Ad Hoc Networks 8 (2010) 489–505

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

PROMPT: A cross-layer position-based communication protocol for delay-aware vehicular access networks

Boangoat Jarupan *, Eylem Ekici

Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH 43210, United States

ARTICLE INFO

Article history:
Available online 1 February 2010

Keywords:
Cross-layer protocols
Delay sensitive applications
Multi-hop communication
V2V
V2I

ABSTRACT

Vehicular communication systems facilitate communication devices for exchange of information among vehicles and between vehicles and roadside equipment. These systems are used to provide a myriad of services ranging from traffic safety application to convenience applications for drivers and passengers. In this paper, we focus on the design of communication protocols for vehicular access networks where vehicles access a wired backbone network by means of a multi-hop data delivery service. Key challenges in designing protocols for vehicular access networks include quick adaptability to frequent changes in the network topology due to vehicular mobility and delay awareness in data delivery. To address these challenges, we propose a cross-layer position-based delay-aware communication protocol called PROMPT. It adopts a source routing mechanism that relies on positions independent of vehicle movement rather than on specific vehicle addresses. Vehicles monitor information exchange in their reception range to obtain data flow statistics, which are then used in estimating the delay and selecting best available paths. Through a detailed simulation study using ns-2, we empirically show that PROMPT outperforms existing routing protocols proposed for vehicular networks in terms of end-to-end packet delay, packet loss rate, and fairness of service.

1. Introduction

In recent years, the awareness of transportation problems has increased due to rising fuel costs, air pollution, and increased number of accidents. In urban environments, where the population density is high, these problems are more pronounced. To alleviate these problems, Intelligent Transportation Systems (ITS) are proposed to utilize information about vehicle traffic through a communication structure. Such systems are useful in many applications including emergency notification systems, vehicle traffic management, and traveler information/support.

An important component of ITS is the vehicular communication network called VANET that enables information exchange among vehicles. VANET is a self-configuring mobile ad hoc network of vehicles interconnected via wireless link. VANETs' services can be classified as Vehicle-to-Infrastructure (V2I) and Vehicle-to-Vehicle (V2V). V2I communication provides connectivity between roadside equipments and vehicles. V2V provides connectivity directly between vehicles without relying on an existing infrastructure. There are increasing number of efforts such as DSRC[1] and 802.11p[2] to standardize communication protocols for VANETs.

VANETs are fundamentally different from other ad hoc networks in several ways. They have geographically constrained but highly dynamic topologies, predictable mobility, and fast varying connectivity. These characteristics

* Corresponding author. Tel.: +1 16142923430.
E-mail addresses: bea@ece.osu.edu (B. Jarupan), ekici@ece.osu.edu (E. Ekici).

[1] http://www.standards.its.dot.gov/Documents/advisories/dsrc_advisory.htm.
[2] http://grouper.ieee.org/groups/802/11/Reports/tgp_update.htm.

1570-8705/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.adhoc.2009.12.006

make it difficult to apply traditional solutions developed for MANETs directly to VANETs. A major focus of recent research on VANETs is to minimize the end-to-end communication delay, which is important both in providing emergency information and also in delay sensitive applications. For example, consider user communication services, which require information exchange among passengers from different cars, and info-tainment applications involving real-time multimedia. They try to improve the convenience and comfort level for travelers and often requires vehicular networks with small end-to-end delay. VANETs may also be used to develop marketing tools for businesses. For instance, restaurants or hotels can broadcast their information over VANETs and interested drivers or passengers can send a query for more information or even make a reservation. VANETs may also be used to develop e-commerce applications wherein travelers can make business transactions. Such applications require networks which are delay-efficient and reliable.

As part of our previous research, we developed a cross-layer protocol CVIA for highways [1] which aimed at providing location-independent throughput and fairness to vehicles. In this work, we focus on urban environments laced with intersections. Such road networks pose several new challenges such as multiple paths, overhead due to routing decisions at intersections, packet congestion, and highly mobile relay nodes. We propose a new protocol called *PROMPT*,[3] which is a cross-layer position-based delay-aware protocol that addresses these challenges and delivers packets over minimum delay paths.

PROMPT adopts a position-based routing approach to eliminate negative effects of high node mobility rates. Paths are determined at source nodes based on network traffic statistics collected during propagation of service advertisements of base stations. Taking advantage of the fact that data traffic along roads vary much slower than vehicles' positions, statistics collected in individual nodes are mapped to locations. We introduce analytical methods to map such statistics to delay estimations at source nodes, which allows us to distribute the traffic over all available paths.

Through an extensive simulation study, we show that PROMPT is able to use local information effectively to predict the end-to-end delay. We also show that PROMPT outperforms DSR [2], CAR [3], GPSRJ+ [4], and VADD [5] protocols in terms of end-to-end delays, success rates, and fairness.

2. Related work

Vehicular ad hoc networks (VANETs) are a type of mobile ad hoc network with special properties. Even though some routing protocols developed for MANETs (e.g., AODV [6] and DSR [2]) can directly be used in VANETs, it has been shown that such direct applications do not deliver good performance due to fast vehicle movement and relatively high speed of mobile nodes in VANETs [7].

Due to continuous vehicular movements, it has been well accepted that *position-based routing* as opposed to topology-based routing is more suitable for VANETs. In MANETs, GPSR [8] is one of the well known examples of position-based routing protocols. However, in urban networks, the greedy forwarding of GPSR often fails and its planar graph-based route discovery mechanism suffers from significant delays. There are several other algorithms which are built on top of GPSR, where the main effort is in improving the route discovery process. Two such examples are GPCR [9] and GPSRJ+ [4]. Since these protocols are position-based, they do not have a global view of the paths. The intensive neighbor management around intersection areas in these protocols makes them very inefficient in urban networks.

Researchers have proposed several position-based protocols to deal with errors in route discovery process of GPSR. The nodes make the decision to either forward the packets or to store them until better forwarding nodes are found. Examples of such protocols include VADD [5] and D-Greedy/D-MinCost [10]. Although these protocols reduce packet delays, they estimate path delays based on vehicle speed and number of intersections. They do not consider more relevant information like packet traffic congestion.

A-STAR [11] incorporates traffic awareness by using *statistically rated maps* where each street is rated based on the city bus routes or by using *dynamically rated maps* where streets are rated based on the number of vehicles. These ratings are used to determine anchor paths with high connectivity for efficient packet delivery. Similarly, CAR [3] maintains network paths through dedicated vehicles called "guards". They try to capture the information on path connectivity in the presence of vehicular mobility. It utilizes route discovery messages as in AODV to discover paths, but uses geographic forwarding principles for data delivery. Both A-STAR and CAR focus mainly on network connectivity issues and are not designed to target delay sensitive applications.

Cross-layer protocol design is another approach to address the mobility issue of the vehicular networks. CCBF [12] is one such protocol delivering routing and MAC services. It takes advantage of cluster-based forwarding mechanism to reduce collisions and packet delay. MAC layer provides cluster and neighbor information to the routing algorithm. However, this study mainly focuses on priority-based data delivery where packets are classified as high and low priority. CVIA [1] is another cross-layer protocol for highways which aims to provide location-independent throughput and fairness to vehicles. CVIA reduces collisions by forming single-hop clusters and propagating data based on road segments. An analytical throughput estimation model is used to provide fairness. Both CCBF and CVIA, which rely on cluster-based topology, are more suitable for highways but not for city roads.

3. Protocol overview

Our proposed protocol PROMPT is designed for V2I systems consisting of vehicles and base stations (BS), each of

[3] PROMPT is an anagram of capitalized letters in "cross-layer Position-based coMmunication PROTcol for delay-aware vehicular access networks".

which is identified with a unique ID. BSs are gateways installed at fixed locations along the road. All vehicles and BSs are equipped with wireless radios for communication. BSs communicate among themselves through wired connections and with vehicles through wireless interfaces. The system supports multi-hop communication to connect base stations with distant vehicles. Vehicles are also equipped with Global Positioning Systems (GPS) to aid in location discovery and with digital roadmaps to assist with street information. In this paper, we assume that the location information provided by GPS is accurate. This is a reasonable assumption as GPS error is typically within 15 m range [13]. Furthermore, Korkmaz and Ekici [14] showed, in the context of position-based MAC protocols, that the effect of such GPS errors is minimal on the protocol performance (e.g., packet delivery rate). In case where GPS signal is blocked by buildings or other environmental factors, vehicles can find their location information by using techniques like *location assist algorithm* [15] which are beyond the scope of this paper. In this work, we assume the road structure to be similar to an urban area with several intersections.

A BS advertises its services by periodically broadcasting beacon messages in its *immediate service area*, which is determined by the BS's communication range. This process is referred to as *Service Advertisement* (Fig. 1). Beacons are propagated outside a BS's communication range (*extended service area*) via a directional multi-hop broadcasting protocol (Section 4.1). In an urban road network, a vehicle may receive the same beacon over multiple paths. Our protocol determines whether to forward the received beacons based on the path traversed so far rather than the BS information alone. Before forwarding, the vehicle updates the beacon with its current location and local data traffic statistics so that the receivers can obtain detailed knowledge about the path to BS (Section 5.1).

We follow a *location-based source routing* approach to send packets to a BS during *Data Delivery* process (Fig. 2). Our strategy is to combine source routing and geographic routing principles. *Relay nodes* are intermediate vehicles which are located along a path from a vehicle to a BS. Whenever a vehicle has a packet to send, it initiates the *route selection* process to select the *best* possible route in terms of total path delay from its path information table (Section 5.2). The vehicle leverages the traffic load information collected via beacons in estimating the total delay along a given path. The selected route is then transformed

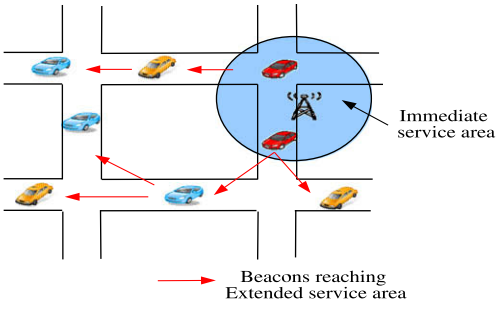

Fig. 1. System architecture.

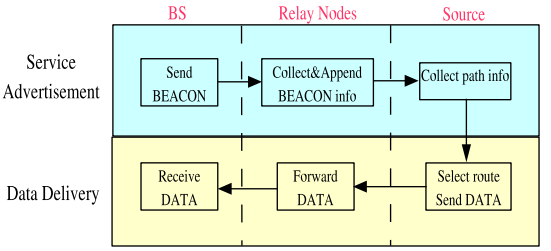

Fig. 2. System diagram.

into a path on the roadmap that consists of streets and directions, and the resulting path is attached to the data packet. The MAC layer then sends packets by selecting the relay nodes along the directions contained in packets (Section 5.3). Any relay node in a given direction can provide the forwarding service independent of the node address. Such a mechanism reduces the neighbor management overhead and minimizes the connectivity problem. To further reduce the average delay, relay nodes try to bundle different packets with the same path and send them in a single batch, which is referred to as a *packet train*. By doing so, the channel is contended only for the first packet in the batch, reducing the average end-to-end delay (Section 5.3.2).

4. Service advertisement

To advertise their services, base stations broadcast beacon messages. Each beacon message contains a *BSid*, *SEQ*, *TTL*, *PATH*, and *PathInfo*. *BSid* is the base station address which uniquely identifies a BS in the system. *SEQ* is the sequence number of the beacon. The BS attaches a different *SEQ* for each beacon that it generates. *TTL* field in the beacon indicates its lifetime or the beacon hop limit. *TTL* is decremented at each forwarding node, and the beacon is discarded when *TTL* becomes zero. *PATH* is the locations of the relay nodes along the beacon propagation path. *PathInfo* is the collection of data traffic statistics indicating the load information at each relay node along the beacon propagation path. Detailed information about *PathInfo* is given in Section 5.1. *BSid*, *SEQ*, and *PATH* uniquely identify a beacon message. The fields *TTL*, *PATH*, and *PathInfo* are updated at each relay node as the beacon is propagated.

When vehicles receive a beacon message, they store the beacon information into their *path information* table which consists of *BSid*, *SEQ*, *mapPATH*, and *PathInfo* fields. *BSid*, *SEQ*, and *PathInfo* are taken directly from the beacon message. *mapPATH* is the sequence of (street, direction) pairs converted from the *PATH* information. The detailed discussion about this *route mapping* is provided in Section 4.2.

4.1. Beacon propagation

To propagate the beacon to all vehicles, we adopt our UMB protocol [16]. UMB is an IEEE 802.11-based directional multi-hop broadcast protocol where sender finds the farthest node to rebroadcast beacon messages using black burst contention. UMB serves a broadcast session as a collection of directional broadcasts along roads.

At intersections, UMB sends new directional broadcasts to all road directions to cover the complete network. This mechanism, however, results in a significant packet overhead, and may potentially slow down the beacon propagation. In this paper, we address this limitation by making relay nodes broadcast beacons only in one direction. When vehicles that are on streets different than the one relay nodes is on receive the beacon, they forward the beacon in a direction that is away from both the previous relay node and the intersection. We refer to such nodes as *passive relay nodes*. As shown in Fig. 3, *C* is a passive relay node where it further propagates the packet to *F* in a different direction from its source (*B*).

Among all vehicles which are in the beacon propagation direction, the vehicle that has the largest distance to the relay node is chosen as the next relay node following UMB's black burst mechanism. Similarly, among vehicles which are on different streets, a vehicle that is farthest from the intersection is chosen as a relay node in that direction. To this end, vehicles choose a backoff value that is inversely proportional to the distance from the intersection before relaying the message. Consequently, a vehicle closer to the intersection selects a longer backoff value and suppresses its own forwarding request if it overhears any requests from vehicles which are farther from the intersection.

Before forwarding a beacon, a vehicle adds its current location to *PATH* and its current local traffic information to *PathInfo*. Adding the location information instead of the vehicle identifier itself helps receivers in deriving the exact path on the roadmap in which the beacon is transmitted. Consequently, the paths are formed independent of vehicle identities and are not effected by vehicle mobility.

4.2. Beacon handling

Consider a vehicle V_i which receives a beacon B_i whose information is represented as a tuple $(bs, seq, ttl, path, pathinfo)$, where *bs* is the *id* of the base station, *seq* is the sequence number of the packet, *ttl* is the *TTL*, *path* contains the location of relay nodes, and *pathinfo* contains statistics collected by relay nodes. These statistics capture the traffic information along the path from V_i to base station B_i. The location of relay nodes is obtained from a GPS system. For the sake of simplicity, we represent these locations by using Cartesian coordinates on the roadmap.

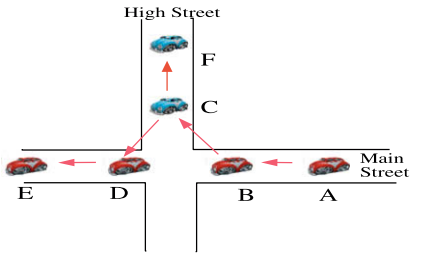

Fig. 3. Beacon propagation example.

Since the roadmap may contain several intersections, a node may receive multiple beacon messages with the same *bs* and *seq* IDs. We treat these beacons differently as long as they differ in the path over which they were received. We maintain paths in the roadmap to deal with vehicular mobility. For this purpose, we first map *path* consisting of locations of relay nodes into a path consisting of streets and directions. We refer to this process as *route mapping*. The resulting mapped path is a sequence of $(street, direction)$ pairs which indicate the path to *bsid* on the roadmap.

We then *reduce* the path to remove *redundant* (street, direction) pairs. This process consists of three steps: (i) We trim the map path by removing identical consecutive pairs. Consider a case where the beacon is propagated on a single long street *s* in a single direction *d*. The map path in such a scenario consists of a sequence of redundant (s, d) pairs, which are reduced to a single pair. (ii) We avoid *triangular* paths which may occur around intersections, as illustrated in Fig. 3. Such paths can arise since we allow vehicles that are not on source street also to accept the beacons. (iii) Finally, we eliminate *loops*.

We then compare the received beacon information with the existing information in the path information table to determine whether to forward the beacon. Whenever the vehicle accepts the beacon, corresponding route information as well as the traffic details (*PathInfo*) are added to or updated in the *path information* table. We also modify Last Update Timestamp (LUT) of the path information table entry to reflect its age. *Stale routes* whose LUT is greater than a threshold are removed from the table at the time of route selection. Section 5.1 describes how a vehicle collects and maintains the traffic information contained in a beacon message.

5. Data delivery

The data delivery process starts with a sender selecting an available *mapPATH* from the *path information* table. We propose a *position-based source routing* approach which is a combination of source routing and geographic routing. Following source routing principles, packet route is attached to the data packets. However, we use geometric information (street and direction) instead of node IDs to indicate routes. In this section, unless mentioned otherwise, a packet refers to a data packet that is to be sent to the base station.

The *PathInfo* obtained from the path information table is used to estimate the path delay for route selection. In Sections 5.1 and 5.2, we define *PathInfo*, introduce the delay estimation model, and their relation to the route selection. The forwarding node selection based on *PATH* and the source route information is described later in Section 5.3.

5.1. Path delay and path quality information (PathInfo)

PROMPT uses the *PathInfo* data in the path information table to select minimum *delay path* during route selection.

The path delay is the total time that a packet spends starting at a source node until it reaches a destination which includes times spent at each relay node. The *per-hop delay* at each node is defined as the sum of the *queuing delay* and the *service delay* at that node. The queuing delay is a time interval from the arrival of a packet at the queue until it reaches the head-of-the-queue. The service delay is the time that a head-of-the-queue packet spends during the channel contention and the actual packet transmission. The per-hop delay depends on the channel capacity and traffic load along the path. In this paper, we assume that other delays such as processing delay and propagation delay are minimal compared to the queuing delay and the service delay.

PathInfo fields:

- Average inter-arrival time of each packet.
- Variance of the inter-arrival time of each packet.
- Average service time of the packet in batch.
- Variance of the service time of the packet in batch.
- Average batch size.

Although vehicles are moving, the data traffic carried in vehicular networks does not fluctuate significantly. Hence, it is not important how much data is forwarded by a given vehicle, but the geographic location to where the data is being forwarded is important. We collect traffic statistics at those geographic locations, and use them in determining the overall path quality. Consequently, specific information such as id of vehicles which relay the data packets is less useful than the traffic information that they collect.

Although beacon forwarders and relay nodes may be different, collected statistics are assumed to remain valid for a given geographical area. We rely on a simple *moving average* analysis that considers only a fixed window of recently collected values. Traffic statistics become *stale* after a period of time due to potential changes in route selection and traffic demands. Therefore, as we show in Section 6.2.1, the choice of *window size*, i.e., the number of recent values we consider affects the accuracy of the estimation model. In the next section, we describe the structure of *PathInfo* and its relationship to the delay estimation.

5.2. Routing decision and delay estimation

PathInfo is the collection of local statistics obtained at each relay node during beacon propagation. Source nodes use this information to estimate the end-to-end delay during the route selection. Unless the network operates at low utilization, reporting average end-to-end delays does not help with the estimation of the end-to-end delay after the addition of a new flow. To illustrate this, consider two paths which are (s, r_1, r_2, d) and (s, r_3, d). Let delay $D[r_i]$ at each relay node r_i be computed using M/M/1 queue analysis:

$$D[r_i] = \frac{1}{(\mu - \lambda)}, \tag{1}$$

where μ is the service rate of the packets at the queue, λ is the arrival rate of the packets at the queue, and $\rho = \frac{\lambda}{\mu}$ is the utilization. Let the load at r_1 and r_2 be $\lambda_1 = \lambda_2 = 0.5C$, where C is the capacity, and the load at r_3 be $\lambda_3 = 0.72C$. For this case, $D_1 = 2,\ D_2 = 2,\ D_3 = 3.6$, and $D_1 + D_2 > D_3$ where $C = \mu = 1$. By using the average delay of the load, the delay of path (s, r_3, d) is smaller than the delay of path (s, r_1, r_2, d). To properly estimate the accumulated path delay, our delay estimation model calculates the total path delay as the sum of delays at each relay node including the delay introduced by the additional packets from the source as shown in the following example. Let s have a flow of rate $\lambda_s = 0.1C$ destined to d. After the addition of this new flow, the delay on (s, r_1, r_2, d) will be $D'_1 = 2.5,\ D'_2 = 2.5$, and $D'_3 = 5.56$ which gives $D'_1 + D'_2 < D'_3$. In this case, path (s, r_1, r_2, d) gives a smaller end-to-end delay than path (s, r_3, d).

As we illustrate in this example, knowledge about the average delay along a path is insufficient to make correct path selections based on delay metric. Hence, our delay model not only analyzes current load in the system but also predicts additional load added by a source.

As a node can bundle multiple packets with same path into a single batch, the inter-arrival times and service times are no longer Poisson processes. We model each relay node as a G/G/1 batch departure queuing system. Curry and Deuermeyer analyze this systems in [17] where they find that the modeling the batch system as a batch process captures the correct behavior and gives a better delay approximation. We adopt this model for our estimations.

PathInfo notations: The following notations are the statistics (*PathInfo*) collected during beacon propagation at each relay node:

- r_i: the ith relay node,
- $E_{r_i}[\tau]$: the expected packet inter-arrival time at r_i,
- $Var_{r_i}[\tau]$: the variance in packet inter-arrival time at r_i,
- $E_{r_i}[S_B]$: the expected batch service time at r_i,
- $Var_{r_i}[S_B]$: the variance in batch service time at r_i,
- $E_{r_i}[b]$: the expected batch size at r_i.

The delay estimation is performed using 5-tuple statistics of *PathInfo*: $[E[\tau], Var[\tau], E[S_B], Var[S_B], E[b]]$, where b is the batch size and B is the batch system. The batch level statistics $E[b]$ are collected at the time of packet arrival at the queue. The following notations are used in developing the queuing model:

- P_j: the jth path obtained from the path information table
- $\vec{E}[\tau]$: the list of expected inter-arrival times of all relay nodes on the *PATH* P_j, which is $(E_{r_1}[\tau], \ldots, E_{r_k}[\tau])$. $\vec{Var}[\tau],\ \vec{E}[S_B],\ \vec{Var}[S_B]$, and $\vec{E}[b]$ are defined similarly.
- $Stats(P_j)$: the statistical information of the *PATH* P_j which consists of $(\vec{E}[S_B],\ \vec{Var}[S_B],\ \vec{E}[\tau],\ \vec{Var}[\tau],\ \vec{E}[b])$.
- $C^2[k]$: the square coefficient of variation (SCV) for a random variable k, $C^2[k] = \frac{Var[k]}{E[k]^2}$.

The additional load at the source is added to $E_{r_i}[\tau]$ which is denoted as $E_{r_i}[\tau']$ and $Var_{r_i}[\tau']$ is assumed to be equal to $Var_{r_i}[\tau]$. The batch utility rate (ρ_{B_i}) at relay node r_i and $SCV(\tau_B)$ are defined as

$$\rho_{B_i} = \frac{E_{r_i}[S_B]}{E_{r_i}[b] \cdot E_{r_i}[\tau']}, \tag{2}$$

$$C^2_{r_i}[\tau_B] = \frac{C^2_{r_i}[\tau']}{E_{r_i}[b]}. \tag{3}$$

The G/G/1 batch departure delay at each relay node is calculated as [17]:

$$D[r_i] = \textit{batch queuing delay} + \textit{batch service delay}$$
$$= \left(\frac{C^2_{r_i}[S_B] + C^2_{r_i}[\tau_B]}{2}\right)\left(\frac{\rho_{B_i}}{1-\rho_{B_i}}\right)E_{r_i}[S_B] + E_{r_i}[S_B].$$

Recall that the traffic information is given per relay node whereas the reduced map path is combined over relay nodes. We estimate the total delay over the entire path P_j as the sum of estimated delays at each relay node r_i. For simplicity in computation, we assume that the delays at different relay nodes are independent of each other:

$$D[P_j] = \sum_{1 \le i \le k} D[r_i]. \tag{4}$$

Once the estimated path delay $D[P_j]$ is calculated for selected paths from the path information table, the path with minimum estimated delay is chosen, and corresponding *mapPATH* is attached to the data packet. To avoid oscillations between paths, the selected path is changed for a flow only when the new minimum delay path has at least 10% less delay than the currently used one, provided that the current path has not expired. Next, we describe the channel access procedure and forwarding relay node selection based on the *mapPATH* information.

5.3. Relay node selection-MAC function

In this paper, we propose a position-based MAC function which uses the positions given (*mapPATH*) to search for the next relay node, which does not require maintaining a neighborhood table.

5.3.1. Channel contention in MAC

The packet that is at the head-of-the-queue in the MAC layer passes through a *contention period* before it is transmitted on the channel. It is in this period that the vehicle with an outstanding data packet discovers the best possible relay node to forward the packet. A vehicle V_s initiates the contention process by sending a FRREQ (forward relay request) packet and waits for a FRREP (forward request reply). The FRREQ packet includes the following information from the data packet: sender id, target *BSid*, and the complete path to the target base station *mapPATH*. If a valid FRREP is not received by a threshold time, then V_s retransmits the FRREQ packet by doubling the contention window. After a pre-set number of retransmissions, the data packet is dropped at V_s.

Whenever a vehicle (say, V_r) receives the FRREQ, it first decides whether to reply by evaluating *qualification criteria* (Section 5.3.3) using the information contained in the FRREQ packet. If V_r is qualified to send a reply, then it selects a backoff (BO) value using the same criteria, and starts the backoff timer. Once that timer expires, V_r sends a FRREP packet to the sender of FRREQ (V_s). The qualification criteria are set up such that the best possible relay node sends the FRREP packet, i.e., the *most qualified relay node selects the smallest backoff value*. Other receivers of the initial FRREQ packet stop their own backoff timers and discards their (yet to be sent) FRREP packets upon hearing the FRREP packet from V_r. This is done so that only one vehicle transmits FRREP to the sender vehicle V_s.

Once V_s successfully receives a FRREP packet, it sends the actual DATA packet to V_r. It then starts a *send timer* and waits for an ACK from the receiver vehicle. If the send timer expires before receiving an ACK, then V_s *restarts* the contention process by sending a *new FRREQ* packet. Here, a new contention period is started instead of retransmitting the DATA packet alone because the receiver vehicle may have moved out of V_s's transmission range which may be the reason for not receiving ACK on time. The data packet is discarded if V_s fails to transmit the packet in a pre-defined number of trials. Throughout this process, all other vehicles which are within the transmission range of V_s and V_r remain idle until the transmission is complete to reduce the interference. This is accomplished by using network allocation vector (NAV) values present in FRREQ, FRREP, and DATA packets.

In case of a FRREQ or FRREP collision, the source vehicle V_s retransmits the FRREQ following the *binary exponential backoff* procedure. Here, instead of letting all receivers of FRREQ to send FRREP packets, we let *only the vehicles which sent FRREP in previous round* contend again to send a new FRREP packet. In these subsequent contentions, the receiver vehicles contend by simply selecting a random BO value from the interval $[0, K]$, where K is a pre-defined constant. Such a method reduces the probability of (re)collision that may otherwise occur due to deterministic evaluation of qualification criteria. Vehicles which did not send a FRREP in previous rounds remain idle until the DATA packet is successfully transmitted.

5.3.2. Packet train mechanism

It is possible that multiple vehicles send packets to a given BS. Therefore, intermediate relay nodes may hold several packets from *different sources* with the *same path*. Instead of transmitting these packets independently, the relay node may choose to bundle multiple packets into a single *packet train* so that the channel is contended only once for the entire packet train. Such a mechanism not only alleviates the contention overhead but also reduces the possibility of collisions, thereby improving the overall average packet delay. We refer to vehicles which bundle the packets as *hyper relay nodes* (HRN).

Hence, in a packet train of size k, a typical sequence of packets transmitted includes FRREQ, FRREP, DATA_1, $\text{ACK}_1, \ldots, \text{DATA}_k$, ACK_k. If the transmission of any packet in the packet train fails, then the vehicle restarts the contention process with data packet that is currently at the head-of-the-queue. Note that the receiver vehicle is the same for all k packets.

To ensure fairness, we limit the size of the packet train to a pre-defined constant value. Also, once the packet train is successfully transmitted, we force vehicles to wait for a period of time that is proportional to the size of the packet train. This strategy helps in reducing the chances of prolonged starvation.

5.3.3. Evaluation of qualification criteria

As mentioned in Section 5.3.1, receivers of FRREQ evaluate certain qualification criteria before they send FRREP packets. These criteria define the backoff time of the receivers. The highest priority receiver chooses the smallest backoff value.

Vehicles choose backoff values from a range of integers $[0, BO_{max}]$. We divide this range into four non-overlapping intervals: $[0,0]$, $[1,k_1]$, $[k_1+1,k_2]$, $[k_2+1,BO_{max}]$, where $0 < k_1 < k_2 < BO_{max}$. The receivers of FRREQ packet evaluates the qualification criteria and selects a backoff value from one of the four intervals as explained below:

- *Case 1: If the receiver is the target base station,* it *immediately* sends the FRREP packet so that the corresponding DATA packet is transmitted to itself by choosing 0 as backoff value.
- *Case 2: If the receiver is a hyper relay node:* If a receiver of FRREQ already has packets with the same path, it is considered as a HRN. HRN acts as a temporary gateway that the sender can reuse for subsequent transmissions. Further, HRNs bundle packets from different sources with the same path into a single packet train. Therefore, they are given higher preference in the interest of smaller delay. HRNs choose the backoff value from the interval $[1,k_1]$. The actual value that is chosen is inversely proportional to the distance between HRN and the source node. The shorter the distance, the higher is the backoff value.
- *Case 3: If the receiver is not a hyper relay node,* vehicles which are in the *same direction* of the packet are given higher preference to become relay node, since they can deliver packets faster than the vehicles which are in different directions than the packet. Such vehicles choose backoff values from the range $[k_1+1,k_2]$. Again farther vehicles choose smaller backoff values. If the streets of the source node and the receiver node are different, then the receiver that is farthest from the intersection chooses the smallest backoff value.
- *Case 4: If the receiver is not a hyper relay node,* receivers which are *not* in the direction of the packet are given lower preferences choosing backoff values from $[k_2+1,BO_{max}]$ interval. Therefore, unlike in Case 3, vehicles which are closer to the intersection (if any) choose smaller backoff values when compared to vehicles which are farther from the intersection.

Note that this interval-based backoff selection is carried out only for the first contention period. In subsequent contentions (especially in Case 3 and Case 4), the backoff values are randomly chosen from the interval $[0,K]$ independent of receiver's position.

6. Evaluation

6.1. Simulation setup

We evaluate our proposed PROMPT protocol through ns-2 simulations considering an example urban road network shown in Fig. 4. Each road has two lanes running in opposite directions and we randomly place 440 vehicles over the entire road network so that each street has a vehicle density about 15 vehicles/km/lane. The vehicle speed is selected randomly from a Gaussian distribution with mean 36 km/h and standard deviation of 5 km/h. For simplicity, we restrict the vehicle movements to linear paths. Vehicles are removed from the current street when they move outside the road network. They reappear in the adjacent lane and move toward the opposite direction.

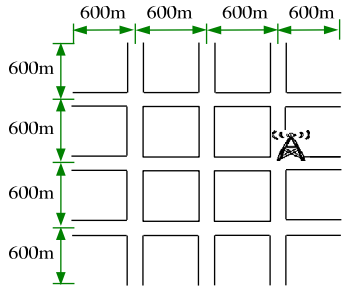

Fig. 4. Example road network in area 2400 m × 24,000 m.

Each simulation lasts 100 s, and the results are averages of 15 independent runs. Base stations generate a new beacon every 20 s. The choice of the beacon interval depends on various parameters such as road layout, application requirements, and vehicle traffic properties. In our simulations, we use 20 s for beacon interval, which appear to be a reasonable choice. For the general settings of our simulation study, we observed diminishing returns of performance of PROMPT that cannot be justified with the increasing in beacon signaling overhead. Focusing on the performance of the protocol, we exclude the detailed discussion of beacon interval from the paper.

We randomly select 20 source vehicles which generate data packets with exponentially distributed inter-arrival times. The default mean of exponential distribution is set to 11.25 pkt/s/src. The data packet payload is of size 500 bytes and the maximum packet train size is 5. The default transmission range of vehicles is 300 m. The data rate is 3 Mbps and the wireless radio frequency is 5.9 GHz (DSRC). The backoff intervals of receivers are $k_1 = 2$, $k_2 = 21$, $BO_{max} = 37$, and $K = 32$. In the delay estimation model, statistics are computed by considering a moving average window of size 25. The packet time-to-live (TTL) is 15 hops for beacon messages.

6.2. Simulation results

We run our protocol under several different parameter settings. For each setting, we consider four different metrics in our evaluation: estimated packet delay ($D[P]$ in Section 5.2); actual end-to-end packet delay observed from simulations; success percentage that represents the fraction of packets that are successfully received at the base station; and fairness index [18] which is computed as $\frac{(\sum x_i)^2}{(n\cdot\sum x_i^2)}$ where n is the total number of flows and x_i is the throughput of flow i.

6.2.1. Evaluation of PROMPT

We study the performance of PROMPT with varying values of number of source nodes, number of base stations, per-node packet generation rate, transmission range of vehicles, vehicle density, and road block size.

6.2.1.1. Effect of number of sources. In Fig. 5, we show the effect of per-node load and the number of source nodes on different performance measures that we consider. With 5 and 10 source nodes, PROMPT delivers low packet delays, high percentage of success, and high fairness. Our delay estimations match with delays from simulations, helping in choosing the best possible path and hence in providing low end-to-end packet delays. As we increase the number of source nodes, the traffic load on the network increases, resulting in high contention overhead and high end-to-end packet delays, especially at high packet arrival rates. Similarly the performance in terms of success percentage and fairness degrade as the load on the network increases.

Next, we study the performance of PROMPT by varying the transmission range of vehicles by fixing the per-node packet generation rate at 11.25 pkt/s/src (Fig. 6). As the transmission range increases, vehicles can cover a larger distance in a single hop thereby increasing the channel contention. Therefore, at very high transmission ranges, *high contention* leads to an increase in the delay. The effect of contention is even more pronounced around intersections and base stations. At low transmission range, vehicles may experience network disconnection in which the search for next relay node may result in multiple retransmissions and increasing the queueing delay. With 20 source vehicles, we found that the average end-to-end delay is optimal around 250–300 m range. This experiment highlights the fact that *transmission power control* is important for non-power-constrained systems for delay minimization.

From Figs. 5a and 6a, it is evident that our model estimates the delay accurately, even at high load conditions. The estimation model relies on communication statistics observed at relay nodes serve as a feedback. This helps the source nodes in adjusting the packet route based on the network packet traffic. Note that the transmission of such statistics does not have a significant impact on the network traffic since they are communicated only during the beacon propagation phase. Once the network reaches its capacity, the success percentage drops significantly due to increased transmission overheads (see Fig. 5b). For 20 source vehicles with transmission range of 300 m, we observed that the network reaches its capacity at 11.25 pkt/src/s.

6.2.1.2. Effect of number of base stations. In our next experiment in Figs. 7 and 8, we study the effect of number of base stations on the performance of PROMPT as we vary the per-node load and transmission range. We fix the number of source nodes as 20. In the case of two base stations, they are placed at (600,600) and (1800,1800) locations on the road network shown in Fig. 4. In case of three base stations, they are placed at (600,600), (1200,1800) and (1800,600) locations.

With high number of base stations, packets can reach the destination in fewer number of hops, thereby providing lower delays, higher success rates, and better fairness (see Fig. 7). However, we observe *diminishing returns* where

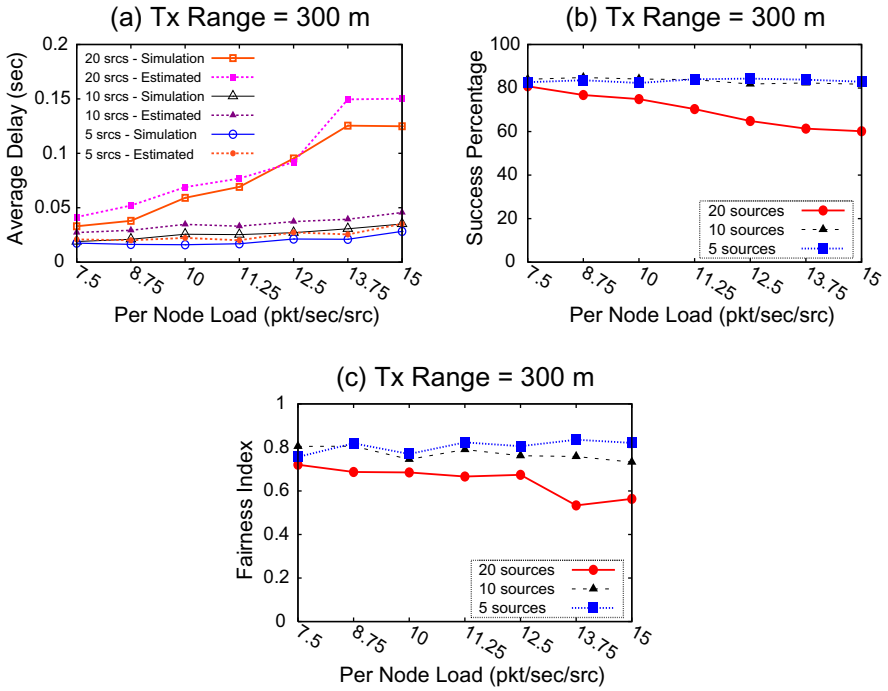

Fig. 5. Varying per-node load for 5, 10, 20 sources.

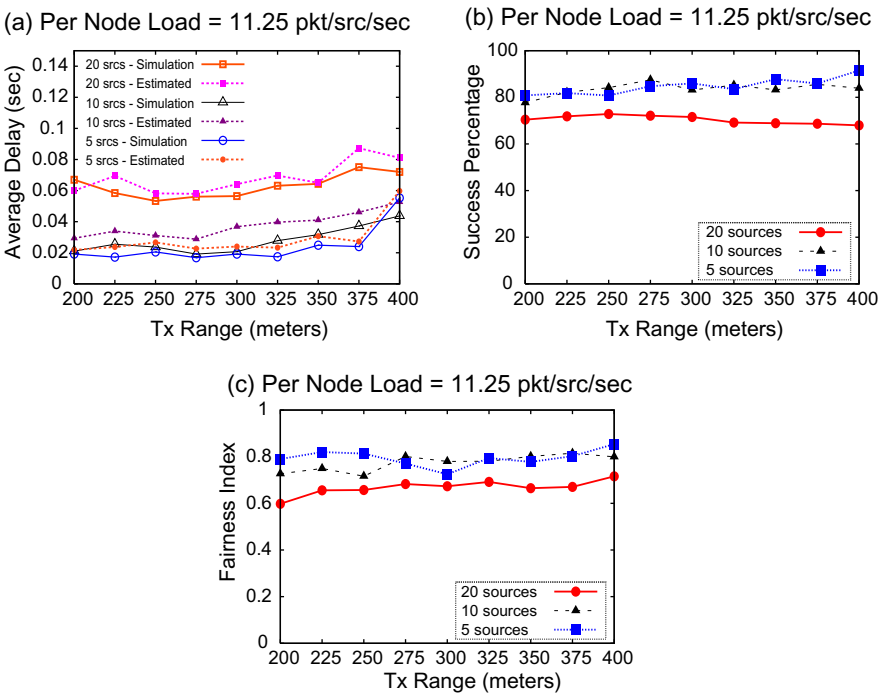

Fig. 6. Varying transmission range for 5, 10, 20 sources.

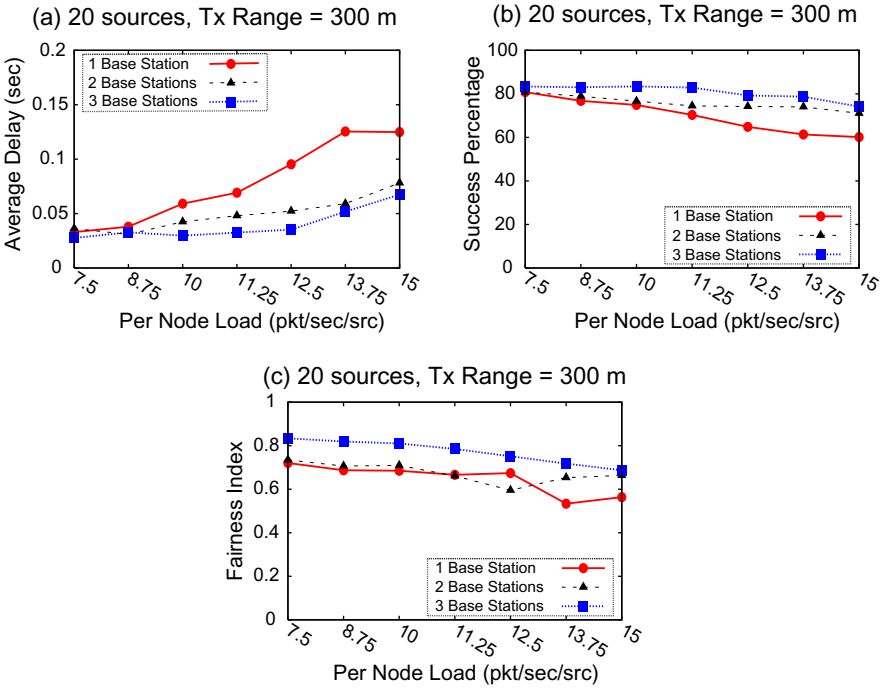

Fig. 7. Varying per-node load with 1, 2, 3 base stations.

each additional base station yields a smaller improvement in performance. While the second base station improved the average end-to-end delay by 40%, the third base station gave only a marginal 20% improvement. Similarly, the second and third base stations have improved the success percentage by 8% and 6%, respectively. Similar conclusions can be drawn from Fig. 8 – the performance improves with the addition of base stations. Note that the benefits

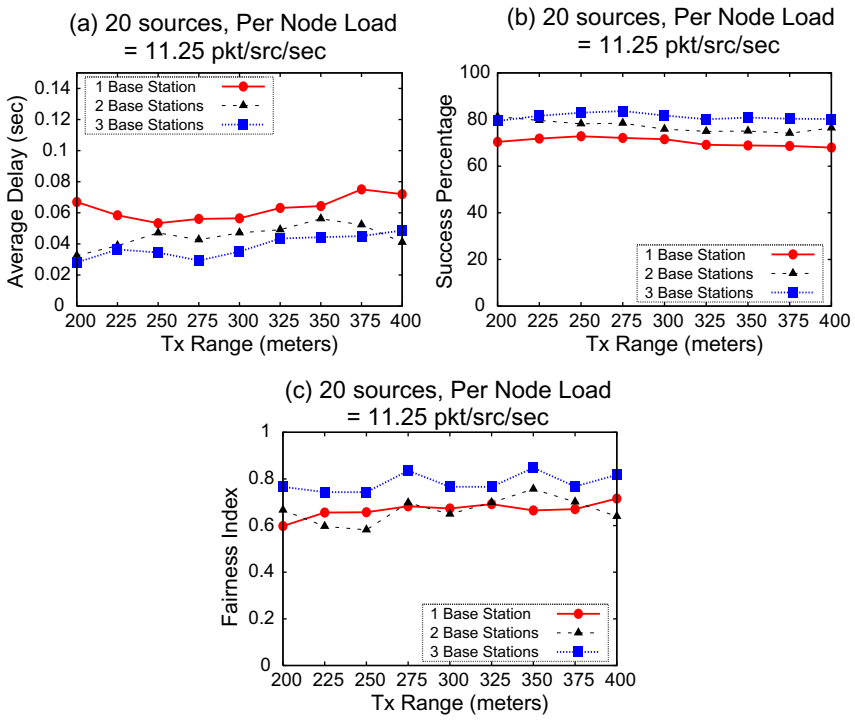

Fig. 8. Varying transmission range with 1, 2, 3 base stations.

from additional base stations overshadow the possible increase in the packet traffic overhead due to extra beacons. This experiment again highlights the fact that by carefully adjusting transmission range, we can tune the system to obtain the best performance for various numbers of base stations.

6.2.1.3. Effect of road block size. In the simulations described above, each block in the road network is assumed to be of 600 m. In this simulation, we study the effect of the road block size on the performance of our protocol. More specifically, we present results when the block size is set to 125 and 250 m. The number of sources (20 sources) and other default configurations are kept the same as the ones presented earlier in this section. Obtained results are shown in Figs. 9–12.

First, our protocol is able to estimate the delay accurately irrespective of the road block size. The difference between estimated and simulated (or observed) delay is small at all transmission ranges and transmission rates. The overall performance of our protocols with the road blocks of size 125 m is slightly higher than that of 250 m, which in turn is marginally higher than the results presented earlier with 600 m road block size. For example, from Figs. 10b, 12b, and 6b, the overall success rates are higher when the road blocks are smaller. This is because the packets can reach the base station quickly as they need to travel shorter distances. As the transmission range increases, the performance initially increases before reaching saturation point. After which, the performance drops slowly. For example, as shown in Fig. 10b and c, the saturation occurs around 300m range when the road block size is 125 m. Note that, after this point, the average packet delay increases. With larger transmission ranges, vehicles can send the packets to a larger distance in a single hop, thereby increasing the interference and collision probability. Both these factors result in slight performance degradation.

Even when we vary the packet generation rate, the results obtained are similar to the ones obtained using road block of size 600 m (see Figs. 9, 11, and 5). As the packet generation rate increases, the network load and the packet congestion increases. As a result, the performance decreases as the per-node load is increased. For example, in Fig. 9a, average packet delay starts to increase when the vehicles generate packets at a rate higher than 10 pkt/s/src. After this point, both the number of successful transmissions and the fairness reduce with increasing packet generation rate. Similar results can be observed when the road block size is set to 250 m (see Fig. 11).

Overall, at all road block sizes (125 m, 250 m, and 600 m), the observed trends are similar while changing the transmission ranges and packet generation rates. The performance of PROMPT is slightly higher when the road blocks are smaller. This is mainly because packets can be delivered over smaller number of hops.

6.2.1.4. Effect of vehicle density. In this experiment, we analyze the performance of our protocol as the vehicle density is varied. We set vehicle density to be 7 veh/km/lane and 15 veh/km/lane, which resulted in a total of 200 and 440 vehicles in the network. The results presented in Fig. 13

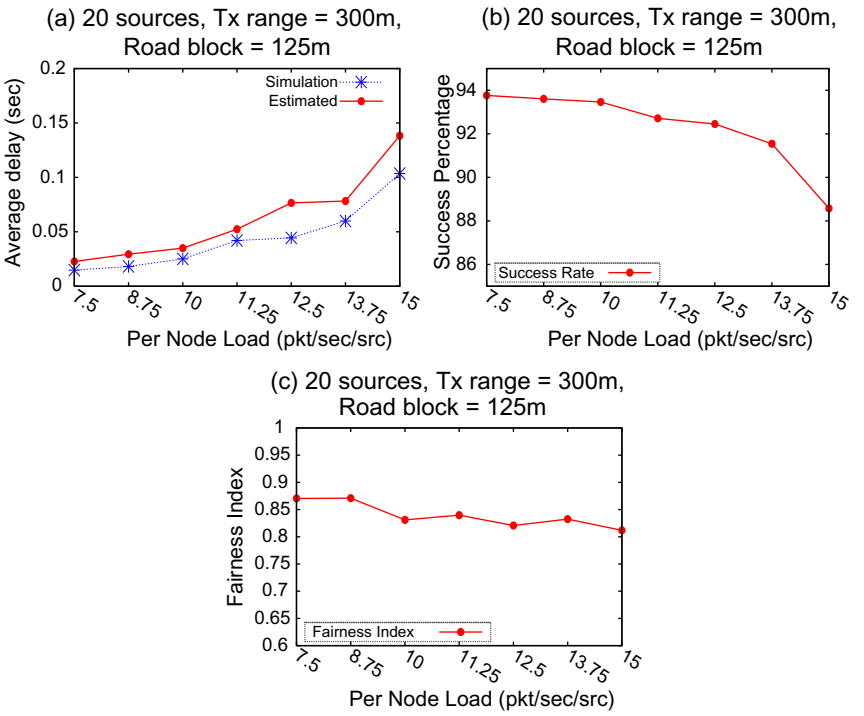

Fig. 9. Road block size = 125 m with varied transmission rate.

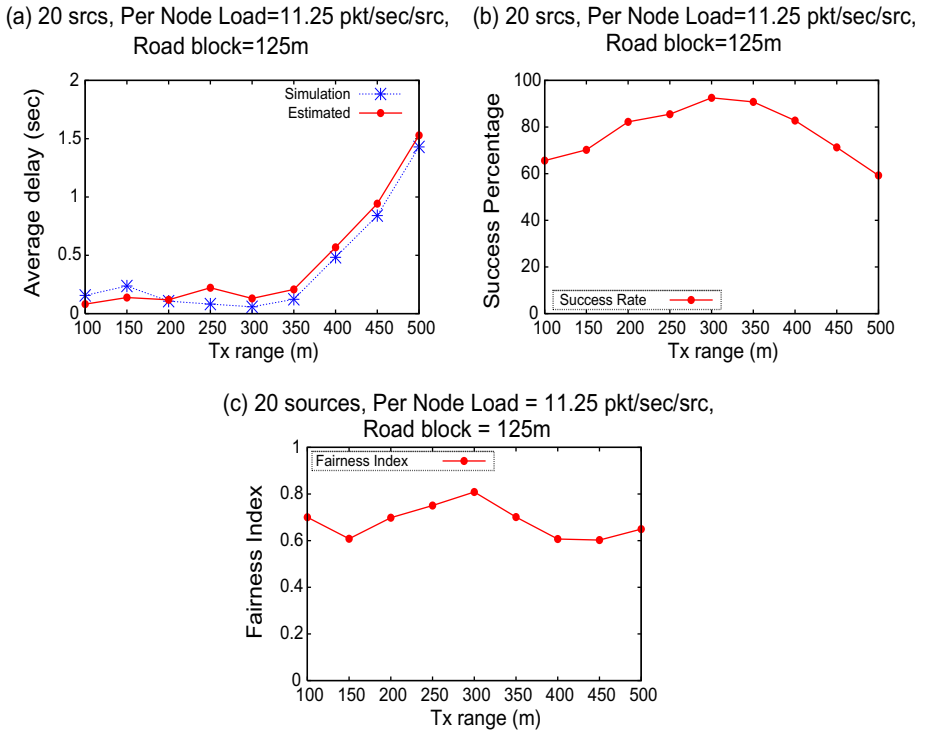

Fig. 10. Road block size = 125 m with varied transmission range.

are obtained using the transmission range of 300 m. Similarly, a packet generation rate of 11.25 pkt/s/src is used to obtain the results shown in Fig. 14. At a fixed transmission range, smaller vehicle densities lead to smaller number of successful transmissions (see Fig. 13b). This is because the nodes are likely to be far from each other when the density

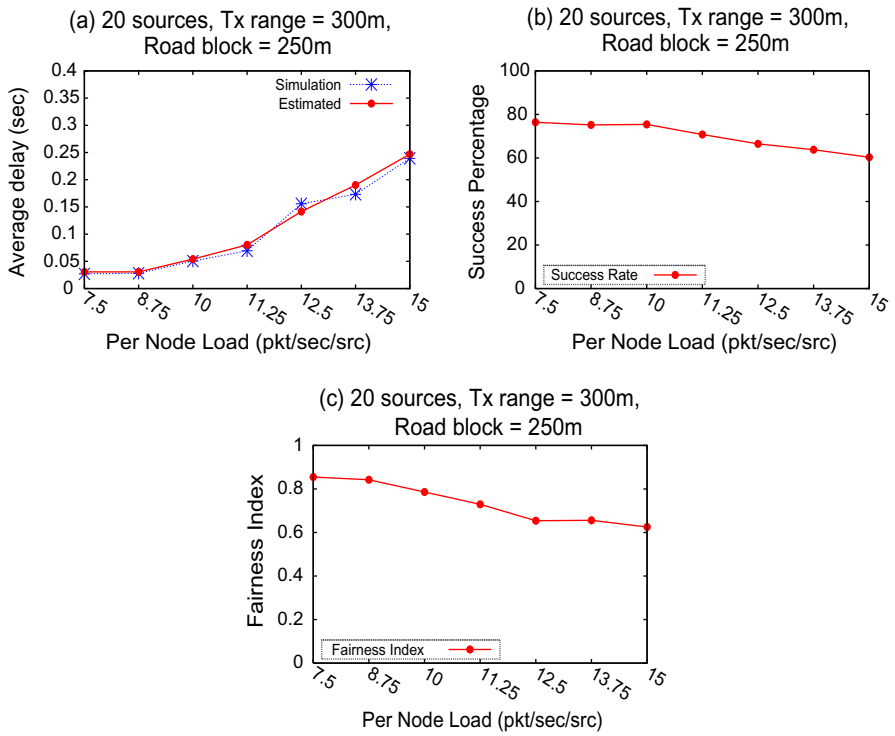

Fig. 11. Road block size = 250 m with varied transmission rate.

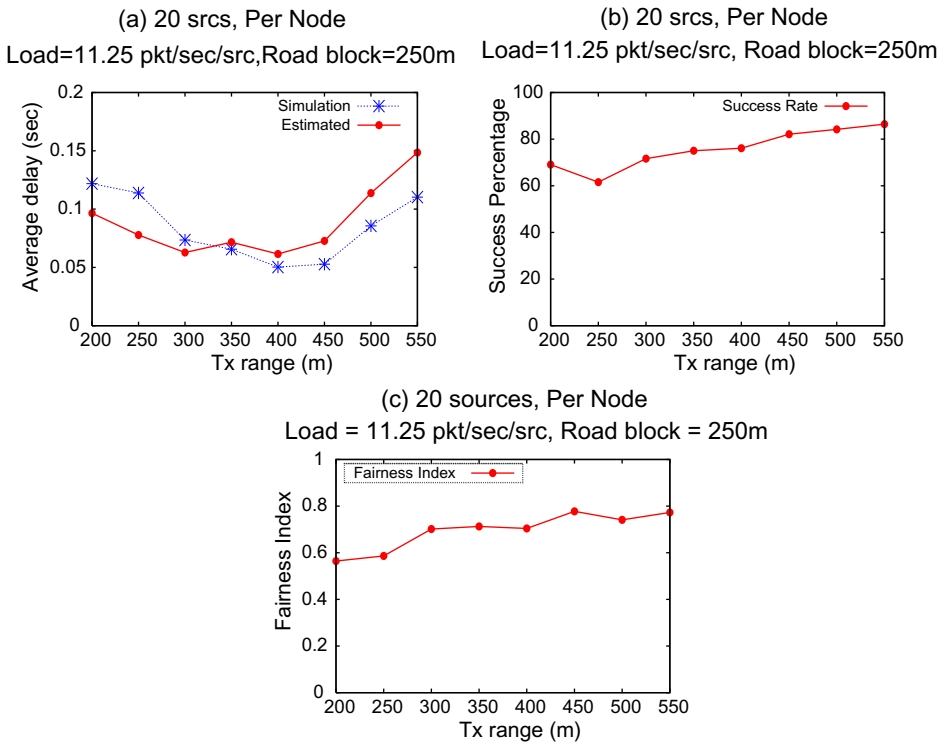

Fig. 12. Road block size = 250 m with varied transmission range.

is smaller. Therefore, a fixed transmission range may not help in finding the next relay node, thereby leading to path disconnection. However, when the transmission range is increased, the vehicles can find relay nodes and therefore the success rates are similar for both vehicle densities of 7 veh/km/lane and 15 veh/km/lane (see Fig. 14b). It is

important to note that the delay performance is similar irrespective of the vehicle density. The effect of vehicle density is only visible in terms of success rate.

6.2.1.5. Packet train. Recall that the delay estimation model relies on statistics such as average inter-arrival time and average service time. They are computed by considering a

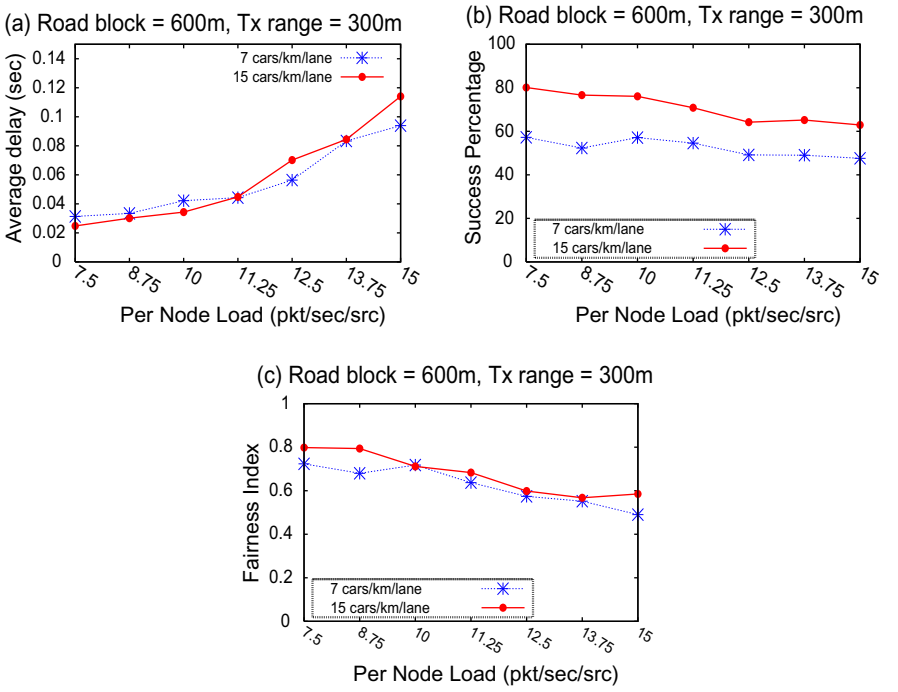

Fig. 13. Vehicle density: 7 veh/km/lane and 15 veh/km/lane with varied transmission range.

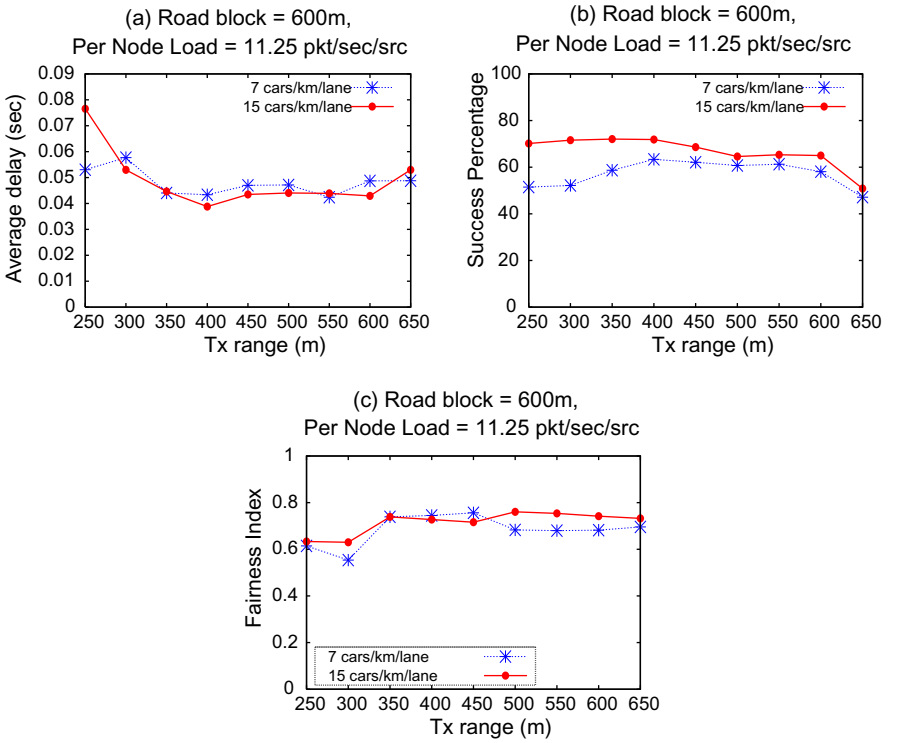

Fig. 14. Vehicle density: 7 veh/km/lane and 15 veh/km/lane with varied transmission range.

set of recent communication traffic values collected at the relay nodes. The number of values we consider, i.e., the *window size* may affect the accuracy of the estimation model. Fig. 15a shows that the average difference between estimated and actual delay increases exponentially as the window size is increased. Note that the *y*-axis is shown in log scale. With higher window size, considered communication traffic values may not reflect the current status of the network, and hence actual delay deviates more from the estimated delay.

Fig. 15b shows that the packet train mechanism improves end-to-end packet delay, especially at high load. When the packet load is high, bundling multiple packets into a single packet train reduces the contention overhead and queueing delay thereby improving the average end-to-end delay. We also found that this mechanism provides only marginal improvements in success percentage and fairness index. This is because the mechanism mainly attempts to improve the delay by reducing the overhead in packet transmission. In our empirical analysis, we found that the average packet train size to be 3.5 at high loads.

6.2.1.6. Packet load in different regions. Next we studied the effect of packet load in different areas in the road network. Fig. 16a shows the different regions (road segments) that we consider in this experiment. We simulated our protocol where 20 source vehicles send packets at the rate of 11.25 pkt/src/sec with 300 transmission range. We then collected the number of packets from the selected regions. The change in this number as we vary the average vehicle speed is shown in Fig. 16b. Note that these are the number of the packets that are *carried across* the selected road segments not the packets that are *generated*.

Evidently, the number of packets in Region 2 is higher than that of Region 1 and Region 3. This is because Region 2 is closer to the base station, the destination for all data packets. Since Region 1 is at the edge of the road, the number of packets in that road segment is smaller than that of Region 3, a road segment that is present in the middle of the road network. Observe that the data traffic in a given region is steady and does not fluctuate significantly with vehicle speeds. We note that the number of packets in fact depends on other relevant parameters like data traffic generation rate (i.e., per node load), transmission range, etc. We ran similar simulations with different configurations, and observed similar trends.

6.2.2. Comparative performance evaluation

We now compare the performance of PROMPT against other state-of-the-art protocols. For this purpose, we implemented CAR [3], GPSRJ+ [4], VADD [5], and DSR [2] protocols in ns-2. All the results described in this section are obtained with a single base station and 20 source vehicles.

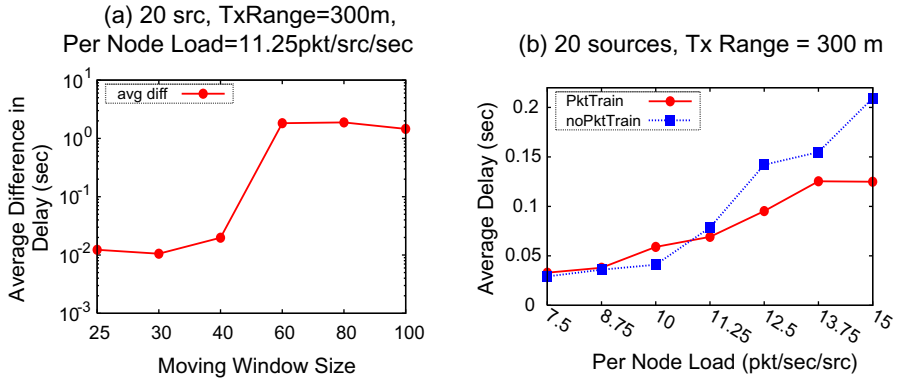

Fig. 15. (a) The average difference in estimated and actual delay, (b) effect of packet train mechanism.

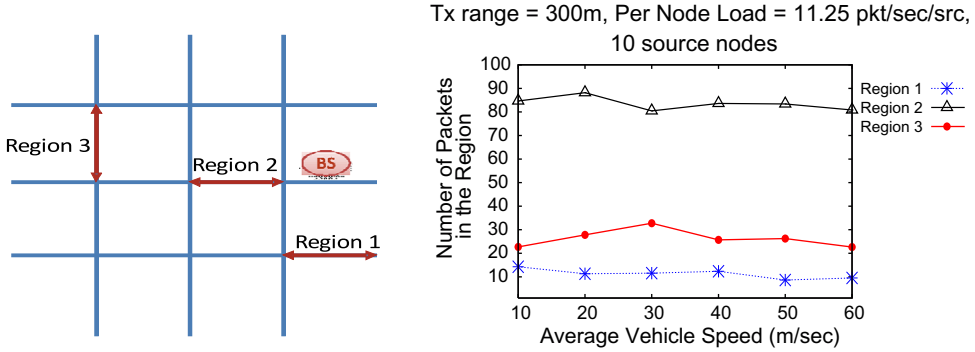

Fig. 16. (a) Regions, (b) packet load in regions.

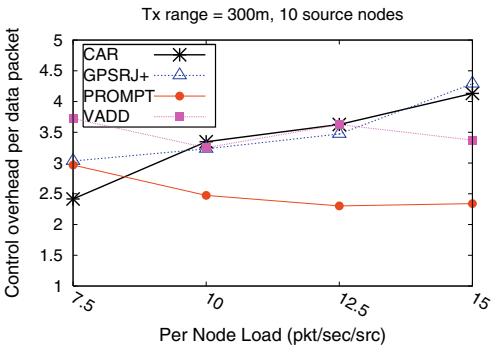

Fig. 17. Protocol overhead.

First, we study the control packet overhead of each protocol. We measure this overhead as the number of *control bytes* sent for each *data byte* that is generated. In PROMPT, there are several types of control packets – RTB, CTB, ACK, FRREQ, FRREP, etc. In Eq. (5), C1 is the control packets of DATA transmission. B1 and B2 are the control packets of the BEACON propagation. Similarly, other protocols involve control packets such as RTS, CTS, ACK, etc. (see Eqs. (6) and (7)). Fig. 17 presents the *overhead* for each protocol that is averaged over 20 different simulation runs. The overhead of each protocol is measured as in Table 1:

Let $n(\cdot)$ be the number of packets of a particular type, and $sz(\cdot)$ be the packet size in bytes. Table 1 gives the size of each control packet

$$\begin{aligned} F1 &= n(FRREQ)sz(FRREQ) + n(FRREP)sz(FRREP) + n(ACK)sz(ACK) \\ B1 &= n(RTB)sz(RTB) + n(CTB)sz(CTB) + n(ACK)sz(ACK) \\ B2 &= n(BEACON)sz(BEACON) \\ PROMPT &= \frac{F1 + B1 + B2}{num(DATA) \cdot size(DATA)}, \end{aligned} \tag{5}$$

$$\begin{aligned} C1 &= n(RTS)sz(RTS) + n(CTS)sz(CTS) + n(ACK)sz(ACK) \\ R1 &= n(RREQ)sz(RREQ) + n(RREP)sz(RREP) \\ H1 &= n(HELLO)sz(HELLO) \\ CAR &= \frac{C1 + R1 + H1}{num(DATA) \cdot size(DATA)}, \end{aligned} \tag{6}$$

$$\begin{aligned} GPSRJ+ &= \frac{C1 + H1}{num(DATA) \cdot size(DATA)}, \\ VADD &= \frac{C1 + H1}{num(DATA) \cdot size(DATA)}. \end{aligned} \tag{7}$$

Table 1

Control packet	
Packet type	Size (Bytes)
RTS, CTS	44
RTB, CTB	44
ACK	38
FRREQ	44
FRREP	44
HELLO	4–12
BEACON	68–544

Each relay node attaches PATH (relay node locations) and PathInfo (statistics) of size 68 bytes into the beacon message (BEACON). All the packet traffic statistics are included in those 68 bytes. In our simulations, we observed that the maximum length of a path, on average, is about 8 hops (TTL is set to 15 in our simulations). In such a case, the beacon will have a maximum of $8 * 68 = 544$ bytes. Note that this is the maximum value, and the actual size of the beacon message depends on how far it has traveled from its originating base station.

The number of control packets for all protocols increases as the system load increases. At higher packet generation rates, CAR protocol needs to send a larger number of route request and maintenance messages, resulting in increased number of control bytes for each data byte that is generated. Similarly, VADD and GPSRJ+ protocol maintains the neighborhood table using *hello* messages. The data transmission is done using *RTS* and *CTS* exchange. Therefore, as the number of generated packets increases the amount of control data transmitted over the network also increases. VADD always attempts to send packets in the path that is given by the source route. It employs a store-and-forward technique whenever a relay node is not found in the chosen route. While the functionality of GPSRJ+ is very similar to VADD, it does not follow a store-and-forward approach. This may cause the transmission over non-optimal paths and may potentially lead to higher control packet overhead, when compared to VADD.

Furthermore, PROMPT exploits the packet train mechanism as the system load is increased. Multiple data packets are bundled into a single packet train, thereby reducing the number of control packets. Therefore, the overhead due to control packets actually *decreases* as the system load is increased. Note that the number of beacon messages in PROMPT and depends only on the simulation time and the road structure.

The results in both Figs. 18 and 19 clearly show that PROMPT outperforms all other protocols with lower delays and high success rates. The average packet delay of PROMPT is too small, and is almost aligned with *x*-axis. We marked the 90% confidence intervals as error bars in Figs. 18 and 19. In some graphs, we avoided these intervals for the sake of clarity and added necessary. Among all four protocols, DSR exhibits the lowest performance. Since DSR relies on address based source routing, it is not suitable for vehicular networks where the constant movement of vehicles makes the address-based routing ineffective.

CAR protocol uses position-based routing, but the source routes are determined using explicit *route request* and *route reply* processes, which exchange metadata between the source vehicle and the base station. In addition, the *neighbor discovery* and the *route error recovery* processes add to the control overhead, resulting in high end-to-end packet delays. The fairness of CAR protocol is marginally better than PROMPT, but only at the expense of significantly higher delays and lower success percentage. CAR achieves this marginal improvement by reducing the load on the network by dropping packets. Further, as we increase the packet rate, the number of packets delivered to the base station drops at a higher rate in CAR when compared to PROMPT (see Fig. 18). Similar performance

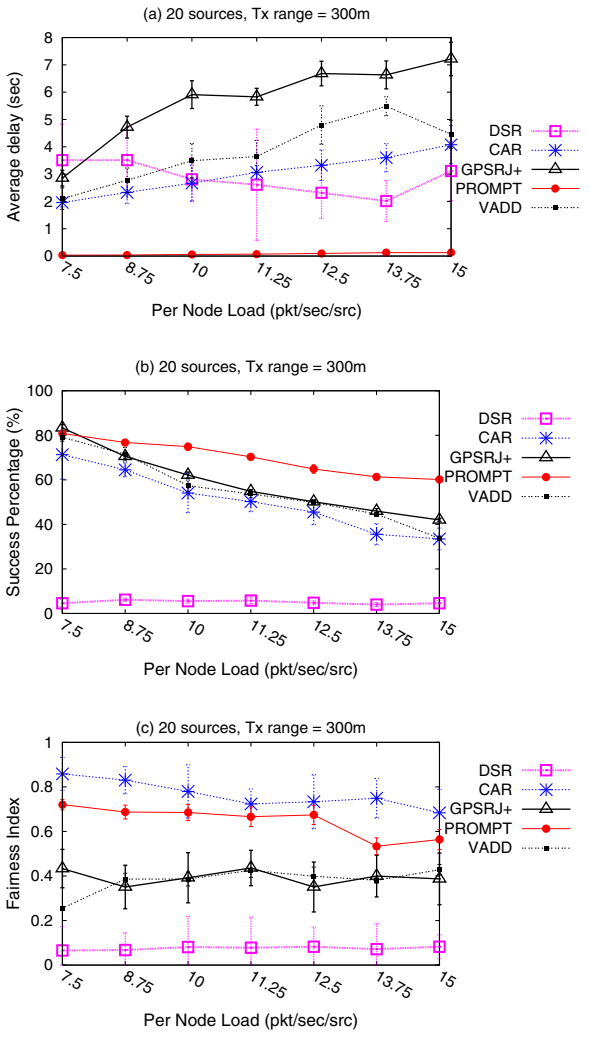

Fig. 18. Varying per-node load for 20 sources.

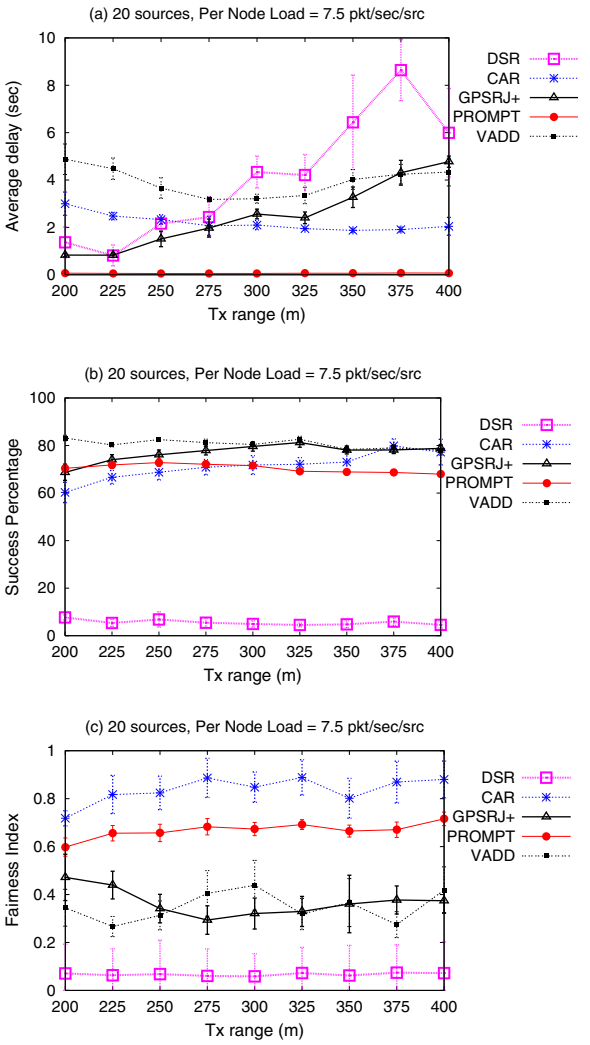

Fig. 19. Varying transmission range for 20 sources.

differences are observed with different transmission ranges as shown in Fig. 19.

GPSRJ+ protocol also uses position-based routes, and employ a neighbor discovery process. Each packet discovers its own route dynamically based on the neighbors of each relay node. Both the control overhead from neighbor discovery and the overhead of route recovery results in very high packet delays when compared to PROMPT. Such dynamic path decisions also do not guarantee any delays, and hence, GPSRJ+ is not suitable for delay sensitive applications. Further, the packets from vehicles far away from the base station suffer more than the those originating at vehicles near the base stations. In contrast, PROMPT treats packets from all vehicles in a more fair manner and takes explicit measures to guarantee fairness such as extra wait time after sending a packet train. Therefore, the fairness of PROMPT is higher when compared to GPSRJ+.

VADD is based on store-and-forward protocol where the relay nodes are selected via neighbor table. If the selected relay node is not reachable due to packet collision, interference, virtual channel sensing, etc., then VADD continues to retransmit the packet until the selected neighbor is found. Such a strategy incurs significant delay due to large number of retransmissions and higher contention delay. On the other hand, the source routes in PROMPT are based on streets and directions, which are derived from positional information. Any qualified node that is present in required packet direction can become the relay node. Thus the time spent in finding the next relay node is relatively small. In addition, our model makes use of packet train mechanism that bundles packets traveling in the same direction, resulting in smaller contention overhead. Although VADD estimates the path delay, the estimation procedure only considers the pre-loaded information, such as vehicle density. In contrast, PROMPT collects and leverages real-time packet traffic information in estimating the delay with good accuracy. This also helps PROMPT in avoiding congested paths while selecting the best available route.

Overall, PROMPT outperforms other state-of-the-art protocols in terms of all three performance measures, i.e.,

delay, success rate, and fairness. It does not have any explicit route discovery process and incurs low control overhead. Since the delay estimation model and the relay node selection process rely on statistics that depend on locations rather than individual nodes, PROMPT chooses best available paths and results in small delays despite network mobility. The position-based routing approach efficiently deals with vehicular mobility during packet delivery.

7. Conclusions and future work

In this paper, we propose a cross-layer protocol for VANETs that improves end-to-end delay based on the path information gathered while propagating beacon messages. We show through simulations that such local traffic data can be used to estimate end-to-end delays. Our extensive results show that our model predicts the delay with high accuracy, and hence can be used in delay-aware data delivery applications. PROMPT outperforms DSR, GPSRJ+, VADD, and CAR in term of the end-to-end delay, the percentage of success, and fairness.

We are currently in the process incorporating a realistic traffic model into our simulation. We plan to use the TIGER maps which provide more realistic road network than our simple grid network.

References

[1] G. Korkmaz, E. Ekici, F. Ozguner, A cross-layer multihop data delivery protocol with fairness guarantees for vehicular networks, IEEE Transactions on Vehicular Technology 55 (3) (2006) 865–875.
[2] D. Johnson, D. Maltz, J. Broch, DSR: the dynamic source routing protocol for multi-hop wireless ad hoc networks, Ad Hoc Networking 1 (2001) 139–172.
[3] V. Naumov, T. Gross, Connectivity-aware routing (CAR) in vehicular ad-hoc networks, in: The 26th IEEE International Conference on Computer Communications (INFOCOM'07), 2007, pp. 1919–1927.
[4] K. Lee, J. Haerri, U. Lee, M. Gerla, Enhanced perimeter routing for geographic forwarding protocols in urban vehicular scenarios, in: IEEE Globecom Workshops, 2007, pp. 1–10.
[5] J. Zhao, G. Cao, VADD: vehicle-assisted data delivery in vehicular ad hoc networks, IEEE Transactions on Vehicular Technology 57 (3) (2008) 1910–1922.
[6] C. Perkins, E. Royer, Ad-hoc on-demand distance vector routing, in: Second IEEE Workshop on Mobile Computing Systems and Applications, 1999, pp. 90–100.
[7] F. Li, Y. Wang, Routing in vehicular ad hoc networks: a survey, IEEE Vehicular Technology Magazine 2 (2) (2007) 12–22.
[8] B. Karp, H. Kung, GPSR: greedy perimeter stateless routing for wireless networks, in: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, 2000, pp. 243–254.
[9] C. Lochert, M. Mauve, H. Füßler, H. Hartenstein, Geographic routing in city scenarios, ACM SIGMOBILE Mobile Computing and Communications Review 9 (1) (2005) 69–72.
[10] A. Skordylis, N. Trigoni, Delay-bounded routing in vehicular ad-hoc networks, in: Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing, 2008, pp. 341–350.
[11] B. Seet, G. Liu, B. Lee, C. Foh, K. Wong, K. Lee, A-STAR: a mobile ad hoc routing strategy for metropolis vehicular communications, Lecture Notes in Computer Science (2004) 989–999.
[12] B. Wiegel, Y. Gunter, H. Grossmann, Cross-layer design for packet routing in vehicular ad hoc networks, IEEE 66th Vehicular Technology Conference (2007) 2169–2173.
[13] The GPS System. <http://www.kowoma.de/en/gps/errors.htm/>.
[14] G. Korkmaz, E. Ekici, Effects of location uncertainty on position-based broadcast protocols in inter-vehicle communication systems, in: Proceedings of the Fifth Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net 2006), 2006.
[15] A. Benslimane, Localization in vehicular ad hoc networks, Proceedings of Systems Communications (2005) 19–25.
[16] G. Korkmaz, E. Ekici, F. Ozguner, Black-burst-based multihop broadcast protocols for vehicular networks, IEEE Transactions on Vehicular Technology 56 (5 Part 2) (2007) 3159–3167.
[17] G. Curry, B. Deuermeyer, Renewal approximations for the departure processes of batch systems, IIE Transactions 34 (2) (2002) 95–104.
[18] R. Jain, D. Chiu, W. Hawe, A quantitative measure of fairness and discrimination for resource allocation in shared computer systems, arXiv preprint cs/9809099, 1998.

Boangoat Jarupan received her BS and MS degrees in Electrical and Computer Engineering from The Ohio State University in 1997 and 2000, respectively. She is pursuing her Ph.D. in the same university. Her research interests include mobile ad hoc communication systems with an emphasis on vehicular networks. She is currently working on cross-layer protocol design and multi-hop communication models for delay sensitive applications and intersection collision warning systems.

Eylem Ekici received his BS and MS degrees in Computer Engineering from Bogazici University, Istanbul, Turkey, in 1997 and 1998, respectively. He received his Ph.D. degree in Electrical and Computer Engineering from Georgia Institute of Technology, Atlanta, GA, in 2002. Currently, he is an Associate Professor in the Department of Electrical and Computer Engineering of The Ohio State University, Columbus, OH. His current research interests include cognitive radio networks, nano-scale networks, vehicular communication systems, and wireless sensor networks, with a focus on routing and medium access control protocols, resource management, and analysis of network architectures and protocols. He is an associate editor of Computer Networks Journal (Elsevier) and ACMMobile Computing and Communications Review. He has also served as the TPC co-chair of IFIP/TC6 Networking 2007 Conference.