Variable Bit Rate Transmission Schedule Generation in Green Vehicular Roadside Units

Abdulla A. Hammad[1], Terence D. Todd[1] and George Karakostas[2]

[1]Department of Electrical and Computer Engineering
McMaster University
Hamilton, Ontario, CANADA
Email: todd@mcmaster.ca

[2]Department of Computing and Software
McMaster University
Hamilton, Ontario, CANADA
Email: karakos@mcmaster.ca

Abstract

Smart traffic scheduling can be used to reduce downlink roadside unit (RSU) energy use in green vehicular roadside infrastructure. In this paper we consider the problem of downlink schedule generation when the RSU-to-vehicle radios use a variable bit rate (VBR) air interface. We first present offline scheduling formulations that provide lower bounds on the energy required to fulfill vehicle requests. An integer linear program (ILP) is introduced which can be solved to find optimal offline VBR time slot schedules. We then prove that this problem is NP-complete by a reduction from the well-known Santa Claus Problem. Two flow graph based models are then used to solve the minimum energy VBR scheduling problem. The first uses Generalized Flow (GF) graphs which represent time slots as individual graph nodes. The second uses time expanded graphs (TEGs) which model the temporal evolution of the system. Both of these models provide lower bounds on energy performance and provide the basis for energy efficient online schedulers. The first scheduler introduced, First Come First Serve (FCFS) is very simple and treats all vehicles equally and in order of arrival. The second, Fastest First (FF), gives priority to faster moving vehicles since their transit time through the RSU coverage area is less than slower moving vehicles. The Greedy Generalized Flow (G-GF) and Greedy Time Expanded Graph (G-TEG) algorithms are two implementations of the two flow graph based models. The proposed algorithm performance is examined under different traffic scenarios and they are found to perform well compared to the lower bound. Our results show that less computational intensive algorithms (FCFS) and (FF) can perform well under light load, but (G-GF) and (G-TEG), while more computational intensive, can provide near optimal throughput and with reasonable drop rates. The results also show that the G-GF and G-TEG algorithms are much more fair than the simpler algorithms in heavy load situations.

I. Introduction

In the future, a wide variety of new systems and applications will be enabled using vehicular ad hoc networks (VANETs). Although the initial focus will be on improving road safety, applications such as traffic management, location aware advertising and infotainment are expected to follow. In the USA, VANETS are licensed to operate using dedicated short range communications (DSRC) in the 5.9 GHz frequency band [1]. Wireless Access in Vehicular Environment (WAVE) is a recent addition to the IEEE 802.11 wireless LAN standards family [2] that is designed specifically for enabling VANET applications.

VANETs can have three modes of communication, vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I), and hybrid V2V-V2I communications. In V2I, fixed roadside units (RSUs) are used to interact with passing vehicles for purposes such as, the relaying of inter-vehicle messages, broadcasting information, and access to Internet information services. Unlike most traditional ad hoc networks, vehicles move quickly and spend a relatively short amount of time inside RSU radio coverage range. This imposes certain constrains on the RSU response time for a vehicle request. Moreover, as multiple vehicles can be moving inside the RSU range with different speeds and communication requirements, the service order becomes crucial in to ensure that satisfactory performance is obtained.

Deploying RSUs in roadside locations can be difficult due to the unavailability of wired electricity. In many locations, power grid connections are not possible, while in others, the connection costs are too prohibitive. As an alternative, the RSUs can be powered using renewable energy sources, such as solar or wind power. A deployment analysis was presented in [3] which was included in the US Department of Transportation Vehicle Infrastructure Integration (VII) Initiative. Cost projections were given for a vehicular infrastructure deployment that would focus initially on safety and monitoring applications. The report estimated that 40% of rural highway infrastructure would have to be solar powered, and that over 63% of the roadside unit costs would be spent on solar energy provisioning. It is clear from this study that costs can be significantly reduced by energy efficient designs.

It has been shown that smart scheduling can be used to reduce roadside unit (RSU) energy costs in green vehicular roadside infrastructure [4]. In this paper we focus on the problem of downlink schedule generation when the RSU-to-vehicle air interfaces use variable bit rate (VBR) transmission, rather than the constant bit rate case considered in [4]. Offline scheduling formulations are first introduced which provide lower bounds on the energy needed to satisfy arriving vehicular traffic demand. An integer linear program (ILP) formulation is given which can find optimal offline VBR time slot schedules. The paper then shows that the scheduling problem is NP-complete by a reduction from the well known Santa Claus Problem. We then introduce two flow graph based models that can solve for minimum energy VBR schedules. The first uses generalized flow (GF) graphs which represent time slots as individual graph nodes. The second is based on time expanded graphs (TEGs) which model the system's temporal evolution. TEGs have been used to model scheduling when vehicle-to-vehicle forwarding is used [5] [6]. Both of these models can be efficiently solved and provide lower bounds on energy performance. They also motivate some proposed online schedulers. The first, First Come First Serve (FCFS) is a simple algorithm which treats all vehicles

equally and assigns time slots in vehicle arrival order. The second, Fastest First (FF), gives priority to faster moving vehicles. This is motivated by the fact that their transit time through the RSU coverage area is less than slower moving vehicles. The Greedy Generalized Flow (G-GF) and Greedy Time Expanded Graph (G-TEG) algorithms are two implementations of the two graph models. The results from various experiments show the good performance of the proposed algorithms in comparison with the energy lower bounds. Our results also show that less computational intensive algorithms, FCFS and FF, can perform well under light loading conditions, but (G-GF) and (G-TEG), while more computational intensive, can provide near optimal throughput and with reasonable values of request dropping. The results also show that the G-GF and G-TEG algorithms are much more fair than the simpler algorithms in heavy load situations.

The remainder of the paper is organized as follows. Section II summarizes related research results. Section III described the assumptions used in our system model and Section IV presents the ILP formulation of our problem for the variable bit rate time slot case. In Section IV-A this scheduling problem is shown to be NP-complete. Following this, in Section IV-B the off-line energy bounds are introduced. In Section V, four online scheduling algorithms are then proposed. The bound and the online algorithm performance are studied in Section VI and in Section VII, the paper presents the conclusions of our work.

II. Related Work

Recent vehicular ad hoc networking work has covered a broad range of topics. For example, application requirements and their relation to networking technologies are explored in [7], and reference [8] surveys routing protocols and related mobility models. The specific challenges relating to vehicular security have been studied in reference [9]. In [10] and [11], the performance of the IEEE 802.11p standard in highway environments was considered, and in [12] its performance was studied. Extending vehicular ad hoc network connectivity, improving utilization and decreasing contention were examined in [13], [14] and [15].

Roadside unit traffic scheduling has been studied using simple schedulers based on message size and their deadlines but without considering the energy consumption of the infrastructure [16]. Reference [17] maximizes RSU throughput given messages with different priorities. The RSU channels are viewed as machines and messages as jobs using a model based on parallel machines. In [18], an RSU throughput optimization is used given the locations and velocities of the vehicles. A scheduler was also proposed which can be used in the IEEE 802.11e contention free period.

Energy efficiency in VANETs has typically not been an issue, since vehicles are usually assumed to have unlimited energy reserves. Also, from the roadside infrastructure viewpoint, most work assumes urban settings where wired power is available at reasonable cost. However, recent work addresses the energy consumption needed by a group of RSUs by switching them on and off in order to maintain connectivity for the vehicles while minimizing RSU energy use [19]. The purpose of this paper is to optimize the RSU energy consumption, but with a focus on smart scheduling at the RSU.

Energy consumption at the RSU was considered in reference [20]. This work assumed a constant bit rate air interface and focused on reducing downlink energy costs in the case where vehicle communication requirements are delay tolerant. Time expanded graphs were used used to model scheduling in vehicle-to-vehicle forwarding [5] [6] and in [21], scheduling was considered when there are multiple RSUs. The variable bit rate model is the starting point for our paper.

III. System Description

The system considered consists of a single Roadside Unit (RSU) which communications with passing vehicles as shown in Figure 1. The figure shows the vehicles moving in one direction, but multi-directional traffic is also easily accommodated. The focus is on the energy required by the radio interface used by the RSU to fulfill vehicle communication requirements in the downlink direction. We are not concerned about vehicular energy efficiency since the vehicle radio is powered by the car's engine.

The objective for the RSU is to generate and execute downlink transmission schedules that fulfill vehicular communication requirements. We assume that a vehicle's communication requirement is delay tolerant in that it can be satisfied at any point during the vehicle's RSU transit time [20]. Upon entering the RSU range, each vehicle announces its requirements, location and velocity to the RSU. This information can then be used to estimate the downlink energy costs of communicating with the vehicle as it passes through the coverage area. The objective is to use these inputs to create a downlink transmission schedule so that energy use is minimized. In [20] schedule generation was considered when the air interface uses a constant bit rate (CBR). In the CBR case, power control is used to maintain a fixed bit rate over the RSU-to-vehicle links as the channel path loss changes. In this case the objective is to schedule downlink transmissions such that the required average downlink *power consumption* is minimized. However, as in [5], we assume a variable bit rate (VBR) air interface where the transmission power is assumed to be fixed, and changes in channel path loss are compensated for by varying the bit rate used over the RSU-to-vehicle links. A packet transmission may therefore occupy a different amount of time on the channel

depending upon the bit rate used when it is transmitted. In this case, the objective is therefore to create a downlink transmission schedule such that the total time that the RSU spends transmitting is minimized. Note that the bit rate needed can be set in a variety of ways such as using a short two-way handshake prior to actual vehicle data packet transmission. The schedule generation problem is more formally stated as follows.

INPUT: The scheduler is given a sample function of V vehicles indexed by the set $\mathcal{V} = \{1, 2, \ldots, V\}$, where each vehicle $v \in \mathcal{V}$ has a residual RSU-to-vehicle communication requirement, d_v, bits. We consider two different transmission arrangements.

1) Time Slotted (TS): The entire time over which scheduling is to occur consists of T fixed duration time slots given by the set $\mathcal{T} = \{1, 2, \ldots, T\}$. The scheduler is given estimates of the number of bits that can be carried in each time slot $t \in \mathcal{T}$ for vehicle v, denoted by $b_{v,t}$. This is referred to as the variable bit rate *time slotted* or TS case.
2) Time Framed (TF): The entire time over which scheduling is to occur consists of transmission frames given by the set $\mathcal{F} = \{1, 2, \ldots, F\}$ and their duration, τ_f, for all $f \in \mathcal{F}$. We are also given estimates of the bit rate that can be achieved in each frame f, for vehicle v, denoted by $B_{v,f}$. This is referred to as the *time framed* or TF case. Time frame durations are such that $B_{v,f}$ is constant for all vehicles inside the RSU range.

OUTPUT: A scheduler output gives a downlink RSU-to-vehicle transmission schedule. Given the inputs defined above, the scheduler will try to generate a downlink transmission schedule so that all vehicle communication requirements are satisfied, and so that the downlink RSU energy cost is minimized, i.e., the scheduler tries to minimize the total time needed to fulfill all downlink vehicular transmissions. In the TS case the scheduler output gives $x_{v,t}$ for all $v \in \mathcal{V}$ and $t \in \mathcal{T}$, which are binary decision variables equal to 1 if vehicle v is allocated to time slot t, and 0 otherwise. In the TF case, the scheduler output gives $x_{v,f}$ for all $v \in \mathcal{V}$ and $f \in \mathcal{F}$, which is the *number of bits* which are sent to vehicle v in time frame f.

The schedules that are generated can be either offline or online. In *offline scheduling*, the entire set of inputs for all $t \in \mathcal{T}$ (or for all $f \in \mathcal{F}$) are provided to the scheduler at once. Optimum offline schedules are derived in Section IV-B and are used as lower bounds on energy performance for comparisons with the online algorithms. In *online scheduling* the inputs are provided to the scheduler in real time and it must create a schedule in real time using these causal inputs. Online scheduling algorithms are given in

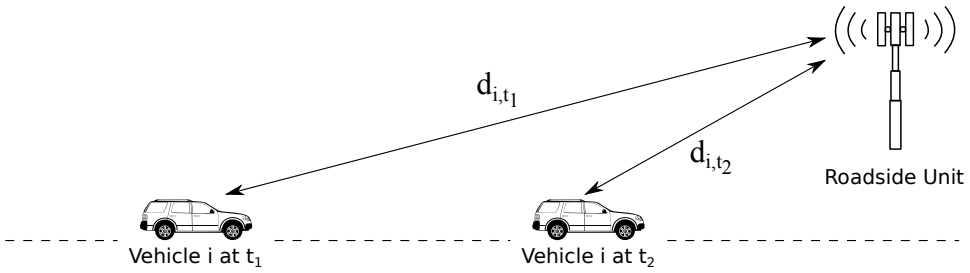

Fig. 1: RSU Example. Vehicle i is shown at two different times t_1 and t_2 at distances d_{i,t_1} and d_{i,t_2} from the RSU, respectively

Section V. In the following section, we formalize these definitions and derive optimum offline schedules for the energy needed for RSU-to-vehicle communication.

IV. Offline Variable Bit Rate Scheduling

The offline scheduling task is modelled as an Integer Programming (IP) optimization. We first consider the time slotted (TS) model where the offline scheduling problem can be written as an ILP, referred to as VBR-OPT, as follows.

$$\begin{aligned}
\underset{x_{v,t}}{\text{minimize}} \quad & \sum_{v\in\mathcal{V}}\sum_{t\in\mathcal{T}} x_{v,t} && && \text{(VBR-OPT)}\\
\text{subject to} \quad & \sum_{v\in\mathcal{V}} x_{v,t} \le 1, && \forall t\in\mathcal{T} && (1)\\
& \sum_{t\in\mathcal{T}} b_{v,t}x_{v,t} \ge d_v, && \forall v\in\mathcal{V} && (2)\\
& x_{v,t}\in\{0,1\}, && \forall t\in\mathcal{T}, v\in\mathcal{V} && (3)
\end{aligned}$$

In VBR-OPT, $x_{v,t}$ are binary decision variables equal to 1 if vehicle v is allocated to time slot t and 0 otherwise. The objective is to minimize the total number of time slots the RSU will be transmitting. Constraint (2) ensures that the number of time slots allocated to each vehicle satisfies the vehicle demand, while Constraint (1) prevents time slots from being assigned more than once.

Note that since optimization VBR-OPT is an ILP, it may have an exponential worse-case complexity, which precludes its use for reasonable problem sizes. In fact, we show next that VBR-OPT is NP-complete. However, we can generate a less complex lower bound on energy by relaxing constraint (3) to $x_{v,t} \ge 0$ for all $t\in\mathcal{T}, v\in\mathcal{V}$. This permits us to obtain a bound by solving a conventional linear program. This is discussed further in Section IV-B.

A. Time Slotted Variable Bit Rate Scheduling Complexity

In this section we prove the NP-completeness of time slotted version of the offline scheduling problem formulated in Section IV. This is done by a reduction from the well known Santa Claus Problem [22]. In the Santa Claus Problem, Santa has a set $\mathcal{G}$ of presents (gifts) that he wants to distribute among a set $\mathcal{K}$, of kids. There is a given level of satisfaction, $p_{k,g}$, associated with giving present g to kid k, for all $g \in \mathcal{G}$ and $k \in \mathcal{K}$. The total satisfaction for a kid consists of the sum of satisfactions of each present received. Given these inputs, the decision version of the Santa Claus Problem must answer the question: *Given a satisfaction threshold, S, is there an assignment of presents to the kids such that each kid receives at least a satisfaction of S?*

The decision version of the Santa Claus Problem can be easily written as an integer linear program feasibility test as follows.

$$\begin{aligned}
&\underset{x_{k,g}}{\text{minimize}} \quad 0 && && \text{(SC-OPT)}\\
&\text{subject to} \quad \sum_{k\in\mathcal{K}} x_{k,g} \leq 1, && \forall g \in \mathcal{G} && (4)\\
&\qquad\qquad \sum_{g\in\mathcal{G}} p_{k,g}\, x_{k,g} \geq S, && \forall k \in \mathcal{K} && (5)\\
&\qquad\qquad x_{k,g} \in \{0,1\}, && \forall g \in \mathcal{G}, k \in \mathcal{K} && (6)
\end{aligned}$$

where $x_{k,g}$ is a binary decision variable, equal to 1 when kid k is given present g and 0 otherwise. Constraint (4) ensures that each present, is assigned to at most one kid and Constraint (5) guarantees that the each kid achieves the minimum satisfaction threshold, S. We now show the following result.

Theorem 1. *The time slotted variable bit rate vehicular scheduling problem is NP-complete.*

Proof: We first divide both sides of Constraint (5) by S, and define $p'_{k,g} = p_{k,g}/S$. This constraint now becomes

$$\sum_{g\in\mathcal{G}} p'_{k,g}\, x_{k,g} \geq 1, \quad \forall k \in \mathcal{K} \tag{7}$$

where $p'_{k,g}$ can be viewed as a normalized satisfaction. Similarly, we divide both sides of Constraint (2) by d_v and define $b'_{v,t} = b_{v,t}/d_v$. This constraint now becomes

$$\sum_{t\in\mathcal{T}} b'_{v,t}\, x_{v,t} \geq 1, \quad \forall v \in \mathcal{V} \tag{8}$$

where $b'_{v,t}$ can be viewed as a normalized time slot capacity. Note the similarity that these two normalized constraints create for ILPs VBR-OPT and SC-OPT.

Now assume that we are given an instance of the Santa Claus Problem defined above. We define an instance of the variable bit rate scheduling problem where $|\mathcal{V}| = |\mathcal{K}|$ and $|\mathcal{T}| = |\mathcal{G}|$. We set $b'_{k,g} = p'_{k,g}$ for all $k \in \mathcal{K}$ and $g \in \mathcal{G}$. Clearly this transformation can be done in linear time.

Using the above inputs we now solve ILP VBR-OPT. If a valid solution is obtained, then from the above construction, the time slot assignments from ILP VBR-OPT are clearly a feasible solution to ILP SC-OPT. Now assume that ILP VBR-OPT is infeasible, i.e., no minimum energy solution to VBR-OPT exists. If a solution to the Santa Claus Problem instance exists however, then from the same construction, the solution from ILP SC-OPT could be used to create a feasible solution for ILP VBR-OPT. This contradicts the infeasibility of the ILP VBR-OPT solution. Therefore, the solution to any instance of the Santa Claus Problem can be answered by testing if a solution to the variable bit rate scheduling problem exists. This establishes the NP-completeness of the variable bit rate scheduling problem. ■

B. Offline Scheduling using Generalized and Time Expanded Graphs

In this section we introduce two methods to compute the minimum energy that an optimal offline scheduler can achieve. Since they are offline, the entire input is given to the scheduler at once, and the schedulers may produce fractional time slot allocations. In Section V we adapt these models for online use. The first proposed method models the time slotted (TS) version of the scheduling problem as a Minimum Cost Generalized Flow Graph. Recent combinatorial algorithms [23] can solve this type of problem in time complexity which is strongly polynomial compared to general purpose LP optimization solvers which at best can achieve weakly polynomial times. The second method models the time framed (TF) version of the scheduling problem as a Time Expanded Graph.

1) Generalized Flow (GF) Graph Model: In this section, we introduce on offline scheduling method using a generalized flow graph construction for the TS model. This technique has advantages over solving an LP relaxation of ILP VBR-OPT in that due to the cycle cancelling algorithms used for its solution [23], it directly manipulates the underlying graph and tends to generate integral time slot allocations. This, combined with careful design of the generalized flow graph model for our problem has led to a dramatic decrease of the number of time slots that the scheduler shares between multiple vehicles. By reducing the percentage of shared time slots, the difference between the solutions obtained by the approximate schedule and an optimal IP scheduler is minimized. This is discussed further in Section V. In addition, when general

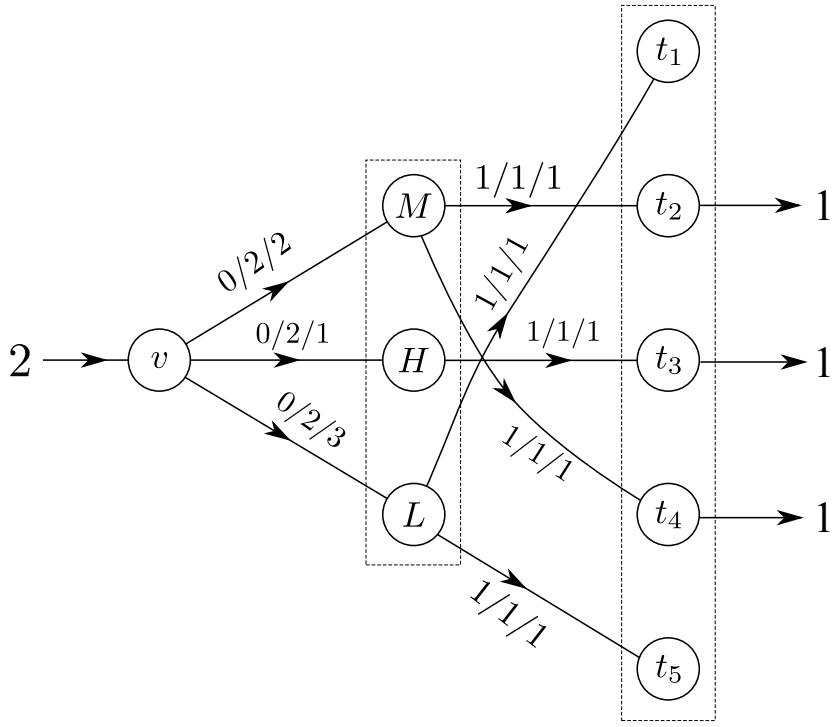

Fig. 2: Generalized Flow Graph Illustration. Part of the graph is shown for a single vehicle v. The format of the edge labels is given by the triplet "cost/capacity/multiplier".

purpose LP methods are used to find an optimal schedule for a problem modeled as a generalized flow, they are weakly polynomial. Their complexity will depend on the size of the cost, capacity and gain factors [23], while combinatorial algorithms designed specifically for Generalized Flow problems can be strongly polynomial.

Generalized Flow Graphs (GFGs) [24] are similar to conventional flow graphs in that they consist of nodes interconnected by directional edges. As in conventional flow graphs, each node can introduce flow into the graph (supply), subtract flow from the graph (demand), or be a transship node (the sum of flows into the node is the same as the sum of the flows out). Each GFG edge has a maximum flow capacity which must not be exceeded. The edges also have costs associated with the use of that edge, and for each unit of flow along an edge, the total edge cost incurred is the product of the flow along the edge and the edge cost. A feasible flow across the graph must satisfy all the nodes supply and demand, as well as flow conservation, while not violating the edge capacities.

Generalized flow graphs differ in that each edge also has a *multiplier*, which is referred to as the edge gain factor. Edge (i, j) has a multiplier, $\gamma_{i,j}$, and flow f entering this edge will emerge at the other end of the edge as flow $f\gamma_{i,j}$. The gain factors can be less (or greater) than 1 in which case the flow reaching the destination is reduced (or magnified), respectively.

We will present a simple example to show how the variable bit rate scheduling problem can be modelled as a GFG. For this example we assume a simplified case where only three bit rate levels are used, high (H), medium (M) and low (L). Clearly, the bit rate that can be used for a given vehicle will depend upon the quality of the downlink during a time slot. In the example we assume that a flow of 2 (in units of high bit rate time slots) is required for vehicle v. We also assume that two and three M and L time slots are needed to accommodate one H bit rate time slot, i.e., a flow of 1 H time slots can be serviced by 2 M bit rate time slots, etc. A section of the final generalized flow graph is shown in Figure 2. The input flow to node v is shown at the left. The L, M and H nodes correspond to using time slots with those bit rates. Each node in the right hand column corresponds to a time slot, and the edges connecting the middle and right hand columns correspond to the bit rate which is possible for v during that time slot. The multipliers on the edges leaving node v correspond to the relative bit rates of the center column nodes used. For example, the v-to-M multiplier is 2 which will cause a unit flow input from v to become a two unit flow leaving node M. This will ensure that two M time slots will be required to satisfy this flow. In the example, the 2 units of flow entering v emerge as three units of flow from the time slot column. This indicates that one unit of flow each is passing along the H-to-t_3, M-to-t_2 and M-to-t_4 edges. The requirement for v has therefore used one H time slot and two M time slots. Note however, that the amount of flow that is leaving the section of the graph shown is not the same as the flow entering. The cost for a particular solution comes from the middle to right hand column edges, each of which are unit costs. The total cost for this solution is therefore 3.

In general a vehicle demand R is represented in terms of multiples of the highest bit rate level used for a full time slot. If vehicle v demand can be served during time slots connected to the H node, it can receive the highest bit rate achievable. Thus, it can be served in exactly R time slots. The multiplier on the edges connecting v to H, $\gamma_{v,H}$, is equal to 1. This is because a unit demand can be served by a unit of flow from H to the time slot node t_3, assuming the second highest bit rate level, M, is half the H bit rate. Serving vehicle v demand using only time slots connected to M would require the use of $2 \times R$ time slots, t_2 and t_4. This is reflected in the multiplier γ_M being equal to 2. The same idea applies to lower bit rates. So we have $\gamma_H < \gamma_M < \gamma_L$. The multipliers over edges connecting nodes H, M and L and time nodes are set to be 1.

For algorithmic convenience reasons, *circulations* instead of flows are used. To achieve this, the above design is "mirrored" as shown in Figure 3. In this example we show a complete graph with the same

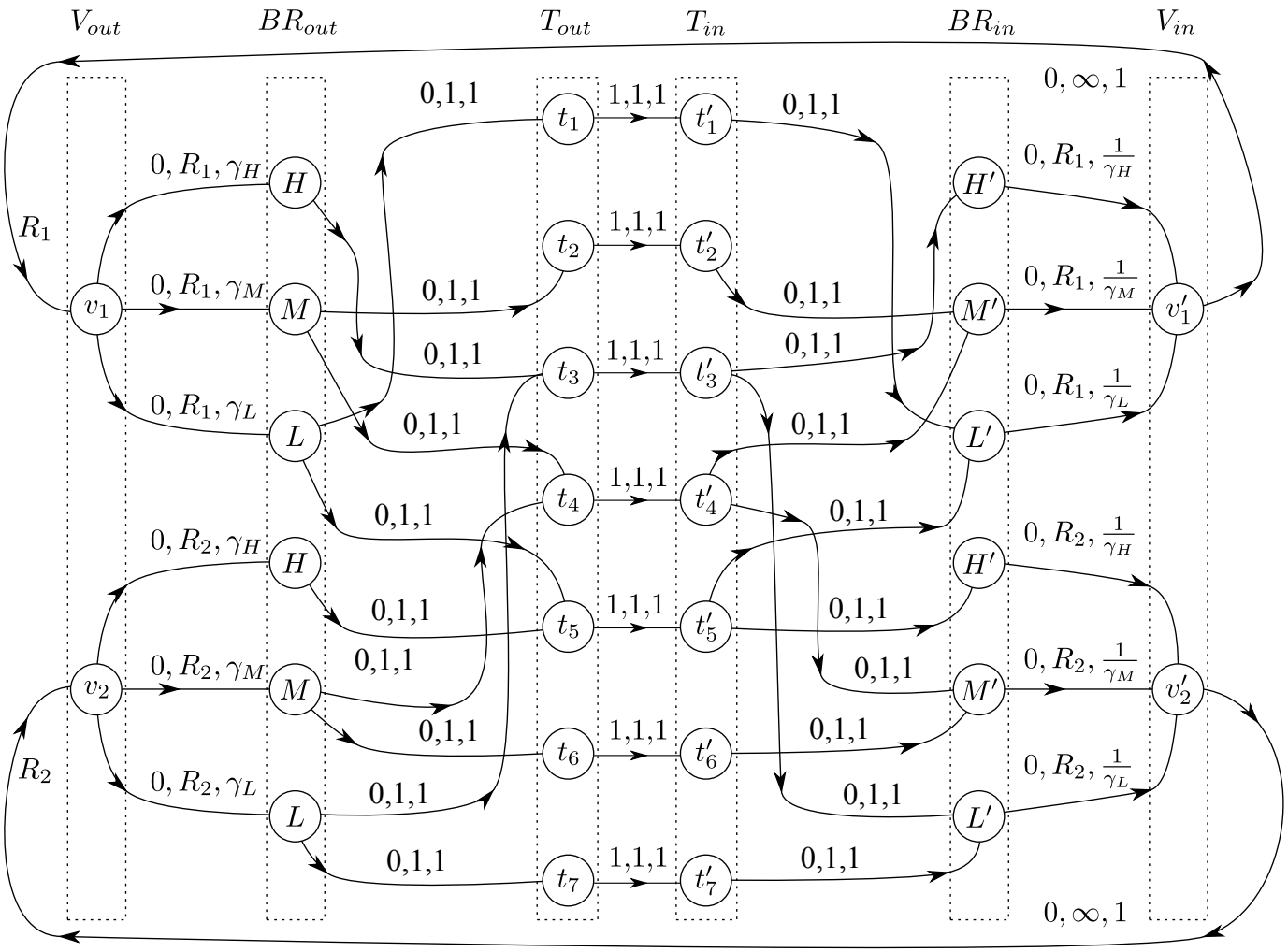

Fig. 3: Generalized Flow Graph Model Example

time slot bit rates as before, showing 7 time slots and two vehicles. By mirroring, the flow coming from vehicle nodes in column $Vout$ flows to bit rate level nodes in column $BRout$ to time slots column $Tout$. At this stage the mirrored design starts where flow from $Tout$ nodes pass through Tin nodes. This in turn is collected in the bit rate level nodes in the column $BRin$ and finally to Vin vehicle nodes. The feedback flow allows excess flow to be fed back into the network, forming a circulation.

The values of capacity and multiplier over the edges in the mirrored part of the graph are kept the same, except for the edges connecting the mirrored bit rate level nodes $BRin$ to vehicles in the Vin column where the value of the multipliers are the inverse of those connecting $Vout$ vehicles nodes to $BRout$. In this design, the same time slot can be used by any vehicle that happens to be inside the RSU range during this time slot. In order to limit the flow from multiple vehicles into a single time slot to one unit, the mirror column of time slots is used where each time slot t'_i is connected to t_i with an edge capacity limited to 1.

This design allows combinatorial algorithms to detect flow generating cycles and flow absorbing cycles [23]. Together they form what are called "bicycles" which can be eliminated when searching for a minimum cost flow solution. After solving this minimum cost flow problem, the final schedule can be obtained from the flows between the bit rate level nodes $BRout$ and time nodes $Tout$. However, note that generalized flow graph solutions do not produce integral flows, so optimum solutions may allocate fractional time slots. In Section V we propose a method based on rounding that generates integral solutions which can be used in practical schedulers.

2) Time Expanded Graph (TEG) Model: The generalized flow graph mode discussed in Section IV-B1 specifically identifies time slots which are then assigned by the GFG optimization. In this section we consider the time framed (TF) version of the problem modeled as a time expanded graph. An advantage of this approach is that fractional flow assignments will be in units of bits rather than units of time slots. This makes it easier to accommodate practical packet constraints. Also, typically the achievable bit rate for a given vehicle does not change over many time slots, which means that the GFG size can be unnecessarily large.

Unlike a conventional flow graph, a TEG is time-expanded, i.e., the graph models the evolution of the system in time [25]. Accordingly, we consider the system to evolve over multiple time frames. An example of a TEG is shown in Figure 4 consisting of three time frames. In this example there are three vehicles V_1, V_2 and V_3 with downlink demands of 7, 5 and 15, respectively. The graph is time-expanded which means that the vehicle nodes and the RSU node are duplicated across the three time frames, as shown. The temporal node replications also model the carrying of flow (i.e., bits) over from one frame to the next, e.g., the RSU may fulfill a vehicle's job requirement over multiple frames. Note that node D is simply a flow collection node. The cost of communicating between the RSU and vehicle i at time frame j is denoted by $c_{i,j}$. When the vehicle is closest to the RSU and the achievable bit rate is highest, the cost is lowest, and as the achievable bit rate drops, the cost increases. The edge capacity denotes the maximum number of bits that can be transmitted to vehicle i if it used the entire frame j. The cost over this arc is inversely proportional to the bit rate $B_{i,j}$ vehicle i can achieve during frame j.

The TEG can easily be expressed in terms of the following LP.

$$\underset{x_{v,f}}{\text{minimize}} \quad \sum_{f\in\mathcal{F}}\sum_{v\in\mathcal{V}} x_{v,f}/B_{v,f} \tag{DNTG-LP}$$

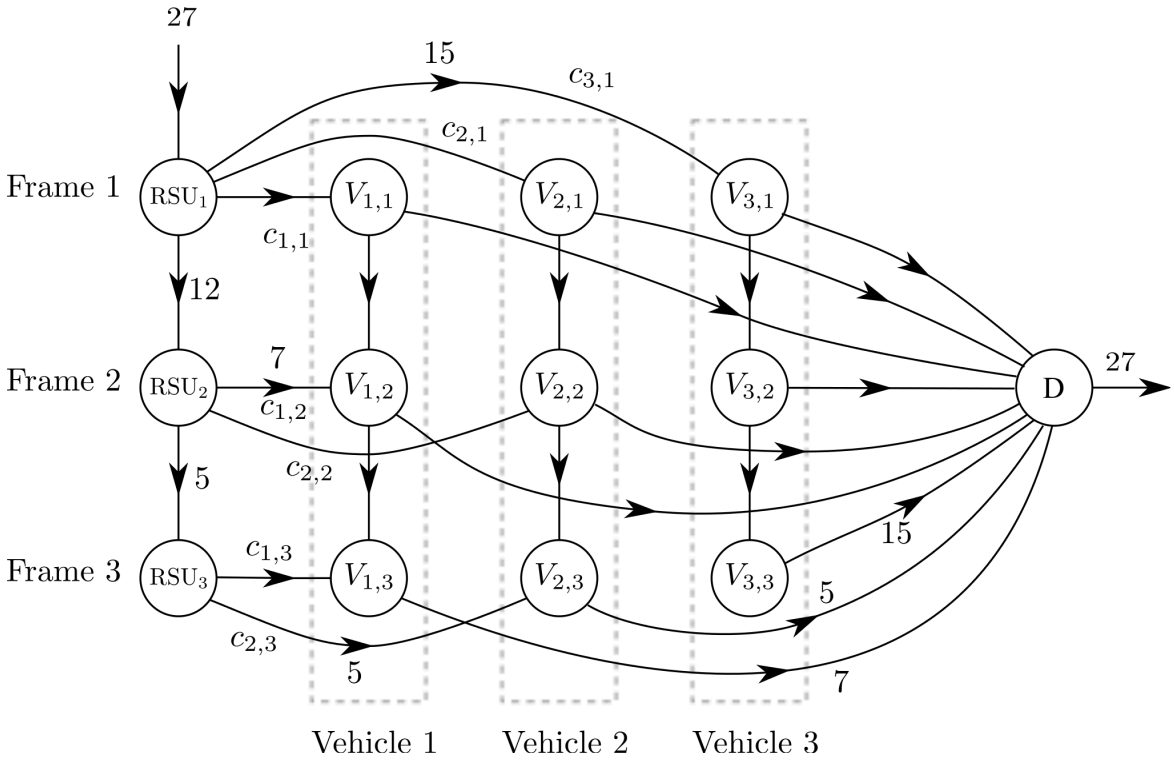

Fig. 4: Time Expanded Graph Example

$$\text{subject to} \quad \sum_{f \in \mathcal{F}} x_{v,f} \geq d_v \qquad \forall v \in \mathcal{V} \tag{9}$$

$$\sum_{v \in \mathcal{V}} x_{v,f} / B_{v,f} \leq \tau_f \quad \forall f \in \mathcal{F} \tag{10}$$

$$x_{v,f} \geq 0 \qquad \forall v \in \mathcal{V}, f \in \mathcal{F} \tag{11}$$

where $\mathcal{V}$ is the set of vehicles, $\mathcal{F}$ is the set of time frames, $x_{v,f}$ is the number of bits sent to vehicle $v \in \mathcal{V}$ during frame $f \in \mathcal{F}$ and $B_{v,f}$ is the achievable bit rate to this vehicle during this time frame. The objective in DNTG-LP minimizes the total time that the RSU transmits. Constraint (10) ensures that the sum of all time durations allocated to vehicles during a frame f does not exceed the frame duration, τ_f. Vehicle v demand d_v (in bits) is guaranteed to be satisfied by Constraint (9). This LP can be solved in polynomial time and provides an offline bound to the energy required to serve the vehicles in $\mathcal{V}$ during time period $\mathcal{F}$.

V. Online Scheduling Algorithms

In this section, algorithms are proposed which perform energy efficient online scheduling in real time. The scheduling decisions made by the online scheduler can only be made using causal input data, i.e., consisting of past and current inputs.

Algorithm 1 FCFS Algorithm

1: **for all** $t \in \{0, 1, \ldots\}$ **do**
2: **if** time slot t is assigned to vehicle k, i.e., $x_{k,t} = 1$ **then**
3: Transmit to vehicle k.
4: **end if**
5: **if** vehicle v arrives **then**
6: Set residual demand, r_v, to the vehicle v demand d_v.
7: **while** $r_v > 0$ **do**
8: Select an unassigned time slot, t', for vehicle v which maximizes its bit rate, i.e., $t' = \arg\max_{t \in \mathcal{T}_v} b_{v,t}$, where $\mathcal{T}_v$ is the set of unused time slots available to v.
9: Assign t' to vehicle v, i.e., $x_{v,t'} = 1$.
10: Reduce r_v by the payload available for vehicle v in time slot t', i.e., $r_v = \max(0, r_v - b_{v,t'})$.
11: **end while**
12: **end if**
13: **end for**

A. First-Come-First-Serve (FCFS) Algorithm

FCFS is a very low complexity scheduler motivated by the CBR version proposed in [4]. The algorithm is executed whenever a new vehicle enters the RSU radio coverage range. Upon arrival, the scheduler assigns enough time slots to satisfy the vehicle's communication requirements. This is done by assigning time slots in "highest bit rate first" order for the arriving vehicle. A more formal description of the algorithm is shown in Algorithm 1.

In step 2 the algorithm executes the current schedule, $x_{k,t}$, for the current time slot, t. If $x_{k,t} = 1$, vehicle k is communicated to by the RSU for time slot t. Vehicle k therefore receives $b_{k,t}$ bits which is the available payload. The demand for vehicle k, d_k, is then reduced by $b_{k,t}$.

In step 5, if a vehicle v arrives, the current schedule is updated. This update does not interfere with anything previously allocated for other vehicles. A temporary variable r_v is initially set to vehicle v demand d_v. In step 8, the algorithm searches for a free time slot t' during which newly arrived vehicle v can receive the highest possible bit rate from the RSU. Once this time slot is found, it is reserved in the schedule $x_{t',v}$ for some future time slot t'. This process is repeated, steps 10 to 14, until vehicle v is assigned enough time slots to satisfy its demand.

The description in Algorithm 1 is written in terms of variable bit rate time slots. However, the algorithm can also operate using the time framed (TF) model with very little change, i.e., instead of assigning maximum data rate time slots, variable length time periods are assigned in frames at the maximum data rate possible.

In Section VI it will be shown that FCFS is most suitable when vehicle speeds and demands are

relatively close and traffic load is light. When vehicle speeds are quite different, there are advantages to distinguishing them on this basis. This is the FF algorithm introduced in the next section, and motivated by a similar algorithm in [4] used in the CBR air interface case.

B. Fastest-First (FF) Algorithm

In this algorithm, the schedule is re-computed upon each vehicle arrival. Time slots are assigned using the same "highest bit rate first" priority as in the FCFS algorithm, however, the vehicles are processed in "fastest-first" order. As before, each vehicle is allocated enough time slots to satisfy its remaining demand. A more formal description of the algorithm is given in Algorithm 2.

Algorithm 2 FF Algorithm

1: **for all** $t \in \{0, 1, \ldots\}$ **do**
2: **if** time slot t is assigned to vehicle k, i.e., $x_{k,t} = 1$ **then**
3: Transmit to vehicle k.
4: **end if**
5: **if** a new vehicle arrives **then**
6: Reset schedule: $x_{k,t} = 0$ for all $k \in \mathcal{K}, t \in \mathcal{T}$
7: **while** unscheduled vehicles exist **do**
8: Select vehicle v with highest speed from unscheduled vehicles.
9: Set residual demand r_v to the vehicle v demand d_v.
10: **while** $r_v > 0$ **do**
11: Select an unassigned time slot, t', for vehicle v which maximizes its bit rate, i.e., $t' = \arg\max_{t \in \mathcal{T}_v} b_{v,t}$, where $\mathcal{T}_v$ is the set of unused time slots available to v.
12: Assign t' to vehicle v, i.e., $x_{v,t'} = 1$.
13: Reduce r_v by the payload available for vehicle v in time slot t', i.e., $r_v = \max(0, r_v - b_{v,t'})$
14: **end while**
15: **end while**
16: **end if**
17: **end for**

As in the FCFS algorithm, in step 2, the algorithm executes the current schedule, $x_{k,t}$, for the current time slot, t. If $x_{k,t} = 1$, vehicle k is communicated to by the RSU. Vehicle k receives $b_{k,t}$ bits during that time slot. The demand for vehicle k, d_k, is then reduced by $b_{k,t}$.

In step 5, if a new vehicle v arrives, the current schedule is re-computed. In step 8, the vehicles are selected to be scheduled according to their speed. A temporary variable r_v is set to the currently selected vehicle v's remaining demand d_v. In step 11, the algorithm assigns a free time slot t' during which vehicle v can receive the highest possible bit rate from the RSU. This process is repeated, steps 10 to 14, until vehicle v is assigned enough time slots to satisfy its remaining demand d_v. All other vehicles are then

processed in the same fashion. As in the FCFS algorithm, FF assumes fixed size time slots.

The description in Algorithm 2 is written in terms of variable bit rate time slots. As in the FCFS case, the algorithm can also operate using the time framed (TF) model with very little change, i.e., instead of assigning maximum data rate time slots, variable length time periods are assigned in frames at the maximum data rate possible.

The idea behind this algorithm is to provide better fairness across vehicles with different speed, since faster moving vehicles spend less time inside the RSU radio coverage range. Note that FF is more computationally expensive than FCFS, but as it will become clear later, it provides better fairness when there are vehicles with different speed.

C. Greedy Generalized Flow (G-GF) Algorithm

In section IV-B we introduced two designs that compute bounds on the minimum energy of the offline scheduling case. These algorithms can be run in polynomial time, and in this section a greedy online version of these algorithms is proposed. The Greedy Generalized Flow Algorithm (G-GF) is based on the Generalized flow construction shown in Figure 3. Since it is an online algorithm it must schedule vehicles as they arrive to the RSU. The algorithm is shown in Algorithm 3 and is described as follows.

Algorithm 3 includes a function declaration, GFG-SCHEDULER, which accepts a set of vehicles $\mathcal{V}$, time slots $\mathcal{T}$, remaining vehicle demand d_i and the number of bits available to vehicle i in time slot j, $b_{i,j}$. When this function is called, it constructs a generalized flow graph as in Figure 3 using the provided inputs. The GFG is then solved and the resulting outputs, $x_{v,t}$, become the new active schedule. As discussed previously, these outputs may be fractional.

When a new vehicle arrives (step 3), GFG-SCHEDULER is called and an updated schedule is obtained using all known information. This includes all known vehicles and the union of time slots available to this vehicle set. The algorithm then tests to see if any vehicles are scheduled for the current time slot, t, and if multiple vehicles are, it selects the one with the highest data rate, and transmits to that vehicle (step 8). Vehicle demand is then updated and if the vehicle in question is finished, the vehicle set is also updated. In step 10, if the vehicle had not been the only one assigned to time slot t, then a new schedule is obtained from GFG-SCHEDULER by excluding time slot t. Note that the same process occurs if a vehicle was assigned a fractional time slot, except that there is no need to call GFG-SCHEDULER since the system will use the existing schedule.

Algorithm 3 G-GF Algorithm

$\mathcal{V}$: set of known vehicles
$\mathcal{T}$: set of time slots available to $\mathcal{V}$
d_v : vehicle demand $\forall v \in \mathcal{V}$
$b_{v,t}$: bits available to vehicle v in time slot t $\forall v \in \mathcal{V}, t \in \mathcal{T}$
$x_{v,t}$: current schedule

for all $t \in \{0, 1, \ldots\}$ **do**
  if a new vehicle arrives **then**
    GFG-SCHEDULER($\mathcal{V}, \mathcal{T}, d_i, b_{i,j}$) $\forall i \in \mathcal{V}, j \in \mathcal{T}$
  end if
  Let $\mathcal{K}_t$ be the set of all vehicles scheduled for time slot t, i.e., $\mathcal{K}_t = \{k | x_{k,t} > 0\}$.
  if one or more vehicles are scheduled for t, i.e., $\mathcal{K}_t \neq \emptyset$ **then**
    Select vehicle $k \in \mathcal{K}_t$ with maximum available bit rate, i.e., $k = \arg\max_{i \in \mathcal{K}_t} b_{i,t}$.
    Transmit to vehicle k and update its demand, i.e., $d_k = \max(0, d_k - b_{k,t})$.
    Let $\mathcal{V}' = \{v | v \in \mathcal{V}, d_v \neq 0\}$
    if t is not completely assigned to k: i.e., $x_{k,t} < 1$. **then**
      GFG-SCHEDULER($\mathcal{V}', \mathcal{T}', d_i, b_{i,j}$), where $\mathcal{T}' = \mathcal{T} - \{t\}, i \in \mathcal{V}', j \in \mathcal{T}'$
    end if
  end if
end for
function GFG-SCHEDULER($\mathcal{V}, \mathcal{T}, d_i, b_{i,j}$) $\forall i \in \mathcal{V}, j \in \mathcal{T}$
  Construct a new GFG as in Figure 3.
  Solve the GFG and create a new schedule $x_{v,t}$ for all $v \in \mathcal{V}, t \in \mathcal{T}$.
  return $x_{v,t}$ for all $v \in \mathcal{V}, t \in \mathcal{T}$
end function

D. Greedy Time Expanded Graph (G-TEG) Algorithm

In this section we introduce an algorithm based on the design described in Figure 4. The idea is to create a time expanded graph for vehicles inside the RSU range. Upon vehicle arrival the scheduler creates a time expanded graph similar in design to the one in Figure 4. The vehicle nodes represent the ones currently inside the RSU range with unmet demands. The time frames set represent the frames during which these vehicles maintain the same achievable bit rate. The frames extend until the last vehicle of those currently inside the range exit. The algorithm steps are similar to the steps shown in Algorithm 3, except for function GFG-Scheduler where a time expanded graph is built and solved instead of a static generalized flow network. As with the G-GF, G-TEG algorithm can be executed in polynomial time.

VI. PERFORMANCE COMPARISON EXAMPLES

In this section we study the performance of the proposed algorithms by showing some representative examples of our results. The input data that was fed to the schedulers assumed a highway scenario

where vehicles maintain constant speed throughout the RSU coverage area [26] [20]. Good estimates of energy costs can be readily made in this type of situation [27][28] and the bit rates used for the downlinks are based on these vehicle position estimates. A distance dependent exponential path loss model was assumed, using a path loss exponent of $\alpha = 3$. In many practical systems however, there will be dominant deterministic propagation with random components due to propagation effects such as shadowing. Therefore, we also include results where the scheduler input data includes random errors due to log-normal shadowing components.

Vehicular traffic models have been extensively studied and in general, vehicle behavior is highly variable [29] [30]. In highway scenarios however, vehicle interaction is considered to be relatively insignificant with vehicles tending to maintain constant speeds for relatively long periods of time [31]. The vehicle arrival process is often modeled to be Poisson [26] [32] and this what is assumed in our experiments. We also model the traffic in different lanes as belonging to different classes each with different speeds as in [27]. The RSU is assumed to be in the middle of the highway segment of interest with vehicles passing by in one or more directions. The physical layer model is based on that obtained from [33] [34] where the bit rates vary according to distance from 3 to 27 Mbps. A 2 Km coverage area is assumed for the RSU.

The theoretical offline minimum energy derived by the models in Section IV-B are referred to as *TEG Bound* in the presented graphs. The values for *TEG Bound* computed either using the GF model or DNTG are very close, and therefore only one of them, i.e., DNTG, is reported in these experiments as the theoretical bound. This Bound is compared to the online algorithms (FCFS, FF, G-TEG and G-GF) we presented in Section V.

Results are presented for experiments using two classes of vehicles, and the experiments show the effects of changing demand and vehicle speed. Later, the fairness of the proposed scheduling algorithms is studied. In the graphs, each plotted point is an average of multiple runs to ensure the results are representative of the performance of the algorithms. To heavily stress the scheduling algorithms, we often subject the RSU to traffic conditions which the RSU cannot fully satisfy. In these experiments we keep track of the vehicle loss rate, i.e., the probability that a vehicle's demand cannot be satisfied. Note that the online algorithms are designed to serve as much demand as possible, i.e., they will not drop vehicle demand in favor of minimizing total energy use. Demand is only dropped when no time slots are available.

In the first experiment, we study the performance of the algorithms under light to medium loading

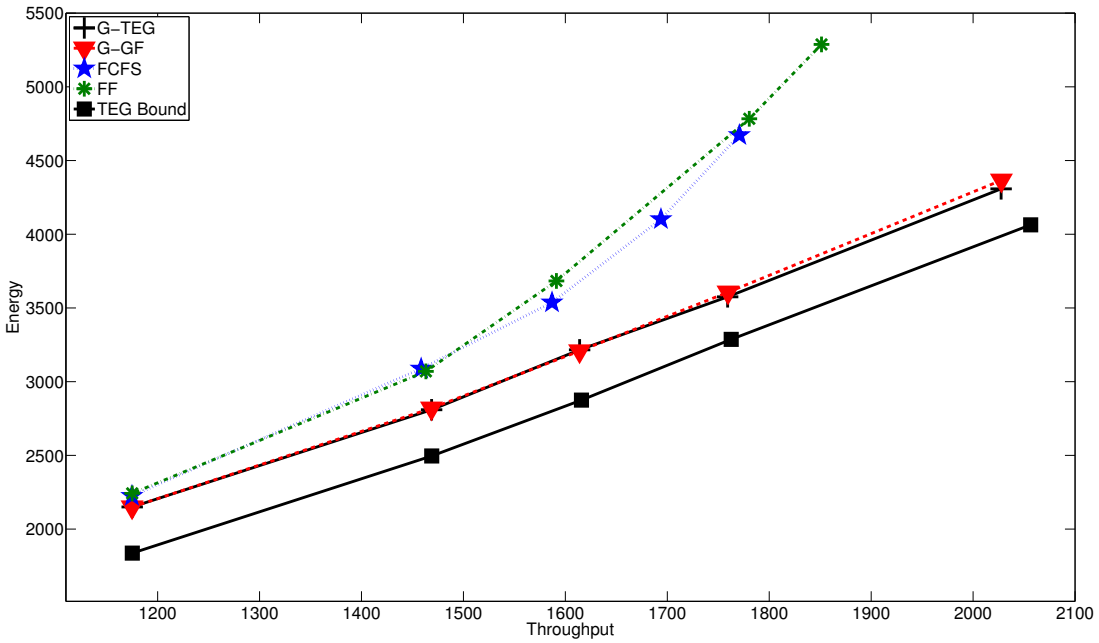

Fig. 5: Throughput and Energy Comparisons. Low to Medium Demand.

demands. The vehicular traffic consists of two classes, c_1 and c_2 where the arrival rate for both is $\lambda = 1/28$ (vehicle/time slot). The two-vehicle platooning percentage is 0.05 which means that 5% of vehicles arrive together. Vehicle speed is c_1 and c_2 are 18 and 30 m/s (65 and 108 KM/h), respectively.

Figure 5 shows the performance of the FCFS, FF, G-GF and G-TEG algorithms versus throughput. It can be seen that the G-GF and G-TEG algorithm performance is very similar and generally follows slightly about that of the Bound. The energy performance is about 25% higher than the bound under lighter throughput values and about 7.5% under the highest values of throughput plotted. Note that the gap between the bound comes from several reasons. The bound is clearly non-causal and benefits from knowledge of future inputs which the online algorithms do not have. Although the G-GF and G-TEG algorithms use myopic versions of the generalized flow and time expanded graphs, the process of rounding the fractional outputs to integer schedules incurs some energy overhead. Finally, at larger values of throughput, the algorithms incur a loss rate, i.e., unfulfilled demand, that decreases their energy use compared to the bound which does not incorporate loss into its formulation. This accounts for the lower fractional spread between the G-GF/G-TEG algorithms compared with the bound at higher values of throughput.

Under light values of throughput, the FCFS and FF algorithms can be seen to require about the same total energy as G-GF and G-TEG. This is clearly because the scheduling problem is very simple when there is almost no contention for downlink time slots. It is not possible to do any better than greedily assigning the best available time slots to incoming vehicles. For this reason, under very light traffic

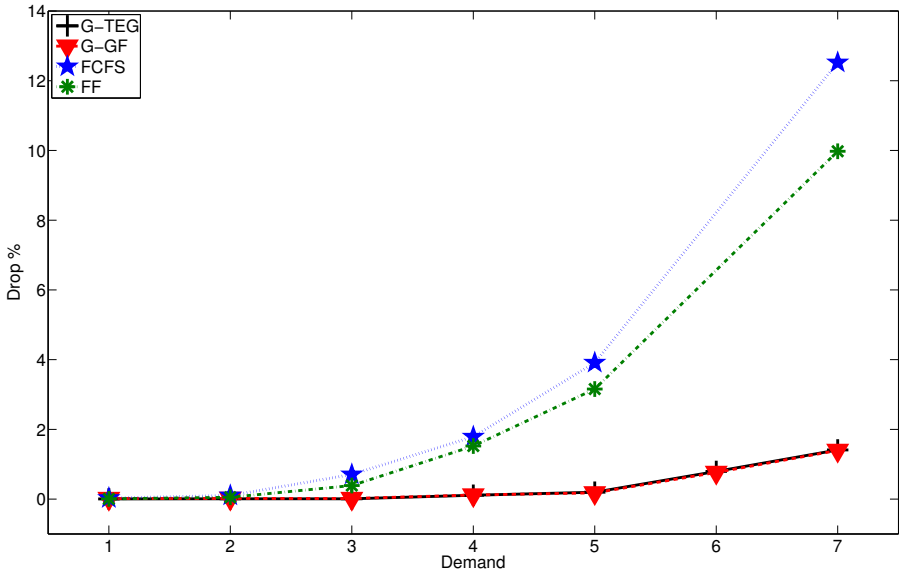

Fig. 6: Demand Drop Percentage Under Low to Medium Demand

conditions there is effectively no value in going to a much more sophisticated scheduling algorithm. Note that at lower values of throughput, all four curves eventually converge. As the throughput increases, however, this situation changes, i.e., the simplistic design of FCFS and FF begins to adversely affect their performance. For example, in the case considered in Figure 5, at the highest values of throughput plotted, the energy required by the FCFS and FF algorithms is over 30% higher than that for the G-GF and G-TEG algorithms. This is a significant improvement in energy performance.

The demand drop percentage for the experiment explained above is shown in Figure 6. Here, the load levels are the same as in Figure 5. Under light demand, the G-GF and G-TEG algorithms experience a very low dropping rate. This is due to the low probability of contention arising from more than one vehicle competing for the same downlink time slots. The G-GF and G-TEG are able to resolve this contention without the need to drop vehicle demand.

Under medium loading, the G-GF and G-TEG dropping rates are in the 1.7% range. This happens for two reasons. First, the Bound has knowledge of all future arrivals so it can anticipate contention and schedule vehicles before and after the contention period to achieve a good schedule. G-GF and G-TEG are causal algorithms and can only identify contention once the vehicles creating it arrive to the RSU. Second, the sharing of a single time slot between more than one vehicle in the Bound model is not allowed in the online algorithms, which renders some schedules created by the Bound as infeasible.

The FF and FCFS algorithms experience zero dropping under very light load as in the first and second

demand levels in Figure 6. Despite their relatively simple design, FF and FCFS in this situation encounter very little contention which they are able to resolve without dropping demand, even if the resulting schedule is less than optimal. As the demand increases (points 3 and 4 on the plot) however, FF and FCFS starts to experience dropping which is still below 2%. This is in contrast to the more computationally expensive G-GF and G-TEG algorithms which are experiencing little or no dropping. If the system was designed to allow for a small percentage of dropping (e.g., 2% or 3%) in exchange for simple to compute schedules, using FF or FCFS would be acceptable.

Under medium loading (last point on the plot), the dropping obtained by FF and FCFS are about 6 and 7.5 times the drop percentage for G-GF, respectively. Under these conditions, even a sophisticated algorithm like G-GF has no option but to start dropping demand while trading performance for computational simplicity. This indicates that under medium load, using a more efficient algorithm like G-GF and G-TEG becomes more reasonable.

It can be seen that the difference in the dropping percentage between FF and FCFS is less than 2% throughout most of these experiments. Although FF is designed to handle faster moving vehicles better than FCFS, under a low platooning percentage of 5%, the frequency of vehicles arriving together is limited. This allows for the simple FCFS algorithm to accommodate the demand of a newly arrived fast moving vehicle without dropping, even if it is in a less optimal location. This encourages the use of the less computationally expensive FCFS algorithm over the FF algorithm in traffic situations where vehicle flow experiences lower levels of arrival fluctuation.

To further study the performance of the online algorithms, Figure 7 shows the throughput and energy performance comparison under higher demand levels than in the previous experiment. Figure 8 shows the corresponding dropping behavior. In this experiment, the traffic consists of two classes of vehicles c_1 and c_2, where c_1 speed is 18 m/s and c_2 speed is 30 m/s. The vehicle arrival rate for both classes is $1/30$ vehicles/time slot and the two-vehicle platooning percentage is 10%. The increased platooning and demand levels allows for studying the algorithm behavior when experiencing heavy load conditions. The figures show that the performance of the two greedy algorithms (G-GF and G-TEG) is very close despite the two different modeling approaches. As slow moving vehicles or the choice of very small time slots can cause the problem size to grow very large in the G-GF case, it is best in this situation to use the G-TEG algorithm. Under heavy load, G-GF and G-TEG experience about 3% demand dropping. Despite the dropping, throughput continues to grow for these algorithms and they do not experience a saturation

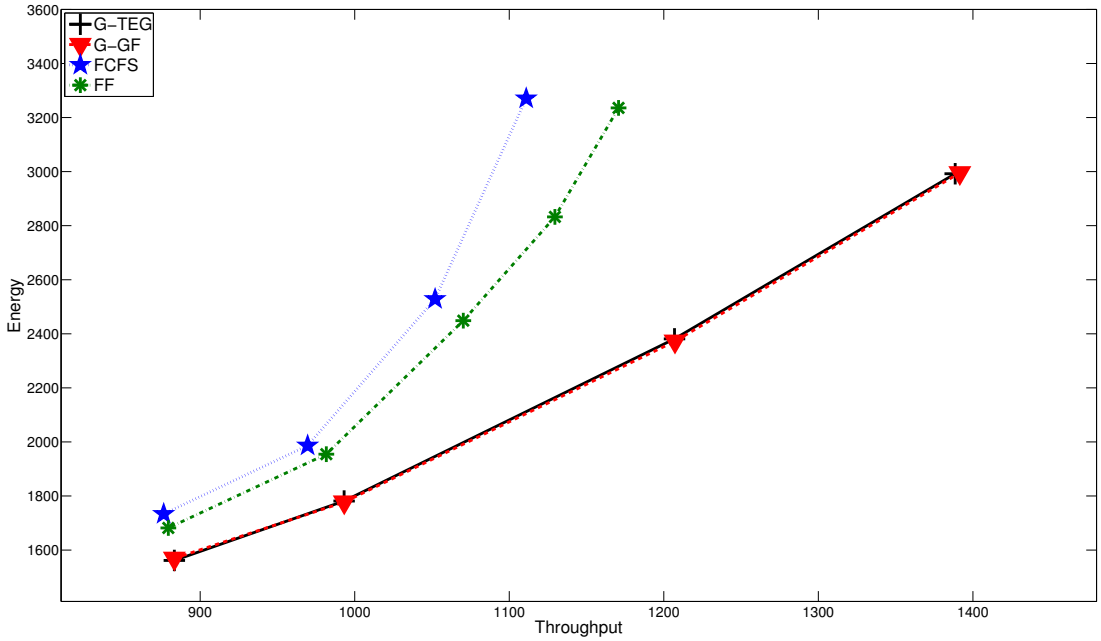

Fig. 7: Energy Performance under High Demand

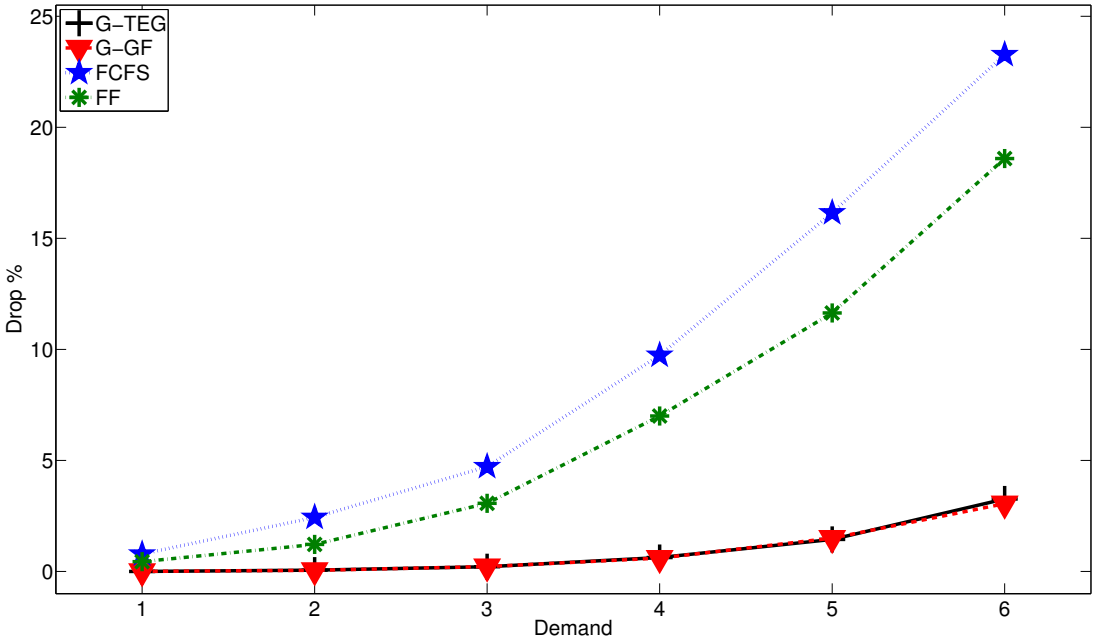

Fig. 8: Dropping Performance under High Demand

phenomenon where the throughput fails to increase while the energy use increases with demand. This is due to the fact that G-GF and G-TEG are able to resolve contention in an efficient way, even while some dropping is needed. On the other hand, the FCFS and FF algorithm throughput saturates and the curve starts to grow vertically under heavy load, indicating that no more demand can be served. This means that the maximum performance for FF and FCFS is lower than their G-GF and G-TEG counterparts. Identifying the demand threshold would be important when using any of these algorithms. We can also

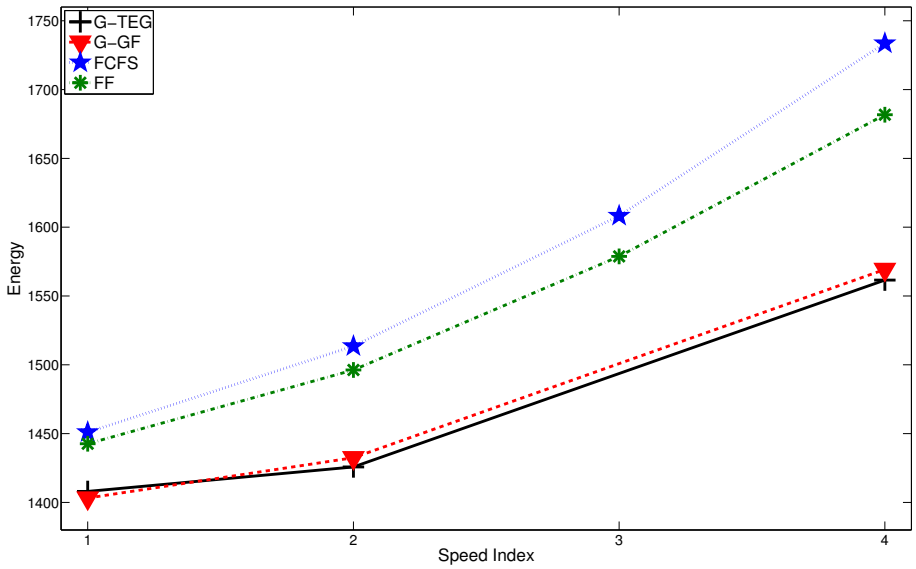

Fig. 9: Energy Use under Light Load for Different Speed Indices

see that in this experiment, there are more significant performance differences between the FF and FCFS algorithms. This is mainly due to the fact that the platooning percentage has increased which creates more frequent contention. In this case, FF offers faster moving vehicles a chance to make use of high bit rate time slots which the statically assigned schedule created by FCFS would deny. This shows that if the traffic to be served by the RSU is characterized by high variations in the arrival of vehicles, it is favorable to use FF over FCFS.

Vehicle speed is strongly related to the amount of energy required and the drop percentage that the vehicles demand experiences. The reasons behind this are that as velocity increases, the number of available time slots decreases for a particular vehicle. Also, in the case of two classes of vehicles traveling at different speeds, faster vehicles are catching up with more vehicles from the slower moving class. This creates more frequent contention periods.

The performance of the algorithms under different speed conditions is shown in Figures 9 and 10. The parameters for these experiments are the same as in the previous cases except for the vehicle speeds. While class c_1 maintains a velocity of 18m/s, class c_2 is changed. The speed index from 1 to 4 refers to testing c_2 under velocities: 18, 22, 26 and 30 m/s. In the figures, the first point on the x-axis represents the two classes traveling at the same speed. In Figure 9, under light loading, the energy required by FF and FCFS is very close. This is expected as it is easy for either the static (FCFS) or the dynamic (FF) algorithms to find a good schedule. But in Figure 10, the difference between them is about 13%. Although

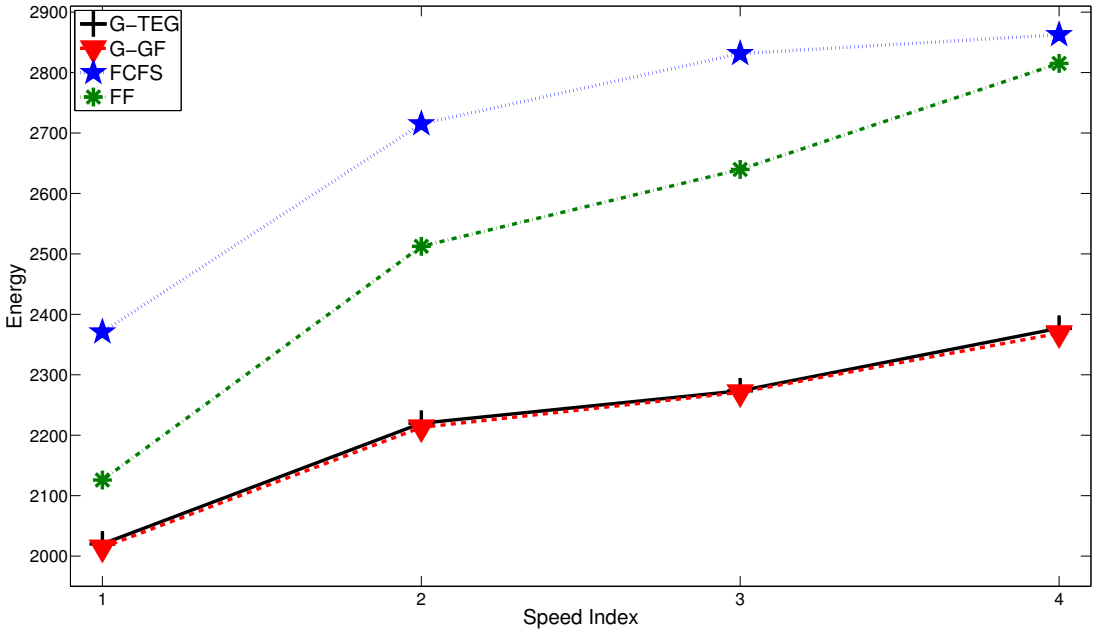

Fig. 10: Energy Use under Medium Load for Different Speed Indices

FF does not differentiate and prioritize vehicles based on their speed, the dynamic nature of FF allows the recalculation of the schedule every time a new vehicle arrives. With no prioritization, a newly arrived vehicle can be given the chance to communicate with the RSU during the time periods when it is enjoying a high bit rate. On the other hand, FCFS is a statically assigned schedule. Previously arrived vehicles keep all previously assigned time slots. In situations where the demand is high even if the traffic flow is steady, using the more computationally expensive FF is favourable. The above reasoning also explains why the gap between FF and FCFS under light load is smaller than the gap under heavier loading.

Table I shows results for the performance of the algorithms under different traffic speed mixtures. The first obvious observation is that under low or medium demand levels, and for all online algorithms, as the speed index increases, the required energy increases. The reason for this is that the algorithms are forced to communicate with faster moving vehicles at further distances and lower bit rates, causing them to use more time slots. However, this does not necessarily translate into a change in the throughput. First we examine the low demand behavior. Both G-GF and G-TEG maintain constant throughput as no dropping occurs and they are able to successfully schedule all requests even as the speed for class c_2 is increasing. In FCFS and FF, the throughput decreases with the speed index increase. FF throughput decreases by about 1% while FCFS throughput decreases by about 2.3%. Under medium load the situation becomes clearer. While G-GF and G-TEG throughput drops by less than 1%, with energy increases of only 17% and 17.6%, respectively, FCFS and FF throughput drops by 7.5% and 6.9% with energy increases of 20%

	Demand		Speed Index 1	2	3	4
FCFS	Low	Throughput	993	983	981	970
		Energy	1549	1816	1883	1987
	Med	Throughput	1186	1147	1101	1096
		Energy	2370	2715	2831	2862
FF	Low	Throughput	993	992	989	981
		Energy	1534	1796	1866	1954
	Med	Throughput	1213	1198	1174	1129
		Energy	2125	2512	2639	2815
G-GF	Low	Throughput	993	993	993	993
		Energy	1513	1698	1731	1777
	Med	Throughput	1214	1213	1211	1207
		Energy	2014	2212	2271	2368
G-TEG	Low	Throughput	993	993	993	993
		Energy	1531	1705	1738	1781
	Med	Throughput	1214	1213	1211	1206
		Energy	2020	2219	2273	2377

TABLE I: Throughput and energy requirement under different demand and velocity conditions for the online algorithms

and 32%, respectively. These results show that G-GF and G-TEG are far more suitable for medium to high demand situations.

In the next set of results we show comparisons where the input data to the schedulers is not ideal. This was done by including random log-normal shadowing in the propagation model estimates. This is done to ensure that when the scheduler input data are not ideal, the algorithms do not produce results which would be biased in some way, due to this randomness. In the results presented, this was done by adding propagation shadowing effects to the extracted data, which result in unpredictable randomness in these estimates. The scheduling is therefore based on this input, but the actual costs incurred include the energy perturbations due to the random components. The random shadowing components are unknown to the online schedulers which base their decisions on deterministic positional information only, while the theoretical bound can make use of the shadowing information to produce a proper bound on the energy required to serve vehicle demand.

Figure 11 shows the performance differences between the online algorithms and the Bound when these effects are introduced with shadowing standard deviation of 4 dB. In this experiment, the two-vehicle platooning percentage is 10%, two classes of traffic are simulated, and the arrival rate for the two classes is both $1/30$ vehicles/time slot. Vehicle speeds for c_1 and c_2 are 18 and 30 m/s, respectively.

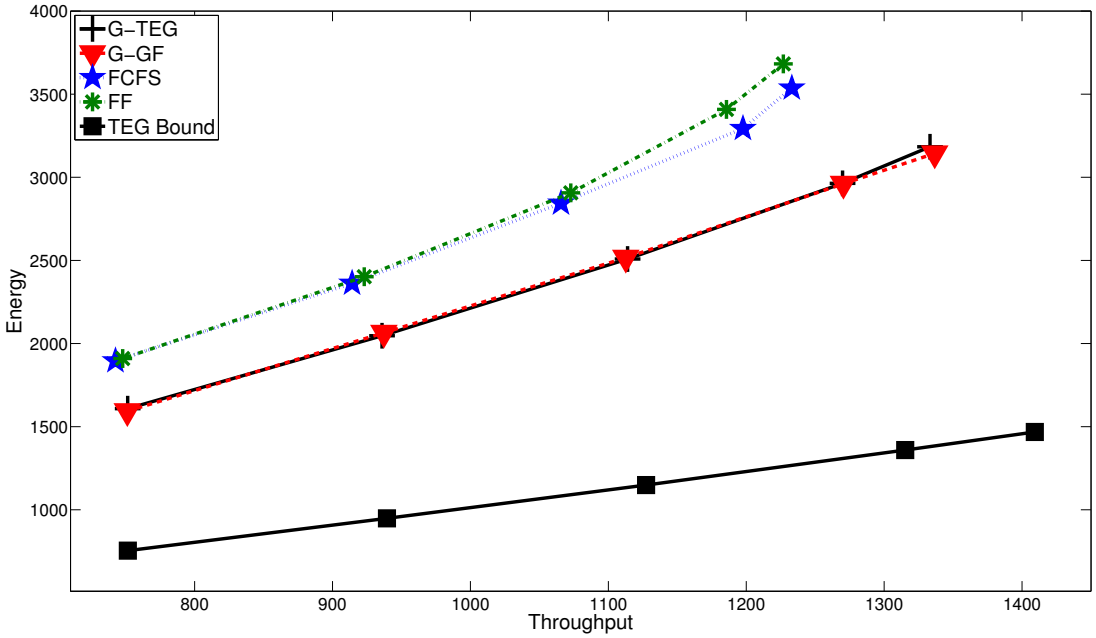

Fig. 11: Throughput Performance with Log-Normal Shadowing

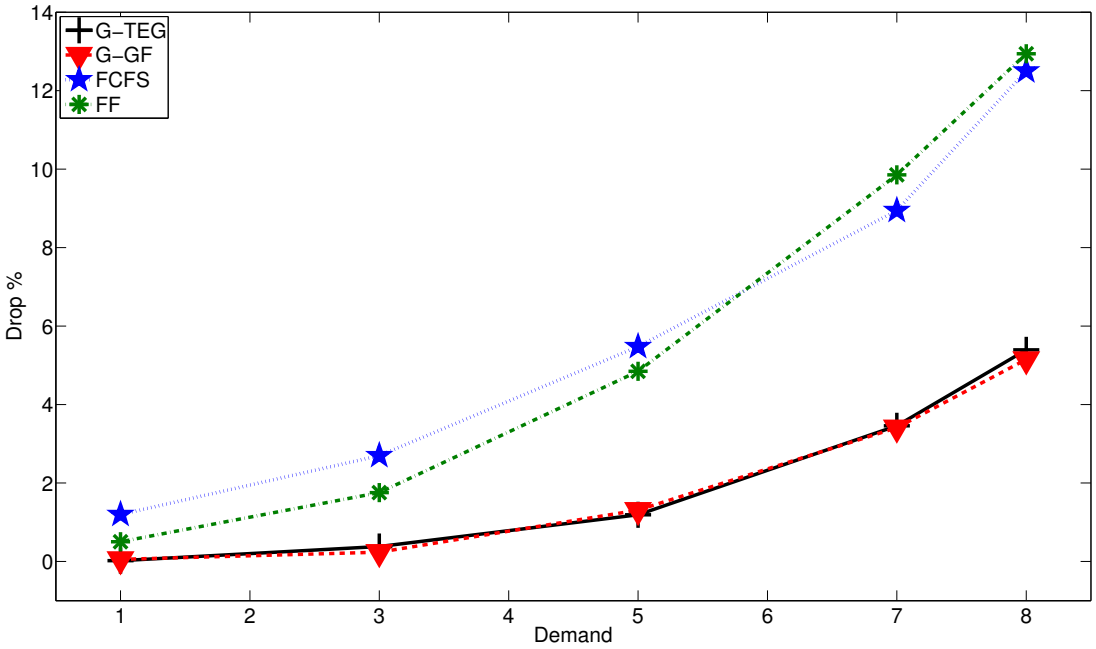

Fig. 12: Demand Drop Percentage with Log-Normal Shadowing

The effect of including the shadowing shows in the increased difference between the Bound and the online algorithms when compared to the case in Figure 5. The reason behind this is that as the theoretical Bound takes into account the instantaneous shadowing effect, whereas the online algorithms are only aware of it at packet transmission time and not when the scheduling occurs. When the actual bit rate at the execution time is different than the one the online algorithm based its decision upon, rescheduling is triggered and extra computations need to be done in order to accommodate the change in achievable

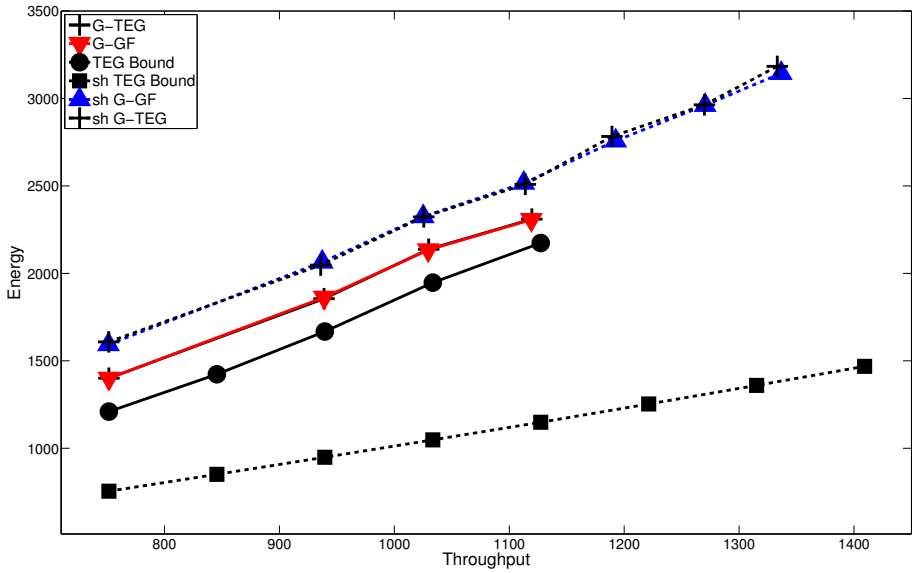

Fig. 13: G-GF and G-TEG Performance with and without Shadowing

bit rate. Figure 12 shows the dropping associated with this experiment. The main effect of including the log-normal shadowing is that the drop percentage is higher compared to the previous cases. For example, G-GF and G-TEG starts dropping at medium demand levels, which did not happen before. Faced with instantaneous bit rates different than the ones used to create the schedule, G-GF and G-TEG are forced to reschedule the rest of the time slots remaining for the vehicles which can result in a poorer schedule and increased dropping. However, these algorithms have maintained their superior performance over the simpler ones, i.e., FCFS and FF.

Figure 13 shows the effect of including the shadowing on the differences between the Bound and the two online algorithms, G-GF and G-TEG, where the throughput for both cases (with and without shadowing) is shown. The experiment is for low to medium demand to ensure the feasibility of the bound calculations. The lines which represent the results of including the shadowing in both the bound and the two online algorithms starts with “sh”. When including the log-normal shadowing, the differences between G-GF and G-TEG on the one side and the bound on the other becomes much bigger than in the deterministic case. We also notice that the bound for the shadowing case is *lower* than the bound for the deterministic case. This is attributed to the fact that, unlike the online cases, the offline algorithms are aware of the random channel fluctuations and can exploit them in their scheduling.

An issue that arises when vehicular demands are not fully fulfilled is one of fairness. For example, an algorithm could choose to drop demands from faster moving vehicles whose fulfillment would result in

Method	Average Load	High Load
G-TEG	0.98	0.99
G-GF	0.98	0.99
FF	0.83	0.91
FCFS	0.53	0.62

TABLE II: Jain's Fairness for Different Algorithms

disproportionately increased energy requirements. Clearly there is a tradeoff between fairness and total energy use. In order to assess whether vehicles of different classes are treated fairly, *Jain's Fairness Index* is used to measure the fairness across the classes. This is defined as

$$f(x_1, x_2, x_3, \ldots, x_n) = \frac{\sum_{i=1}^{n} {x_i}^2}{n \sum_{i=1}^{n} x_i^2} \tag{12}$$

where the index value is in the range $0 \leq f() \leq 1$. x_i represents the percentage loss each traffic class encounters individually. If all traffic classes are treated equally, Jain's Fairness Index will be 1. The worst case unfairness when there are two classes of traffic, is 0.5. We will show some representative examples from our experiments.

Table II shows that at high speed index, G-GF and G-TEG achieve almost perfect fairness across the vehicles. They are followed by the FF algorithm, while the FCFS suffers the most. The algorithms G-GF and G-TEG treat all vehicles equally when it comes to dropping demand. For example, in G-GF, the cost over the edges between vehicles and bit rate nodes depends on which bit rate is higher and does not depend on vehicle velocity. Thus, when the dropping is needed, no distinction between vehicles is made, which translates into good levels of fairness.

The fairness of FF compared to FCFS is shown in Tables III and IV. As mentioned earlier, the speed index number increase refers to the difference in velocity between classes c_1 and c_2. Table III shows that FF fairness actually increases as the difference in vehicle speed increases. This is because as the speed difference increases, the priority given to faster moving vehicles becomes higher. Therefore, despite the much shorter time they spend inside the RSU coverage range, they obtain treatment comparable to their slower moving counterparts. Table IV shows that FCFS suffers as the difference between vehicle speeds increases. This is due the static nature of the FCFS algorithm where it treats all vehicles as jobs in a queue, despite the fact that faster vehicles spend less time inside the RSU coverage range, which increases their probability of dropping.

Speed Index	Low Load	Average Load	High Load
2	0.54	0.61	0.66
3	0.59	0.69	0.78
4	0.72	0.85	0.92

TABLE III: FF Fairness for Different Speed Indices

Speed Index	Low Load	Average Load	High Load
2	0.74	0.82	0.87
3	0.59	0.69	0.73
4	0.52	0.58	0.62

TABLE IV: FCFS Fairness for Different Speed Indices

VII. Conclusions

In this paper we have considered the energy efficient roadside unit (RSU) scheduling problem when the RSU-to-vehicle radios use a variable bit rate (VBR) air interface. Offline scheduling formulations were given which provide lower bounds on the energy needed to satisfy arriving vehicular demand. An integer linear program (ILP) was introduced which can be used to find optimal offline variable bit rate time slot schedules. It was shown that this problem is NP-complete by a reduction from the well-known Santa Claus Problem.

Two flow graph based models were introduced to solve the minimum energy VBR scheduling problem. The first uses Generalized Flow (GF) graphs which represent time slots as individual graph nodes. The second uses time expanded graphs (TEGs) which model the system as a time expanded flow graph. Both of these models can be efficiently solved and provide lower bounds on energy performance. They also provide the basis for some proposed online schedulers. Four energy efficient online scheduling algorithms were considered. The first, First Come First Serve (FCFS) is very simple and treats all vehicles equally and in order of arrival. The second, Fastest First (FF), gives priority to faster moving vehicles since their transit time through the RSU coverage area is less than slower moving vehicles. The Greedy Generalized Flow (G-GF) and Greedy Time Expanded Graph (G-TEG) algorithms are two online implementations of the two flow graph models. Results from a variety of experiments showed that the proposed algorithms perform well compared to the energy lower bound. Our results show that less computational intensive algorithms, i.e., FCFS and FF, can perform well under light load, but G-GF and G-TEG, while more computational intensive, can provide near optimal throughput and minimum drop percent in demand. The results also show that the G-GF and G-TEG algorithms are much more fair than the simpler algorithms

in heavy load situations.

REFERENCES

[1] Federal Communications Commission, "FCC 03-024. FCC Report and Order," February 2004.

[2] R. Uzcategui and G. Acosta-Marum, "Wave: A Tutorial," *IEEE Communications Magazine*, vol. 47, no. 5, pp. 126–133, 2009.

[3] S. Peirce and R. Mauri, "Vehicle-Infrastructure Integration (VII) Initiative: Benefit-Cost Analysis: Pre-Testing Estimates," *Intelligent Transportation Systems Joint Program Office, United States Department of Transportation, Washington, DC.*, March 2007.

[4] A. A. Hammad, G. H. Badawy, T. D. Todd, A. A. Sayegh, and D. Zhao, "Traffic Scheduling For Energy Sustainable Vehicular Infrastructure," *IEEE Global Communications Conference (IEEE GLOBECOM'2010), Miami, Fla.*, December 2010.

[5] M. Azimifar, "Vehicle-to-Vehicle Forwarding in Green Vehicular Infrastructure," Master's thesis, McMaster University, September 2013.

[6] M. Azimifar, T. D. Todd, and G. Karakostas, "Vehicle-to-Vehicle Forwarding in Green Vehicular Infrastructure," *submitted for publication*, 2013.

[7] Y. Khaled, M. Tsukada, J. Santa Lozano, and T. Ernst, "On The Design of Efficient Vehicular Applications," *Proc. of IEEE VTC09, Barcelona, Spain, April*, 2009.

[8] F. Li and Y. Wang, "Routing in Vehicular Ad Hoc Networks: A Survey," *IEEE Vehicular Technology Magazine*, vol. 2, no. 2, pp. 12–22, 2007.

[9] L. Zhang, Q. Wu, A. Solanas, and J. Domingo-Ferrer, "A Scalable Robust Authentication Protocol For Secure Vehicular Communications," *IEEE Transactions on Vehicular Technology*, 2009.

[10] V. Bychkovsky, B. Hull, A. Miu, H. Balakrishnan, and S. Madden, "A Measurement Study of Vehicular Internet Access Using In Situ Wi-Fi Networks," in *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking*. ACM, 2006.

[11] J. Ott and D. Kutscher, "Drive-thru Internet: IEEE 802.11b for Automobile Users," *INFOCOM 2004*, vol. 1, March 2004.

[12] J. Mittag, F. Schmidt-Eisenlohr, M. Killat, J. Harri, and H. Hartenstein, "Analysis and Design of Effective and Low-Overhead Transmission Power Control for VANETs," in *Proceedings of the Fifth ACM International Workshop on Vehicular Inter-Networking*. ACM, 2008.

[13] M. F. Jhang and W. Liao, "On Cooperative and Opportunistic Channel Access for Vehicle to Roadside (V2R) Communications," *GLOBECOM*, Dec. 2008.

[14] J. Zhao, T. Arnold, Y. Zhang, and G. Cao, "Extending Drive-Thru Data Access by Vehicle-to-Vehicle Relay," *VANET '08: Proceedings of the Fifth ACM International Workshop on Vehicular Inter-Networking*, 2008.

[15] A. Nandan, S. Das, G. Pau, M. Gerla, and M. Sanadidi, "Co-operative Downloading in Vehicular Ad-Hoc Wireless Networks," *WONS*, Jan. 2005.

[16] Y. Zhang, J. Zhao, and G. Cao, "On Scheduling Vehicle-Roadside Data Access," in *Proceedings of the Fourth ACM International Workshop on Vehicular Ad Hoc Networks*, 2007.

[17] F. Chen, M. P. Johnson, Y. Alayev, A. Bar-Noy, and T. F. La Porta, "Who, When, Where: Timeslot Assignment to Mobile Clients," *IEEE Transactions on Mobile Computing*, vol. 11, no. 1, pp. 73–85, 2012.

[18] J. Alcaraz, J. Vales-Alonso, J. García-Haro *et al.*, "Control-Based Scheduling With QoS Support for Vehicle to Infrastructure Communications," *IEEE Communications*, 2009.

[19] F. Zou, J. Zhong, W. Wu, D.-Z. Du, and J. Lee, "Energy-Efficient Roadside Unit Scheduling for Maintaining Connectivity in Vehicle Ad-Hoc Network," in *Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication*. ACM, 2011, p. 64.

[20] A. A. Hammad, T. D. Todd, G. Karakostas, and D. Zhao, "Downlink traffic scheduling in green vehicular roadside infrastructure," vol. 62, no. 3, pp. 1289–1302, March 2013.

[21] A. Kezrian, "Scheduling in Green Vehicular Infrastructure with Multiple Roadside Units," Master's thesis, McMaster University, September 2013.

[22] N. Bansal and M. Sviridenko, "The Santa Claus Problem," in *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*. ACM, 2006, pp. 31–40.

[23] K. D. Wayne, "A Polynomial Combinatorial Algorithm For Generalized Minimum Cost Flow," *Mathematics of Operations Research*, vol. 27, no. 3, pp. 445–459, 2002.

[24] R. Ahuja, T. Magnanti, and J. Orlin, "Network Flows: Theory, Algorithms, and Applications," *Journal of the Operational Research Society*, vol. 45, no. 11, pp. 1340–1340, 1994.

[25] B. Kotnyek, "An Annotated Overview of Dynamic Network Flows," 2003.

[26] M. Khabazian and M. Ali, "A Performance Modeling of Connectivity in Vehicular Ad Hoc Networks," *IEEE Transactions on Vehicular Technology*, vol. 57, no. 4, pp. 2440–2450, 2008.

[27] S. Wang, "The Effects of Wireless Transmission Range on Path Lifetime in Vehicle-formed Mobile Ad Hoc Networks on Highways," in *IEEE International Conference on Communications, 2005. ICC'2005.*, vol. 5. IEEE, 2005, pp. 3177–3181.

[28] R. G. C. Sommer, D. Eckhoff and F. Dressler, "A Computationally Inexpensive Empirical Model of IEEE 802.11p Radio Shadowing in Urban Environments," in *Eighth International Conference on Wireless On-Demand Network Systems and Services*, 2011, pp. 84–90.

[29] J. Harri, F. Filali, and C. Bonnet, "Mobility Models for Vehicular Ad Hoc Networks: A Survey and Taxonomy," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 4, pp. 19–41, 2009.

[30] F. Martinez, C. Toh, J. Cano, C. Calafate, and P. Manzoni, "A Survey and Comparative Study of Simulators for Vehicular Ad Hoc Networks (VANETs)," *Wireless Communications and Mobile Computing*, vol. 11, no. 7, pp. 813–828, 2011.

[31] D. Helbing, "Traffic and Related Self-driven Many-particle Systems," *Rev. Mod. Phys.*, vol. 73, pp. 1067–1141, Dec 2001. [Online]. Available: http://link.aps.org/doi/10.1103/RevModPhys.73.1067

[32] K. Bilstrup, E. Uhlemann, E. Strom, and U. Bilstrup, "Evaluation of the IEEE 802.11p MAC Method for Vehicle-to-Vehicle Communication," in *IEEE 68th Vehicular Technology Conference, 2008. VTC 2008-Fall.*, September 2008.

[33] S. Demmel, A. Lambert, D. Gruyer, A. Rakotonirainy, and E. Monacelli, “Empirical IEEE 802.11p Performance Evaluation on Test Tracks,” in *IEEE Intelligent Vehicles Symposium (IV) 2012*, 2012, pp. 837–842.

[34] V. Shivaldova, G. Maier, D. Smely, N. Czink, A. Alonso, A. Winkelbauer, A. Paier, and C. Mecklenbrauker, “Performance Evaluation of IEEE 802.11p Infrastructure-to-Vehicle Tunnel Measurements,” in *7th International Wireless Communications and Mobile Computing Conference, IWCMC 2011*, 2011, pp. 848–852.