Adaptive Forwarding Delay Control for VANET Data Aggregation

Bo Yu, Cheng-Zhong Xu, *Senior Member*, *IEEE*, and Minyi Guo, *Senior Member*, *IEEE*

Abstract—In-network data aggregation is a useful technique to reduce redundant data and to improve communication efficiency. Traditional data aggregation schemes for wireless sensor networks usually rely on a fixed routing structure to ensure data can be aggregated at certain sensor nodes. However, they cannot be applied in highly mobile vehicular environments. In this paper, we propose an adaptive forwarding delay control scheme, namely Catch-Up, which dynamically changes the forwarding speed of nearby reports so that they have a better chance to meet each other and be aggregated together. The Catch-Up scheme is designed based on a distributed learning algorithm. Each vehicle learns from local observations and chooses a delay based on learning results. The simulation results demonstrate that our scheme can efficiently reduce the number of redundant reports and achieve a good trade-off between delay and communication overhead.

Index Terms—Vehicular networks, data aggregation, distributed learning.

1 Introduction

Vehicular Ad hoc Networks (VANETs) have been regarded as an emerging and promising field in both industry and academia. It has 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, redundant reports can be generated and forwarded in the network, consuming a considerable amount of bandwidth.

Bandwidth utilization is an important issue for VANET communications. For example, in Dedicated Short Range Communications [1] (DSRC), the communication range can be up to 1,000 m, while the data rate is only 6 to 27 Mbps. Imagine the rush hour traffic, it is very likely that there are more than a hundred vehicles within the range of 1,000 m, all of which share a quite limited bandwidth.

Data aggregation is a potential approach to improving communication efficiency. 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 reports are generated.

- *B. Yu and C.-Z. Xu are with the Department of Electrical and Computer Engineering, Wayne State University, Detroit, MI 48092. E-mail: {boyu, czxu}@wayne.edu.*
- *M. Guo is with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Computer Science SJTU, 800 Dongchuan Road, Shanghai 200240, China. E-mail: guo-my@cs.sjtu.edu.cn.*

Manuscript received 15 Aug. 2010; revised 27 Dec. 2010; accepted 6 Jan. 2011; published online 17 Mar. 2011.
Recommended for acceptance by S. Rangarajan.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2010-08-0484.
Digital Object Identifier no. 10.1109/TPDS.2011.102.

One challenge in data aggregation is to ensure that reports from different sources can be delivered to the same node at the same time so that they can be merged together. In sensor networks, researchers have proposed a number of structure-based aggregation schemes [5], [6], [7], [8], which rely on a fixed routing tree to ensure reports can be merged at the tree forks. Certainly, these schemes are not suitable for dynamic vehicular environments. Fan et al. [9] proposed a structure-free aggregation protocol based on randomized waiting. However, this probabilistic approach cannot guarantee the aggregation of all reports from a single event source. Later, the authors presented a semistructured approach [10] to improve the aggregation degree, but this approach still needs to maintain a routing structure. Several VANET projects, such as Self-Organizing Traffic Information System (SOTIS) [11], [12] and TrafficView [13], use periodical broadcasting for data aggregation. As time elapses, neighboring vehicles can get an overview of current traffic conditions in the vicinity by periodically exchanging information. However, content exchange based on periodical broadcasting may not be an efficient way in terms of communication overhead, and what is an optimal broadcast interval is still an unsolved issue.

In this paper, we propose an adaptive forwarding delay control scheme for VANETs, namely, Catch-Up. The basic idea of the Catch-Up scheme is to insert an adaptive delay before forwarding a report to the next hop. Since reports are propagated along a straight road segment, when different delays are applied to different reports, nearby reports can have a better chance to meet each other and be aggregated together. We formulate in-network data aggregation in VANETs as a distributed learning problem. Vehicles cooperate with each other to increase the chance of report encounters. We propose a new paradigm of distributed learning, "Learning-From-Others." Traditional distributed learning encourages local information exchange to facilitate the learning process [18]. However, this introduces extra communication overhead, which actually is opposite to the original purpose of data aggregation. We propose a new

1045-9219/12/$31.00 © 2012 IEEE Published by the IEEE Computer Society

paradigm of reinforcement learning, in which a learner learns from other learners' action/reward pairs so as to avoid exchanging the entire local knowledge base. We design a Q-learning-based scheme under this paradigm, and we also propose a fuzzy-rule-based function approximation method to speed up the learning process. We conduct simulation experiments with NS2 [2] and GrooveNet [3], [4], which demonstrate the effectiveness of our scheme.

The remainder of this paper is organized as follows: After discussing the related work in Section 2, we present the system model for the Catch-Up scheme in Section 3. Section 4 describes the design details of the Catch-Up scheme. Section 5 presents simulation results. We conclude the whole paper in Section 6. In addition, Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2011.102, discusses the use case and implementation issues, and Appendix B, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2011.102, analyzes the overall performance of the Catch-Up scheme.

2 Related Work

In mobile ad hoc networks, data aggregation is studied in two main aspects: report routing and data management. Report routing focuses on routing problems such as when and where two (or more) reports can meet each other and be aggregated together. Data management studies data representation, data processing, and data merging calculation. In this paper, we focus on the routing-related issues in VANET data aggregation.

In the report routing aspect, great efforts have been made in sensor networks in the past few years. Since sensor nodes are usually static, it is feasible to establish a routing structure to facilitate data aggregation. In [6] and [8], a fixed forwarding tree is established in advance, and sensor nodes periodically measure the environments and generate reports, which, later, are aggregated at tree forks. In [10], a semistructure approach was introduced to deal with event-based applications. However, in VANETs, it's not feasible to maintain a structure or semistructure due to the highly mobile nature of VANETs. Fan et al. [9] proposed a structure-free data aggregation protocol. This is a probabilistic approach in which each node applies a randomized delay before forwarding the report to the next hop. Due to the differences in propagation speed, a report has a chance to be aggregated with other reports. This approach helps to increase the opportunity of report encounters, but there is no guarantee on when or where reports from a single event source can be aggregated together.

In VANETs, several studies have implemented data aggregation mechanisms. In the Self-Organizing Traffic Information Systems [12], vehicles on a road segment periodically send out reports containing the traffic information on the current road segment. During the broadcast interval, a vehicle collects and aggregates information received from neighboring vehicles. This approach helps generate an overview of current traffic conditions by periodical broadcasting. However, periodical report broadcasting is not an efficient way for report propagation, and there is no guarantee that redundant reports from the same road segment can be aggregated together. TrafficView [13] is another similar system which uses periodic report broadcasting for disseminating traffic information. Caliskan et al. [14] proposed a hierarchical aggregation scheme for free parking place discovery. In this scheme, a city is divided into grids, which are further organized in a hierarchical grid-tree structure. Each vehicle maintains such a structure and periodically broadcasts this structure at a predefined interval. As stated by the authors, an optimal broadcast interval dynamically depends on a few factors, which the authors didn't further study.

In general, the challenge is to ensure reports can be delivered to the same node at the same time in a distributed environment. The key is to apply different delays to different reports so that one can catch up with another. Essentially, Randomized Waiting [9] and periodical broadcasting [12], [13], [14] are all different delay strategies. A natural question is how we can adaptively control the forwarding delay so that a report has a better chance to encounter other reports.

3 System Model

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

Basic assumptions. We assume each vehicle is equipped with an on-board computing device, a wireless radio, a GPS device, and digital maps. Time in different vehicles can be synchronized by GPS technologies. For the purpose of neighbor discovery, each vehicle periodically broadcasts beacons so that neighboring vehicles are aware of its identity, speed, and GPS location.

Applications. Our work focuses on providing generic data aggregation service for various applications, such as traffic congestion detection, road condition (i.e., icy road surface) detection, pothole detection. A modern vehicle is usually equipped with a number of sensors, including speedometer, camera, stability control sensor, suspension system sensor, etc. The sensor readings from a single vehicle may be inaccurate or even erroneous. For example, when a vehicle runs over a pothole, the on-board computer may obtain a 80 percent probability that there is a pothole. Therefore, it is desired that information from different vehicles can be merged to increase its accuracy. From the perspective of applications, it is important to efficiently aggregate raw data to improve the accuracy of observation.

In the remaining part of this paper, we will take traffic congestion detection as a striking example to show how our scheme works.

Reports. A report can be generated according to various standards. For example, a simple standard for traffic congestion can work as follows: A vehicle calculates the average speed of its neighboring vehicles. When the calculated speed is much less than the speed limit for that road, the vehicle generates a traffic congestion report.

Event sources and aggregatable reports. In this work, we consider event-driven sources. We divide a road into road segments. If two reports come from the same road segment and their birth time is within T_{max}, these two reports are

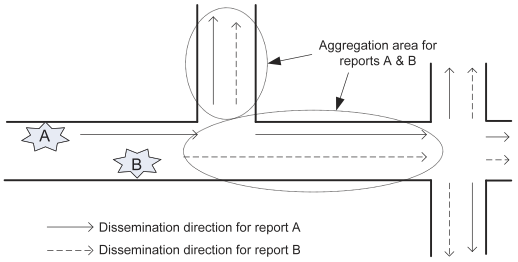

Fig. 1. Dissemination tree. If data from different events are propagating in the same direction, they can be aggregated for reducing communication overhead.

regarded as *aggregatable reports*. An *aggregated report* is the aggregation result of multiple other reports.

However, redundant reports may be generated for two reasons. 1) Two vehicles, out of the signal range of each other, may generate two reports which are supposed to be aggregated together. 2) A vehicle may miss a report broadcast 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 copies and improve communication efficiency.

Data dissemination. As soon as a report is generated for a local event, it will be disseminated to any vehicles coming toward this event source so that the coming vehicles may take an alternate route. Reports will have the same maximal dissemination distance for an event source. On a straight road, reports can be propagated with the help of data forwarding protocols, such as MDDV [15]. At road intersections, a report is duplicated, and one copy goes to one branch. In this way, a dissemination tree, rooted at its birth vehicle, extends to all road segments leading to the event source. Without data aggregation technologies, we could have multiple dissemination trees rooted at different vehicles in the same road segment. These overlapped dissemination trees may cause a considerable amount of redundant communication overhead. Therefore, it is desired that redundant reports can be merged as soon as they are generated. Fig. 1 shows an example of overlapped dissemination trees. We suppose that report B happens earlier than report A. If report B waits for a few seconds, reports A and B may be merged together, and redundant dissemination can be reduced.

4 Catch-Up: Adaptive Forwarding Delay Control

In this section, we introduce the Catch-Up scheme—an adaptive forwarding delay control scheme for VANET data aggregation. The scheme is designed based on a customized distributed learning algorithm.

4.1 Distributed MDP Model

In this section, we introduce a distributed Markov Decision Process (MDP) model. This model is designed for individual vehicles with the objective of improving global performance through distributed cooperation.

In our model, each vehicle is a learner. Each vehicle maintains its local learning knowledge base and makes a decision on how much delay should be applied before forwarding a report to the next hop.

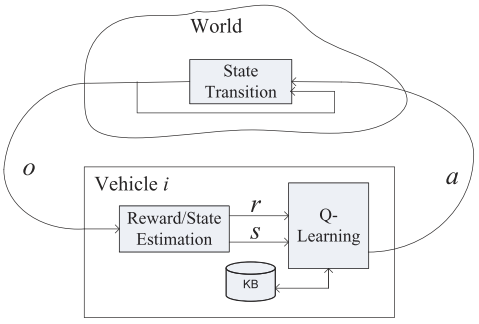

Fig. 2. Distributed MDP model.

We use Markov Decision Process to describe this learning model. The model is defined as a tuple $\langle S, A, P, R\rangle$, where

- S is a finite set of the world states;
- A is the set of actions a vehicle can perform;
- $P_a(s, s')$ is the transition probability that action a in state s will lead to state s' at time $t+1$;
- $R_a(s, s')$ is the immediate reward a vehicle may receive after taking action a in state s at time t.

Fig. 2 shows the major components of this model. In this model, each vehicle receives observation o from the world. The observation consists of two parts: received reports and eavesdropped reports. A vehicle can receive a report which has itself as the destination, and it can also eavesdrop reports being transmitted between neighboring nodes. The vehicle uses this observation o to estimate the current world state s and the reward r.

We define state S as a tuple $<\rho_1, \rho_2, T_w>$, where ρ_1 is the vehicle density ahead, ρ_2 is the vehicle density behind (For the meaning of ahead/behind, we use the report propagation direction as the reference direction), and T_w is the last-seen interval, which indicates the interval between the arrival of the report and the moment when the vehicle eavesdropped a passing-by aggregatable report. We define vehicle density as the number of neighbors within the signal range of a vehicle. A vehicle can calculate the number of its neighbors by receiving periodical beacons from its neighbors. We suppose that every time a vehicle sends a report, it attaches its local vehicle density to the report. Therefore, either by eavesdropping or normally receiving reports, a vehicle may know the vehicle density ahead and behind.

We define action A as a set $\{WALK, RUN\}$, which means that the vehicle applies a short (for RUN) or long (for WALK) delay before forwarding the report to the next hop. Vehicles apply different delays (WALK or RUN) for different nearby reports so that these reports have a better chance to meet each other sooner or later.

A report in our model includes four parts:

$$\{\rho, s, a, payload\}, \tag{1}$$

where ρ is the vehicle density at the sender's location, s is the sender's state, and a is the action applied by the sender, and $payload$ describes the details of an event report, such as a traffic congestion report, or a pothole detection event. Before a report is forwarded to the next hop, ρ, s, a are updated for the purpose of learning. $payload$'s are merged into an aggregated report, if two reports arrive at the same vehicle.

Reward R is used to represent whether a report gets closer to another potential report ahead of it. It implies how successful the previous action is. Suppose that vehicle i receives a report r_1, and, T_w^i seconds before vehicle i receives this report, vehicle i eavesdrops another report which is aggregatable with r_1. After applying a delay, short (RUN), or long (WALK), vehicle i forward the report to vehicle j. Suppose that T_w^j seconds before vehicle j receives this report, vehicle j also eavesdrops another aggregatable reports. In this way, we can use relative time intervals, T_w^i and T_w^j, to represent whether report r_0 is getting closer to another aggregatable report ahead of it. Reward R can be defined as follows:

$$R = \begin{cases} \frac{T_w^i - T_w^j}{T_w^i}, & \text{if } T_w^i - T_w^j > 0, \\ 0, & \text{otherwise}, \end{cases} \tag{2}$$

where $T_w^i - T_w^j > 0$ means report r_1 is getting closer to another aggregatable report. If vehicle i receives another aggregatable report before sending out report r_1, we set $R = 1$.

Given the current state and the reward for the received report, the vehicle updates its local knowledge base and finds the next optimal action (WALK or RUN) with its learning algorithm. After a delay (WALK or RUN), the report is forwarded to the next hop.

The action has an impact on the world state; the impact can be represented by state transition model P, which describes the probability of state change after taking an action. However, it's difficult for us to define or estimate model P. A typical method to solve such MDP problems with unknown models is to use reinforcement learning algorithms. In the next section, we will introduce our reinforcement learning algorithm.

In general, the MDP model works as follows: Suppose that we have vehicles i and j. Before vehicle i forward a report to vehicle j, vehicle i attaches its local variables ρ^i, s^i, a^i to the report (as shown in (1)). After vehicle j receives the report, it calculates the reward R^i with (2). Evidently, reward R^i is obtained based on vehicle i's action a^i. Vehicle j then uses s^i, a^i, R^i to update its local knowledge base and also calculates the optimal action (WALK or RUN) based on its local knowledge base. After a delay (WALK or RUN), vehicle j forward the report to the next hop. Please note that in this process vehicle j uses vehicle i's action/reward pair to update its local knowledge base. In other words, vehicle j learns from other vehicles' action/reward pairs. This important feature guarantees that little extra communication overhead is introduced by the learning process. Further analysis of this feature is presented in Section 4.4.

4.2 Q-Learning

In this section, we propose a reinforcement learning algorithm for our MDP learning model. In reinforcement learning, an agent learns to optimize an interaction with a dynamic environment through trial and error. There have been a number of reinforcement learning algorithms, such as direct utility estimation, temporal difference learning, and Q-learning, etc. [19]. Since the state transition model is unknown in our problem, we are motivated to use model-free Q-learning.

With Q learning, each vehicle maintains a two-dimensional Q table. Given state s and action a, we can locate a cell in the Q table. The Q value in each cell implicates the average reward value over time. Therefore, every time we need to make a decision, we just look up the maximal Q value in the table, and the corresponding action is the optimal action. A vehicle periodically updates the Q table to improve the optimality of its actions. The Q table serves as the local knowledge base for each vehicle.

The updating of Q function can be defined as

$$\begin{aligned} Q(s_{t-1}, a_{t-1}) \leftarrow Q(s_{t-1}, a_{t-1}) + \eta(R_{t-1} + \gamma Q(s_t, a_t) \\ - Q(s_{t-1}, a_{t-1})), \end{aligned} \tag{3}$$

where s_t is the current state and a_t is the next action to take, γ is the future discount ($0 < \gamma < 1$), and η is the learning rate($0 < \eta < 1$). η is decreased slowly during learning to ensure that the learning converges to optimal Q values.

We can regard the Q table as a function of variables ρ_1, ρ_2, T_w, a. To find an optimal action, we just need to calculate $a = \arg\max_a Q(\rho_1, \rho_2, T_w, a)$ given ρ_1, ρ_2, T_w. Since we have four variables and especially variable T_w may vary in a wide range, the state space of Q function is very large. It may take very long time for the learning algorithm to achieve the final optimal policy. On the other hand, it's not necessary to explore every corner in the whole state space. Therefore, we need a technique to approximate to those unexplored state space. Function approximation is a good potential approach.

4.3 Function Approximation

In this section, we first discuss the challenge behind our problem and then propose a fuzzy-rule-based function approximation method.

Theoretically, we can use any regression method to train the Q function, but our particular task has several unique properties. Vehicle densities ρ_1 and ρ_2 have a significant impact on network connectivity, but it's difficult to find an accurate transition threshold; if higher than the threshold, the network is connected; if lower than the threshold, the network is disconnected. Similarly, T_w implicitly indicates how far away the recent aggregatable report is. It's also difficult to define what is near and what is far. In general, it's difficult to directly define a linear or polynomial model to represent the relation between the Q function and its variables ρ_1, ρ_2, T_w, a.

Based on above analysis, we propose to use fuzzy rules to represent the relation between the Q function and its variables. Fuzzy logic is good at representing inaccurate values. In fuzzy theory, a rulebase is a function f that maps an input vector $\overline{x}$ into a scalar output y. A rulebase is formed out of fuzzy rules, where a fuzzy rule i is a function f_i that maps an input vector $\overline{x}$ into a scalar p_i. Vector $\overline{p} = (p_1, \ldots, p_i, \ldots, p_M)$ can be regarded as a set of coefficients of the rulebase function f. Given a series of input $\overline{x}$ and output y, we can perform supervised learning on the rulebase function f, during which coefficients $\overline{p}$ are tuned.

We need to represent our Q function as a series of fuzzy rules. For our problem, we suppose $Q^*(\rho_1, \rho_2, T_w, a)$ to be the optimal Q function, and we can use $Q(\rho_1, \rho_2, T_w, a|\overline{p})$ to approximate to $Q^*(\rho_1, \rho_2, T_w, a)$, where vector $\overline{p} = (p_1, \ldots, p_i, \ldots, p_M)$ is a set of coefficients of the Q function.

$Q(\rho_1, \rho_2, T_w, a|p)$ is able to approximate to Q^* by varying $\overline{p}$. ρ_1, ρ_2, T_w, a can be regarded as the input vector of the rulebase, i.e., $\overline{x} = (\rho_1, \rho_2, T_w, a)$, and p_i is the output scalar of fuzzy rule i.

The following 16 fuzzy rules are used for our problem:

- Rule 1: IF $(\rho_1$ is $S_1^1)$ and $(\rho_2$ is $S_2^1)$ and $(T_w$ is $S_3^1)$ and $(a$ is $S_3^1)$ THEN (p_1);
- Rule 2: IF $(\rho_1$ is $L_1^2)$ and $(\rho_2$ is $S_2^2)$ and $(T_w$ is $S_3^2)$ and $(a$ is $S_3^2)$ THEN (p_2);
- ...;
- Rule 16: IF $(\rho_1$ is $L_1^{16})$ and $(\rho_2$ is $L_2^{16})$ and $(T_w$ is $L_3^{16})$ and $(a$ is $L_3^{16})$ THEN (p_{16}) .

These fuzzy rules establish a mapping from fuzzy logic conditions to the output scalars $p_1, \ldots, p_{16}$. In the rules, $(x$ is $S)$ is a fuzzy precondition, representing "if variable x is small," and $(x$ is $L)$ has the corresponding meaning. For example, the second rule can be interpreted as "*IF (the front vehicle density is high) and (the back vehicle density is low) and (the last-seen time is short) and (the last action is applying a short delay) THEN we get the coefficient value* p_2."

For each precondition used in the 16 fuzzy rules, we define a membership function μ to indicate the degree to which a precondition is satisfied. For our problem, the membership functions are defined as follows:

- the degree to which $(\rho_1$ is $S)$: $\mu_S(\rho_1) = \rho_1/\rho_{max}$;
- the degree to which $(\rho_1$ is $L)$: $\mu_L(\rho_1) = 1 - \rho_1/\rho_{max}$;
- the degree to which $(\rho_2$ is $S)$: $\mu_S(\rho_1) = \rho_2/\rho_{max}$;
- the degree to which $(\rho_2$ is $L)$: $\mu_L(\rho_1) = 1 - \rho_2/\rho_{max}$;
- the degree to which $(T_w$ is $S)$: $\mu_S(T_w) = T_w/T_{w_max}$;
- the degree to which

$$(T_w \text{ is } L)\text{: } \mu_L(T_w) = 1 - T_w/T_{w_max};$$

- the degree to which $(a$ is $S)$: $\mu_S(a) = 0$;
- the degree to which $(a$ is $L)$: $\mu_L(a) = 1$.

In these functions, ρ_{max} is the maximal vehicle density considered in the calculation, and T_{w_max} is the maximal last-seen interval considered. Both ρ_{max} and T_{w_max} are system parameters.

The output of the fuzzy rulebase $f(\overline{x})$ is a weighted average of p_i

$$y = Q(\overline{x}|\overline{p}) = \frac{\sum_{i=1}^{M} p^i w^i(\overline{x})}{\sum_{i=1}^{M} w^i(\overline{x})}, \tag{4}$$

where $w^i(\overline{x}) = \sum_{j=1}^{4} \mu_{i,j}(x_j)$, i is the index of the rule, and j is the index of the μ value in a rule. Actually, p_i controls the contribution of each rule to the final output of $f(\overline{x})$.

We can let function $Q(\overline{x}|\overline{p})$ approximate to the optimal Q^* function by iteratively updating the coefficients p_i as follows:

$$p_{t+1}^i - p_t^i = \eta[R_t + \gamma Q(\overline{x}_{t+1}|\overline{p}_t) - Q(\overline{x}_t|\overline{p}_t)]\frac{\mathrm{d}Q(\overline{x})}{\mathrm{d}p^i}.$$

The increment of p^i is equal to the total increment of the Q value times the fraction of contribution of rule i. The formula can be reformatted as

$$p_{t+1}^i = p_t^i + \eta[R_t + \gamma Q(\overline{x}_{t+1}|\overline{p}_t) - Q(\overline{x}_t|\overline{p}_t)]\frac{w^i(\overline{x})}{\sum_{i=1}^{M} w^i(\overline{x})}. \tag{5}$$

After a vehicle receives a report, it can update the Q function by updating all p^i values with (5). Then, it can calculate the optimal action $a = \arg\max_a Q(\rho_1, \rho_2, T_w, a|\overline{p})$.

4.4 Main Feature of Catch-Up

The main feature of our scheme is that a vehicle indirectly learns from other neighboring vehicles' action/reward pairs, whereas, in traditional learning algorithms, the learners learn from themselves' trial and error, and they usually encourage local information exchange for better learning performance [18].

Our method minimizes the communication overhead in distributed learning in two aspects. First, we don't need the vehicles ahead, perhaps multiple hops away, to send back a message to show the reward for the previous action. In other words, we avoid the communication overhead for reward information exchange. Second, this model also avoids the exchange of Q tables among vehicles. Q table (function) is actually a local knowledge base, usually of large volume. Therefore, this learning-from-neighbors paradigm effectively reduces the communication overhead for distributed learning algorithms.

The extra communication overhead incurred by our learning process is little. As aforementioned, each vehicle attaches only three variables ρ, s, a to a report before forwarding it to the next hop. Since they are simple variables, the incurred communication overhead is little. We don't have extra message exchanges for the learning process. Actually, it is more important that our scheme can effectively reduce the number of redundant reports, which is a major contributing factor in wireless channel congestion and collision.

In Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2011.102, we introduce how our Catch-Up scheme can be integrated into a data aggregation system. We discuss a use case as well as the related implementation issues. In Appendix B, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2011.102, we analyze the overall performance of our Catch-Up scheme.

5 Simulation Evaluation

In this section, we use simulations to evaluate our aggregation scheme and compare our scheme with Randomized Waiting [9].

5.1 Simulation Design

In our experiment, we simulated a scenario of morning rush hour in Detroit. Suppose that 200 vehicles are rushing on I94 (Interstate Highway) in both directions. However, traffic congestion happens at one end of this highway, which is close to Woodward Avenue. This traffic congestion information is propagated and aggregated along I94 so that vehicles in I94, 10 km away from the congestion segment, will know the congestion ahead. The map of the scenario is shown in Fig. 3. Due to the size of the scenario and the simulator's constraints, we mainly focused on how traffic reports were propagated along a 10 km-long distance in I94.

Our simulation was based on NS2 [2] and GrooveNet [3], [4]. GrooveNet is a VANET simulator, which provides a

Fig. 3. Simulation map.

variety of useful models for VANET simulations, such as mobility models, trip models, etc. On the other hand, NS2 presents a series of well-developed low-layer protocols as well as easy programming interfaces. We first used GrooveNet to design the simulation scenario and to generate mobility trace files, and then used NS2 to load these trace files and to run our aggregation scheme. Our aggregation scheme was implemented on NS2.

The system parameters used in our simulation are shown in Table 1. The event source range was set to 1 km, and the aggregatable time frame (T_{max}) was set to 30 s. The communication range was about 250 m. We ran the simulation for 1 hour (in virtual time) and had all the report transmissions logged.

We compared our scheme with Randomized Waiting (RW) [9]. With this scheme, each vehicle applies a randomized delay before forwarding a report to the next hop.

In order to investigate the effects of learning, we also compared our scheme to a *nonlearning version* of our scheme and a *discrete-Q-table version* of our scheme. A naive method to solve our problem is to probabilistically have a part of reports "walk" but the others "run." In our simulations, we suppose that, at each hop, a report has a 80 percent probability of "running," and 20 percent "walking." In other words, this is a nonlearning version of the scheme we proposed, so we call this method the *nonlearning version*. For the discrete-Q-table version of our scheme, we simply replace the function approximation method (proposed in Section 4.3) with a standard discrete Q table. Therefore, we can investigate the real effects of our function approximation method.

To make a fair comparison, we scaled our scheme and Randomized Waiting to the same total delay level, that is, the total end-to-end delay in 10 km. The motivation of doing this is to scale these schemes to the same drivers' expectations. For example, a driver hopes that the traffic updates 10 km away are fresh enough within 60 s. For our

TABLE 1
Simulation Configurations

System Parameter	Value
Road Length	10km
Vehicle Number	200
Communication Range	250m
MAC layer	802.11
Mobility Model	StreetSpeedModel[3]
Trip Model	DjikstraTripModel[3]
WALK	100m/s
RUN	1000m/s

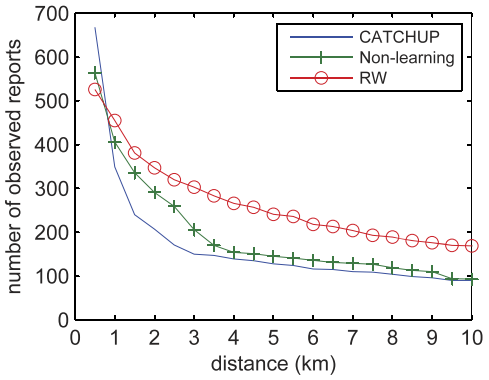

Fig. 4. Number of reports observed at different locations.

scheme, given a set of system parameters, we can calculate the total propagation delay for the 10 km road segment, T_{dism}, using Proposition 2 in Appendix B, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2011.102. For Randomized Waiting, we assume that the physical distance of each hop is about 200 m, so 10 km 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 scaled these two schemes to the same users' expectation, and then evaluated their communication overhead to the system.

Our simulation focused on two metrics: communication overhead and delay. They are two important evaluation metrics for aggregation protocols, which have been widely used in exiting work such as [5], [9], [10].

5.2 Number of Reports

Fig. 4 shows the number of reports observed at different locations. All schemes show a reduced report number with increased distance, which proves the contribution of aggregation operations. However, after the point at about 3 km, our aggregation scheme shows apparent better performance, reducing as much as 50 percent reports than Randomized Waiting. The majority of aggregation operations in our scheme are performed close to the event source, 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.

Our scheme and the nonlearning method have the same waiting time if they perform the same action. However, essentially, the nonlearning method is a probabilistic method. From Fig. 4, we can see that the curve for our scheme drops quickly until 3 km, which indicates that the convergence distance for our scheme is about 3 km in this simulation. The convergence distance for the nonlearning method is about 4 km. The optimized decisions help our scheme outperform the nonlearning method.

Fig. 5 shows the number of final reports observed at different locations. A final report is a report which includes all the aggregatable reports from the same event source. Corresponding to Fig. 4, we can find that most of the

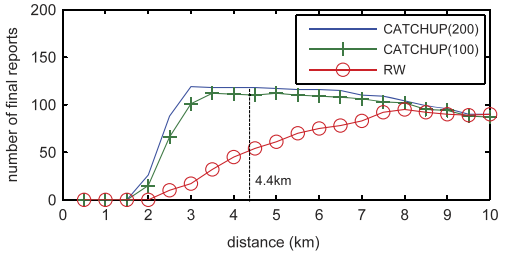

Fig. 5. Number of final reports observed at different locations. The vertical line shows the convergence distance calculated by Proposition 1 in Appendix B, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2011.102. The simulation was conducted with 100 and 200 vehicles, respectively, as indicated by CATCHUP (100) and CATCHUP (200).

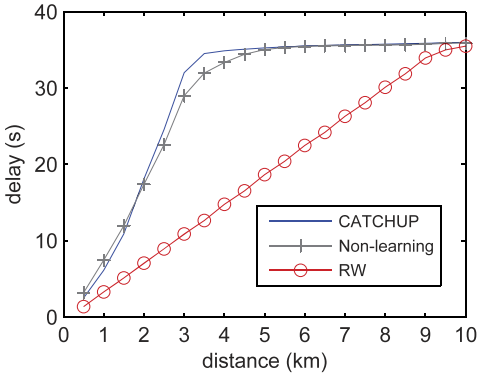

Fig. 6. Delay observed at different locations.

aggregation operations happen between 1 and 3 km. There is a knee point at 3 km for both CATCH (200) and CATCH (100), where the number of final reports reaches its peak. It is about 1.5 km less than the convergence distance (4.4 km) calculated by Proposition 1 in Appendix B, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2011.102. These results show that Proposition 1 is statistically meaningful. The convergence distance calculated by Proposition 1 indicates an upper bound, and in most cases it takes less distance to achieve convergence. In comparison, the curve of Randomized Waiting increases slowly throughout the 10 km distance.

The number of total report transmissions is shown in Table 2. Our scheme reduces nearly 40 percent transmissions compared to Randomized Waiting.

5.3 Delay and Delay Jittering

Fig. 6 investigates the relation between average propagation delay and distance. The delay is defined as the period between the birth of a report and the moment when the report is observed at a given location. For our scheme, after the distance of 4 km, the increase in delay is little. That is because after the theoretical convergence distance (introduced in Appendix B, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2011.102), reports in our scheme enter FLY mode (introduced in Appendix B, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2011.102) and therefore the later delay is almost insignificant. The delay incurred in our scheme is mainly associated with the initial part of the propagation. For randomized waiting, its aggregation operations are performed all over the trip. Therefore, its delay is proportional to the distance. The performance of our scheme 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 segment close to the event source and the total delay time is up bound by a computable limit. For the nonlearning method, since its delay settings is the same as our scheme, they show similar average delays over distance.

Fig. 7 shows the delay jittering observed at a given location. This figure shows the same simulation as Fig. 6, but at each point, an error bar is added to show the delay jittering range. Jittering indicates some reports arriving early but the others late. Therefore, the bigger the jittering, the bigger difference in report propagation speed, and correspondingly the more aggregation operations are associated with that location. Compared to Randomized Waiting and the nonlearning method, our Catch-Up scheme has a wider range of jittering in the initial part of the propagation process, which implies that effective aggregation operations are performed in the early phase. For Randomized Waiting, the operations are distributed all over the path.

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.

5.4 Learning Effects

In this section, we investigate the effects of learning. Fig. 8 compares our scheme to the nonlearning version and the discrete-Q-table version. In our simulation, we allocate 30 cells for each dimension (ρ_1, ρ_2, T_w) in the Q table, and two cells for dimension a. Therefore, the training space is $30 \times 30 \times 30 \times 2$. The huge space slows down the learning process and increases the probability of making a wrong

TABLE 2
Total Report Transmissions

Scheme	CATCHUP	Non-learning	RW
Total Transmissions	8544	9859	13386

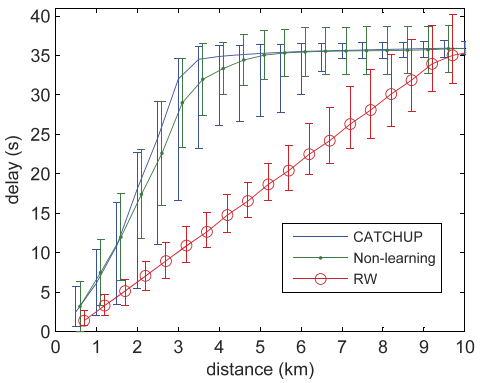

Fig. 7. Delay jittering observed at different locations. Error bars show the range of 90 percent observations.

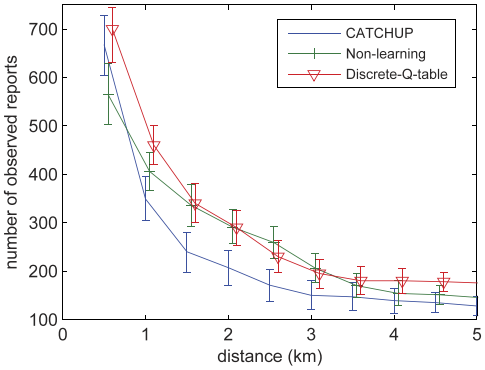

Fig. 8. The effects of learning with different learning algorithms. Error bars show the range of 90 percent observations.

decision. The figure clearly distinguishes our scheme from the nonlearning version and the discrete-Q-table version. In all the test cases, the discrete-Q-table version performed worse than the nonlearning version. This is because the not-well-trained algorithm may frequently make wrong decisions, such as always WALKing or RUNning, due to the unexplored space.

6 Conclusion

In this paper, we have presented a data aggregation scheme for VANETs based on distributed learning. Essentially, the difference in propagation speed helps reports encounter each other, and we formulate this issue as a distributed learning problem where vehicles adaptively choose forwarding delays to make nearby reports have a better chance to meet each other. In order to avoid introducing extra communication overhead, we propose a new paradigm of distributed learning—"Learning-From-Others." We design a Q-learning-based algorithm to implement this new paradigm, and our simulation results demonstrate the effectiveness of our scheme.

References

[1] "Dedicated Short Range Communications Project," http://www.leearmstrong.com/DSRC/DSRCHomeset.htm, 2011.
[2] "The Network Simulator—ns-2," http://www.isi.edu/nsnam/ns/, 2011.
[3] "GrooveNet Project," http://www.seas.upenn.edu/rahulm/Research/GrooveNet/, 2011.
[4] 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," *Proc. Second ACM Int'l Workshop Vehicular Ad Hoc Networks (VANET '05)*, 2005.
[5] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," *Proc. ACM MobiCom '00*, pp. 56-67, 2000.
[6] S. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong, "TAG: A Tiny Aggregation Service for Ad-Hoc Sensor Networks," *Proc. Fifth Symp. Operating Systems Design and Implementation (OSDI '02)*, 2002.
[7] S. Madden, R. Szewczyk, M.J. Franklin, and D. Culler, "Supporting Aggregate Queries over Ad-Hoc Wireless Sensor Networks," *Proc. Fourth IEEE Workshop Mobile Computing Systems and Applications*, 2002.
[8] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann, "Impact of Network Density on Data Aggregation in Wireless Sensor Networks," *Proc. 22nd Int'l Conf. Distributed Computing Systems*, 2002.
[9] K. Fan, S. Liu, and P. Sinha, "On the Potential of Strucuture-Free Data Aggregation in Sensor Networks," *Proc. IEEE INFOCOM*, 2006.
[10] K. Fan, S. Liu, and P. Sinha, "Scalable Data Aggregation for Dynamic Events in Sensor Networks," *Proc. Fourth Int'l Conf. Embedded Networked Sensor Systems (SenSys '06)*, 2006.
[11] L. Wischhof, A. Ebner, H. Rohling, M. Lott, and R. Halfmann, "SOTIS—A Self-Organizing Traffic Information System," *Proc. 57th IEEE Vehicular Technology Conf. (VTC '03-Spring)*, pp. 2442-2446, Apr. 2003.
[12] L. Wischhof, A. Ebner, and H. Rohling, "Information Dissemination in Self-Organizing Intervehicle Networks," *IEEE Trans. Intelligent Transportation Systems*, vol. 6, no. 1, pp. 90-101, Mar. 2005.
[13] T. Nadeem, S. Dashtinezhad, C. Liao, and L. Iftode, "TrafficView: Traffic Data Dissemination Using Car-to-Car Communication," *ACM SIGMOBILE Mobile Computing and Comm. Rev.*, vol. 8, no. 3, pp. 6-19, 2004.
[14] M. Caliskan, D. Graupner, and M. Mauve, "Decentralized Discovery of Free Parking Places," *Proc. ACM Third Int'l Workshop Vehicular Ad Hoc Networks (VANET '06)*, 2006.
[15] H. Wu, R. Fujimoto, R. Guensler, and M. Hunter, "MDDV: A Mobility-Centric Data Dissemination Algorithm for Vehicular Networks," *Proc. ACM First Int'l Workshop Vehicular Ad Hoc Networks (VANET '04)*, 2004.
[16] F.J. Ros, P.M. Ruiz, and I. Stojmenovic, "Reliable and Efficient Broadcasting in Vehicular Ad Hoc Networks," *Proc. IEEE Vehicular Technology Conf.*, 2009.
[17] A. Broder and M. Mitzenmacher, "Network Applications of Bloom Filters: A Survey," *Internet Math.*, vol. 1, no. 4, pp. 485-509, 2005.
[18] L. Panait and S. Luke, "Cooperative Multi-Agent Learning: The State of the Art," *Autonomous Agents and Multi-Agent Systems*, vol. 11, no. 3, pp. 387-434, Nov. 2005.
[19] S.J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, second ed. Prentice Hall, 2003.

Bo Yu received the PhD degree in computer science in 2007 from Fudan University, China. Currently, he is a postdoc fellow in the Department of Electrical and Computer Engineering, Wayne State University. His research interests include Wireless Sensor Networks, Mobile Ad hoc Networks, and Vehicular Networks, with a focus on Information Dissemination and Resource Management.

Cheng-Zhong Xu received the PhD degree in computer science from the University of Hong Kong in 1993. He is currently a professor in the Department of Electrical and Computer Engineering of Wayne State University, and the director of Cloud and Internet Computing Laboratory (CIC) and Sun's Center of Excellence in Open Source Computing and Applications (OSCA). His research interest is mainly in scalable distributed and parallel systems and wireless embedded computing devices. He has published two books and more than 160 articles in peer-reviewed journals and conferences in these areas. He is a senior member of the IEEE.

Minyi Guo received the PhD degree in computer science from the University of Tsukuba, Japan. He is currently a distinguished professor and head of the Department of Computer Science and Engineering, Shanghai Jiao Tong University (SJTU), China. He received the national science fund for distinguished young scholars from NSFC in 2007. His present research interests include parallel/distributed computing, compiler optimizations, embedded systems, pervasive computing, and bioinformatics. He has more than 230 publications in major journals and international conferences in these areas. He is a senior member of the IEEE, member of ACM, IEICE IPSJ, and CCF.