Governing Energy for Parked Cars

Riccardo Crepaldi, Ryan Welsh, and Robin Kravets
University of Illinois at Urbana-Champaign
{rcrepal2,rjwelsh2,rhk}@illinois.edu

***Abstract*—Vehicular networks offer service coverage for urban environments that would otherwise be too expensive for infrastructure-based networks to provide. While many services have already been proposed based on collaboration between moving cars, the inclusion of parked cars in these systems extends their reach, coverage and stability. However, despite the presence of a large car battery, cars still suffer from limited energy availability when they are parked. Essentially, if there is not enough energy to power the system for the whole duration of the stop, energy must be saved by shutting down the services for part of the stop duration.**

In this paper, we propose an energy management framework that governs the energy provisioning of a service. Our governor guarantees the most effective energy utilization for the system, requiring minimum effort from the service designers to attach to the system. We demonstrate the effectiveness of our governor through simulation of our LoadingZones Internet connectivity service and the feasibility of the architecture through the evaluation of our prototype system.

I. Introduction

The availability of mobile services, including accessing content or connectivity, has exploded, and with it, the demand for users to access those services over limited wireless and cellular infrastructures from wherever they may be, especially in their cars. With the increasing number of devices and the high rates at which data is generated and requested, the combined user bandwidth requirements will quickly overwhelm the current infrastructures. In response, distributed networks of vehicles have been proposed to enhance local services, ultimately providing relief of some of the stress on the infrastructure [1], [2]. Since connectivity between cars is challenged by high mobility and uncoordinated deployment, the inclusion of parked cars has been proposed to improve network stability and so system and service performance [1], [2]. While introducing parked cars seems to be a straightforward extension, despite the large battery in every car, energy management becomes a serious challenge.

Although energy management is a well-researched field and effective energy management technologies and algorithms are commonly implemented in existing systems, optimizing energy for services supported on a vehicle that may be mobile or stationary faces new and difficult challenges. The major challenge comes from the goal to provide a service continuously, which requires different optimizations than for a single user using their own device. Additionally, when a car is parked for a long time, the energy stored in the battery might not be enough to continuously run the desired services. To enable the inclusion of parked cars, it is necessary to devise an energy-efficient schedule that governs when a service should be turned on based on the demand for that service over time.

Although there are many services that can benefit from the use of parked cars, in this paper we focus on our Internet connectivity service, LoadingZones, which is an opportunistic internet connectivity service using parked cars as relays [1]. To support LoadingZones, it is necessary to determine a *schedule* that indicates when the system should be active based on monitored system metrics (i.e., time or battery level) and metrics specific to LoadingZones (i.e., density of cars) that determine the utility of the service. For example, a good schedule for LoadingZones would be to provide the service when the traffic flow is high. Essentially, the schedule does not specify exact times at which the system should be active, but rather what conditions should trigger its activation.

In this paper, we present the design and evaluation of a governor that provides energy efficiency using the schedule to determine when the system, and so LoadingZones, should be activated or deactivated based on the utility of the service. Although utility functions are often used to hide the complexity of a system, it is only necessary to approximate the utility of the service (i.e., density of cars) and then apply a linear utility function to that metric. Given the dynamic nature of any monitored metric and the challenge of accurately estimating stop duration, the governor takes an adaptive approach that recomputes the schedule as the system evolves. We evaluate the effectiveness of the governor and this scheduled version of LoadingZones through simulation and validate our design on a prototype system. Our results show that the governor can effectively allocate energy for LoadingZones in dynamic environments.

The rest of this paper is organized as follows. In Section II, we discuss the difference between application- and service-oriented energy management and discuss why current service-oriented techniques cannot be directly applied to services in vehicular networks. Section III presents our LoadingZones service and its associated utility function. In Section IV, we describe the system model and architecture of the governor. In Section V, we provide a test case for proving optimal service for LoadingZones and present the results from our prototype in Section VI. Finally, in Section VII, we discuss ongoing work.

II. Distributed network of services

Mobile systems were originally designed for *application*-oriented devices that focused on responding to user input and requests. However, as mobile devices become more powerful, they have been used to build *service*-oriented distributed systems, such as wireless mesh, sensor or vehicular networks, where nodes cooperate to provide network-wide services to monitor the environment [3], react to specific events, or create a distributed network for users to access the Internet or each other [4], [5], [1], [6], [7].

For vehicular networks, recent proposals have suggested the use of parked cars to improve system performance by introducing more stability into the network [1], [2]. However, none of these approaches have considered the fact that although services in parked cars do have access to the car battery, that battery is not unlimited and must always provide the power to start the engine when required. To ensure no chance of interfering with the car's ability to start, a dedicated battery can be used to exclusively power the networking and other service equipment. However, independent batteries are even more limited in size and so also limit the amount of time any services can be active. When a device does not have enough energy to provide continuous service, it is necessary to plan a schedule and alternate periods of activity with periods of sleeping to save energy. Careful planning can mean the difference between beneficial use of energy to provide services that are actually used, or wasting it when requests for the service are low.

Energy management for *application-oriented* systems focuses on improving the efficiency of the service when it is running and so extending the amount of time a service can be available. However, it does not address the problem of scheduling service availability to maximize service utilization. In comparison, energy management for *service-oriented* systems is a complex problem that is still being researched. The main goal of service-oriented systems is to provide a service until a specific deadline. For example, a monitoring system may be needed for a week after its deployment before batteries can be replaced, or should guarantee a service overnight while waiting for the sun to rise and recharge its batteries. If the devices are active until there is no energy available, the service will work at full potential initially and eventually be completely silent. If there is not enough energy to last until the next recharge, devices can alternate periods of activity and periods of sleeping to save energy and spread service availability over time.

In many mobile service-oriented systems, such as sensor networks, the main source of energy consumption is the wireless network interface. Synchronization mechanisms such as PSM, or signaling-based protocols such as B-MAC [8], X-MAC [9], or NPM [10] save energy by enabling long periods of sleep during idle times. If the network is dense enough and cooperation can be exploited, more aggressive approaches, such as temporarily shutting down entire nodes [11], can be applied without disrupting the service [12]. When a sleeping node needs to be woken up, a trigger is used to turn the system back on (i.e., Wake on Lan [13], Wake on Wireless [14] or passive [15]), significantly reducing energy consumption of the wireless component.

While existing techniques can be used to reduce network energy consumption, they might not be as effective in vehicular networks since the nodes are large vehicles equipped with powerful and expensive hardware capable of providing multiple, complex services. For example, consider the embedded system designed and deployed for LoadingZones (see Table I). The energy required by the network interface is just a fraction of the overall energy consumption. Additionally, while signaling-based methods reduce idle energy consumption, they assume that the switching time is almost immediate, so that a trigger signal can wake up the radio in a matter of milliseconds. Vehicular networks introduce two complications to applying such solutions. First, in a complex architecture like what we would expect in a vehicular network, most of the components must be off to effectively save energy. Switching from an energy saving state to an active state implies booting or resuming a complex operating system, establishing network connections and activating one or more services. This transition could take at least a few seconds. Second, the average contact duration between two cars is also only a few seconds [1]. Essentially, if the system waits until two cars come in contact with each other to take the system out of sleep mode, by the time the system is finally active and ready to provide the services, the contact is almost at its end.

Component	Power consumption
CPU - High Freq	220mA
CPU - Low Freq	350mA
WiFi - Full Power	100mA
WiFi - Low Power	50mA
GPS receiver	$\approx 40mA$

TABLE I
TYPICAL POWER CONSUMPTION VALUES FOR AN EMBEDDED SYSTEM BASED ON THE AMD GEODE CPU, SUITABLE FOR VEHICULAR-BASED APPLICATIONS.

III. SERVICE-DEPENDENT SCHEDULING

The benefits of activating a service can be measured by introducing the concept of the *utility* of a service, which depends on the particular use or performance of a given service. For example, the utility of a routing service can be quantified by the number of messages being routed, while the utility of a sampling service can be quantified by the number of samples per second. Given knowledge of the amount of energy available for a given time period, and the utility function for a given service, the goal of an optimal energy management strategy is to determine a active-sleep schedule for the system that satisfies the following two properties: (1) all available energy from the battery should be used during a stop (i.e., no available energy is left unused); (2) the chosen active times for a service maximizes the utility of the system.

One of the biggest challenges is that it is often very difficult to come up with a perfect utility function for a service. However, given the dynamics of our target systems, perfection is not needed. Instead, it is beneficial just to have an approximation of the utility of a service.

A. System and Energy Model

Each vehicle uses an embedded device that can be active, requiring power P_a, or idle, requiring a much lower power, P_i. The vehicles are either moving, in which case no energy management is needed, or parked for time δ_s and have available energy e_{av}. In this section, we assume that the duration

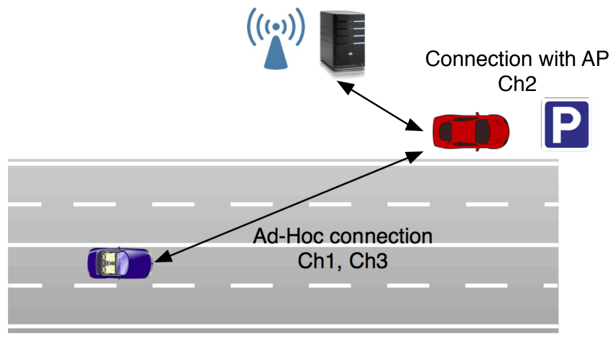

Fig. 1. LoadingZones architecture

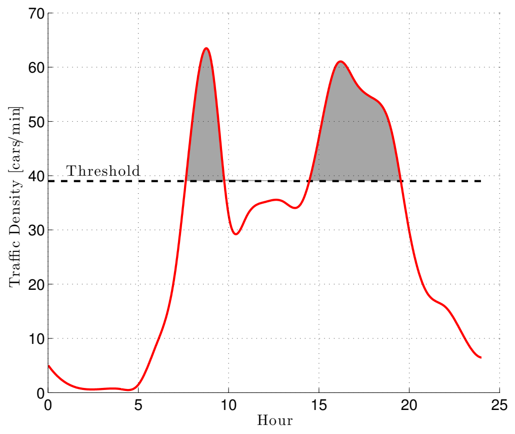

Fig. 2. Traffic density averaged over a year of monitoring for a road in Hampshire county, MT. The gray area represents the interval during which the density exceeds a hypotetical threshold, and the system is active.

of the stop is known to the system. Since the stop duration in a deployed system must be estimated, in the rest of this paper we consider the duration of the stop to be a random variable with normal distribution, $\mathcal{N}(\mu_s, \sigma_s)$. Although it might seem difficult to accurately estimate the duration of a stop for a parked vehicle, it has been shown that it is strongly correlated to the parking location and time of day. An even more accurate estimate can be made if historical data stored on a specific vehicle can be queried to locate previous stops in the same area at similar times. However, an analysis of how statistical data can improve the accuracy of this estimate is not within the scope of this paper, and we leave it for future work.

To determine an optimal schedule for an individual service, it is first necessary to determine what fraction of the stop the service can be active and then select a schedule of active and idle times. The total time the system can spend in the active state is a function of e_{av}, P_a, P_i and δ_s. The total energy spent during a stop, E, is given by: $E = \delta_a \cdot P_a + \delta_i \cdot P_i$, where δ_a and δ_i are the times spent in the *active* and *idle* states, and $\delta_s = \delta_a + \delta_i$. The maximum length of δ_a can be calculated using the maximum amount of energy that can be spent, $E = e_{av}$:

$$\delta_a = \frac{e_{av} - \delta_i \cdot P_i}{(P_a - P_i)}. \tag{1}$$

This time accounts for a fraction $f = \frac{\delta_a}{\delta_s}$ of the total estimated duration of the stop.

B. LoadingZones

Our target service is LoadingZones [1], a communication system that uses parked cars as relay agents between moving vehicles and APs as illustrated in Figure 1. LoadingZones divides what would be a single connection between a moving car and an access point into a two hop link: the moving vehicle communicates with a parked car, which relays the packets to and from an indoor AP. Dual radio hardware enables concurrent communications on separate (orthogonal) channels. Additionally, LoadingZones uses its local storage for caching the data that is uploaded by the moving vehicle in case the connectivity with the AP is the bottleneck. When parked vehicles participate in the network, the amount of data that a moving vehicle can transfer is several times higher than that achievable with a direct connection to APs, an approach used by early systems like Cartel [7].

The utility of running LoadingZones is determined by the number of passing cars that need connectivity and the amount of data they need to transfer. One way to approximate this is to use a utility function that is proportional to the density of vehicles (Figure 2 shows as an example of hourly average density [16]). For example, if rush hour is known to be between 5 and 6 PM, a car parked between 12pm and 8pm could save energy from 12pm - 5pm to make sure that LoadingZones can be provided during the most demanding time. LoadingZones is just one of many examples where the traffic density can be used as a measure of the potential benefit of the service. A similar service is PVA [2] that uses parked cars as relays between moving vehicles to enhance the delivery of messages in a DTN fashion. While PVA has a different goal than LoadingZones, car density is a good metric for its utility as well.

If the amount of available energy cannot sustain running LoadingZones for the entire stop, the goal is to select active intervals so that the sum of the active times is equal to δ_a (i.e., the system is active for as long as possible), and the service is provided when its utility is maximum (i.e., when more moving vehicles are present to use it). Given an estimate of the traffic density, the system should identify a threshold Th such that when the density is above this value, it is activated and will be active for a total of δ_a during the stop, thus optimizing the utility of the service and using all of the available energy.

IV. An Architecture for Governing Services

Our governor is based on the principle that a service is only as good as the benefits it provides to the users and the system itself. The basic idea is to control the energy-saving modes of the device (i.e., when it is active or sleeping). With information about a service's costs and utility functions, a schedule is determined for when to activate the system. To achieve this goal, we need a system model that describes the operating

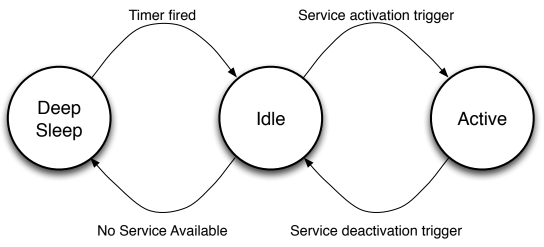

Fig. 3. State machine model and transition triggers.

modes of the device and a system architecture that is capable of measuring the available energy, the costs of a service and monitoring the benefits that a service provides. Together, these components can be used to determine an optimal schedule for a service, to govern activation and deactivation during a stop.

In this section, we first describe our system model and then provide an overview of the architecture.

A. System Model

To save energy, a system can be designed with a number of states, each with different capabilities and energy consumption. The base system model includes three states, Active, Idle and Deep Sleep (see Figure 3). However, the design is not limited to these states and can be expanded as needed. While these states are certainly differentiated by their energy consumption, it is also important to understand what functionality is available in each state.

In **Active** mode, all hardware components are active and the target service is enabled and available. In this state, the system power is P_a. For the sake of simplicity, we only consider one active state and one power level. In reality, different sets of hardware could be activated based on what service must be provided, thus creating multiple active states with different power characteristics.

For **Idle** mode, only low-power hardware is active, which reduces the energy of the system to $P_i << P_a$ but limits its functionality. In this mode, the system can monitor a number of environmental characteristics or *primitives* that are only dependent on the low-power hardware. Essentially, primitives can be based on simple system parameters such as time, enabling the system to track elapsed time since the beginning of the stop or time left until departure, or battery levels, enabling the system to determine available energy for a given stop. Additionally, other simple metrics can be added using low-power hardware, such as acoustic sensors that expose environmental noise levels. In our prototype, we add a low-power ZigBee radio to monitor traffic density. It is important to note that metrics that require powerful hardware, such as a value stored on a server accessible through WiFi, cannot be used as primitives.

The final state is **Deep Sleep** mode, which is the most energy saving state that runs at $P_{ds} \approx 0$. In this state, no functionality is provided. The system can only be woken up by a timer.

The governor integrates knowledge of the system states to ultimately determine a schedule that alternates between the different states. Since state transitions are not instantaneous, it is important to include state transition times [17].

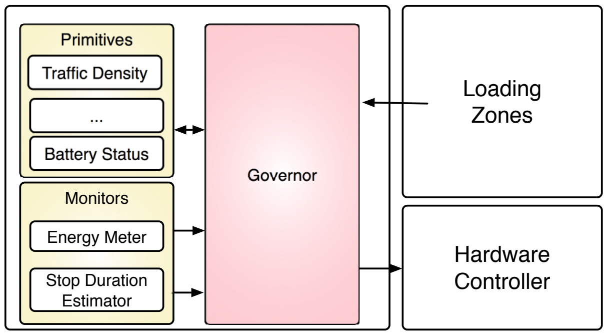

Fig. 4. System Architecture

B. System Architecture

The governor sits between the system controller, which controls the system state transitions, and the service, Loading-Zones. Monitors track the level of the battery and estimate the duration of a stop. The main primitives in our current system are traffic density based on monitoring the number of passing cars and timing information about the time of the stop.

One of the most important components is the *stop duration estimator*, which uses local history, GPS position, and statistics collected by a central server, to predict an estimate for the duration of a new stop, δ_s. The habits of a driver are known to be a determining factor in the duration of a stop, especially when visiting regular locations. In addition, many studies in urban planning research have shown a high correlation between location, time of the day and duration of a stop even across different vehicles. While fine grained statistics are hard to obtain, although some promising effort is being pursued, the complexity of estimation and the verification of its accuracy are out of the scope of this paper. Instead, we consider the output of the stop duration estimator to be a random Gaussian variable $\mathcal{N}(\mu_s, \sigma_s)$. The *stop duration estimator* uses a conservative approach by computing the estimated length of a stop as:

$$\delta_s = \mu_s + \sigma_s. \tag{2}$$

which guarantees that, for 80% of stops, parking duration is not underestimated.

For each primitive, the estimators compute a cumulative distribution function (CDF) over the estimated duration of the stop. For some metrics, such as time passed since the beginning of the stop or until the departure, this metric is easily computed and solely based on the duration of the stop. Other primitives, such as traffic density, have more complex behavior and their distribution is a function of location, time of day and duration of the stop. For traffic distribution, when the vehicle is stopped, it is possible to connect to a central server that stores traffic logs, such as those described in Figure 2. This information is enough to compute an estimated CDF. The level of detail of this information can be enriched over time, especially for the most popular locations, by updating the data on the server as well as using a cache of the vehicle history on the system itself.

Finally, the governor collects the stop duration estimation, the value of the primitives and the properties of the service that needs to be be scheduled. The full system architecture is illustrated in Figure 4.

C. Governor

For many services, the utility of the service, $\mathcal{U}_s$ depends on the value of a specific metric that can be captured as a primitive and so monitored in the idle state. For example, for LoadingZones, this metric is traffic density. To use the governor, a service need only register which primitive to use and provide a utility function for that primitive. The service can also specify a minimum continuous active time, int_{min}. If no specific utility function is provided, a linear utility function is used.

As described in Section III, an optimal schedule is achieved when the sum of all active times is equal to δ_a, as defined in (1), and guarantee that if the device is active at time t_0, it is also active at any time t_1 such that:

$$\mathcal{U}_s(t_1) > \mathcal{U}_s(t_0). \tag{3}$$

This is equivalent of defining the threshold Th for the utility and activating the service every time $\mathcal{U}_s(t) > Th$. The identification of the right threshold that satisfies the properties of an optimal schedule is easier if we leverage the characteristics of the CDF of the metric. The CDF is a monotonic function and is much less susceptible to the noise that perturbs metrics such as traffic density. After determining what fraction f of the stop the service can be active, using (1), the threshold is:

$$Th = CDF^{-1}(f), \tag{4}$$

where CDF^{-1} is the inverse of the CDF, or quantile function. Algorithm 1 describes the steps followed by the governor.

Algorithm 1: Governor for service s

```
$E$ : available energy();
$\delta_s$ : stopEstimator→get Stop Duration();
$int_{min}$ : $s$ →get minimum active time();
$p$: $s$ →get primitive();
$CDF$: $p$ →get CDF();
/* compute f using (1)                    */
$f$: get active frac($e_{av}$,$\delta_s$);
/* compute threshold using (4)            */
$Th$: get threshold($CDF$,$f$);
while parked do
    $u$: $p$ → get reading(t);
    if $u > Th$ then
        SetActive();
        wait($int_{min}$);
    else
        SetIdle();
    end
    /* repeat test after $int_{min}$             */
    wait(interval);
end
```

To better understand the use of the governor, consider a system running LoadingZones where a car is parked for 9 hours starting at 7am (see Figure 5 and Figure 6). For this example, the battery can provide up to $3.5Ah$, which, with $P_a = 6W$ and $P_m = 0.5W$, can sustain up to 2 hours active (a fraction $f = 0.295$ of δ_s) plus 7 hours idle. The evaluation uses the average density shown in Figure 2, shifting it by a random value in the interval [+30, -30] minutes, and adding noise with values within 10% of the expected average value. The governer uses the CDF based on the average values described in Figure 2 (blue line in Figure 5) to compute a threshold for $f = 0.295$. Despite the random duration of the stop and the offset and noise in the density, which cause the actual CDF (red line in the figure) to be different from the estimate, at the end of the stop, the system uses $\approx 100\%$ of the total available energy.

V. Evaluation

To evaluate the performance of our governed version of LoadingZones, we compare it to two non-adaptive strategies. The first and most simple one, FrontLoad, activates the system at the beginning of the parking period and runs until the battery is completely discharged. The DutyCycle approach alternates active and sleep periods at regular intervals to guarantee an even distribution of the active time over the estimated duration of the stop. We evaluated different period lengths for DutyCycle, but only show the results for a period of 20 minutes. For space constraints we do not display the results for other settings of these parameters. However, the adaptivity of our governor consistently achieved near-optimal schedules for every configuration we tested, while the performance of DutyCycle was extremely dependent on the choice of the two parameters.

We implemented the system model, the framework and the two other strategies in a MATLAB simulator. The flexibility of the simulated environment allowed us to generate a large number of scenarios and evaluate performance in ideal conditions. Furthermore, using our simulator, we were able to control the level of precision of our stop duration estimation, as well as the noise that affects the traffic density with respect to its average value provided by online databases.

A. System Performance

To measure the performance of each schedule, we compute the similarity with the optimal schedule, computed *a posteriori*, as the MSE of the difference between two binary vectors, one of which represents the status (active or not) of the service over time of the optimal schedule while the other represents the schedule produced by the evaluated strategy. The schedule computed by FrontLoad and DutyCycle are fixed and do not depend on the service properties. This has a negative impact on the effectiveness of their schedules even in ideal conditions (see Figure 7 left-most grouping, where a value equal to 1

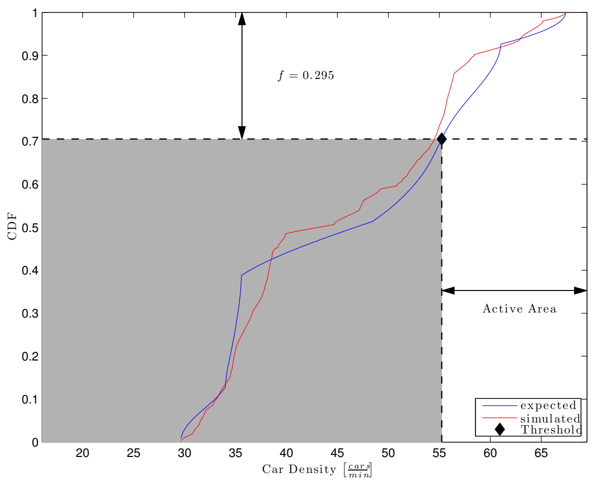

Fig. 5. CDF of the traffic density used for the example in Figure 2, with a Threshold value computed for $f \approx 0.3$.

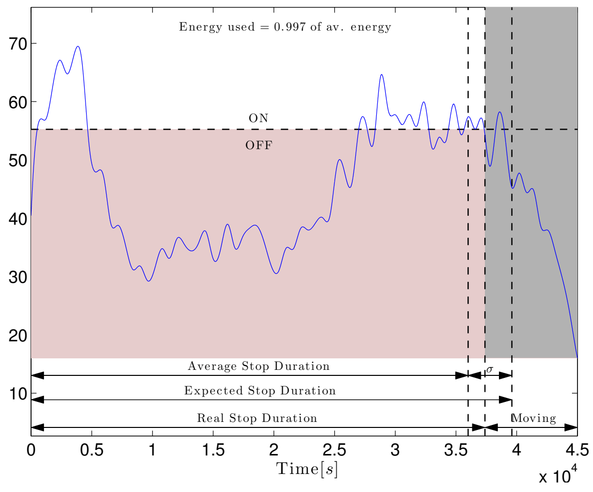

Fig. 6. Density evolution over time for one simulated stop of $\approx$ 8 hours, with a random offset and noise with respect to the average density.

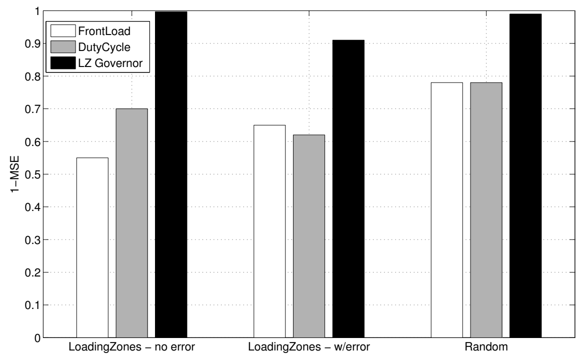

Fig. 7. Proximity of the schedule produced by each strategy to the optimal schedule for LoadingZones with perfect knowledge of traffic density and stop duration, estimated knowledge of traffic density and stop duration and for a random traffic density (i.e., not time dependent)

identifies a perfect match between a schedule and the optimal solution). We individually scheduled LoadingZones simulating many stops of different lengths, keeping the battery level constant. To limit the dependency of the specific traffic distribution that we used for the performance of LoadingZones, we also simulated a service based on a metric whose values are a random variable based on the same CDF (see Figure 7 rightmost grouping).

In all cases, the schedule produced by the adaptive approach is almost perfectly overlapping with the optimal schedule. FrontLoad and DutyCycle perform equally poorly in both configurations. For LoadingZones, not only the duration of the stop is important, but also the time of the day, which influences the traffic density values during the stop. The value shown in Figure 7 for LoadingZones are an average over multiple simulations of the same duration, starting at different times of the day. We computed the standard deviation of the results for each run, which show an almost null value for our adaptive solution, while the proximity between the DutyCycle and FrontLoad schedule and the optimal one show a very large dependency on the time at which the schedule starts.

B. *Effect of Approximation*

The duration of the stop, δ_s and in some case the values of the metrics, such as traffic density, are approximations based on statistical analysis of historical data. To investigate the negative impact of approximations on the performance of the various scheduling strategies, we ran our simulations generating δ_s as a random Gaussian variable $(N)(\mu_s, \sigma_s)$.

As described in our system architecture, in an effort to guarantee service availability throughout the whole stop, we used a conservative estimate for duration, $\delta_{s,est} = \mu_s + \sigma_s$. We used the same heuristic for DutyCycle, while FrontLoad keeps using all energy at the beginning of the stop. The average results for several simulations with random values of δ_s, shown in Figure 7 middle grouping for $\sigma = 0.1 \cdot \delta_s$, confirm the benefits of the adaptive approach used by our governor, which consistently achieves a schedule that is closest to optimal with respect to the non-adaptive approaches. FrontLoad has some advantage with respect to the non-random simulations because it always uses all energy, while the conservative approach and DutyCycle cause an under-utilization of the available energy, when the stop is shorter than the estimated duration. This is still preferable to using all energy before the stop ends, since this latter approach would cause an unfair distribution of the resources. We evaluated the effect of errors between the primitives and their expected distribution (the CDF used for scheduling), and we observed a similar behavior, with our governor outperforming the other approaches and finding a schedule that overlaps with the optimal one for the largest part.

VI. PROTOTYPE

While simulations can provide extensive exploration of the system space, it is important to understand how a system performs in a real environment. Therefore, we implemented a prototype of the governor and LoadingZones and tested it

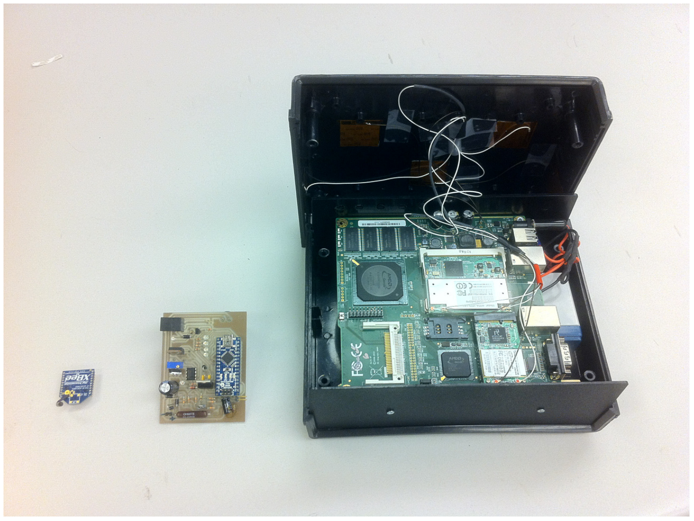

Fig. 8. From right to left, the main prototype board, our custom-made micro-controller board, and the low-power ZigBee radio for traffic density monitoring during the *idle* state.

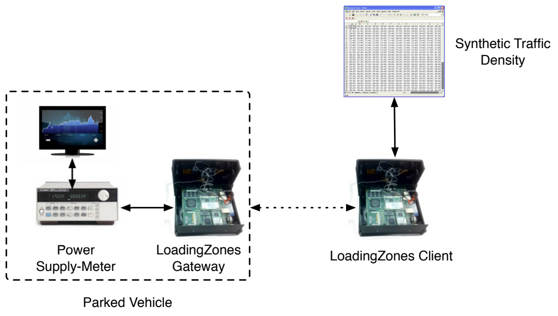

Fig. 9. Experimental system configuration

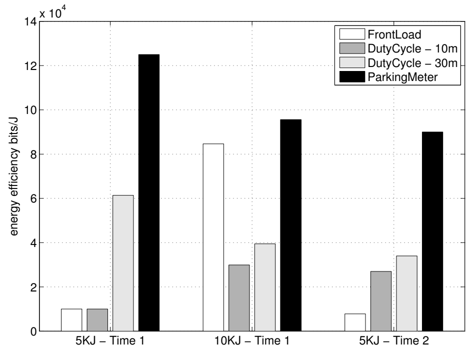

Fig. 10. Energy efficiency of our governor compared to FrontLoad and DutyCycle schedulers in our experiments. Each grouping refers to different settings.

as part of the Illinois Vehicular Project with LoadingZones as a service.

An evaluation of the efficiency in a real deployment is challenging due to the long time that each experiment would require and the large number of devices that would be required to test the governor for services like LoadingZones. Additionally, it would be very hard to monitor the power consumption with a fine granularity in an actual deployment. For these reasons, we ran the experiments in our lab, using a remotely controlled power supply as energy source and meter, as described in Figure 9. We tested effectiveness by connecting one IVP device to an Agilent 66319D power supply, remotely controlled by a laptop that measured the consumed energy during the experiment. This device is configured as a parked car and is running the LoadingZones service, offering a Internet gateway for other vehicles.

A second IVP system was used to emulate passing vehicles that connect to the parked car using the LoadingZones client. To emulate the changing traffic density, we use a trace similar to that described in Figure 2. The client, based on the density pattern described in this trace, maintained a number of constant rate data streams to the gateway equal to the expected number of vehicles. In our experiments, we emulated stops at different times of the day. We tested our system with a low data rate ($\approx 40Kbps$ per simulated car) and a high rate ($\approx 500Kbps$ per simulated car).

The hardware platform is designed around low power Geode processors with an emphasis on efficiency. In lieu of traditional hard disk drives, our systems utilize flash memory for local storage and require limited read/write accesses to reduce slow downs. On each unit, the 802.11n radio is dedicated to Internet connection by attaching to existing access points, while the 802.11g is used for car-to-car communication. Power is provided by the car generator when the engine is on, and by a dedicated lead-acid battery when the car is parked. This enables our experiments without the risk of impairing the ability of the main vehicle battery to turn on the engine. In the active state, when the device is fully powered, it consumes 6W. To enable the energy saving states *idle* and *deep sleep*, we designed and built a board based on the low power Atmel atmega328 micro-controller. This board, equipped with a low-power Zigbee radio to monitor traffic density using broadcast presence packets, enables low power access to the timers and primitives ($\approx 0.4W$).

The main system board is shown on the right side of Figure 9, in the enclosure that also contains the battery and the antennas. The micro-controller board that we designed is shown in the center of the figure, while on the left there is the small Zigbee radio used to measure traffic density when the state is *idle*. If access to primitives is not required (e.g., when duty-cycle is enforced), our micro-controller can switch to the *deep sleep* mode where the consumption is only $0.08W$. Our custom-made micro-controller board is also designed to constantly monitor the battery level and switch the system to the *deep sleep* state until the battery is recharged if the maximum discharge level is reached. This feature prevents damaging the battery.

To be able to repeat the experiments several times and use different schedulers, we limit the energy available on the battery and run each experiment for one hour. We used two different values for the available energy: $10KJ$ and $5KJ$ capable of powering the system for $\approx$ 30 or 15. We implemented and tested three approaches: FrontLoad, which uses all the available energy at the beginning of the stop, DutyCycle, which alternates active and sleeping times according to a regular schedule, and a schedule determined by our governor. For DutyCycle, we studied the impact of a long period (30m) and a shorter one (10m). The average number of successfully received bits per joule for each configuration is displayed in Figure 10, for two stop times, with different traffic characteristics. The results highlight how the density-aware schedule is making better use of the available energy, since it always keeps the service active when the density is higher and it can serve more clients. More importantly, the performance of FrontLoad and the DutyCycle are heavily affected by the traffic distribution as shown by the large differences between the results for the two parking times. On the contrary, our governor is aware of the expected density and then adjusts the active times accordingly, constantly achieving the largest number of bits per joule. Since our governor always activates LoadingZones when the density is higher, its energy efficiency is actually higher when the available energy is less, since the active times are concentrated in this case during the density peaks.

In real hardware, switching among states never comes at a free cost: energy and a finite time must be spent in doing so. This is a well known issue in wireless devices and even more so in a complex system that can provide complex services like the ones described here. A schedule that requires too frequent switches, although theoretically optimal, might end up with a dramatic performance degrade. For this reason, the performance of the DutyCycle scheduler with a shorter period, which requires more switches among the idle and active states, has a worse energy efficiency.

VII. Conclusion

The successful deployment of a distributed networked service is a promising opportunity that vehicular networks, and in particular parked vehicles, can contribute to. However, a number of challenges must be addressed to make this vision a reality. One of the most important is to manage the challenging task of maximizing service availability even when limited energy and long stop durations make it necessary to enforce a sleep-active schedule.

In this paper, we presented our schedule-based system that is capable of providing near-optimal scheduling. We demonstrate that the adaptive strategy achieves near-optimal solutions in an ideal scenario with perfect estimates of the stop duration. We also verified that the system is able to react to random errors with less performance loss with respect to its non-adaptive competitors. Finally, we provide the results of our experiments on a custom-designed hardware which confirms the improved performance.

As future work, we are working on expanding our system to manage multiple simultaneous services. In a powerful device like a car, many services can be provided simultaneously, which introduces an interesting problem of resource sharing. Essentially, it is important to balance the use of the available energy fairly across the multiple services. However, given the dynamics of the systems and the environment it is deployed in, this fair share must be based on the utility of offering the service at a given time.

Acknowledgment

This work was supported in part by the National Research Foundation (NSF CNS 1059628 and CRI 1059294). The authors would like to thank the reviewers for their valuable comments.

References

[1] R. Crepaldi, R. Beavers, B. Ehrat, M. Jaeger, S. Biersteker, and R. Kravets, “LoadingZones: Leveraging Street Parking to Enable Vehicular Internet Access,” in *Proceedings of the seventh ACM international workshop on Challenged networks - CHANTS ’12*, 2012.

[2] N. Liu, M. Liu, W. Lou, G. Chen, and J. Cao, “PVA in VANETs: Stopped cars are not silent,” in *2011 Proceedings IEEE INFOCOM*, 2011.

[3] K. Ibrahim and M. C. Weigle, “CASCADE: Cluster-Based Accurate Syntactic Compression of Aggregated Data in VANETs,” in *Proc. of IEEE Globecom Workshops*, 2008.

[4] I. Leontiadis, P. Costa, and C. Mascolo, “Extending Access Point Connectivity through Opportunistic Routing in Vehicular Networks,” in *Proceedings of IEEE INFOCOM*, 2010.

[5] R. Crepaldi, M. Bakht, and R. Kravets, “QuickSilver: Application-driven Inter- and Intra-cluster Communication in VANETs,” in *Proceedings of the third ACM international workshop on Mobile Opportunistic Networks - MobiOpp ’12*, New York, New York, USA, 2012.

[6] N. Thompson, R. Crepaldi, and R. Kravets, “Locus: a location-based data overlay for disruption-tolerant networks,” in *Proc. of CHANTS*, 2010.

[7] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden, “CarTel: a distributed mobile sensor computing system,” *Proc of ACM SenSys*, 2006.

[8] J. Polastre, J. Hill, and D. Culler, “Versatile low power media access for wireless sensor networks,” in *Proceedings of the 2nd international conference on Embedded networked sensor systems - SenSys ’04*, 2004.

[9] M. Buettner, G. V. Yee, E. Anderson, and R. Han, “X-MAC: A Short Preamble MAC Protocol for Duty-Cycled Wireless Sensor Networks,” in *Proceedings of the 4th international conference on Embedded networked sensor systems - SenSys ’06*, 2006.

[10] F. Ashraf, R. Crepaldi, and R. H. Kravets, “Know your neighborhood: A strategy for energy-efficient communication,” in *The 7th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (IEEE MASS 2010)*, 2010.

[11] C. Sengul and R. Kravets, “Heuristic approaches to energy-efficient network design problem,” in *The 27th International Conference on Distributed Computing Systems (ICDCS)*, 2007.

[12] S. Das, “Avoiding Energy Holes in Wireless Sensor Networks with Nonuniform Node Distribution,” *IEEE Transactions on Parallel and Distributed Systems*, 2008.

[13] R. A. Williams and J. R. Dwark, “Apparatus and method in a network interface for enabling power up of a host computer using magic packet and on-now power up management schemes,” 1999.

[14] E. Shih, P. Bahl, and M. J. Sinclair, “Wake on wireless: An Event Driven Energy Saving Strategy for Battery Operated Devices,” in *Proceedings of the 8th annual international conference on Mobile computing and networking - MobiCom ’02*.

[15] L. Gu and J. Stankovic, “Radio-Triggered Wake-Up for Wireless Sensor Networks,” in *Real-Time Systems*, 2005.

[16] (2012) Massachusetts Department of Transportation. [Online]. Available: http://mhd.ms2soft.com/tcds/

[17] C. Sengul, A. Harris, and R. Kravets, “Reconsidering power management,” in *Fourth International Conference on Broadband Communications, Networks and Systems - BROADNETS 2007.*, 2007.