehicles for piggybacking are scheduled. Suppose v_i in Fig. 3 is requesting for m_k^1 and v_6 is requesting for m_s^5. p=$[v_1 - v_2 - v_3 - v_i]$ is explored as the RPP for recovering m_k^1. The piggybacking strategy and scheduler are $\boldsymbol{A}_t(p)$=$\{a_t^2(m_k^1) = 1, a_t^3(m_k^1) = 1\}$, $\boldsymbol{X}_t(p)$=$\{v_2, v_3\}$. Similarly, p'=$[v_5 - v_4 - v_i - v_6]$ is explored as the RPP for recovering m_s^5. The piggybacking strategy and schedule are $\boldsymbol{A}_t(p')$=$\{a_t^4(m_s^5) = 1, a_t^i(m_s^5) = 1\}$, $\boldsymbol{X}_t(p')$=$\{v_4, v_i\}$. As p and p' are non-conflict RPPs, the proposed strategies and schedulers can be merged regarding time stamps, i.e., $\boldsymbol{A}_t$=$\{a_t^2(m_k^1) = 1, a_t^3(m_k^1) = 1, a_t^4(m_s^5) = 1, a_t^i(m_s^5) = 1\}$, $\boldsymbol{X}_t$=$\{v_2, v_3, v_4, v_i\}$ if $k > s$.

(8) each vehicle checks whether any other vehicle in its RoI is scheduled at the same time slot. If yes, the two vehicles will be rescheduled regarding the time stamp to avoid interference. As shown in Fig. 3, suppose v_8 is on a RPP to recover m_t^{10} in v_9's RoI, and v_4 is on a RPP to recover m_s^5 in v_5's RoI. If v_8 is close to v_4 and happens to be scheduled in the same slot of v_4, v_i can infer v_8's link connectivity in v_5's RoI, but it cannot figure out whether v_8 is scheduled in v_9's RoI, which can cause interference with v_4's broadcast. To solve this problem, the time stamps of m_s^5 and m_t^{10} should be compared, and v_4 will be scheduled first if $s > t$ and vise versa.

V. Theoretical Analysis

For each vehicle in v_i's RoI, we assume it can find at least one RPP to reach v_i, and the CAM's TTL is long enough to allow each vehicle in the RoI to have at least one chance to make a broadcast. Then, we can have the following Theorems.

Theorem 1. *Consider a RoI with h vehicles and m lost CAMs. If the RPPs for all lost CAMs are non-conflict and the vehicle's broadcast does not interfere with other vehicles outside the RoI, then Algorithm 1 can generate the optimal solution to recover all lost CAMs before their TTL's deadlines, where the number of piggybacked CAMs at each vehicle is no larger than $m/\lfloor \frac{\tau}{\Delta(h)} \rfloor$.*

Proof: Suppose m_k^i is a CAM requested by v_j at time t, and $d_t(i,j)$=$[v_i - v_p \ldots - v_q - v_j]$ is a RPP from v_i to v_j. According to Algorithm 1, the piggybacking strategy and broadcast scheduler for m_k^i are developed as $\boldsymbol{A}_t(ij)$=$\{a_t^p(m_k^i) = 1, \cdots a_t^q(m_k^i) = 1\}$, $\boldsymbol{X}_t(ij)$=$[v_p, \cdots v_q]$, which should be

merged with the piggybacking strategies and broadcast schedulers for other lost CAMs in v_j's RoI.

When merging the piggybacking strategies, if there is no shared link in all RPPs, each lost CAM can be recovered individually in the corresponding RPP, and each vehicle on the RPPs only has to piggyback one CAM. However, if there exists a shared link on all RPPs, it becomes bottleneck of the CAM recovery as each piggybacked data should be forwarded through this link. Since each vehicle has $\lfloor \frac{\tau}{\Delta(h)} \rfloor$ chances to broadcast before its deadline, the maximum number of piggybacked CAMs at each vehicle is $m/\lfloor \frac{\tau}{\Delta(h)} \rfloor$. Besides, Constraint (2c) can also be satisfied when $w = m/\lfloor \frac{\tau}{\Delta(h)} \rfloor$.

When merging the broadcast schedulers, we use $d_t(h,j)$=$[v_h - v_x \ldots - v_y - v_j]$ to indicate the RPP for another lost CAM m_s^h in v_j's RoI, and $\boldsymbol{X}_t(hj)$ is the corresponding scheduler. As all RPPs are non-conflict, we have:

(1) if there is no shared vehicle in $\boldsymbol{X}_t(ij)$=$[v_p, \cdots v_q]$ and $\boldsymbol{X}_t(hj)$=$[v_x, \cdots v_y]$, then $\boldsymbol{X}_t$=$\boldsymbol{X}_t(ij) \cup \boldsymbol{X}_t(hj)$=$[v_p, \cdots v_q, v_x, \cdots v_y]$ is a feasible scheduler to recover both m_k^i and m_s^h, where $\boldsymbol{X}_t(ij)$ and $\boldsymbol{X}_t(hj)$ are merged without conflict;

(2) if there is only one shared vehicle v_a in $\boldsymbol{X}_t(ij)$=$[v_p, \cdots v_a \cdots v_q]$ and $\boldsymbol{X}_t(hj)$=$[v_x, \cdots v_a, \cdots v_y]$, then $\boldsymbol{X}_t$=$[v_p, \cdots v_x, \cdots v_a \cdots v_q, \cdots v_y]$ is a feasible scheduler to recover both m_k^i and m_s^h, where $\boldsymbol{X}_t(ij)$ and $\boldsymbol{X}_t(hj)$ are merged without conflict;

(3) if there exist more than one shared vehicles in $\boldsymbol{X}_t(ij)$ and $\boldsymbol{X}_t(hj)$, the relative positions of any two shared vehicles $\{v_a, v_b\}$ must be the same according to non-conflict RPPs' definition. Considering $\boldsymbol{X}_t(ij)$=$[v_p, \cdots v_a, v_m \cdots v_n, v_b, \cdots v_q]$ and $\boldsymbol{X}_t(hj)$=$[v_x, \cdots v_a, v_r \cdots v_z, v_b, \cdots v_y]$, $\boldsymbol{X}_t$=$[v_p, \cdots v_x, \cdots v_a, v_m \cdots v_n, v_r \cdots v_z, v_b, \cdots v_q, \cdots v_y]$ is a feasible scheduler to recover both m_k^i and m_s^h, where $\boldsymbol{X}_t(ij)$ and $\boldsymbol{X}_t(hj)$ are merged without conflict;

As shown in $\boldsymbol{X}_t$, each vehicle in the RoI is scheduled in a specific time slot, and no vehicle has to compete for a slot with others, where Constraint (2d) is satisfied.

Based on all conclusions above, a feasible piggybacking strategy and broadcast scheduler are developed in Algorithm 1 to recover all lost CAMs, where Eq. (2a) also achieves the maximum value as 1 with all Constraint (2b)-(2d) satisfied. ■

One assumption in Theorem 1 is that all RPPs are non-conflict, which does not always hold as the RPP for one lost CAM may conflict with the RPPs for other lost CAMs. We have the following two theorems to justify that such conflict can be solved by the dual scheduling in Algorithm 1.

Theorem 2. *If two RPPs $d_t(i,j)$ and $d_t(h,j)$ conflict with each other and the vehicle's broadcast does not cause interference with vehicles outside the RoI, such conflict can be solved in Algorithm 1 by dual scheduling of one conflict vehicle with a maximum piggybacking latency of $\delta \cdot (\epsilon_{ij} + \epsilon_{hj} - 5)$, where ϵ_{ij} and ϵ_{hj} are the number of hops in $d_t(i,j)$ and $d_t(h,j)$ respectively, and δ is the length of the time slot.*

Proof: Suppose $d_t(i,j)$=$[v_i - v_m \ldots v_a \ldots v_b \ldots - v_n - v_j]$ and $d_t(h,j)$=$[v_h - v_p \ldots v_b \ldots v_a \ldots - v_q - v_j]$ are two RPPs in v_j's RoI, which conflict at $\{v_a, v_b\}$'s scheduling. A sub-scheduler $\boldsymbol{X}_t^1$=$[v_m, \cdots v_p, \cdots]$ can be developed by merging the sub-paths of $[v_m \ldots v_a) \subset d_t(i,j)$ and $[v_p \ldots v_b) \subset d_t(h,j)$ sequentially, where the source vehicles v_i and v_h are excluded. Similarly, another sub-scheduler $\boldsymbol{X}_t^2$=$[\cdots v_n, \cdots v_q]$ can be developed by merging $(v_b \cdots v_n] \subset d_t(i,j)$ and $(v_a \cdots v_q] \subset d_t(h,j)$ sequentially, where the sink vehicle v_j is excluded. Then, we have both $\boldsymbol{X}_t$=$\boldsymbol{X}_t^1 \cup [v_a \cdots v_b \cdots v_a] \cup \boldsymbol{X}_t^2$ and $\boldsymbol{X}_t^{'}$=$\boldsymbol{X}_t^1 \cup [v_b \cdots v_a \cdots v_b] \cup \boldsymbol{X}_t^2$ as feasible schedulers to solve the conflict by dual scheduling of v_a and v_b respectively, where both the path sequences of $d_t(i,j)$ and $d_t(h,j)$ are satisfied.

To recover m_s^h in $\boldsymbol{X}_t$, $(\epsilon_{ij} - 2 + \epsilon_{hj} - 2 - 1)$ vehicles are scheduled because $\{v_i, v_h, v_j\}$ are excluded and only v_a is dual scheduled. As the piggybacking latency is proportional to the number of vehicles and more vehicles are scheduled to recover m_s^h than m_k^i, it has the maximum piggybacked latency as $\delta \cdot (\epsilon_{ij} + \epsilon_{hj} - 5)$. Similarly, To recover m_k^i in $\boldsymbol{X}_t^{'}$, $(\epsilon_{ij} - 2 + \epsilon_{hj} - 2 - 1)$ vehicles are scheduled, and the maximum piggybacked latency is also $\delta \cdot (\epsilon_{ij} + \epsilon_{hj} - 5)$. ■

Theorem 3. *If the TTL is long enough to allow each vehicle to have two chances to make a broadcast and the vehicle's broadcast is interference-free with other vehicles outside the RoI, Algorithm 1 can recover all lost CAMs before deadline even if the RRPs conflict with each other.*

Proof: If all RPPs are non-conflict and the vehicle's broadcast is interference-free, Theorem 1 has proved that Algorithm 1 can recover all CAMs before deadline even if each vehicle has only one chance to make a broadcast.

If RPPs conflict with each other and the vehicle's broadcast is interference-free, Theorem 2 has proved that the dual scheduling in Algorithm 1 can also solve this conflict. Consider two RPPs conflict at $\{v_a, v_b\}$'s scheduling, the link $l_t(a,b)$ can be scheduled at most in two directions, i.e., from v_a to v_b or from v_b to v_a. Hence, a slot scheduler can be developed to traverse all links in two directions by scheduling each vehicle twice. For any conflict RPPs, if each vehicle has two chances to make a broadcast, a feasible scheduler can be developed by scheduling the conflict links from two different directions. ■

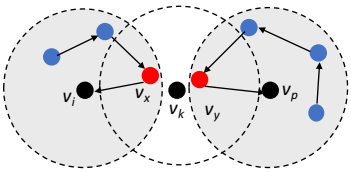

Fig. 4. Interference outside the RoI

A non-conflict scheduler does not mean interference-free as it may cause interference with the vehicles outside the RoI. As shown in Fig. 4, assume v_x and v_y are both in v_k's RoI, which overlaps with both v_i and v_p's RoIs. Considering v_p and v_i cannot hear from each other, v_i can neither infer the link connectivity in v_p's RoI nor determine whether v_y is scheduled by v_p. Similarly, v_p cannot confirm whether v_x is scheduled

in v_i's RoI. Hence, they may schedule v_x and v_y at the same slot which can cause interference outside each RoI.

Theorem 4. *The interference in Fig. 4 can be solved by scheduling the vehicles with a tighter deadline first in the CAM recovery. Suppose v_x is a vehicle with interference outside v_i's RoI, and ϵ_{xi} is the number of hops from v_x to v_i. Λ_t^i is the set of vehicles in the RoIs that overlap with v_i's RoI, and $|\Lambda_t^i|$ is the number of vehicles in Λ_t^i. The lower and upper bounds of piggybacking latency are $\delta \cdot \epsilon_{xi}$ and $\delta \cdot (|\Lambda_t^i| + \epsilon_{xi})$ respectively.*

Proof: As shown in Fig. 4, although v_i and v_p cannot realize the potential interference, v_x and v_y can identify the risk as they share the same RoI. To solve this problem, v_x and v_y should check the time stamps of the two piggybacked CAMs, and schedule the vehicle with a tighter deadline in the CAM recovery first. The lower bound of the latency is $\delta \cdot \epsilon_{xi}$ when v_x has the highest priority and is always scheduled first without any delay; the upper bound is $\delta \cdot (|\Lambda_t^i| + \epsilon_{xi})$ when: (1) v_x interferes with all vehicles in Λ_t^i and (2) v_x has the lowest priority and is always scheduled after others. ■

VI. Simulations

A. Simulation Setup

We evaluate the performance of DCB by comparing it with following broadcast schemes.

(1) *Probabilistic Flooding (PF)* [11] : The vehicles forward all received data with a probability of p, which is calculated based on the driving speed and vehicle density;

(2) *Selective Piggybacking (SP)* [9]: Each vehicle selects n closest vehicles and piggybacks the data in the broadcast;

(3) *Greedy Piggybacking (GP)* [15]: Each vehicle broadcasts a request vector to seek for help. The more times a CAM is requested, the higher priority it will be piggybacked.

(4) *Centralized Piggybacking (CP)*: Assume that a central node can obtain the global network information and develop the piggybacking strategy. Other vehicles make the piggybacking following the developed strategy.

As a hypothetical solution, CP is hardly practical in decentralized mobile ad hoc networks. We use CP as a benchmark in the evaluation due to its high performance in reliability, latency and communication overhead [12], and regard it as an optimal solution in the cooperative broadcast.

We run the simulation based on real trace data in Gatech/Vehicular Dataset [4], which was obtained from a series of V2V communication experiments in road tests. In each experiment, a sending vehicle follows a receiving vehicle with a fixed distance, driving in the northwest sector of Atlanta, GA along I-75 between Exit 250 and Exit 255. The sending vehicle makes broadcast every 2 ms, and the receiving vehicle generates a communication record with the same frequency, which contains the GPS data, vehicle speed, receiving status (0 for lost and 1 for received), etc. All records generated in the experiment make up a trace file. Such experiments are conducted repeatedly on the same trajectory with different setting on the follow-up distances, and all generated trace files form the real trace dataset.

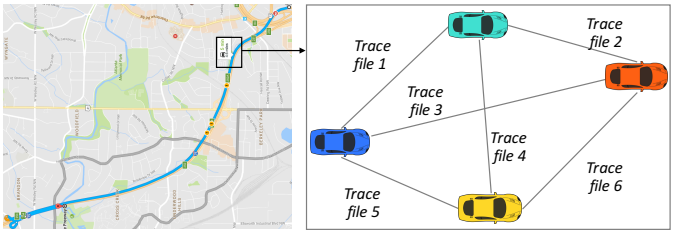

Fig. 5. Extended V2V Communication Trace

As shown in Fig. 5, we extend the experiments to a small-scale VANET simulation. Each channel link between two vehicles is configured with a trace file randomly, which contains all communication records on the whole trajectory with a specific follow-up distance d_f. Whenever a vehicle makes a broadcast, the simulator reads its GPS data, and calculates each receive vehicle's position based on the GPS location and the follow-up distance d_f. Then, the link connectivity between the transmit vehicle to each receive vehicle can be obtained by accessing the communication record corresponding to the calculated position in each allocated trace file. If the receiving status in the record is 1, the link between the transmit vehicle and the receive vehicle is reliable, and the simulator will deliver the data to the receive vehicle; otherwise, the simulator will discard the data to imitate the data loss in the broadcast. The CAM's TTL is set to 100 ms; for SP, GP, CP and DCB, each vehicle can piggyback at most two CAMs in one broadcast.

B. Broadcast Reliability

We use the average PDR to evaluate the broadcast reliability in different schemes. As shown in Fig. 6, DCB has an average PDR around 99.6%, which is very close to CP and barely decreases when the vehicle number increases. GP achieves around 96% in the average PDR and decreases slightly when the vehicle number increases. However, the average PDRs in PF and SP drop significantly against the vehicle number, which fall to 87.5% and 71.5% respectively with 50 vehicles. The reason can be explained as the excessive forwarding in PF and SP has caused a severe data collisions in the broadcast.

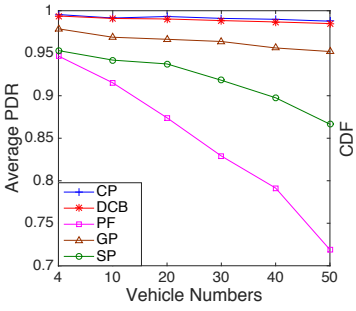

Fig. 6. Broadcast Reliability

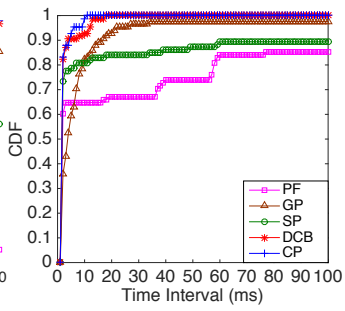

Fig. 7. Piggybacking latency

C. Piggybacking Latency vs. Network Size

In Fig.7, we compare the piggybacking latency in different broadcast schemes, by calculating the Cumulated Distribu-

tion Function (CDF) of time intervals between each CAM's generation and its recovery (received) within a 10-vehicle network. The piggybacking latency of DCB is also very close to CP, where all CAMs can be received in 20 ms, 90% of which are completed in less than 10 ms. The latency in GP is slightly behind, where it takes more than 40 ms to recover the lost CAMs. In contrast, SP only recovers 85% lost CAMs before the deadline as only a limited number of CAMs are piggybacked. PF has the worst performance in piggybacking latency and only recovers 80% lost CAMs before TTL's expiration. The reason is similar to that of the broadcast reliability, where the high collision caused by redundant broadcast deteriorates the CAM recovery.

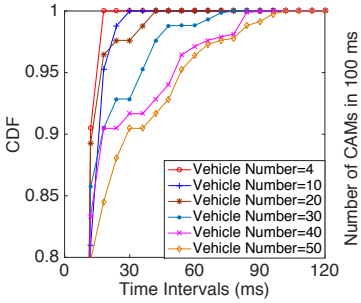

Fig. 8. CDFs *V.S.* Vehicle numbers

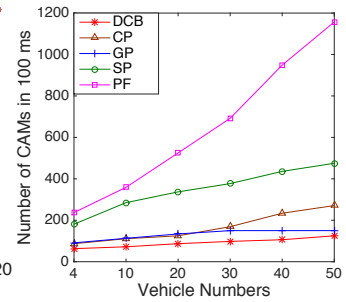

Fig. 9. Network load

Fig.8 illustrates the CDFs of piggybacking latency in DCB under different settings of the network size. Although the piggybacking latency in large networks is slightly longer than that in small networks, most of the CAMs can be received/recovered in 100 ms even in a VANET with 50 vehicles, still satisfying the requirement in the ITS standards and the DSRC standard.

D. Network Load

The average number of CAMs in the broadcast is counted every 100 ms to evaluate the network load. As shown in Fig.9, DCB broadcasts fewer CAMs than CP as the vehicles work in a decentralized model and no frequent strategy distribution is required comparing to CP. GP has a moderate network load which climbs to the maximum point at 150 CAMs /100 ms, where two CAMs are piggybacked at each time slot using TDMA. SP and PF broadcast 475 and 1157 CAMs per 100 ms respectively, which infers that SP and PF have a high data collision ratio in the broadcast, even with broadcast storms in VANETs.

VII. Conclusion

To achieve accident-free driving on the road, we propose a cooperative broadcast scheme (DCB) to enhance the PDR with short latency and lightweight overhead in VANETs. We discuss the link inferring and RPP exploring in the CAM recovery with only local network information, based on which the piggybacking strategies and the broadcast schedulers are developed. We prove that DCB is an optimal solution to maximise the PDR of the CAM broadcast when it is interference-free. We also justify that DCB can solve the interference with little cost of a bounded delay. Simulation results also indicate that it can achieve a higher efficiency comparing to related solutions, which can be extended to other applications with similar requirements on reliability, latency and overhead.

References

[1] K. Alodadi, A. H. Al-Bayatti, and N. Alalwan, "Cooperative Volunteer Protocol to Detect Non-line of Sight Nodes in Vehicular Ad hoc Networks," *Vehicular Communications*, vol. 9, pp. 72–82, 2017.

[2] R. Arshad, M. A. Shahid, D. Khan, and S. H. H. Shah, "Study and Analysis of Shortest Path Algorithms," in *Proceedings of 2nd International Multi-Disciplinary Conference*, vol. 19, 2016, pp. 20–24.

[3] M. Boban and P. M. d'Orey, "Exploring the Practical Limits of Cooperative Awareness in Vehicular Communications," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 3904–3916, 2016.

[4] R. M. Fujimoto, R. Guensler, M. P. Hunter, H. Wu, M. Palekar, J. Lee, and J. Ko, "CRAWDAD Data Set Gatech/Vehicular (v. 2006-03-15)," 2006.

[5] P.-F. Ho and J.-C. Chen, "WiSafe: WiFi Pedestrian Collision Avoidance System," *IEEE Transactions on Vehicular Technology*, vol. 1, pp. 15–15, 2016.

[6] J. Huang, Q. Li, S. Zhong, L. Liu, P. Zhong, J. Wang, and J. Ye, "Synthesizing Existing CSMA and TDMA Based MAC Protocols for VANETs," *Sensors*, vol. 17, pp. 338–355, 2017.

[7] S. Kaul, R. Yates, and M. Gruteser, "On Piggybacking in Vehicular Networks," in *2011 IEEE Global Telecommunications Conference (GLOBECOM 2011)*, vol. 1, 2011, pp. 1–5.

[8] J. B. Kenney, "Dedicated Short-Range Communications (DSRC) standards in the United States," *Proceedings of the IEEE*, vol. 99, no. 7, pp. 1162–1182, 2011.

[9] F. Librino, M. E. Renda, and P. Santi, "Multihop Beaconing Forwarding Strategies in Congested IEEE 802.11p Vehicular Networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 9, pp. 7515–7528, 2016.

[10] W. Lou and J. Wu, "Toward Broadcast Reliability in Mobile Ad Hoc Networks with Double Coverage," *IEEE Transactions on Mobile Computing*, vol. 6, no. 2, pp. 148–163, 2007.

[11] Y. Mylonas, M. Lestas, A. Pitsillides, P. Ioannou, and V. Papadopoulou, "Speed Adaptive Probabilistic Flooding for Vehicular Ad Hoc Networks," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 5, pp. 1973–1990, 2015.

[12] M. Seyedebrahimi, A. Raschellà, F. Bouhafs, M. Mackay, Q. Shi, and M. H. Eiza, "A Centralised Wi-Fi Management Framework for D2D Communications in Dense Wi-Fi Networks," in *2016 IEEE Conference on Standards for Communications and Networking (CSCN)*, 2016, pp. 1–6.

[13] P. Soldati, H. Zhang, and M. Johansson, "Deadline-constrained transmission scheduling and data evacuation in wirelesshart networks," in *Control Conference (ECC), 2009 European*. IEEE, 2009, pp. 4320–4325.

[14] G. Xiao, N. Sun, L. Lv, J. Ma, and Y. Chen, "An HEED-based Study of Cell-clustered Algorithm in Wireless Sensor Network for Energy Efficiency," *Wireless Personal Communications*, vol. 81, no. 1, pp. 373–386, 2015.

[15] G. Xiao, H. Zhang, Z. Huang, and Y. Chen, "Decentralized Cooperative Piggybacking for Reliable Broadcast in the VANET," in *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*. IEEE, 2016, pp. 1–5.

[16] L. Zhang, B. Hassanabadi, and S. Valaee, "Cooperative Positive Orthogonal Code-based Forwarding for Multi-hop Vehicular Networks," *IEEE Transactions on Wireless Communications*, vol. 13, no. 7, pp. 3914–3925, 2014.