Adaptive Traffic Signal Control With Vehicular Ad hoc Networks

Kartik Pandit, Dipak Ghosal, *Member, IEEE*, H. Michael Zhang, and Chen-Nee Chuah, *Senior Member, IEEE*

Abstract—**In this paper, we propose to use vehicular ad hoc networks (VANETs) to collect and aggregate real-time speed and position information on individual vehicles to optimize signal control at traffic intersections. We first formulate the vehicular traffic signal control problem as a job scheduling problem on processors, with jobs corresponding to platoons of vehicles. Under the assumption that all jobs are of equal size, we give an online algorithm, referred to as the oldest job first (OJF) algorithm, to minimize the delay across the intersection. We prove that the OJF algorithm is 2-competitive, implying that the delay is less than or equal to twice the delay of an optimal offline schedule with perfect knowledge of the arrivals. We then show how a VANET can be used to group vehicles into approximately equal-sized platoons, which can then be scheduled using OJF. We call this the two-phase approach, where we first group the vehicular traffic into platoons and then apply the OJF algorithm, i.e., the oldest arrival first (OAF) algorithm. Our simulation results show that, under light and medium traffic loads, the OAF algorithm reduces the delays experienced by vehicles as they pass through the intersection, as compared with vehicle-actuated methods, Webster's method, and pretimed signal control methods. Under heavy vehicular traffic load, the OAF algorithm performs the same as the vehicle-actuated traffic method but still produces lower delays, as when compared with Webster's method and the pretimed signal control method.**

Index Terms—**Conflict graphs, online job scheduling, traffic signal control, vehicular ad hoc network (VANET) simulation, vehicle-actuated traffic signal control, Webster's algorithm.**

I. Introduction

INTELLIGENT traffic signal control has been extensively studied in the literature [9], [25], [28]. Current methods of implementing intelligent traffic signal control include roadside sensors, such as loop detectors and traffic monitoring cameras. Loop detectors can only detect the presence or absence of vehicles [15], [16], which is a serious limitation. These loop detectors are physically connected to the traffic signal controller, and this connection is used to communicate the information gathered from the loop detectors to the traffic signal controller. The traffic signal controller then uses the data to schedule traffic through the intersection by cycling through preset phases and assigning appropriate amounts of GREEN time or skipping phases altogether. More recently, video-based traffic detection systems employing traffic monitoring cameras have been considered for traffic signal control. A prominent example of this is in Reno, NV, USA, where traffic data from video cameras is aggregated, and duration of red lights are adjusted based on current traffic volumes [1], [2]. While these have been effective, particularly to coordinate traffic conditions with known events, they require a high degree of human intervention.

In this paper, we examine the possibility of deploying an intelligent and real-time adaptive traffic signal controller, which receives information from vehicles, such as the vehicle's position and speed, and then utilizes this information to optimize the traffic signal scheduling at the intersection. This approach is enabled by onboard sensors in vehicles and standard wireless communication protocols specifically for vehicular applications. For example, all vehicles are already equipped with a speed sensor. In addition, new vehicles are increasingly being equipped with Global Positioning System (GPS) units that can provide location information with accuracy of a few meters [25]. Furthermore, vehicles can use wireless communications for vehicle-to-vehicle (V2V) or vehicle-to-infrastructure (V2I) communications, as described in the dedicated short-range communications/wireless access in vehicular environments standards operating in the spectral range of 5.85–5.95 GHz [19]. We refer to the transient mesh networks formed via V2V or V2I communication links as vehicular ad hoc networks (VANETs).

A. VANET Applications

The speed and location information on vehicles that can be disseminated to the traffic signal controller using VANETs [10] are both spatially and temporally fine-grained. Such precise per-vehicle speed and location information can enable additional capabilities such as being able to predict the time instance when vehicles will reach the stop line of the intersection. This is in comparison with roadside sensors such as loop detectors that can only detect the presence or absence of vehicles and, at best estimate, the size of vehicle queues. Furthermore, it is cheaper to equip vehicles with wireless devices than to install roadside equipment [25].

Traffic adaptive signal control has been widely studied. Examples include the well-known Split, Cycle, and Offset

Manuscript received June 15, 2012; revised October 15, 2012 and December 16, 2012; accepted January 9, 2013. Date of publication January 18, 2013; date of current version May 8, 2013. The review of this paper was coordinated by Dr. G. Mao.

K. Pandit and D. Ghosal are with the Department of Computer Science, University of California at Davis, Davis, CA 95616 USA (e-mail: kdpandit@ucdavis.edu; ghosal@ucdavis.edu).

H. M. Zhang is with the Department of Civil and Environmental Engineering, University of California at Davis, Davis, CA 95616 USA (e-mail: hmzhang@ucdavis.edu).

C.-N. Chuah is with the Department of Electrical and Computer Engineering, University of California at Davis, Davis, CA 95616 USA (e-mail: chuah@ucdavis.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVT.2013.2241460

Optimization Techniques (SCOOT) [16] and Sydney Coordinated Adaptive Traffic System (SCATS) [26]. SCOOT uses a loop detector as a sensor that is placed at the entry point of every link to an intersection. SCATS also relies on loop detectors, which are immediately placed before the stop line of an intersection. RHODES and its successor MILOS [14] are probably the most sophisticated traffic adaptive control systems that have been recently developed. They are also based on loop detectors, and they optimize lost times on a global scale. In MILOS, the traffic signal scheduling is done for a network of traffic controllers, including freeway ramp controllers. Loop detectors provide vehicle's position information to a central controller, which then generates schedules for the entire traffic network. In [7], a detailed survey of vehicle-actuated traffic signal control methods is given, both for one-way and two-way streets. The VANET-based vehicle-actuated traffic method is based on the study presented in [7], with additional enhancements that take advantage of the finer grain information enabled by a VANET. These enhancements take advantage of the ability of the VANET infrastructure to estimate when a vehicle is going to approach the stop line. The controller uses this information to extend the GREEN time by an appropriate amount so that the vehicle can pass through the intersection. Another example of VANET-based traffic signal control is Traffic View [25]. This work modified the Webster's method to leverage VANETs to communicate with the traffic signal controller.

VANETs have also been used to enhance other traffic control and management applications. The study in [10] presents a VANET-based method for variable speed limits to improve the flow of vehicles in freeways. In [21], VANETs are used to detect highway incidents and broadcast this information to drivers. In an extension to this work, [12] examines the "memory" that platoons of vehicles can keep to more efficiently broadcast freeway incident messages. VANETs have also been used in many driver experience improvement applications. For example, VANETs have been used to monitor road conditions in [18]. In addition to VANET, cellular communications have been used to design a system that estimates traffic delays in [11].

B. *Our Contributions*

In this paper, we present an algorithm, which we call the oldest arrival first (OAF) algorithm, that makes use of the per-vehicle real time position and speed data to do vehicular traffic scheduling at an isolated traffic intersection with the objective of minimizing delays at the intersection. This simple algorithm leads to a near optimal (delay minimizing) schedule that we analyze by reducing the traffic scheduling problem to a job scheduling problem, with conflicts, on processors. The scheduling algorithm captures the conflicts among opposing vehicular traffic with a conflict graph [9], and the objective of the algorithm is to minimize the latency values of the jobs. If the condition that all jobs require equal processing time is enforced, we can show that the OAF algorithm becomes the oldest job first (OJF) algorithm in the job scheduling domain with conflicts between jobs and the objective of minimizing job latency values. We present a 2-competitive (with respect to job latencies) online algorithm that does nonclairvoyant scheduling [27] with conflicts of the jobs on the processors and then prove a stronger result that the best possible nonclairvoyant scheduling with conflicts algorithm is 2-competitive.

We leverage a VANET to implement the OJF algorithm. An important requirement for the OJF algorithm is that all jobs require equal processing time. We give an algorithm that uses the VANET to divide up the approaching vehicular traffic into platoons that can be treated as jobs in the job scheduling with conflicts. The traffic signal controller can then use the conflict-free schedule from the OJF algorithm to schedule platoons of vehicles in a safe conflict-free manner. This two-phase approach, where we first use the platooning algorithm to divide up the traffic into platoons and then treat each platoon as an equal-sized job and then apply the OJF algorithm on the jobs to generate a conflict-free schedule, leads to what we call the OAF algorithm.

To ascertain the performance of the algorithm, we choose the average delay per vehicle that has passed through the intersection as the measure of effectiveness. We compare the performance of the OAF algorithm against an vehicle-actuated traffic signal controller, Webster's algorithm, and a fixed-time algorithm. The vehicle-actuated algorithm and Webster's algorithm are well-known traffic algorithms [9] that traditionally utilize fixed road-based sensors such as loop detectors. We have modified these methods to also utilize the information from VANETs, and we give the details in Section III. We also test the performance of the OAF algorithm in the scenario where only a proportion of the vehicles are VANET enabled. We conduct the experiments on a closed-loop VANET simulator that couples the realistic and the well-known Simulation of Urban Mobility (SUMO) traffic simulator [3] and the INET/OMNET++ [4] wireless simulator. This novel simulator realistically simulates the closed-loop interaction between the wireless communication characteristics and the mobility of the vehicles. The vehicular traffic simulator has a proven mobility model and is a widely used tool in industries and laboratories to do experiments on urban traffic operations research. The wireless simulator INET/OMNET++ also realistically simulates the wireless channel and wireless packet traffic delivery probabilities in the presence of multiple radio transmitters.

The rest of this paper is organized as follows. In Section II, we outline the OJF traffic signal algorithm and analyze its optimality. In Section III, we describe how a VANET and, more specifically, V2I communications can be leveraged to implemented the OAF algorithm. In Section IV, we describe the simulation tool that has been implemented to study the performance of OAF and compare it with other VANET-enabled traffic light scheduling algorithms. In Section V, we discuss the simulation results. Finally, in Section VI, we conclude with a discussion on future research directions.

II. Traffic Light Scheduling Reduced to Job Scheduling (OJF Algorithm)

Here, we propose a method to reduce traffic signal control problem to the problem of scheduling jobs on processors, and we propose an online job scheduling algorithm called the OJF

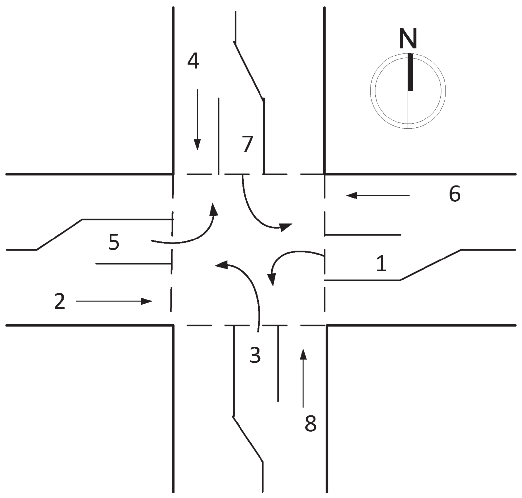

Fig. 1. Typical four-leg intersection showing the different movements on the approaches.

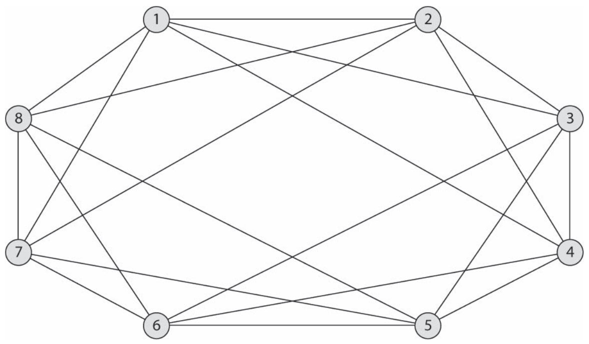

Fig. 2. Conflict graph for the intersection in Fig. 1.

algorithm. This is phase two of the OAF two-phase traffic signal control algorithm.

Fig. 1 shows a typical four-leg intersection with eight traffic movements numbered 1–8. This type of intersection is the most common and well-studied type [7], [8]. There are conflicts among some of these movements. For example, traffic movements 1 and 2 cannot simultaneously occur. We can reduce the problem of traffic signal control to scheduling of jobs on a processor, where a job is a platoon of one or more vehicles. We classify jobs as follows. A job is of type i if and only if the platoon of vehicles that it represents is part of traffic movement i. A pair of jobs of type i and j are said to be in conflict if the traffic movements i and j are in conflict; hence, jobs of type i and j cannot be scheduled to be simultaneously processed. For the intersection in Fig. 1, we can build a conflict graph $G(V, E)$, where V is a set of vertices, and E is a set of arcs. There is a vertex for each job type, i.e., for each job type i, $\exists$ vertex $i \in V$. The arc set E is constructed as follows. If jobs of type i, j are in conflict (and cannot be scheduled simultaneously), then there exists an arc (i, j) in E. E does not contain any other arc, and V does not contain any other vertex. The conflict graph for the four-leg intersection in Fig. 1 is shown in Fig. 2. Conflict graphs have been studied by traffic engineers to build safe traffic signal control plans. In [9], methods of developing safe signal control plans are shown for more complicated traffic intersections. We will assume that jobs are of equal size, and each job i of type j has an arrival time a_i^j, which would correspond to the instance of time when the first vehicle of platoon i arrives at the stop line in movement j. We will assume that time is divided into slots, and since all jobs are equal, without loss of generality, we can assume that all jobs need 1 unit of time to complete. Thus, if a job is scheduled at time t, it will complete at time $t + 1$. The ability to divide the oncoming traffic into platoons that require approximately equal amount of GREEN time (the green time represents the amount of processing time required) is achieved using a VANET. We discuss how this is done in Section III. At the beginning of time unit t, jobs of any type j can arrive, and we can think of them as arriving at vertex j in G. A group of vertices is chosen that do not conflict, and a job from each of these is scheduled in time t. Now, our objective would be to minimize the *maximum latency* over all jobs. For a particular job a_i^j, the latency is $d_i - a_i^j - 1$, where d_i is the time unit at the beginning of which job i has disappeared (completed), and a_i^j is the time unit at the beginning of which job i of type j arrived. Therefore, the objective is simply to minimize the maximum latency. In the context of vehicular traffic, minimizing maximum latency is equivalent to minimizing the maximum time that any vehicle spends at rest at an intersection waiting for green light. An important simplification that we make here is that all jobs need equal service time.

Job scheduling with conflicts is a well-studied problem. It was first identified in [20] as one of the famous 21 NP-hard (Complete) problems. It was shown that even approximating the problem is hard [23]. It is easy to see that the graph coloring problem can be reduced to job scheduling with conflicts. In particular, bipartite graphs can be colored in polynomial time, and the minimum number of colors needed is 2. A bipartite graph (or bigraph) is a graph whose vertices can be divided into two disjoint sets U and V, such that every edge connects a vertex in U to one in V, i.e., U and V are each independent sets. However, in our problem setting, we have no prior knowledge of the time instance at which jobs arrive. Consequently, any algorithm that schedules jobs in this setting cannot make any assumptions on the arrival times of jobs and can only schedule jobs that have already arrived at the vertices. Such a type of algorithm is called an *online algorithm*. In contrast, if an algorithm has prior knowledge of arrival times of all jobs, it might use this information to compute a better schedule. This type of algorithm is called an *offline algorithm*. An online algorithm is said to be c-competitive if, for any sequence of jobs, its cost on the sequence is at most c times the cost of the optimal offline algorithm on the same sequence plus an additive constant [13].

The techniques that are used to analyze online algorithms are described in [13]. In [27], a 2-competitive algorithm was shown that minimizes the makespan in the case of bipartite conflict graphs. More recently, in [17], it has been proven that, for the problem of minimizing maximum latency on conflict graphs, no algorithm can be better than a $n/4$-competitive algorithm, and they actually devise an algorithm that has maximum latency of $O(n^2T^2)$, where T is the maximum latency of any job in the schedule returned by the optimal offline algorithm, and n is the number of vertices. We give a 2-competitive algorithm for

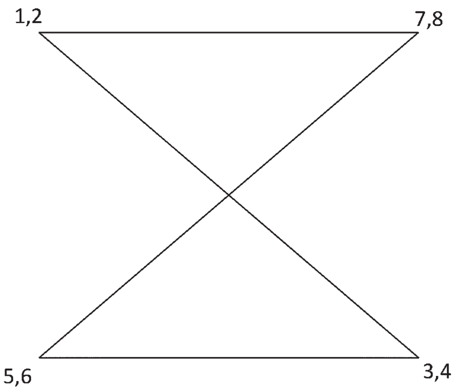

Fig. 3. Bipartite graph for the conflict graph G' in Fig. 2.

the much simpler case of a bipartite graph with $n = 2$. The proof in [17] can then be used to show that there cannot exist a better online algorithm.

A. 2-Competitive Algorithm for Job Scheduling

Having made the reduction from vehicular traffic scheduling to job scheduling with conflicts, we present a 2-competitive algorithm that minimizes latency for each job that we call the OJF scheduling algorithm. In addition, it turns out that, under the assumption of no future knowledge, this is the best possible online algorithm. The OJF scheduling algorithm can only be applied to bipartite conflict graphs; therefore, we need to do this transformation first. Graph G in Fig. 2 can be transformed into a bipartite graph G' by merging vertices 1 and 2, 3 and 4, 5 and 6, and 7 and 8. Fig. 3 shows the bipartite graph. We describe the OJF scheduling algorithm as follows.

Algorithm 1: OJF scheduling algorithm.

```
Let $a_i^r$, $a_j^{r'}$, $a_k^l$, and $a_m^{l'}$ be the earliest arrival times on each of the vertices of $G'$;
while $r$, $r'$, $l$, $l'$ have jobs waiting, do
    Let $a_t^s$ be the earliest arrival time among $a_i^r$, $a_j^{r'}$, $a_k^l$, and $a_m^{l'}$;
    Let $S$ be the side of $G'$ on which vertex $s$ lies;
    for Each vertex $s'$ on side S in $G'$, do
        Schedule the job with the earliest arrival $a_t^{s'}$;
```

Let r and r' be the vertices on the right side, and let l and l' be the vertices on the left side of the bipartite graph. Let L be the list of jobs that would arrive at the vertices in some time interval. Since we have no prior knowledge of the composition of L, the OJF algorithm aforementioned in Algorithm 1 makes decisions on the fly to reduce the maximum latency and is hence an *online algorithm*. For example, there exists an algorithm A^* that, given L, generates the optimal schedule (a schedule that minimizes maximum latency). A^* is the optimal *offline algorithm* (see Table I for notations). Let us compare the performance of OJF and A^* when it comes to minimizing the maximum latency. We claim that the OJF scheduling algorithm is 2-competitive, i.e., for any L, OJF produces a schedule where the maximum latency experienced by any job is at most twice the maximum latency experienced by any job in a schedule produced by A^*. Thus, the OJF algorithm is 2-competitive. Furthermore, it turns out that there cannot exist a better than 2-competitive algorithm for job scheduling under the assumption of no future knowledge.

To prove that OJF is 2-competitive, we need the following lemma.

Lemma 2.1: Let the *weight* of a vertex be the number of jobs waiting on it. The weight of an arc in G' is the sum of the weights of its two vertices. For example, T is the maximum latency in the schedule for L returned by A^*. Then, OJF always maintains the following for all time t.

1) If A^* has an arc of weight w at some time unit t, then the optimal schedule has at least $w - T$ jobs on the same arc at time t.
2) If A^* has a vertex of weight w at some time unit t, then the optimal schedule has at least $w - T$ jobs on the same vertex at time t.

Proof: See the Appendix. ■

Theorem 2.2: OJF is 2-competitive.

Proof: We will prove that, for any L, if the schedule generated by A^* for L has maximum latency T, then OJF will generate a schedule that has latency at most $2T$. As long as the two conditions specified in the lemma are maintained, there can never be an arc of weight $2T + 2$ or more as algorithm OJF runs, since otherwise (by the lemma) there would be an arc of weight at least $T + 2$ and then the schedule produced by A^* would have a latency of at least $T + 1$ on some job. Therefore, OJF never has more than $2T + 1$ jobs on an arc, and when job j arrives on vertex l, there are never more than $2T$ other jobs on any arc going into to l. Let X be the number of jobs already on l when the job j arrives, i.e., $0 \leq X \leq 2T$. There are at most $2T - X$ jobs on any vertex on the right side. Once the left side has been chosen x times by OJF, j will be the oldest job on vertex l; therefore, it will be scheduled the next time the left side is chosen by OJF. If we can prove that the right side is not chosen more than $2T - X$ times before j, then we know that j incur latency at most $2T$ before it is scheduled. After the right side has been chosen $2T - X$ times, if j has not yet been chosen, then the left side has the oldest job in the system. This gives us our result. ■

The given discussion shows that, if we can do the reduction from vehicular traffic scheduling to job scheduling correctly, we can employ the OJF algorithm to generate schedules that will then be applied to schedule vehicular traffic at intersections while maintaining the 2-competitive performance bounds.

B. Optimality of the OJF Algorithm

Here, we prove that the algorithm presented earlier obtains the optimal competitive ratio. We adopt the proof for a bipartite graph with n vertices given in [17]. The method is based on an adversary technique in which the adversary creates a sequence of job arrivals based on the behavior of the online algorithm. At the beginning of each time unit, the adversary can determine how many jobs arrive and on which vertices of the conflict graph. After the entire job sequence has been determined, the

TABLE I
SEMANTICS OF NOTATIONS USED IN THE OJF SCHEDULING ALGORITHM

L	A sequence of jobs where for job $j \in L$, job j arrives at the start of time unit $t = a_j$
A^*	The optimal off line job scheduling algorithm
$G = (V, E)$	The conflict graph shown in Figure 2. V is the set of vertices and E is the set of arcs
G'	The Bipartite graph in Figure 3 formed by merging equivalent vertices in G
V	A set of vertices such that for each traffic movement v, there exists vertex v in V
E	A set of arcs such that $(u, v) \in V$ iff the traffic movements corresponding to vertices u, v conflict

adversary then can determine a schedule for the jobs in an offline manner. The adversary tries to create the worst sequence possible for the online scheduling algorithm. The cost of the online algorithm is then compared with the cost of the adversary determined offline algorithm. For a four-leg intersection, we consider to reproduce the same proof for the much easier case of $n = 4$, which makes the proof shorter.

Lemma 2.3: For example, OJF is an arbitrary scheduling (online) algorithm. Suppose that, at the end of time unit t, the adversary has no jobs and OJF has i jobs on a single arc (l, r). Then, the adversary can create a sequence of jobs where OJF has an arc of weight $i + 1$ while the adversary has no jobs left on any vertices. Furthermore, the adversary never has a job with latency more than $i + 1$.

Proof: See the Appendix. ■

Theorem 2.4: For the case of bipartite conflict graphs, there is no online algorithm, which is c-competitive with $c < 2$.

Proof: Let OJF be an arbitrary scheduling (online) algorithm. We will show that, for T, an adversary can force the algorithm to have a job of latency at least $2T$, whereas the adversary only has a latency of at most T.

Let j vary from 0 to $T - 1$. The algorithm will start each phase with j jobs on an arc. Lemma 2 is used to obtain an arc with $j + 1$ jobs. At the end of the whole process, OJF has an arc, e.g., (l, r), with T jobs, whereasthe adversary's graph is empty. For the next T time units, the adversary has a job arrive on l and r. By always scheduling the oldest job, the adversary never has a latency value of more than that of OJF. OJF, on the other hand, will have $2T + 1$ jobs on arc (l, r) and must incur a latency of at least $2T$. This proves the theorem. ■

We have proved in Theorem 2.2 that OJF is 2-competitive on G', i.e., the bipartite conflict graph. Thus, we need to prove that OJF is also 2-competitive on G, i.e., the original conflict graph. This is proven in lemma 2.5 as follows.

Lemma 2.5: Suppose that OJF produces a maximum delay of $2L$ on G'. Then, an optimal job scheduling algorithm on G will produce a maximum delay of at least L.

Proof: Since OJF produces a maximum delay of $2L$, using Theorem 1, we know that the optimal algorithm on G' produces a maximum delay of at least L. Suppose there exists an optimal algorithm that produces a maximum delay of less than L on G. Because of the way we did the reduction from G to G' and the fact that only one job can be scheduled at a time whenever there is a conflict, this optimal algorithm on G can also be applied on G' to produce a maximum delay of less than L. This contradicts Theorem 1. Hence, the optimal algorithm on G will produce a maximum delay of at least L. ■

III. VEHICULAR AD HOC NETWORK-BASED TRAFFIC INTERSECTION CONTROL

Here, we show how we implemented the platooning phase (phase one) of the OAF algorithm and how we implemented the other traffic light control schemes, such as the vehicle-actuated logic and Webster's method using VANETs. We first explain some of the terms used in describing our adaptive traffic control algorithms that may differ slightly from their conventional definitions.

- MAX-OUT: The maximum amount of GREEN time that can be allocated to the current phase.
- GAP-OUT: If a vehicle is more than the GAP-OUT units of time away from the stop line, then the signal goes to the next phase.
- EXTENSION: If a vehicle is detected less than GAP-OUT units of time away from the stop line, then the GREEN time is extended by EXTENSION units of time.

A. System Description

In this paper, we only study an isolated intersection. Fig. 1 shows the single traffic intersection under consideration. It is a typical four-leg intersection with eight traffic movement groups represented by the arrows. Each of the legs of the intersection is L meters long, and each of the left turning bays is B meters long. The numbered arrows show the directions of the various traffic movements. For this type of traffic intersection, we now describe the system architecture of the VANET-based traffic signal controller. In the single traffic intersection scenario, the traffic signal controller is connected to a wireless receiver that is placed at the intersection. The wireless receiver listens to information being broadcast from the vehicles. The broadcast medium is the 5.9–5.95-GHz radio spectrum, and the communication standards are defined in the IEEE 802.11p standards [19]. This system architecture is shown in Fig. 4. The information consists of speed and position data collected from vehicles. Speed data can be gathered from the vehicle speedometers, and position data can be gathered using GPS receivers fitted to the vehicles. In our implementation, the following data are gathered and encapsulated in data packets that are broadcast over the wireless medium. This is what we call the data dissemination phase.

- Vehicle ID: Every vehicle is uniquely identified by its Vehicle ID#. In our traffic simulator SUMO, every vehicle is identified by a unique unsigned integer. In practice, the medium access control (MAC) address of the network

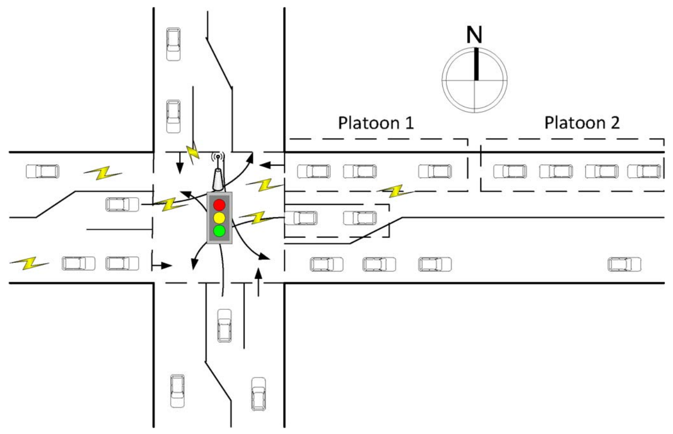

Fig. 4. VANET-based traffic signal control architecture.

interface card in the wireless receiver would serve the same purpose.

- Location: In SUMO, the location of each vehicle is specified by the LINK NUMBER#, Lane#, and position from a point of reference. The position from a point of reference is a subfield containing (x, y), which are floating point quantities. We chose to use the stop line as a point of reference; therefore, the stop line has position (0, 0) for each Link Number# and Lane#. Thus, collectively, these three fields describe vehicle location. In practice, it is assumed that each vehicle is equipped with a GPS receiver; therefore, vehicles always know their locations. It is possible to convert the GPS coordinates of each vehicle to the format that we described earlier. We will show later that we can compute the distance from each vehicle to the stop line from this information.
- Speed: Speed of a vehicle is a floating point quantity expressed in meters per second and is obtained from the in-vehicle speedometer sensor.
- Current Time: The time at which the packet was created. The format is (hh:mm:ss). Because of the nature of the traffic control application, there is no need for a finer grain time. However, we need to assume that all clocks are synchronized. The current time is required to distinguish between old packets and new packets.

After the data dissemination phase, we have the data aggregation and processing phase where we actually make use of the transmitted information to do traffic signal control. The processing logic that does this consists of the adaptive traffic signal control algorithms, such as the adaptive Webster's method and the vehicle-actuated traffic control algorithm. These algorithms are contained in the traffic signal controller. The details of the data aggregation phase and the processing phase are closely linked with the type of adaptive traffic signal control algorithm used, and we describe these details in Section IV.

B. Platooning Algorithm

In Section II, we obtained the lower bounds on how well an online algorithm can perform when it comes to minimizing the maximum latency. These lower bounds were achieved by an online algorithm that had no knowledge of future inputs. Can we use information gathered from the VANET to obtain future knowledge of traffic and use this to obtain a better-than-2-competitive algorithm? Unfortunately, this is unlikely; since due to radio range limitations, the VANET can only provide a relatively myopic view of the future, and in the long run, we will fall back to a 2-competitive performance. However, we can use the information from the VANET in a different way. One of the conditions under which the performance bounds hold is that all jobs, which represent platoons of vehicles, are of equal size and hence require equal processing time. This means that, for the OJF algorithm to be effective, all platoons must require equal amounts of time to pass through the intersection. We can achieve this requirement by using the vehicle position and speed data obtained via the VANET to compute the spatial headways between the vehicles. We can then divide the vehicles into platoons using this headway information, where each platoon takes equal amount of time to pass through the intersection. This platooning phase will be the phase one of the OAF algorithm, with the OJF being phase two of the OAF algorithm.

In addition, in our job scheduling transformation, we assumed that a job completes at the end of the slot in which it is scheduled. Under this assumption, the OJF scheduling algorithm produces optimal schedules. However, if we treat each vehicle as a job (platoons of size 1 vehicle) and apply the OJF scheduling algorithm, then in the worst case, the OJF algorithm would behave similar to a stop sign and produce very high delays. This is because there is a delay experienced by vehicles as they accelerate through the intersection once they are scheduled, which is called the startup delay. By scheduling platoons that contain a large number of vehicles, we can amortize the startup delay over a large number of vehicles.

To solve the two problems discussed earlier, we propose to formulate an optimization problem that decides platoon sizes with the objectives being the following.

1) Select the platoon size that minimizes the difference in times required to service platoons of vehicles.
2) Maximize the size of the platoons.

The way we solve this optimization problem is to estimate the amount of GREEN time that a platoon needs and find the platoon configuration that minimizes the difference between the maximum and minimum GREEN times for that configuration. We can estimate the GREEN time necessary to service a platoon as follows.

- If platoon has stopped at the stop line: Green time = start-up time + time for platoon to pass through the intersection.
- If Platoon is in motion: Green time = time for platoon vehicles to pass through the intersection.

We can estimate the time for a platoon to pass through the intersection as

$$1.5 + h_1 + \cdots + h_n \tag{1}$$

where the h_i values are the headways of the $1 \leq i \leq n$ vehicles in the platoons, and 1.5 is a constant that accounts for the startup delay of the very first vehicle in the platoon. Headways are defined as either the distance between two vehicles or the time between two vehicles. We define the headway h_i as the time

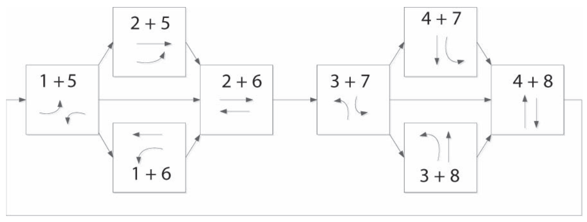

Fig. 5. Traffic signal phase sequence for the VANET-based vehicle-actuated traffic signal control. The numbers refer to the traffic movements shown in Fig. 1.

between vehicle i and $i + 1$ in a platoon. We can estimate h_i by measuring the distance between vehicle i and vehicle $i + 1$ and divide by the current speed of the vehicles.

The platooning algorithm is an exhaustive search over all the platoon configurations to determine the platoon combination that minimizes the difference between the maximum and minimum GREEN times. For n vehicles, we first generate all the platoon combinations using IntegerPartitions$[n]$, which generates all partitions of an integer n. Each partition represents a platoon configuration. For example, $n = 10$, then a possible partition is 3, 2, 2, and 3, which would represent a platoon configurations containing platoons of size 3, 2, 2 and 3. Since the vehicles arrive on a leg of the intersection, only a platoon size is required to identify a particular platoon. The constraint on the search result is that the maximum service time for a platoon in the configuration is less than or equal to MAXGREEN. Once the platoon size of the head-of-line platoon is determined, it does not change. The head-of-line platoons are then scheduled using the OJF scheduling algorithm shown in Algorithm I. We show that the platooning algorithm in Algorithm 2, where Estimate_Green_Time(j) is computed from (1).

Algorithm 2 Platooning Algorithm

```
for each approach $k$ do
    Configuration = IntegerPartitions$(n)$
    for each platoon configuration $i$ in Configuration do
        for each platoon $j$ in $i$ do
            Platoon_Green_Time$[j]$ =
            Estimate_Green_Time$(j)$;
        Add Platoon_Green_Time$[j]$ to the list
        Config_Green_Time$[i, k]$;
Min_Diff =
$\min_{i\in k,k=\{1,\ldots,4\}}\{\max\{$Config_Green_Time$[i,k]\} -$
$\min\{$Config_Green_Time$[i,k]\}\}$;
Final_Platoon_Configuration =
    $\arg\min_{i\in k,k=\{1,\ldots,4\}}\{\max\{$Config_Green_Time$[i,k]\} -$
$\min\{$Config_Green_Time$[i,k]\}\}$;
```

C. Vehicle-Actuated Traffic Signal Control

Here, we explain how the vehicle-actuated traffic signal control method is implemented in a VANET environment. Fig. 5 shows the phase sequence of the traffic signal controller. We initialize the traffic signal controller to the initial phase and initially set the EXTENSION time for the phase to 0. Next, we search for a vehicle that is closest to the stop line by examining the location field of all the vehicles. We compute the approximate traveling time to the stop line using the Compute_Traveling_Time() function as follows. The packet broadcast by the closest vehicle contains its position and speed data. These data are extracted, and since the position data consist of a Cartesian coordinates, we can compute the Euclidean distance of the vehicle from the stop line. Given the distance of a vehicle from the stop line, we can use the current speed information to compute the traveling time to the stop line. This traveling time is an approximation of the actual traveling time. If the vehicle closest to the stop line indicates a speed of 0, Compute_Traveling_Time() returns a traveling time of 2 s [28]. We set variable GAP to be equal to the traveling time returned by Compute_Traveling_Time(). If GAP is less than GAP-OUT, then the phase is the allocated EXTENSION units of GREEN time. If there is no close vehicle, then all packets received from vehicles would indicate that all vehicles are more than the GAP-OUT amount of time away from the stop line, and the signal controller would GAP-OUT and go to the next phase, indicating 0 GREEN time.

The magnitude of the EXTENSION is set to EXTENSION + GAP. Then, the GAP for the next closest vehicle is computed using Compute_Traveling_Time(), and the process is repeated. The EXTENSION accumulates while it is less than or equal to MAX-OUT. Once it crosses the MAX-OUT threshold, the signal controller switches to the next phase. The RED plus GREEN times are set to 5 s. Fig. 5 shows the sequence of phases. The sequence of phases is represented as a graph.

This represents the set of vertices consisting of the phases, which are pictured as the square boxes in Fig. 5. Each phase is a combination of traffic movements represented by the numbers within the square boxes. The edges between the vertices represent phase transitions.

D. Webster's Method

Webster's algorithm [29] is the most quoted method of determining a delay minimizing cycle time or evaluating delay for a cyclic fixed signal control scheme. In the Webster's algorithm, simulation tools are used to generate random vehicle arrival times to the intersection at a given average arrival rate. Arrivals to the stop line are added to a queue estimate and dispersed during the effective GREEN time at a constant departure rate called the saturation flow rate. Delay is calculated as the integral of the queue over the cycle, and an average value is obtained by dividing the delay by the volume. Webster used the result of simulation analysis to deduce a model of average delay per vehicle as a function of the cycle time, GREEN split, saturation flow rate, and arrival rate. In particular, the average delay per vehicle on the particular leg of the intersection, which is denoted by d, is given by

$$d = \frac{c(1-\lambda)^2}{2(1-\lambda x)} + \frac{x^2}{2q(1-x)} - 0.65\left(\frac{c}{q^2}\right)^{0.5} x^{(2+5\lambda)} \quad (2)$$

where c is the cycle time; λ is the proportion of the cycle, which is effectively GREEN for the phase under consideration;

q is the flow rate; s is the saturation flow; and x is the degree of saturation. This is the ratio of the actual flow to the maximum flow that can be passed through the intersection. The first two terms are theoretically derived, and the last term is a correction factor to account for the difference between empirical and theoretical results. The first term is the delay for uniform arrivals, and the second term is the additional delay for Poisson arrivals [24].

In this paper, we consider a simplified alternative, suggested by Webster, where the third term is dropped, which generally reduces the value by about 5% to 15%, and 0.9 is multiplied by the sum of the first two terms [24]. Among approaches served by a given phase, the approach with the highest degree of saturation is often referred to as the critical lane group or the critical movement. The following equation gives the delay minimizing cycle time as a function of lost time per cycle and critical movement saturation levels. The optimal cycle time C_o is given by

$$C_o = \frac{1.5L + 5}{1 - Y} \tag{3}$$

where y_i is the degree of saturation for the critical movement relative to phase i, $Y = \sum_i y_i$ for all phases i; and L is the total lost time per cycle. This is the sum all-red clearance intervals and lost time (due to startup or yellow) over all phases in the sequence. Phases are allotted GREEN time in proportion to the degree of saturation on their corresponding critical lane groups. This simple rule, known as the critical movement approach, has been found effective in minimizing vehicle delay.

We have extended Webster's method to make use of the data collected by the wirelessly enabled vehicles. This works by measuring traffic flow on each of the lanes of the intersections. The optimal cycle times are computed using the expression given, and the optimal GREEN time G_i is given by

$$G_i = \frac{(C - L)y_i}{\sum_j y_j}. \tag{4}$$

One parameter that needs to be chosen is the length of the interval during which to measure the traffic flow. During periods of high variance of flow, we choose short intervals (5 min in our experimental studies) and longer intervals (2 h) when the variance is low. We can also choose to measure traffic flow during a cycle, e.g., C_i, and then use the data collected to determine the length of the next cycle c_{i+1}. However, our results show that there is negligible difference between this and a measuring interval of 5 min.

IV. Simulation Model

We have developed a simulator that integrates a vehicular traffic simulator and a wireless network simulator to produce a closed-loop simulation environment. The vehicular traffic simulator is the SUMO traffic simulator [3], [5], which is a space continuous microscopic simulator for vehicular traffic. The SUMO simulator is a C++-based open-source highly portable microscopic road traffic simulation package designed to handle large road networks. The underlying vehicular traffic model has been validated in [22]. The wireless network simulator is the OMNET++/INET [4] wireless network simulator, which is a component-based modular open-architecture discrete-event network simulator implemented in C++. One important reason for choosing OMNET is that it implements the IEEE 802.11p standard at both the physical and the MAC layers.

To connect the discrete-event simulator OMNET++/INET with the continuous simulator SUMO, we used the TRACI interface [6]. SUMO runs as a process that cannot be accessed during the execution of a simulation step. TRACI uses a client/server architecture; SUMO is configured as a server, and using methods provided by TRACI allows client applications to connect SUMO to TRACI, providing access to SUMO. The architecture is actually based on Transmission Control Protocol/Internet Protocol (TCP/IP); therefore, applications connect to SUMO via a TCP socket. The client send commands to SUMO to control the simulation run, to influence a single vehicle movement, or to request for environmental details. SUMO responds with a status response to each command and a traffic trace after executing a single step. Both the commands to SUMO and the traffic traces from SUMO are transported using TCP. Each TCP segment consists of a small header that gives the overall message size and a set of commands or traffic traces contained in the segment. As shown in Fig. 6, TRACI can be used to allow a connected OMNET++ instance to send a series of commands to the traffic signal controller, influencing the mobility of SUMO vehicle instances.

Every vehicle in SUMO is mapped to a mobile node in OMNET++. We have extended OMNET++ with a module that allows creation of new nodes as vehicles are injected into SUMO, deleting nodes when their corresponding vehicles reach their destination and reflecting the movement of their corresponding vehicles in SUMO. SUMO executes in discrete time steps, and the traffic trace generated by SUMO is parsed by a manager module. The manager module then communicates the traffic trace to the OMNET wireless nodes. OMNET then executes one time step, and the wireless packet delivery trace generated by OMNET is parsed by the data aggregation module. It is the data aggregation module that actually generates the traffic light commands that are sent to the traffic simulator. Therefore, in our implementation, the data aggregation module encapsulates the adaptive traffic signal control algorithms for the traffic lights. The timing sequence diagram of these events is shown in Fig. 6. OMNET++ also uses the TRACI interface to send the vehicle control information in the form of TRACI commands, as shown in Fig. 6. The commands here are SIM_SIGNAL_GREEN and SIM_SIGNAL_RED, which identify a traffic signal controller and make it cycle through its phases. During the simulation, at regular intervals, the manager module triggers the execution of one time step of the road traffic simulation, receives the resulting mobility trace, and triggers position updates for all wireless nodes it has instantiated.

A. *Vehicular Traffic Simulation Parameter Set*

The four-leg traffic intersection is the most commonly found intersection, and it is also considered to be a canonical intersection used in most studies. We set the following vehicular traffic characteristics in our simulations. There are four

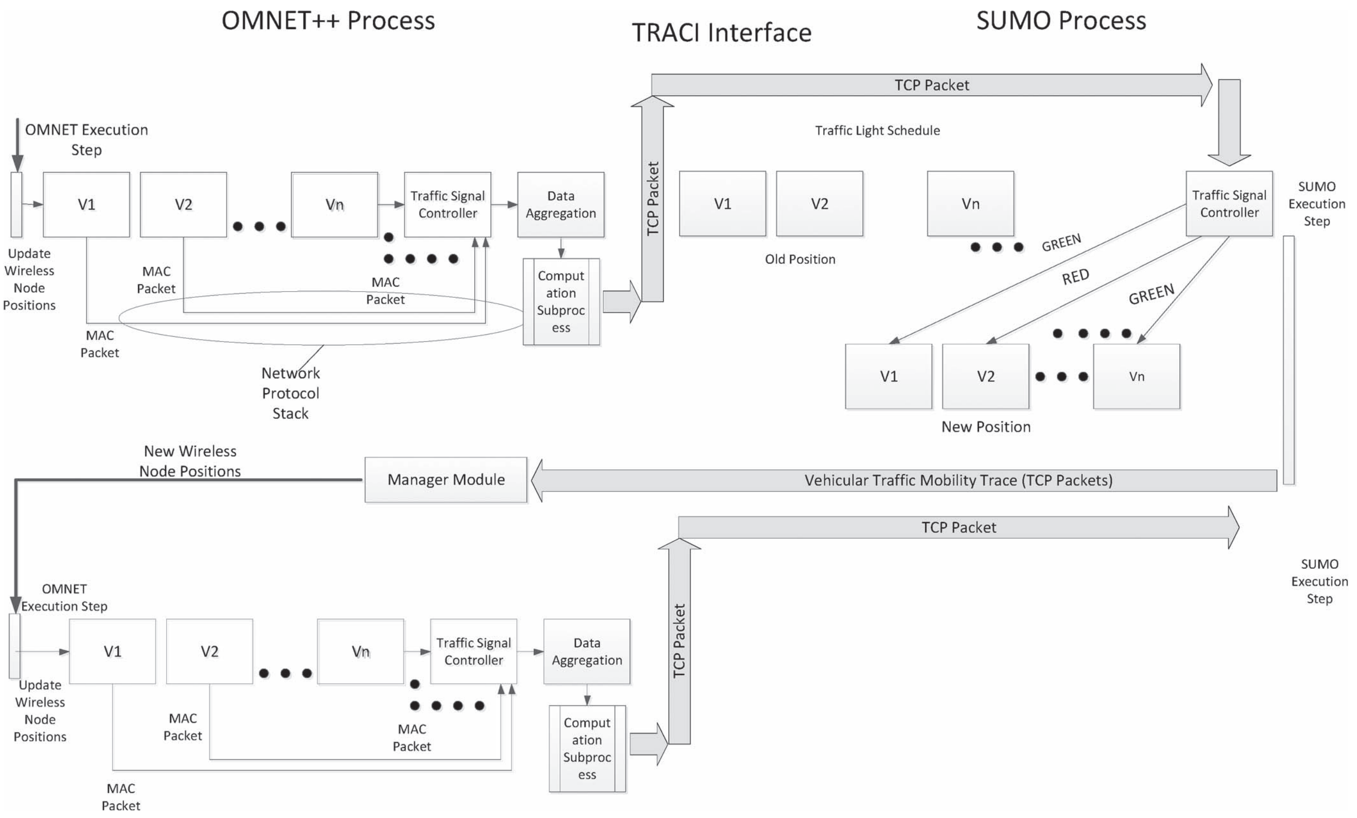

Fig. 6. Sequence diagram of TRACI message exchange between SUMO and OMNET++. OMNET++ acts as a application connecting to SUMO via a TCP socket.

approaches, which are referred to as the north, south, east, and west approaches. The saturation flow for each lane in the link is 1800 vehicles/h, and the approaches are 1000 m long. We modeled the vehicular traffic arrival process first as a Poisson arrival process and second as a uniform arrival process. For the Poisson arrival process, vehicles were injected at the beginning of each approach, following a Poisson distribution with rate λ. Thus, the vehicle interarrival times are exponentially distributed with inverse rate parameter r, where $r = \lambda^{-1}$. Similarly, for the uniform arrival process, vehicles were injected at the beginning of each approach where the interarrival time follows a uniform distribution on $(0, 1) \times 2\lambda^{-1}$. Again, there is a 0.15 probability that a vehicle makes a left turn and enters the left-turn bay.

We conducted experiments under four traffic conditions. In the first condition, we have model the traffic arrival as a Poisson arrival process with identical traffic arrival rate on all four approaches to the intersection, and we vary the traffic arrival rates. We called this the homogeneous traffic condition. We classified traffic arrival rates into three categories: heavy with $\lambda = 1700$ vehicles/h, medium with $\lambda = 800$ vehicles/hour, and light with $\lambda = 400$ vehicles/h. We start at light traffic arrival rate and then increase the traffic arrival rate as the simulation continues. All Vehicles are of the same type. In our second traffic condition, we model the traffic arrival as a Poisson arrival process, keep the arrival rate on the north and south approaches at $\lambda = 800$ vehicles/h, and vary the traffic arrival rates on the east and west approaches. We call this the heterogeneous traffic condition. In the third traffic condition, we model the traffic arrival process as a Poisson arrival process, and we have a very low arrival rate, i.e., $\lambda = 100$ vehicles/hour, on the north and south approaches and vary the traffic on the east and west approaches. In our fourth traffic condition, we model the traffic arrival process as a uniform arrival process with identical traffic arrival rate on all four approaches to the intersection, and we vary the traffic arrival rates.

B. Wireless Network Simulation Parameters

As aforementioned, the wireless network simulator implements the IEEE 802.11p. The mobile nodes transmit messages, encapsulated in MAC packets, to the traffic signal controller that include the following information: vehicle ID, location of the vehicle, speed, and current time. The size of the MAC packets transmitted, including overhead, is 2 kB. Transmission rate was set to 0.1 Mbps, which corresponds to 50 packets/s.

V. Results and Discussion

We compare the performance of the OAF algorithm against the VANET-enabled vehicle-actuated control, VANET-enabled Webster's method, and an optimized fixed-time signal control. For the fixed-time approach, the controller has been optimized for the current traffic parameters, following the guidelines in [7]. The timing parameters were 60 s of GREEN for the through traffic and 30 s of GREEN for the left turning for the heavy traffic condition, 40 s of GREEN for the through traffic and 20 s of GREEN for the left turning traffic for the Medium traffic condition, and 35 s of GREEN for the through traffic and 15 s

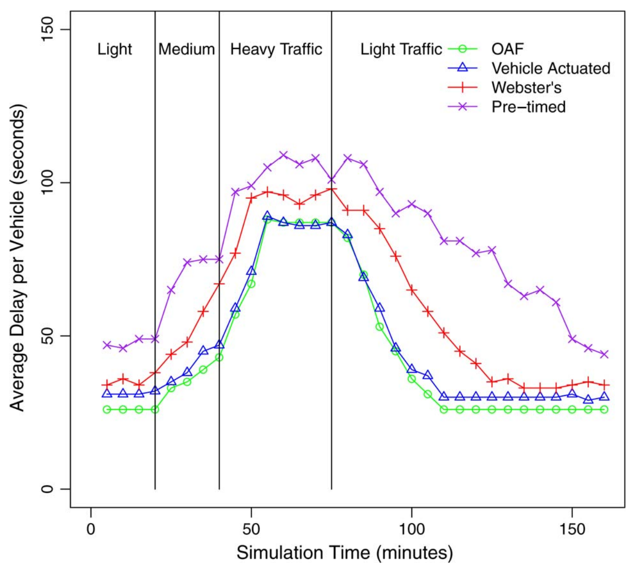

Fig. 7. Performance of OAF algorithm compared with other VANET-based traffic signal scheduling methods when all four approaches have equal vehicle arrival rates.

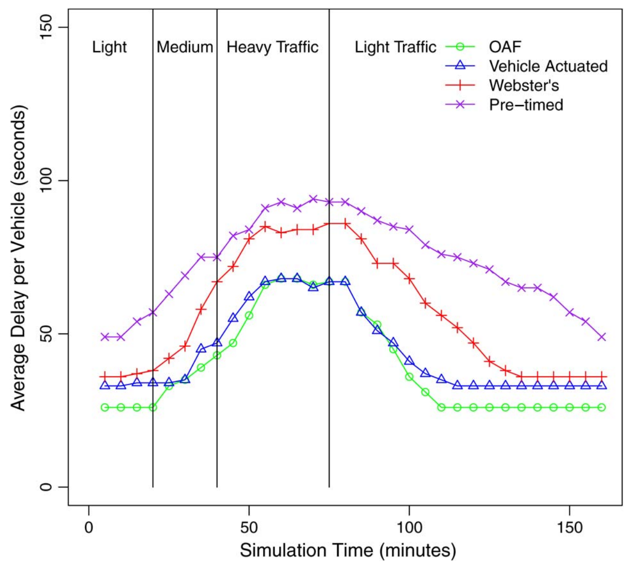

Fig. 8. Performance of the OAF algorithm compared with other VANET-based traffic signal scheduling methods when north–south approaches have constant (800 vehicles/hour) arrival rates and east–west approaches have a varying vehicle arrival rate.

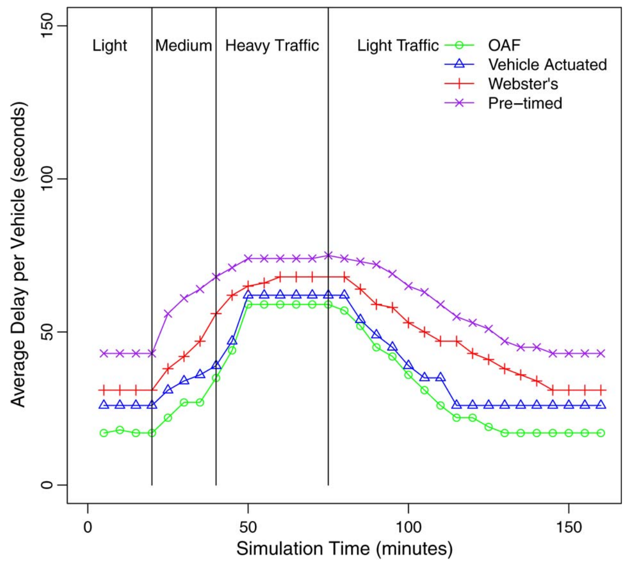

Fig. 9. Performance of the OAF algorithm compared with other VANET-based traffic signal scheduling methods when north–south approaches have constant (100 vehicles/hour) arrival rates and when the east–west approaches have a varying vehicle arrival rate.

of GREEN for left turning traffic for the light traffic. We also tested the effectiveness of the platooning algorithm, which is part of the OAF signal control algorithm.

A. Comparison of Traffic Signal Control Methods During Homogeneous Traffic Arrival Rates

In our first experiment, we have homogeneous traffic arrival rate on all four approaches to the intersection, and we vary the traffic arrival rates. Fig. 7 shows the performance of the OAF in comparison with the vehicle-actuated logic, the pretimed logic, and the Webster's logic. The performance parameter that we measured was the average delay per vehicle in terms of seconds, and we plot this delay value at 5-min intervals for all the traffic signal control methods.

The labels light, medium, and heavy indicate the time intervals during the simulation with different traffic arrival rates. We started out the simulation at a light traffic arrival rate, and 20 min into the simulation, we switched to the medium traffic arrival rate. At the 40-min mark, we switched to a heavy arrival rate, and at 75 min mark, we switched to the light traffic arrival rate, and let the simulation run to the 160-min mark. We can see that, at every instant, the OAF Algorithm performs better than both Webster's and the pretimed algorithm. During a heavy traffic arrival rate, we see that the OAF algorithm degenerates to the vehicle-actuated control since the MAX-OUT times for both the OAF algorithm and the vehicle-actuated algorithm become the same. However, when we switch back to the Light traffic arrival rate, we see that the OAF Algorithm recovers from congestion much faster; hence, the delays experienced decreased much faster. This is because the OAF algorithm is able to take advantage of the gaps that occur among vehicles and create platoons, and then, it minimizes the maximum delay that each platoon experiences. The delay is then amortized among all the vehicles in the platoon. In effect, it is much more efficient at discharging the queues. Intuitively, this is because Webster's method and the pretimed control method both react very slowly to the changes in the traffic arrival rate.

B. Comparison of Traffic Signal Control Methods for Heterogeneous Traffic Arrival Rates

Next, we study the performance of the OAF and the other three traffic control algorithms for the case with heterogeneous traffic arrivals. We set up the experiment in the following way. In Figs. 8 and 9, the east-to-west traffic and the west-to-east traffic are set at 800 and 100 vehicles/h, respectively, but the north-to-south traffic and the south-to-north traffic vary from 400 (light), 800 (medium), and 1700 (heavy) vehicles/h and then back to 400 (light) vehicles/h. Once again, there is a 0.15 probability that a vehicle makes a left turn. Once again, we compare the average delay per vehicle of the OAF algorithm and the vehicle-actuated traffic control algorithm against the

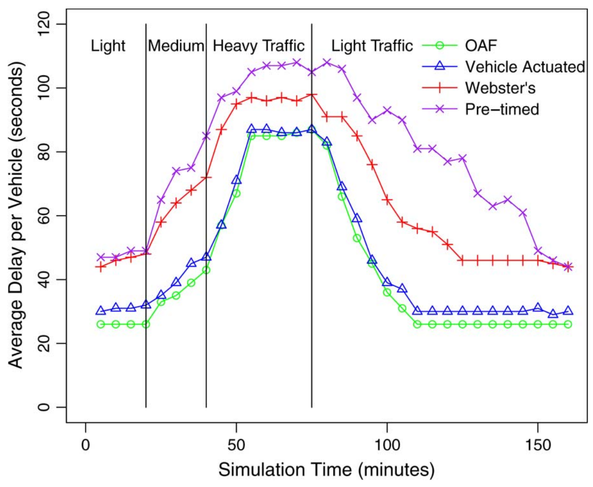

Fig. 10. Performance of the OAF algorithm compared with other VANET-based traffic signal scheduling methods when all four approaches have equal vehicle arrival rates, and the arrival process is modeled by a uniform arrival process.

Webster's and pretimed traffic signal control methods. Again, the OAF outperforms all the other three algorithms. Because of the lower variance in the traffic arrival rates, the delay curves are flatter than in the identical traffic arrival rate experiment, and the overall delays are lower for all the traffic signal control methods. The OAF and vehicle-actuated traffic control algorithms perform better than both Webster's method and the pretimed logic. However, because of the slower variance of the traffic arrival rates, the vehicle-actuated traffic control algorithm exhibits a flatter average delay curve.

C. Comparison of Traffic Signal Control Methods for Identical Traffic Arrival Rates Modeled as a Uniform Arrival Process

Fig. 10 shows the performance of the OAF in comparison with the vehicle-actuated logic, the pretimed logic, and the Webster's logic under a uniform traffic arrival process. The performance parameter that we measured was the average delay per vehicle in terms of seconds, and we plot this delay value at 5-min intervals for all the traffic signal control methods. OAF performs better than the other traffic control methods and achieves delays comparable with the Poisson arrival case, indicating that no assumption on the traffic arrival process is needed to achieve good performance.

D. Performance of the Platooning Algorithm

The performance of the OAF algorithm depends on the ability of the platooning algorithm to divide the vehicular traffic on the approach into platoons that require equal amounts of GREEN time. We show the distribution of the platoon size generated by the OAF algorithm in Fig. 11 for different traffic arrival rates. Here, at an arrival rate of 400 vehicles per platoon, we see that 33% of the platoons were platoons of size 4, 45% were platoons of size 5, and 22% of the platoons were of size 6. This implies that the platooning algorithm produces approximately equal-sized platoons, which is a necessary condition for the OAF algorithm to maintain the 2-competitiveness of the

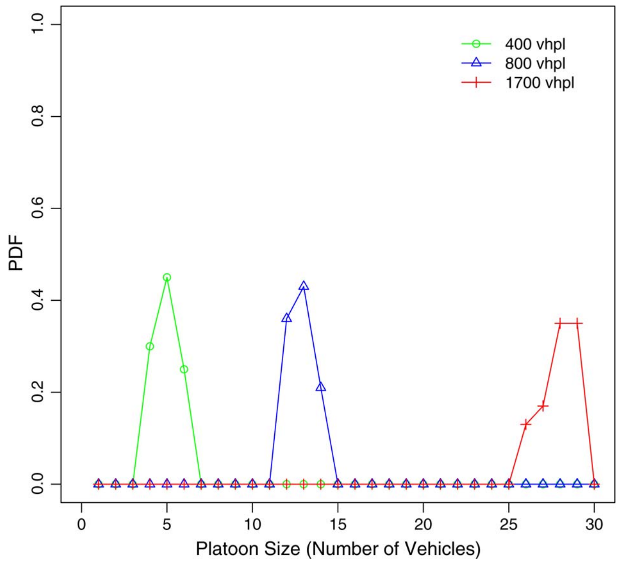

Fig. 11. Platoon size distribution under identical traffic arrival rate.

OJF algorithm. The figure shows that, at relatively heavy traffic arrival rates, we see platoons containing approximately equal number of cars, and they take approximately equal amount of GREEN time to pass through the intersection. For the OAF algorithm to perform well, we want to see this desired property in order for the real performance to approach the theoretical bound.

E. Performance of OAF Under Varying Penetration Rates

We define the penetration rate as the proportion of vehicles that are VANET enabled. The OAF algorithm depends upon the vehicle speed and position data to form and detect platoons; therefore, if some proportion of the vehicles are not detected, then we expect a reduction in performance. An example of this would be a VANET-enabled vehicle that is waiting at the end of a long queue of vehicles that are not VANET enabled. Then, this vehicle would be in a one-vehicle platoon, which is at the head of the line. Then, when the OAF algorithm schedules this platoon, all the vehicles in front of the platoon would need to clear the intersection before the platoon can pass through the intersection. This creates nondeterministic delays that severely reduce the effectiveness of the OAF algorithm. To quantify these delays we tested the performance of the OAF algorithm at various penetration rates: 90%, 70%, 50%, and 30%. We compare the delays produced under these scenarios with the delays produced under the 100% penetration scenarios in Fig. 12. We consider a homogeneous traffic condition, and we vary the traffic arrival rate during the course of the simulation.

The results in Fig. 12 show that, under a 90% penetration rate, the delays produced by the OAF algorithm are the same as those produced under the 100% penetration case while the traffic arrival rate is heavy. There is, however, a degradation in performance under light and medium traffic arrival rates. This is because, when we have a heavy traffic arrival rate, the platoons formed by the OAF algorithm, under 90% penetration rate, require approximately equal amounts of GREEN, but this is not the case for light and medium arrival rates. As we further

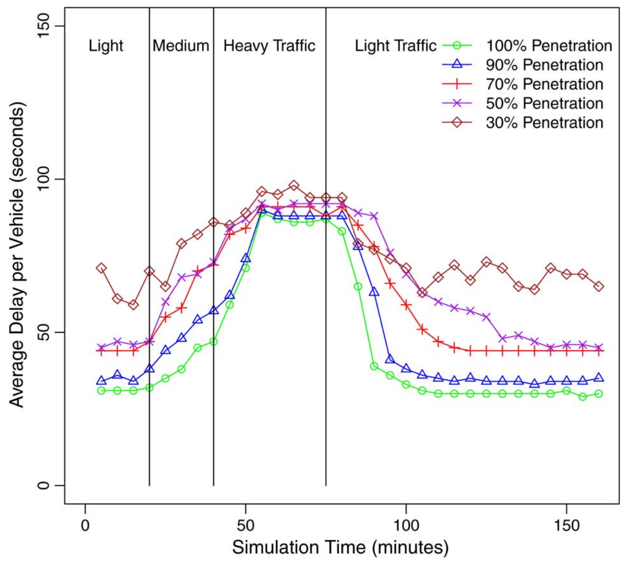

Fig. 12. Average delays produced by the OAF algorithm under various penetration rates.

reduce the penetration rate, we see increasing delays being produced by the OAF algorithm. Particularly at a 30% penetration rate, the delays experienced by vehicles are even higher than the delays produced by a nonadaptive fixed-time signal controller. This is because, at 30% penetration, we observe large numbers of vehicles that are not grouped into a platoon. The OAF algorithm assigns the MINGREEN amount of green time for all traffic phases, whereas the pretimed algorithm assigns the GREEN amount of green time computed using Webster's method, following guidelines in [7]. To compute this value, we need to measure the traffic arrival rate, which we would not be able to accurately estimate at 30% penetration. In the OAF algorithm, MINGREEN was set to 15 s, resulting in stop-and-go traffic that showed increased amount of startup times. This is why, at 30% penetration, the pretimed method performed better. If MINGREEN was suitably computed, then the OAF method would perform at least as well as the pretimed method. This indicates that the OAF method is not suitable under low penetration rates.

VI. Conclusion

In this paper, we have shown how a VANET can be used to aid in traffic signal control, including a new job-scheduling-based online algorithm, i.e., the OAF algorithm. We implemented several adaptive traffic signal control algorithms that use the fine grain information broadcasts by the vehicles. We implemented and compared these algorithms under various traffic conditions. Our experimental results show that the OAF algorithm reduces the delays experienced by the vehicles as they pass through the intersection, as compared with the other three methods under light and medium vehicular traffic loads. Under heavy vehicular traffic load, the performance of the OAF algorithm degenerates to that of the vehicle-actuated traffic method but still produces lower delays, compared with Webster's method and the pretimed signal control method. This is because, under lighter traffic, the OAF algorithm can dynamically skip through phases and minimize the delay of vehicles whenever there is a gap in the traffic. However, when the traffic gets heavier, the gaps in traffic disappear, and we always have queues on the approaches, reducing the advantage that a dynamic scheduling algorithm may have.

Appendix A
Proof of Lemma 2.1

The proof is by induction. Assume that OJF has maintained the two conditions up to and including time $t-1$. Therefore, at the beginning of time t, they are still maintained. We will prove that, after new jobs have arrived and scheduled by both OJF and A^*, the conditions will still hold at the end of time t. (Therefore, they also hold at the beginning of time $t+1$.) Let l be the vertex in G' with the oldest job at the beginning of time t. If OJF has at least one job on l' at the start of time unit t, then it schedules a job from both arcs connected to l and both arcs connected to l', and condition 1 will still hold at the end of time t. Notice that A^* can only schedule one job from each arc in one time unit.

On the other hand, for OJF, if there are no jobs at the beginning of time t on l', then if condition 2 is satisfied at the end of t, we are guaranteed that condition 1 will also be satisfied. If there are at most T jobs on a vertex on the right side for OJF, then Condition 2 holds; therefore, the only problem with guaranteeing condition 2 is if OJF has a vertex on the right side with at least $T+1$ jobs. For example, r is this type of vertex (symmetrically, we can argue for r'). For example, OJF has X jobs on l, and Y jobs on r at the start of time unit t. Therefore, in the schedule output by A^*, by the inductive hypothesis, there are at least $(X+Y-T)$ jobs on the arc (l,r), and of these jobs, at least $Y-T$ jobs are on r.

We will now prove that the A^* schedule has at least $Y-T$ jobs on r at the end of time unit t. This maintains condition 2. There are two cases.

Case 1: OJF has job j on l with latency of at least T at the start of time unit t. If the schedule generated by A^* still has j on l, it must schedule that one; therefore, $Y-T$ jobs will remain on r at the end of time t. Suppose that the optimal schedule does not still have job j on l. First, there can be no more than $X-1$ arrivals on l in the last $T-1$ units of time. This is because OJF has at most $X-1$ jobs that are more recent than j, and OJF would not schedule any of these over job j. Therefore, the schedule generated by OJF can have at most $X-1$ jobs on l, and this means that it has at least $Y-T+1$ jobs on r at start of time unit t. Therefore, at least $Y-T$ jobs remain at the end of time unit t.

Case 2: OJF does not have a job on l with latency of at least T at the start of time t. Remember that l has the oldest job; therefore, OJF does not have a job on r with latency of at least T. Therefore, Y jobs have arrived on r in the last $T-1$ time units. A^* (in fact any algorithm) could only have scheduled at most $T-1$ of these by start time unit t. This means that the optimal schedule has at least $Y+1-T$ jobs on r before time t and at least $Y-T$ jobs at the end of time unit t. ■

Appendix B
Proof of Lemma 2.3

The adversary creates jobs on both l and r until the OJF has $i+1$ jobs on l or on y. This will happen by the beginning of some time unit $t' \leq t+i+1$. Suppose that l is the vertex on which the OJF has $i+1$ jobs. The adversary can schedule the jobs with a maximum latency cost of at most $i+1$ so that, after time t', l is empty. The adversary can do the following. Complete all jobs on its own graph while at the same time forcing the OJF to keep $i+1$ jobs on an arc. Let r' be the vertex on the right side of the graph, noting that $r' \neq r$. For the next $i+1$ time units, the adversary has a job arriving on r'. The adversary schedules the oldest job on r and the new job on r'. After the $i+1$ time units, the adversary's graph is empty. Since a job arrived on r in each time unit, OJF still has $i+1$ jobs on arc (l, r'). ■

References

[1] The City of Reno Public Works Department. [Online]. Available: http://www.reno.gov/index.aspx?page=658

[2] America Revealed: Nation On The Move. PBS documetary. [Online]. Available: http://video.pbs.org/video/2223774770/

[3] The vehicular traffic simulator. [Online]. Available: http://sumo.sourceforge.net/

[4] The wireless simulation framework. [Online]. Available: http://www.omnetpp.org/

[5] The German Aerospace Research Laboratory. [Online]. Available: www.dlr.de/en/

[6] The TRACI interface can be found. [Online]. Available: http://sourceforge.net/apps/mediawiki/sumo/?title=TraCI

[7] G. F. Newell, *Theory of Highway Traffic Signals*, 6th ed. Berkeley, CA, USA: Univ. California, 1989.

[8] D. C. Gazis, *Traffic Science*, 1st ed. New York, NY, USA: Wiley, 1989.

[9] Optimal Traffic Control: Urban Intersections, 1st ed. Boca Raton, FL, USA: CRC, 2008, pp. 400–401.

[10] C. N. Chuah, D. Ghosal, A. Chen, B. Khorashadi, and M. Zhang, "Smoothing vehicular traffic flow using vehicular_based ad hoc networking amp; computing grid (VGrid)," in *Proc. IEEE ITSC*, Sep. 2006, pp. 349–354.

[11] K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, A. Thiagarajan, L. Ravindranath, and J. Eriksson, "Vtrack: Accurate, energy-aware road traffic delay estimation using mobile phones," in *Proc. 7th ACM Conf. Embedded Netw. SenSys*, New York, NY, USA, 2009, pp. 85–98.

[12] D. Ghosal, C. N. Chuah, B. Liu, B. Khorashadi, and M. Zhang, "Assessing the VANET's local information storage capability under different traffic mobility," in *Proc. INFOCOM*, 2010, pp. 1–5.

[13] A. Borodin and R. El-Yaniv, *Online Computation and Competitive Analysis*. New York, NY, USA: Cambridge Univ. Press, 1998.

[14] K. L. Mirchandani, D. Head, and P. B. Sheppard, "Hierarchical framework for real-time traffic control," *Transp. Res. Rec.*, Traffic Operations, vol. 16, no. 1360, pp. 1420–1433, Dec. 2008.

[15] V. Gradinescu, C. Gorgorin, R. Diaconescu, V. Cristea, and L. Iftode, "Adaptive traffic lights using car-to-car communication," in *Proc. IEEE 65th VTC-Spring*, Apr. 2007, pp. 21–25.

[16] N. Hounsell, J. Landles, R. D. Bretherton, and K. Gardener, "Intelligent systems for priority at traffic signals in London: The INCOME project," in *Proc. 9th Int. Conf. Road Transp. Inf. Control*, Number 454, 1998, pp. 90–94.

[17] S. Irani and V. Leung, "Scheduling with conflicts," in *Proc. 7th Annu. ACM-SIAM SODA, Soc. Ind. Appl. Math*, Philadelphia, PA, USA, 1996, pp. 85–94.

[18] B. Hull, R. Newton, S. Madden, J. Eriksson, L. Girod, and H. Balakrishnan, "The pothole patrol: Using a mobile sensor network for road surface monitoring," in *Proc. 6th Int. Conf. MobiSys*, New York, NY, USA, 2008, pp. 29–39.

[19] D. Jiang and L. Delgrossi, "Ieee 802.11p: Towards an international standard for wireless access in vehicular environments," in *Proc. IEEE VTC Spring*, May 2008, pp. 2036–2040.

[20] R. M. Karp, "Reducibility Among Combinatorial Problems," in *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, Eds. New York, NY, USA: Plenum, 1972, pp. 85–103.

[21] B. Khorashadi, F. Liu, D. Ghosal, M. Zhang, and C. N. Chuah, "Distributed automated incident detection with VGRID," *IEEE Wireless Commun.*, vol. 18, no. 1, pp. 64–73, Feb. 2011.

[22] S. Krauss, P. Wagner, and C. Gawron, "Metastable states in a microscopic model of traffic flow," *Phys. Rev. E*, vol. 55, no. 5, pp. 5597–5602, May 1997.

[23] C. Lund and M. Yannakakis, "On the hardness of approximating minimization problems," in *Proc. 25th Annu. ACM STOC*, New York, NY, USA, 1993, pp. 286–293.

[24] W. R. McShane, R. P. Roess, and E. S. Prassas, *Traffic Engineering*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1998.

[25] C. Priemer and B. Friedrich, "A decentralized adaptive traffic signal control using v2I communication data," in *Proc. 12th Int. IEEE ITSC*, Oct. 2009, pp. 1–6.

[26] M. Besley, R. Akcelik, and E. Chung, "An evaluation of SCATS master isolated control," in *Proc. 19th ARRB Transp. Res. Conf.*, May 1998, pp. 1–24.

[27] S. Phillips, R. Motwani, and E. Torng, "Non-clairvoyant scheduling," in *Proc. 4th Annu. ACM-SIAM SODA*, Soc. Ind. Appl. Math., Philadelphia, PA, USA, 1993, pp. 422–431.

[28] S. G. Shelby, "Design and evaluation of real-time adaptive traffic signal control aalgorithms," Ph.D. dissertation, Univ. Arizona, Tucson, AZ, USA, 2001.

[29] F. V. Webster and B. M. Cobbe, "Traffic signals," Road Research technical paper. H. M. S. O. Road Res. Lab., Berkshire, U.K., 1966.

Kartik Pandit is currently working towards the Ph.D. degree in computer science with the Department of Computer Science, University of California, Davis, CA, USA.

His research interests are resource allocation in wireless networks, transportation research, optimization, and operations research.

Dipak Ghosal (M'08) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Kanpur, India, in 1983, the M.S. degree in computer science and automation from the Indian Institute of Science, Bangalore, India, in 1985, and the Ph.D. degree in computer science from the University of Louisiana, Lafayette, LA, USA, in 1988.

He is currently a Professor with the Department of Computer Science, University of California, Davis, CA, USA. His main research interests include high-speed networks, wireless networks, vehicular ad hoc networks, next-generation transport protocols, and parallel and distributed computing.

H. Michael Zhang received the B.S.C.E. degree from Tongji University, Shanghai, China, and the M.S. and Ph.D. degrees in engineering from the University of California, Irvine, CA, USA.

He is currently a Professor with the Department of Civil and Environmental Engineering, University of California, Davis, CA, USA. He is an Area Editor for the *Journal of Networks and Spatial Economics* and Associate Editor for *Transportation Research—Part B: Methodological*. His research interests include transportation systems analysis and operations.

Chen-Nee Chuah (M'01–SM'06) received the B.S. degree from Rutgers University and the M.S. and Ph.D. degrees in electrical engineering and computer sciences from the University of California, Berkeley, CA, USA.

She is a Professor of electrical and computer engineering with the University of California, Davis (UC Davis), CA, USA. Her research interests include Internet measurements, network management, anomaly detection, online social networks, and vehicular ad hoc networks. She is an ACM Distinguished Scientist.

Dr. Chuah received the NSF CAREER Award in 2003 and the Outstanding Junior Faculty Award from the UC Davis College of Engineering in 2004. In 2008, she was named a Chancellor's Fellow of UC Davis. She has served on the executive/technical program committee of several ACM and IEEE conferences and is currently an Associate Editor for the IEEE/ACM Transactions on Networking.