

Catch-Up: A Data Aggregation Scheme for VANETs

Bo Yu Jiayu Gong Cheng-Zhong Xu
Department of Electrical and Computer Engineering
Wayne State University, Detroit, Michigan, USA
{boyu,jygong,czxu}@wayne.edu

ABSTRACT

In-network data aggregation is a useful technique to reduce redundant data and improve communication efficiency. One challenge in data aggregation is how reports can be routed to the same node so that the reports can be merged. Most of existing approaches rely on maintaining a routing structure to achieve this purpose. However, these approaches are not applicable to the mobile environment of Vehicular Ad hoc Networks (VANETs). In this paper, we design a cooperative model to facilitate the aggregation of adjacent traffic reports. The basic idea behind this work is that we can adaptively change the forwarding delay of individual reports in a manner that a report can have a better chance to meet other reports. The decision is made distributedly by each vehicle based on local observations. Actually, our scheme is also a tradeoff between communication overhead and propagation delay. Simulation results based on realistic map data and traffic models demonstrate that our scheme can effectively reduce communication overhead with acceptable delay.

Categories and Subject Descriptors

H.4.3 [Information Systems Applications]: Communications Applications; C.2.1 [Computer-Communication Networks]: Network Architecture and Design—*Network communications, Wireless communication*

General Terms

Algorithms, Design, Performance

Keywords

Vehicular networks, Data aggregation, Routing

1. INTRODUCTION

Vehicular Ad hoc Networks have been regarded as an emerging and promising field in both industry and academia.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

VANET'08, September 15, 2008, San Francisco, California, USA.
Copyright 2008 ACM 978-1-60558-191-0/08/09 ...\$5.00.

It has the potential to improve the efficiency and safety of future highway systems. One good example is traffic congestion detection. Once a vehicle detects the number of its neighboring vehicles exceeds a certain limit, it will broadcast a warning to the vehicles following behind. This warning could travel a rather long distance so that vehicles, possibly several kilometers away, can have enough time to choose an alternate route. Since each vehicle in VANETs is able to detect traffic conditions and generate traffic reports distributedly and independently, a great amount of redundant reports can be generated and forwarded in the network, consuming considerable bandwidth.

Bandwidth limitation and channel collision could be serious issues for VANET communications. Dedicated Short Range Communications (DSRC) [1] is a physical and MAC layer communication protocol for VANETs. In DSRC, the communication range (or the interference range) can be up to 1000m, while the data rate is only 6 to 27 MBPS [1]. A growing concern is bandwidth limitation and channel collision. Imagine the rush hour traffic, it is very likely that there are hundreds of vehicles within the range of 1000m, all of which compete and interfere with each other for the limited channel resource. Moreover, the channel is shared with a variety of applications including safety applications. Bandwidth limitation and channel collision may even impair the availability of safety applications.

Data aggregation could be a potential approach to relieve this problem. It consists of a variety of adaptive methods which can merge information from various data sources into a set of organized and refined information. The process of data aggregation can be performed in-network so that communication overhead can be effectively reduced soon after redundant information is generated. It can be applied to any distributed information collection application, such as sensor networks and vehicular networks.

Existing data aggregation techniques cannot meet the requirements of VANETs environments. In sensor networks, researchers have proposed a number of structure-based aggregation schemes [9][8][10][12][13], which need to establish a fixed routing structure in advance. Certainly, they are not applicable to dynamic VANETs environments. Fan et al. [6] proposed a structure-free aggregation protocol based on randomized waiting. However, this probabilistic approach cannot guarantee aggregation of packets from a single event source. Later, the authors presented a semi-structured approach [7] to improve the aggregation degree, but this approach still needs to maintain a routing structure. Several VANET projects, such as Self-Organizing Traffic Informa-

tion System(SOTIS) [20][19] and TrafficView [15], are also related to data aggregation. However, their simplest approach, periodical rebroadcasting, could further increase the chance of channel collision [18].

This paper is motivated by two unique features of VANETs. First, unconstrained by power supply, the powerful on-board computer can eavesdrop the channel and log every channel transmission into its storage. Later, when a traffic report arrives at this node, previous local observations can facilitate aggregation decisions. Second, traffic information is not delay sensitive, and traffic conditions don't change a lot within a short period. Even a delay of tens of seconds is still acceptable. This delay-insensitive property provides us an opportunity to trade off increased delay for reduced communication overhead.

In this paper, we propose a data aggregation scheme for VANETs, namely, Catch-Up. First, we define the aggregation problem in VANETs, which is quite different from that in sensor networks. The basic idea in this scheme is to insert a delay before forwarding a report to the next hop. However, our scheme makes this delay more controllable in order to increase the probability that a report can be merged with reports ahead or reports behind. Intelligent delay control policies are made based on local observations of individual vehicles. We design a future reward model to define the benefits of different delay-control policies, and then we establish a decision tree to help a vehicle to choose an optimal policy from the perspective of long-term rewards. The Catch-Up scheme has a desirable property that for a given time frame and a given road section, the scheme can guarantee to aggregate all traffic reports into a single report within a certain distance. This property is particularly appealing for the facts that traffic reports could travel tens of kilometers, and a highly aggregated report could save a lot in communication overhead. We conducted simulation experiments with NS2 [3] and GrooveNet [2][14], which demonstrate the effectiveness of the scheme we developed.

The remainder of this paper is organized as follows. Section 2 introduces related work. Section 3 presents the system model for our scheme. Section 4 describes the details of our aggregation scheme. Section 5 discusses two properties of our scheme. Section 6 presents simulation results. Section 7 concludes the whole paper.

2. RELATED WORK

In mobile ad hoc networks, data aggregation is studied in two main aspects: routing-related and data-related aspects. The routing-related aspect focuses on routing problems such as when and where two (or more) packets can meet each other and then be aggregated, while the data-related aspect focuses on the coding, calculation, and compression of aggregatable data from multiple packets. In this paper, we only consider the routing-related aggregation in VANETs.

In the routing-related aspect, great efforts have been made in ad hoc networks in the past few years. We classify the existing data aggregation schemes in the literature into three categories: 1) *Structured Aggregation*. In this category of approaches [9][8][10][12][13], a fixed forwarding structure, such as forwarding tree, is established in advance, and then, packets can be aggregated at the tree forks. These fixed-structure approaches can meet the requirement of simple queries in sensor networks, but they incur too much communication overhead in constructing and maintaining tree structures

and are not applicable to dynamic environments; 2) *Structureless Aggregation*. Boulis et al. [4] proposed a flooding-based periodic aggregation scheme, but his scheme is constrained to simple exemplary queries such as MIN MAX. Fan et al. [6] proposed a structure-free data aggregation protocol. However, his approach is a probabilistic approach and can not guarantee optimal aggregation of packets from a single event source. In VANETs, two projects, SOTIS [20][19] and TrafficView [15], also implemented simple data aggregation mechanisms. In these schemes, traffic information is collected, aggregated, and rebroadcasted by each node in a periodical and stochastic manner. However, periodical broadcast could dramatically increase communication overhead, potentially impairing the functionality of other applications in VANETs, as pointed out in [17][18]. 3) *Semi-structured Aggregation*. Fan et al. later proposed a semi-structure aggregation scheme [7], intended to combine the benefits of structured and structureless aggregation. However, actually, this approach introduces more complicated aggregation structures, which are not suitable for dynamic environments.

In the data-related aspect, the work focuses on data representation and process in data aggregation. In TAG [12], the authors studied the properties of aggregation functions (MAX, MIN, AVERAGE, etc.). Nath [16] and Considine [5] use probabilistic counting to improve the resilience of data aggregation in sensor networks. Lochert et al. [11] also applied probabilistic counting to a duplicate-insensitive aggregation scheme for VANETs.

Based on above analysis, we can find that existing approaches cannot meet the application requirements in the dynamic environment of VANETs, and we are motivated to study the problem of communication-efficiency in data aggregation in VANETs.

3. SYSTEM MODEL

In this section, we introduce the system model which is applicable to our aggregation scheme.

Aggregatable Reports. A physical *event* can be a change in traffic conditions. For example, from an individual vehicle's point of view, a change in the number of neighboring vehicles can be an event. The generation of reports is event-driven, and any vehicle traveling on a road can generate a report. If there are no changes in traffic conditions, no reports need to be generated. We divide roads into road sections and divide the time axis into time frames. If two reports are generated from the same road section and from the same time frame, they are called two *aggregatable reports*. We define *Event Frame* as a tuple $\langle p_1, p_2, t_1, t_2 \rangle$, where p_1 and p_2 are the starting and ending positions of a road section and t_1 and t_2 are the starting and ending time of a time frame. In other words, two aggregatable reports originate from the same event frame.

However, redundant event reports may be generated for two reasons. 1) The length of a road section can be longer than a vehicle's signal range. Two vehicles, out of the signal range of each other, may generate two reports, which represent two traffic conditions at two locations respectively. 2) A vehicle may miss a report from its immediate neighbor due to packet loss and then generate a second redundant one. For these reasons, in-network data aggregation is necessary to reduce redundant reports and improve communication efficiency.

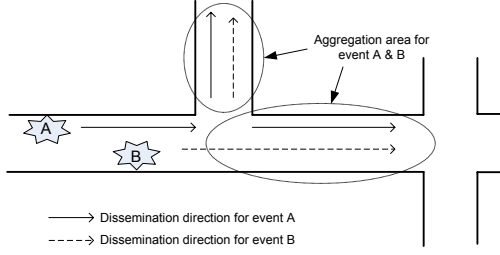


Figure 1: Dissemination tree. If data from different events are propagating in the same direction, they can be aggregated for reducing communication overhead.

Our aggregation goal is to generate a single overview report for all aggregatable events from an event frame $\langle p_1, p_2, t_1, t_2 \rangle$. Of course, if no events happen in an event frame, no overview report is generated.

Data Dissemination. As soon as a report is generated for a local event, it will be disseminated to any traffic (vehicles) coming toward this road section. Reports have the same predefined maximal dissemination distance. On a straight road, reports can be propagated with the help of data forwarding protocols in VANETs (such as MDDV [21]). At road intersections, a report is duplicated, and each copy goes to one branch. We can have a dissemination tree rooted at the vehicle which generates the report. We can have multiple dissemination trees rooted at different vehicles in the same road section, and our work is to merge these tree branches. These trees are partially overlapped and aggregation can only happen at the overlapped sections, as shown in Fig.1.

Aggregation. There are two aspects for aggregation: data-related and routing-related. In the data-related aspect, two or more reports can be merged into one report with aggregation functions such as MAX, MIN, AVG, or the probabilistic aggregation [11]. We also assume that the aggregation operation is not reversible. That is, once two or more reports are aggregated, they can no longer be split. The routing-related aspect deals with how two aggregatable reports can meet each other at the same node (vehicle) and at the same time, and that's exactly our task in this paper.

4. DISTRIBUTED COORDINATED AGGREGATION

The motivation behind our design is that, in order to make reports meet each other, we must have some reports go faster and others go slower. We can control the forwarding speed by applying a delay before forwarding a report to the next hop. Then, we need to design a distributed scheme to help individual vehicles control forwarding delays in an intelligent manner that reports have a better chance to meet each other.

The uniqueness of our problem make it different from those problems in existing decision making algorithms. First of all, each vehicle can only obtain a partial observation of the world, and the observation could be incomplete and outdated. Second, this is also a distributed cooperation problem. Vehicles need to make local decisions to achieve better global performance (reduced total communication overhead). Finally, we cannot introduce much extra communication overhead for node coordination. In other words, if

we allow vehicles to communicate with each other for better cooperation, this may counteracts the original purpose of reducing communication overhead.

In Section 4.1, we introduce a global Markov Decision Process (MDP) model to describe the problem from a global perspective. In Section 4.2, we introduce a distributed MDP model which is applicable to our problem. The model explains the relation among internal states, local observations, and actions. In Section 4.3, we define the reward function to evaluate potential future rewards, and we also introduce the concept of virtual report to facilitate the cooperation among vehicles. In Section 4.4, we propose to use the decision tree to find an optimal policy. In Section 4.5, we discuss several practical issues when implementing our scheme in real VANET environments.

4.1 Global MDP Model

We first introduce a global Markov Decision Process (MDP) model for better understanding of the problem from the global perspective.

The global MDP model can be described as a tuple $\langle S, A, U, R_g, T \rangle$, where

- S is a finite set of the world states;
- A is a finite set of actions a vehicle can perform, and in our scheme we define $A = \{WALK, RUN\}$;
- U is the set of all possible action vectors that can be performed by all vehicles in the system, and $U = A^n$, where n is the total number of vehicles;
- $R_g : S \times A \rightarrow \mathbb{R}$, is the global reward function;
- $T : S \times U \rightarrow S$, is the global transition function.

A world state can be regarded as a vector of states of all reports in the system, i.e., $s = (r^1, r^2, \dots, r^n)$, $s \in S$. We define two actions for a vehicle to process each individual report: WALK and RUN. The purpose of WALK and RUN is to insert different delays before the vehicle forwards the report to the next hop. The RUN action applies a short delay and the WALK applies a longer delay. WALK is intended to let the vehicle wait longer for future arrivals of aggregatable reports, while RUN is intended to forward the report quickly in the hope that the report can be merged with reports ahead.

We define v_{WALK} and v_{RUN} as the physical speed (in meters per second) at which a report can be forwarded among vehicles. In Section 4.5, we introduce how this is implemented in practice.

Actually, it is infeasible for individual vehicles to obtain global state s , global reward R , as well as the transition model $T(s, u, s')$, and it is infeasible to implement such calculation in practice. Based on this global model, we design a distributed MDP model for each individual vehicle.

4.2 Distributed MDP Model

In this subsection, we introduce a distributed MDP model. This model is designed for individual vehicles and tries to improve global performance through distributed cooperation. We define the model and introduce the main components in the model.

Our distributed MDP model can be described as a tuple $\langle S, A, U, T, \Omega, R_i, B, \Pi \rangle$, where

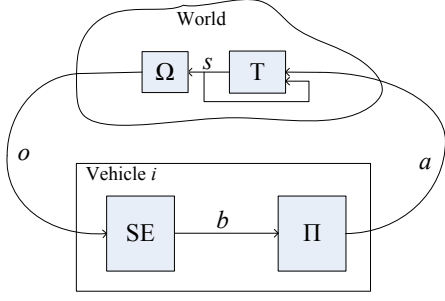


Figure 2: Distributed MDP model.

- S, U, A, T have the same meanings as them in the global model;
- Ω is a finite set of local observations a vehicle could obtain;
- B is a finite set of internal states, which is an incomplete estimation of the world state.
- $R_t: B \rightarrow \mathbb{R}$, is the expected reward function;
- $\Pi: B \rightarrow A$, is the policy, which establishes a mapping from current internal states to the next action.

In general, a vehicle interact with its environment via observation and action, as depicted in Fig. 2. The outside world can provide each vehicle with an incomplete observation. Each vehicle has two components: State Estimator(SE) and Decision Maker (Π). The state estimator uses local observations as input, and then calculates an estimate of the world state, which is represented as internal state B . Then, the decision maker uses the internal state to estimate potential future rewards, and finally make a decision a , $a \in A$.

The observation is defined as a set of observed reports, $o = (o^1, \dots, o^i, \dots, o^q)$, where o^i is a tuple $\langle r, t_{time_stamp}, v_{rcv} \rangle$. Actually, each observation is an overheard report, which was transmitted in the channel. r is the report; t_{time_stamp} is the time stamp when this report was overheard; v_{rcv} is the next hop receiver (vehicle) of this report.

The internal state is defined as a set of estimated report positions, $b = (b(r^1), \dots, b(r^i), \dots, b(r^q))$, where $b(r^i)$ is a probabilistic distribution of report r^i 's position. Since we know the minimal and maximal speeds of a report (v_{WALK} and v_{RUN}), we can calculate the position range of report r^i at a given time t , $[p(v_{WALK}, t), p(v_{RUN}, t)]$. For simplicity, we can assume that the position of report r^i is uniformly distributed over range $[p(v_{WALK}, t), p(v_{RUN}, t)]$.

4.3 Expected Future Reward and Coordination

In this subsection, we first define the expected future reward function for an arbitrary policy, and then we introduce the concept of virtual report in order to improve cooperation among vehicles.

We can use π to represent a complete policy from now to the future, i.e. a sequence of actions. Then, we can use the reward function to evaluate the expected reward value of this policy over current internal belief state b . The local

reward function can be written as

$$R_b(\pi) = E\left(\sum_{t=1}^{\infty} \gamma^t w_t\right), \quad (1)$$

where γ is a discount factor for future rewards, and w_t , $t \geq 1$, is a future reward based on policy π . In our model, if no aggregation happens at time t for the current report, then the reward, w_t , is ZERO; otherwise, the reward is the saved communication overhead for two merged reports, described as follows:

$$w_t(r^i, r^j) = \frac{\text{cost}(r^i) + \text{cost}(r^j) - \text{cost}(\text{agg}(r^i, r^j))}{2}, \quad (2)$$

where $\text{cost}(r^i)$ calculates the rest communication overhead for report r^i , and $\text{agg}(\cdot)$ is the merge function, which merges two reports into one.

Since we know the predefined maximal propagation distance for each report, we can calculate the rest propagation distance for each report. We can also approximately regard the communication overhead function $\text{cost}(\cdot)$ as a function of propagation distance.

Further, given a policy $\pi = (a_1, a_2, \dots, a_t, \dots)$, report r^i can calculate a sequence of its future positions, $(p_1, p_2, p_3, \dots, p_t, \dots)$. Based on the current vehicle's internal states, the vehicle can calculate the expected reward if report r^i is merged with report r^j at time t :

$$b(r^j, p_t) w_t(r^i, r^j), \quad (3)$$

where $b(r^j, p_t)$ is the probability that report r^j is at position p_t at time t . Finally, we get the total average reward for policy π based on current belief states:

$$R_b(\pi) = \sum_{t \geq 1} \gamma^t \frac{\sum_{1 \leq j \leq q} b(r^j, p_t) w_t(r^i, r^j)}{q}, \quad (4)$$

where q is the number of overheard reports.

We can regard our problem as a cooperative game, where some vehicles need to slow down, while the others need to speed up. A vehicle should have both reports ahead and possible reports behind considered so that it can make a better decision. Based on this motivation, we introduce a concept of *virtual report* to improve the cooperation among vehicles. The virtual report is an estimation of a possible report behind.

The virtual report is assumed to be trying to catch up with the current vehicle. Then, the problem is what's the expected reward if the current report slows down (WALK) and waits for the follower.

First we define a report arrival model as follows. The arrival event of a report is described as a random variable A . Here, "arrival" means that a vehicle receives a report. Let $\text{arr}(t)$ be the density function of random variable A . In queuing theory, the most common choice for A is the exponential distribution, so we can describe the density function as

$$\text{arr}(t) = \lambda e^{-\lambda t},$$

where λ is the arrival rate.

The practical meaning of λ can be the event generation rate of a road section. Since each vehicle maintains a traffic-state list of the road sections ahead, a vehicle can calculate λ with the history data in its memory.

Based on this arrival model, we can insert an internal state, $b(r^0)$, into the vehicle's internal states, i.e. $b = (b(r^0), b(r^1), \dots, b(r^q))$. Here, r^0 represents the virtual report.

We suppose report r^i is on the current vehicle. Given policy π , we can calculate report r^i 's future position p_t at time t . We suppose that at time t_1 the virtual report arrives at the current vehicle, and at time t_2 the virtual report catches up with report r^i . Through simple deductions, we can easily obtain

$$t_1 = t_2 - \frac{p_{t_2} - p_0}{v_{RUN}}, \quad (5)$$

where p_0 is the current vehicle's position and p_{t_2} is the future position where the virtual report catches up with report r^i .

Then, the probability that report r^i and the virtual report can meet at time t is

$$b(r^0, t) = \int_{t_2 \in [t, t+1]} \text{arr}(t_2 - \frac{p_{t_2} - p_0}{v_{RUN}}) dt_2. \quad (6)$$

We can further change the form of this equation:

$$b(r^0, t) = \int_{t_2 \in [t, t+1]} \text{arr}(\frac{v_{RUN} - \bar{v}}{v_{RUN}} t_2) dt_2, \quad (7)$$

where \bar{v} is the average forwarding speed of report r^i between p_0 and p_{t_2} , which can be calculated based on the policy π . It's evident that the more closer to v_{RUN} the average speed \bar{v} is, the less possibility that report r^i could encounter the virtual report. In other words, the equation proves that the policy π for report r^i will affect the possibility of encountering the virtual report.

Finally, we establish the belief state for both overheard reports (local observations) and the virtual report. The next step is to determine the optimal policy π based on the belief states.

4.4 Decision Trees

In this section, we use a decision tree to find the optimal forwarding policy π over the current vehicle's belief state.

Fig. 3 is a good example of decision tree. As shown in the figure, the decision tree is a binary tree. At each node, we have two choices, WALK and RUN, which lead us to the left child node or the right. A path from the root node to a leaf node represents a possible policy π .

At a node in a given path, we can calculate the corresponding node reward :

$$w_{node}(t) = \frac{\sum_{1 \leq j \leq q} b(r^j, p_t) w_t(r^i, r^j)}{q}. \quad (8)$$

The path reward can be obtained as follows:

$$w_{path}(\pi) = \sum_t \gamma^t w_{node}(t), \quad (9)$$

where γ is the future reward discount. After we explore all possible paths, we can obtain an optimal path, i.e. optimal policy π^* .

In Fig. 3, we can find an example of optimal policy. In this policy, report r^i first chooses WALK; at time t_1 , report r^i could be aggregated with the virtual report r^0 with reward w_1 ; after that, report r^i changes to RUN; at time t_3 , report r^i could be aggregated with report r^j with reward w_3 . Of course, report r^i is not really merged with the virtual report. This is only intended to encourage report r^i to cooperate in some conditions.

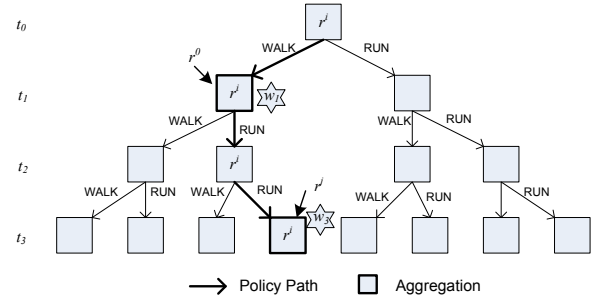


Figure 3: Decision tree for potential forwarding policies.

After the vehicle obtains the optimal policy π for report r^i , the vehicle will execute the first action in the policy, i.e., WALK or RUN to the next hop. Since we don't want to increase the communication overhead, report will be forwarded without a decision tree or anything else. After receiving the report, the next-hop vehicle will set up a new decision tree based on its local observations.

The computational complexity of this search algorithm is $O(2^n)$, where n is the height of the tree. We can add some constraints to the search path to reduce the complexity. For example, if at most k WALK actions are allowed in one policy, then through computational complexity analysis, the complexity is reduced to $O(n^k)$ (k is a constant). Since there is only one virtual report behind the current vehicle, k can be used to serve as a parameter to control the degree of cooperation. The bigger k is, the more willing a vehicle is to cooperate with reports behind it.

4.5 Other Issues

In this subsection, we discuss several other issues in our aggregation scheme.

In order to control report forwarding speed (WALK or RUN), a vehicle can insert an intentional delay, Δt , before it forwards the report to the next host. Since vehicles are equipped with GPS devices, and their time can be synchronized, it's easy to calculate an appropriate delay Δt to match the designated speed of a report. In this way, v_{WALK} or v_{RUN} can be set to a physical speed in meters per second.

Since the main task of our scheme is only to control the forwarding delay, our scheme runs very well with existing VANET forwarding protocols, such as, MDDV [21], VADD [22], etc. A vehicle chooses a next-hop candidate from its neighbor list, and forwards a report to the next hop. The only requirement of our scheme is that each vehicle keeps eavesdropping the channel and logs all transmissions in the channel. Actually, MDDV [21] already requires that the forwarding is broadcast-based (each neighbor is supposed to receive the packet being forwarded) in order to increase system resilience.

Since each vehicle independently chooses a next-hop forwarding candidate, it's possible that a report may overtake another report, and in this case, these two aggregatable reports are still in two vehicles. For example, in Fig.4, vehicle B is holding report r^1 and vehicle A forwards report r^2 to vehicle C . However, meantime, vehicle B also receives report r^2 by eavesdropping, so vehicle B performs the aggregation operation and generates the aggregated report $r^1 + r^2$. Then,

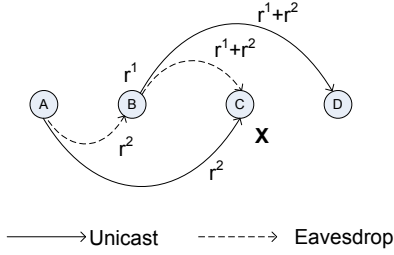


Figure 4: Forwarding and aggregation.

the next step is to eliminate the redundant copy of report r^2 .

We can use *bloom filter* (or its variants) to eliminate the redundant report copies. Bloom filter is a space-efficient probabilistic data structure that is usually used to test whether an element is a member of a set. Then we can have each report attached with a short bloom filter table, which includes information of the set of previous reports which are merged into the current report. Later, a vehicle just needs to look up the table to make a judgement. For example, in Fig.4, vehicle C is holding report r^2 and later vehicle C receives report $r^1 + r^2$ by eavesdropping. By checking the bloom filter table in report $r^1 + r^2$, it is convinced that report r^2 is already merged in report $r^1 + r^2$. In this way, vehicle C just discards report r^2 . Of course, there is always a possibility that vehicle C may miss report $r^1 + r^2$ due to varying channel conditions. In this case, report r^2 will later be discarded by any vehicle which receives report $r^1 + r^2$ by eavesdropping.

In addition, redundant report copies may also appear due to multipath forwarding. For example, when a report arrives at an intersection, we have to duplicate this report and send a copy to each direction. However, somehow two copies may meet each other again at another intersection due to the special road topology. In this situation, we can also use the bloom filter method to eliminate the redundant copies.

5. ANALYSIS

In this section, we explore several properties of our aggregation scheme.

Definition 1. Given an event frame $\langle p_1, p_2, t_1, t_2 \rangle$, convergence time is the time period from time t_1 to the moment when all reports from a single event frame are merged into one report (an overview report for this event frame), and convergence distance is the distance between the road section and the position where the final overview report is born.

Based on above two definitions, our scheme have two properties. To simplify the analysis complexity, we assume the communication channel is ideal. That is, no extra latency or packet loss is caused due to the channel reason, and reports can be forwarded freely over the underlying communication networks.

Proposition 1. Given an event frame $\langle p_1, p_2, t_1, t_2 \rangle$, the convergence time has an upper bound:

$$T_{cvg} \leq \frac{(p_2 - p_1)v_{RUN} + (t_2 - t_1)v_{RUN}v_{WALK}}{v_{WALK}(v_{RUN} - v_{WALK})},$$

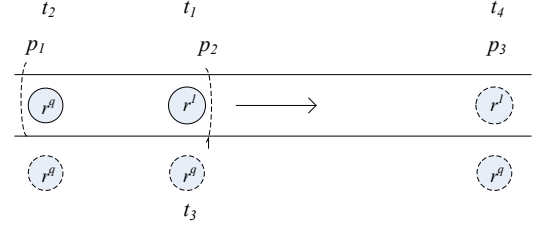


Figure 5: Convergence time and distance.

and the convergence distance also has an upper bound:

$$D_{cvg} \leq \frac{(p_2 - p_1)v_{RUN} + (t_2 - t_1)v_{RUN}v_{WALK}}{v_{WALK}(v_{RUN} - v_{WALK})} \cdot v_{WALK}.$$

PROOF. Consider an extreme example shown in Fig. 5. This event frame generates a sequence of reports (r^1, \dots, r^q) . We suppose that at time t_1 report r^1 is generated at position p_2 ; at time t_2 report r^q is generated at position p_1 ; all other reports are generated between time t_1 and time t_2 . Therefore, we can regard report r^1 as the far-ahead report, and report r^q as the far-behind report, and then we only need to consider the behavior of reports r^1 and r^q to determine the convergence distance and convergence time.

We consider the worst case, in which all reports, except report r^q , are born close to position p_2 . Thus, when report r^q goes through the road section (p_1, p_2) , it cannot find any observation, so it just WALKs in the hope of some reports existing behind it. When report r^q arrives at position p_2 , it get all observations of other reports and begin to catch up, RUN. On the other side, since there is no other report from the same event frame ahead of report r^1 , report r^1 would always keep WALKing.

We claim this is a worst case, because, first, reports r^q and r^1 have the maximal difference in both time and space; in addition, the last report r^q doesn't have a choice except WALKing slowly which further extends the convergence time to the extreme value. Based on this reasoning, we can get the above inequalities through simple algebraic calculations. \square

From Proposition 1, we observe that when the lifetime of a report has exceeded the convergence time upper bound, the report doesn't need to WALK slowly any more. Instead, the report can move at the fastest speed. Here, we define a new action *FLY*. With this action, when a report arrives at a vehicle, it will be immediately forwarded to the next hop with no deliberate delay. The only delay is MAC-layer transmission delay. We can calculate v_{FLY} given MAC-layer delay and transmission range. We can easily derive the following proposition.

Proposition 2. The total latency for a report to be propagated to the maximal dissemination range D_{dism} has an upper bound:

$$T_{dism} \leq \max(T_{cvg}) + \frac{D_{dism} - \max(D_{cvg})}{v_{FLY}}.$$

For example, we set $t_1 = 0s$, $t_2 = 30s$, $p_1 = 0m$, $p_2 = 500m$, $v_{WALK} = 100m/s$, $v_{RUN} = 1000m/s$, $D_{dism} =$

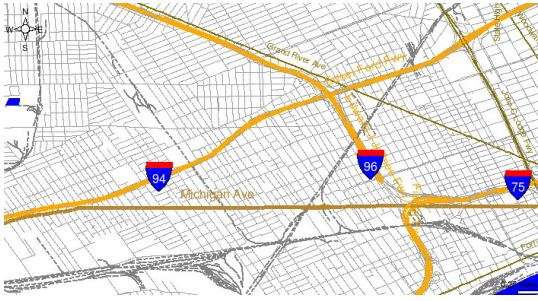


Figure 6: Simulation map.

Table 1: Simulation Configurations

System Parameter	Value
Road Length	10km
Vehicle Number	300
Road Section Length	1km
Time Frame	30s
Communication Range	250m
MAC layer	802.11
Mobility Model	StreetSpeedModel[2]
Trip Model	DijkstraTripModel[2]

10km, $\Delta t_{MAC} = 50ms$, $transmission_range = 200m$. We can get $T_{cvg} \leq 44.4s$, $D_{cvg} \leq 4.4km$, $T_{dism} \leq 45.4s$. We find that $D_{cvg} \ll D_{dism}$, which shows that the communication overhead after the convergence point could be effectively reduced, because all reports from one event frame are aggregated together.

In this section, for the simplicity of analysis, we assume that there are heavy traffic on the roads and the network is connected. If the network is partitioned, the total dissemination latency is very difficult to model. However, our scheme still can facilitate the aggregation operations within a connected part of the network.

6. SIMULATION EVALUATION

In this section, we use simulations to evaluate our aggregation scheme and compare our approach with Randomized Waiting [6].

6.1 Simulation Design

In our experiment, we simulate a scenario of morning rush hour in Detroit. 300 vehicles are rushing on I94 (Interstate Highway) in both directions. However, traffic congestion happens at one section of I94, which is close to Woodward Avenue. This traffic congestion information is propagated and aggregated along I94 so that vehicles in I94, 10km away from the congestion section, will know the congestion ahead. The map of the scenario is shown in Fig. 6. Due to the size of the scenario and the simulator's constraints, we mainly focus on how traffic updates are propagated along a 10km-long distance in I94.

Our simulation is based on NS2 [3] and GrooveNet [2][14]. GrooveNet is a VANET simulator, which provides a variety of useful models for VANET simulations, such as mobility models, trip models, etc. On the other hand, NS2 presents a lot of well-developed low-layer protocols as well as easy programming interfaces. We think a combination of these

tools would be a good choice. We first used GrooveNet to design the simulation scenario and generate mobility trace files, and then we used NS2 to load these trace files and run our aggregation protocol. Our protocol is implemented on NS2.

The system parameters used in our simulation are shown in Tab. 1. The road section length is set to 1km, and the aggregation time frame is set to 30s, meaning that we are expecting an overview report of a 1 km-long road section every 30 seconds. Of course, if no events happen in a 30s time frame, no overview reports are generated. The communication range is set to 250m, which means that even when vehicles are in the same event frame, they may not communicate with each other directly so that aggregation protocols can make their contributions. We run the simulation for 10 minutes and have all the packet transmissions logged.

We compared our scheme with Randomized Waiting [6]. To our knowledge, this is the only existing aggregation technique that could be adapted to highly mobile environments. Since Randomized Waiting has a different application background, we slightly change it in order to fit it into the VANET environments. We suppose that, with this scheme, each vehicle applies a randomized delay before forwarding a report to the next hop. During this delay, there is a probability that the vehicle can receive other reports for aggregation. This is a probabilistic approach where reports can be probabilistically aggregated with other reports behind.

To make a fair comparison, we scale these two protocols to the same total delay level, that is, the total end-to-end delay in 10km. The motivation of doing this is to scale these two protocols to the same drivers' expectations. For example, a driver hopes that the traffic updates 10km away are fresh enough within 30s. For our scheme, given a set of system parameters, we can calculate the total delay of our protocol, T_{dism} , using Proposition 2 in Section 5. We assume that the physical distance of each hop is about 200m, so 10km needs about 50 hops. Given a total delay, we can obtain an average one-hop delay, $\Delta t = T_{dism}/50$, for each hop. For Randomized Waiting, we choose a random number in $(0, 2\Delta t)$ as the delay in each hop. In this way, we scale these two protocols to the same users' expectation, and then evaluate their communication overhead to the system.

6.2 Simulation Results

Our simulation focuses on two important metrics for aggregation protocols: communication overhead and delay.

Fig. 7 presents the relation between propagation distance and the number of packets pass that location. *CATCHUP* (100,1000) represents our protocol with a WALK speed at 100m/s and a RUN speed at 1000m/s; *CATCHUP*(200,2000) has corresponding meanings. All protocols show a reduced report number with increased distance, which proves the contribution of aggregation operations. However, after the point at about 3km, our aggregation protocol shows apparent better performance, reducing as much as 50% reports than Randomized Waiting. The majority of aggregation operations in our protocol are performed close to the road section, because as soon as reports find evidence of other reports, they will make an optimal decision to catch up them, whereas Randomized Waiting cannot control its waiting time, and its aggregation operations are performed probabilistically throughout the propagation trip. We can see that the RW curve in the figure goes down more smoothly.

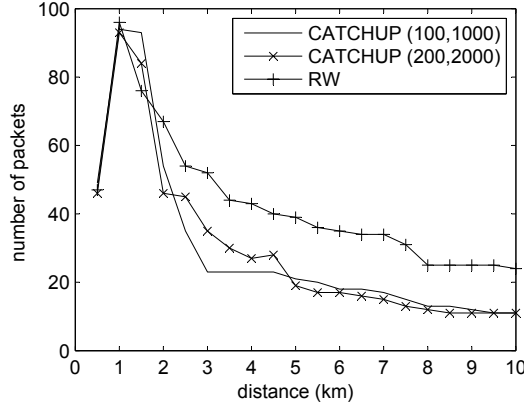


Figure 7: Number of packets vs. distance.

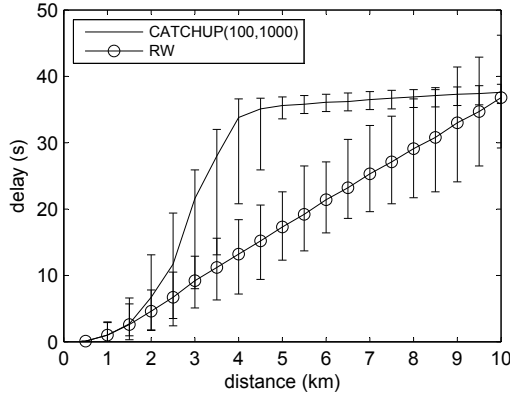


Figure 8: Delay vs. distance.

In our simulation, we also observed that a number of packets were dropped during propagation, and this is the reason which makes the curves of our protocol continue going down slightly even after the maximal convergence distance (The theoretical maximal convergence distance is 4.4km for CATCHUP(100,1000), and 7.6km for CATCHUP(200,2000)).

Table 2 shows the total transmissions in our simulation. It's evident that our aggregation scheme outperforms Randomized Waiting by a significant amount in total communication overhead. Table 2 also shows the theoretical maximal convergence distance and convergence time for CATCHUP(100,1000) and CATCHUP(200,2000). These theoretical results are worst-case upper bounds. From Fig. 7, we can estimate that the actual convergence distance is about 3km for CATCHUP(100,1000), and about 5km for CATCHUP(200,2000), which are much less than the theoretical values.

Fig. 8 shows the relation between delay and propagation distance. The curves describe the average delay observed at a given location, and at each location point, a jittering bar is presented to show the observed minimal delay and maximal delay. We can observe this figure together with Fig. 7. It's easy to find that the majority delay incurred by our protocol is bound with aggregation operations before the convergence time. Since after the convergence time, reports in our protocol enter FLY mode, the later delay is almost insignificant. Also, the delay of Randomized Waiting

is bound with its aggregation operations. However, since its aggregation operations are performed all over the trip, the delay is proportional to the distance. The performance of our protocol has nothing to do with propagation distance. Even when it applies to a longer distance propagation, the majority of aggregation operations are still focused on the road section close to the event source and the total delay time is up bound by a computable limit.

Please note that traffic information is not delay sensitive, and even tens of seconds of delay are still acceptable. Although it seems that Randomized Waiting causes less delay than our scheme, our scheme can reduce the communication overhead significantly, which is more important to bandwidth-limited VANETs.

In general, all existing aggregation approaches are trying to achieve a better tradeoff between delay and communication overheads. Structure-based aggregation protocols, such as [12], make a forwarding schedule in advance to achieve temporal convergence. Randomized Waiting [6] uses random delays to remove the requirements of routing structures and to apply to the mobile environments. Our protocol further makes the delay time more controllable based on heuristic local observations. Randomized Waiting uses random delay to 'wait' for reports behind. But our protocol not only 'waits' for reports behind, but also 'catches up' with reports ahead.

7. CONCLUSION

In this paper, we study the aggregation problem in highly mobile environments, VANETs. We propose an aggregation scheme, Catch-Up, in which traffic reports can 'catch up' other related reports based on heuristic local observations. Our contribution mainly focuses on introducing the idea of adaptively controlling forwarding delay to facilitate in-network data aggregation. However, extended work remains to be done to implement this idea in real applications. For example, one problem is what are the optimal delay values for WALK and RUN for our scheme; also, the performance of our scheme in partitioned networks is to be evaluated and improved.

8. ACKNOWLEDGEMENT

We would like to thank the anonymous reviewers for their constructive comments and suggestions. This research was supported in part by U.S. NSF grants CCF-0611750, DMS-0624849, CNS-0702488, and CRI-0708232.

9. REFERENCES

- [1] Dedicated short range communications project. <http://www.leearmstrong.com/DSRC/DSRCHomeset.htm>.
- [2] Groovenet project. <http://www.seas.upenn.edu/rahulm/Research/GrooveNet/>.
- [3] The network simulator - ns-2. <http://www.isi.edu/nsnam/ns/>.
- [4] A. Boulis, S. Ganeriwal, and M. B. Srivastava. Aggregation in sensor networks: an energy-accuracy trade-off. *Ad Hoc Networks*, 1(2-3):317–331, 2003.
- [5] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate aggregation techniques for sensor databases. In *Proceedings of IEEE ICDE*, 2004.

Table 2: Total Transmissions and Convergence Distance

Protocol	TotalTrans	MaxCvgDist	MaxCvgTime
CATCHUP(100,1000)	1204	4.4km	44s
CATCHUP(200,2000)	1231	7.6km	38s
Randomized Waiting	1944	-	-

- [6] K. Fan, S. Liu, and P. Sinha. On the potential of structure-free data aggregation in sensor networks. In *Proceedings of IEEE Infocom*, 2006.
- [7] K. Fan, S. Liu, and P. Sinha. Scalable data aggregation for dynamic events in sensor networks. In *Proceedings of ACM SenSys*, 2006.
- [8] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann. Impact of network density on data aggregation in wireless sensor networks. In *Proceedings of 22nd International Conference on Distributed Computing Systems*, 2002.
- [9] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed siffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of ACM MobiCom'00*, pages 56–67, 2000.
- [10] B. Krishnamachari, D. Estrin, and S. B. Wicker. The impact of data aggregation in wireless sensor networks. In *Proceedings of the 22nd International Conference on Distributed Computing Systems*, pages 575–578, 2002.
- [11] C. Lochert, B. Scheuermann, and M. Mauve. Probabilistic aggregation for data dissemination in vanets. In *Proceedings of ACM VANET'07 Workshop*, 2007.
- [12] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. In *Proceedings of the 5th symposium on Operating systems design and implementation (OSDI'02)*, 2002.
- [13] S. Madden, R. Szewczyk, M. Franklin, and D. Culler. Supporting aggregate queries over ad-hoc wireless sensor networks. In *Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications*, 2002.
- [14] R. Mangharam, D. S. Weller, D. D. Stancil, R. Rajkumar, and J. S. Parikh. Groovesim: A topography-accurate simulator for geographic routing in vehicular networks. In *Proceedings of ACM VANET'05 Workshop*, 2005.
- [15] T. Nadeem, S. Dashtinezhad, C. Liao, and L. Iftode. Trafficview: Traffic data dissemination using car-to-car communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 8(3):6–19, 2004.
- [16] S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *Proceedings of ACM SenSys*, 2004.
- [17] M. Torrent-Moreno, D. Jiang, and H. Hartenstein. Broadcast reception rates and effects of priority access in 802.11-based vehicular ad-hoc networks. In *Proceedings of ACM VANET'04 Workshop*, 2004.
- [18] M. Torrent-Moreno, P. Santi, and H. Hartenstein. Fair sharing of bandwidth in vanets. In *Proceedings of ACM VANET'05 Workshop*, 2005.
- [19] L. Wischhof, A. Ebner, and H. Rohling. Information dissemination in self-organizing intervehicle networks. *IEEE Transactions on Intelligent Transportation Systems*, 6(1):90–101, March 2005.
- [20] L. Wischhof, A. Ebner, H. Rohling, M. Lott, and R. Halfmann. Sotis: a self-organizing traffic information system. In *Proceedings of the 57th IEEE Vehicular Technology Conference (VTC'03-Spring)*, pages 2442–2446, April 2003.
- [21] H. Wu, R. Fujimoto, R. Guensler, and M. Hunter. Mddv: A mobility-centric data dissemination algorithm for vehicular networks. In *Proceedings of ACM VANET'04 Workshop*, 2004.
- [22] J. Zhao and G. Cao. Vadd: Vehicle-assisted data delivery in vehicular ad hoc networks. In *Proceedings of IEEE Infocom'06*, April 2006.