IT Infrastructure for Providing Energy-as-a-Service to Electric Vehicles

Smruti R. Sarangi[1], Partha Dutta[2], Member IEEE, Komal Jalan[3]
[1]Department of Computer Science, IIT Delhi, India
[2]IBM Research, Bangalore, India
[3]Google, Bangalore, India

***Abstract*—Plugin Hybrid Electric Vehicles (PHEVs) are set to be the defining trend in road transportation for the 21st century. Unfortunately, the process of charging a large number of PHEVs will put a prohibitive amount of stress on conventional power grids. As per current estimates, it will not be possible to satisfy the peak demand of charging millions of PHEVs with current infrastructure. Additionally, to ensure stability of the power grid, the demand needs to be closely matched with supply all the time. This is difficult to ensure in a dynamic scenario like PHEV charging, where it is not possible to predict the load in the future. This problem is further complicated by the introduction of intermittent sources of alternate energy like wind power.**

This paper proposes an IT infrastructure for managing energy usage of PHEVs. The paper introduces a generalized currency of energy called a *token*, which entitles its owner to produce or consume a certain amount of electrical energy. Furthermore, the paper propose an IT infrastructure to manage tokens produced by millions of PHEVs and power stations. Simulation results for the system that consider the behavior of millions of PHEVs using data in prior work and traces from the Australian power grid for the last five years are also presented.

***Index Terms*—PHEV scheduling, IT infrastructure, power distribution**

ACRONYMS

EV	Electric Vehicle
PHEV	Plug-in Hybrid Electric Vehicle
TMS	Token Management System
LM	Local Module
ACS	Admission Control and Scheduling
FIFO	First-In-First-Out
PACS	PHEV Admission Control and Scheduling
ILP	Integer Linear Program
genToken	Generation Token
consToken	Consumption Token

I. INTRODUCTION

A. Motivation

DUE to the dwindling supply of fossil fuels, hazardous levels of urban air pollution, and other economic reasons, the usage of electric vehicles (EVs) is expected to increase substantially over the next few years [1], [2], [3], [4]. The next generation of EVs, Plug-in Hybrid Electric Vehicles (PHEVs), will contain a conventional combustion engine as well as a motor driven by a rechargeable electric battery. PHEVs can consume power from the power grid as well as provide power back to the grid. As per the estimates in a report by Oak Ridge National Laboratories [5], almost 30% of the vehicles in the European Union will be PHEVs by 2020. Another study [6] predicts that there will be 1.7 million PHEVs on the road by 2015. United States alone will have a million charging points [6] with over a half of them being public charging stations.

A major trend in energy production is expected to be the large scale adoption of renewable energy like solar, wind and tidal energy. Unlike conventional power plants, sources of energy such as wind power are intermittent in nature. Since the power grid cannot store a significant amount of energy, PHEVs can act as energy storage devices in this case.

Unfortunately, at present the power generation and distribution infrastructure is not equipped to handle such intermittent sources of energy and millions of PHEVs as consumers [6], [7], [2], [3]. Furthermore, due to the heavy capital and operational expenditure, the electricity production and distribution grids are expected to change slowly as compared to the rapid increase in the number of PHEVs. Finally, to maintain the stability of the power grid, the energy supply needs to closely match the energy demand at all instants of time [7], which is difficult to ensure in this situation because of its dynamic nature.

Recently, [6], [8], [9], [2] have observed that a solution to the above problem that consists merely of physical systems may turn out to be extremely expensive. Therefore, an IT infrastructure is needed for the power generation and distribution system, which can efficiently manage the millions of consumers and intermittent producers, and can provide an acceptable level of service.

B. Contributions

In this context, this paper presents an IT system to manage the charging and discharging of a large number of PHEVs. The system proposed in this paper is based on the concept of a *token*. A token is an electronic record that entitles its owner to consume or produce a certain amount of electrical energy. The attributes of a token abstract the energy related

parameters for PHEVs' charging and discharging operations. Using operations on the tokens, this paper shows that it is possible to seamlessly negotiate the exchange of energy between producers and consumers through the power grid.

The token-based system architecture that is presented in this paper can be divided into two parts: communication and scheduling. The communication part consists of a distributed communication protocol for creating and managing tokens. The scheduling part of the system performs both admission control and scheduling of tokens by matching the tokens from consumers with token from producers. Through extensive numerical simulations using real power generation traces, this paper demonstrates that the proposed system is scalable for large number of PHEVs.

C. Organization of the Paper

The rest of the paper is organized as follows. An overview of the proposed IT system in given in Section II-A. A system to create and manage tokens is presented in Section II-B and Section II-C. Section III present the scheduling aspects of the system. The simulation results are presented in Section IV. Section V summarizes the related work in this area, and Section VI presents some concluding remarks.

II. Token-based System for PHEVs

A. Overview

Our token-based IT system is designed for a PHEV charging and discharging network overlaid on an electricity distribution grid. The network is composed of electricity generators, consumers, and intermediaries. The electricity generators can be the power stations (owned by the electricity companies), or the PHEVs. PHEVs may generate electricity when they discharge their stored electrical energy to the grid [10], [11]. The electricity consumers are the PHEVs when they are charging from the grid. The intermediaries are charging stations where the PHEVs can plug-in to the electricity grid to charge or discharge.

As shown in Figure 1, our token-based system is composed of a central Token Management System (TMS) and multiple Local Modules (LM). The TMS maintains the tokens and controls the usage of the tokens. The LMs at the electricity generators, consumers and intermediaries communicate with the TMS to create and modify the token. We now describe various components of the PHEV network and our IT system.

Nodes. There are three *types* of nodes in the PHEV network: (1) PHEVs, (2) power stations, and (3) PHEV charging/discharging stations. Each node in the PHEV network has two mandatory attributes: a node identifier and a node type. A node identifier uniquely identifies a node. A node can perform one of the three roles: electricity generator, electricity consumer and intermediary. Each node may have one or more optional attributes, such as its current role, current location, or token usage history.

Tokens. A token is an electronic record that entitles a node to generate or consume a certain amount of electrical power for a certain amount of time. Each token has the following mandatory attributes: (1) token identifier, (2) token type, (3) generator or consumer node id, (4) the electrical power level, (5) the duration of power supply, (6) start time, (7) expiration time, and (8) current status. A token identifier uniquely identifies a token; i.e., no two tokens have the same identifier. Generation and consumption are two types of tokens. The power denotes the amount of electrical power that the generator or consumer node of the token is entitled to generate or consume, and the duration states the duration for which the power can be generated or consumed. The start time and expiration time provide the start and end times of the period within which the token can be used for production or consumption, and we call this period the *validity period* of the token. (For example, if the start time, end time, and duration of a token is $t1$, $t2$, and δ, respectively, then the token can be used for at most δ time between the time instants $t1$ and $t2$. The interval from $t1$ to $t2$ (also denoted by $[t1, t2]$) is called the validity period of the token. Obviously, the duration should be less than or equal to the length of the validity period.) Finally, the status of the token states whether the request for this token is admitted or rejected. Like the nodes in the PHEV network, a token can have one or more optional attributes, such as the set of charging stations where the token is valid, or the usage history of the token. The *total energy* of a token is defined as the product of its power level and its duration. We call the tokens of type generation and consumption as *genTokens* and *consTokens*, respectively.

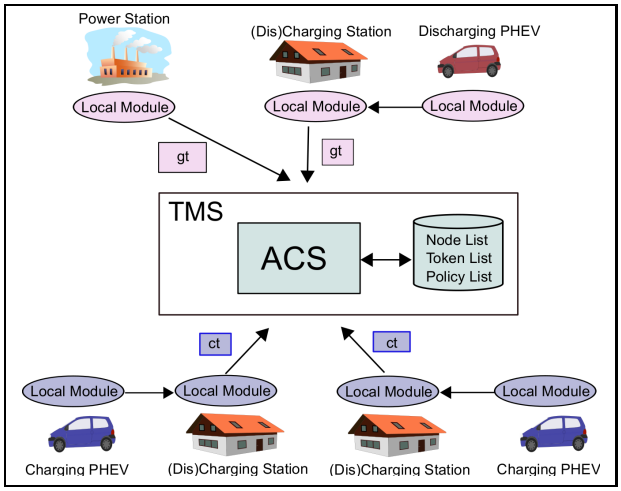

Fig. 1. The PHEV Network (TMS: Token Management System, ACS: Admission Control and Scheduling, gt: generation token, ct: consumption token)

Token Management System (TMS). The Token Management System maintains the information about the nodes, tokens and the token usage policies. It also does the admission control of the tokens (i.e., whether the token is admitted to be scheduled or rejected), and schedules the admitted tokens. It maintains three lists: (1) a list of nodes in the PHEV network along with their attributes, (2) a list of tokens along with their attributes, and (3) a list of policies that control the usage of the tokens.

Optionally, it may maintain some overall system statistics, e.g., the number of charging tokens in a given geographical region.

Local Modules (LM). A local module is an application running at a node through which the node participates in the PHEV network. In particular, through the local module, a node creates and modifies a token, and sends requests to the TMS for using the token. In the next subsections, we describe the main functions of the TMS.

B. Admission Control and Scheduling of Tokens (ACS)

In a PHEV network, the electricity generators and consumers may send requests for using tokens depending on their production and consumption requirements, respectively. However, at any given instant of time in the PHEV network, the total amount of power consumed should be less than the total amount of power allocated for charging PHEVs. (We call this the *power constraint*.) On the other hand, if the power consumed is significantly less than the power generated then the excess power may potentially be wasted. (This power constraint aims to decouple the PHEV usage from the rest of the electricity grid, which may be useful if the power utilities want the PHEV usage to not adversely impact other consumers. However, it is possible to extend the power constraint to the whole power grid, which we plan to study as part of future work.) At any given time, we define the *(power) utilization* of our system as the ratio of power consumed to power allocated for charging. We would like the utilization to be close to 1, but it should never be higher than 1. Thus, the accounting system needs to perform admission control on the token usage request, as well as schedule the admitted tokens based on the usage policies. In Section III, we present a detailed description of the ACS.

C. Token usage policies

Even if a token can be scheduled while satisfying the power constraint, the accounting system admits the token only if it satisfies the *usage policies*. A usage policy is a function of the token attributes, the local module attributes, the current state of the token system (such as the current load), and the usage histories. A policy evaluates to either true or false, and it can be applicable to single or multiple tokens. We now describe various usage policies that can be considered in our system.

- Temporal policy: A token should be used within its start time and expiration time. For example, using the validity period of a token, a PHEV user can indicate how long she is willing to wait for charging her PHEV. In our evaluation, we extensively study the effect of this temporal policy on admission control and scheduling.
- Spatial policy: A consumption token may have a set of valid station ids (i.e., set of the charging station ids where the token can be used) as an optional attribute. A spatial (or geographical) policy requires that a token is used at one of its valid stations. For example, this policy can be used to load-balance users across geographical regions.
- Spatio-Temporal policies: A spatio-temporal policy is a combination of spatial and temporal policies that involve restrictions due to both the valid charging stations and validity periods of tokens. Another spatio-temporal policy can restrict the number of PHEVs that are simultaneously charging in a region. This restriction can be useful when there is a limit on the amount of power that can be supported by the local power distribution grid (e.g., the power capacity of an electrical transformer).
- Usage history based policies: A policy can be based on the usage history of the node that is trying to use a token. There may be a restriction on the amount of energy used by a PHEV in a given period of time. Similarly, producing electricity for the grid (by an electricity company or by a discharging PHEV) can be denied based on a maximum generation limit for a given time period.

We would like to note that our token system is designed in a way that makes it easy to introduce and enforce new token policies.

III. The Admission Control and Scheduling System

A. Problem Formulation and Hardness

Based on the above discussion, this section presents the primary PHEV Admission Control and Scheduling (PACS) problem as an Integer Linear Program (ILP). The objective of the PACS problem is to maximize the average system utilization over all available genTokens while satisfying the validity periods of each scheduled consToken, and never exceeding the power level of a genToken. Our ILP formulation follows that of [12], [13] for a scheduling problem with time-varying resources. We first introduce some notation and assumptions for the ILP formulation.

We assume that time is divided into discrete intervals, and each interval is denoted by an integer $t \in [1, D]$, where D is the maximum activation time over all genTokens. Let GS be the set of available genTokens, and let CS be the set of available consTokens. For each generation token $gt \in GS$, let I_{gt} denote the time interval in which the token is active (which is assumed to be of the same length as its validity period), and P_{gt} denotes the power level of the token. To capture the property that a consToken can be possibly scheduled anywhere in its validity period, we model each consToken $ct \in CS$ as a set of (consToken) instances. For each instance $ci \in ct \in CS$, let I_{ci} denote the time interval in which the instance is active, and P_{ci} denotes the power level of the token. (In other words, through token instances, we capture the different periods in which a token can be possibly activated, e.g., if a consToken has a duration of 5 and a validity period of [1,10], then it consists of six instances that have activation intervals $[1, 5], [2, 6], \ldots, [6, 10]$.)

Let $|I|$ denote the size of an interval I, S_{ins} denote the set of all consToken instances, and $x_{ci,gt}$ be a 0-1 decision variable that is 1 if and only if instance ci is packed in genToken gt. We now specify the ILP for PACS.

Objective:

$$Maximize \frac{\sum_{gt \in GS} \sum_{ci \in S_{ins}} x_{ci,gt}.P_{ci}.|I_{ci}|}{\sum_{gt \in GS} P_{gt}.|I_{gt}|} \quad (1)$$

Subject to:

$$\sum_{ci:t\in I_{ci}} x_{ci,gt}.P_{ci} \leq P_{gt}, \forall gt \in GS, \forall t \in I_{gt} \quad (2)$$

$$\sum_{gt\in GS} \sum_{ci\in ct} x_{ci,gt} \leq 1, \forall ct \in CS \quad (3)$$

$$x_{ci,gt} \in \{0,1\}, \forall ci \in S_{ins}, \forall gt \in GS \quad (4)$$

The objective function in Equation 1 maximizes the average utilization over all active genTokens, which is the ratio of the total energy of the selected consTokens to the total energy of the genTokens. Note that, $P_{ci}.|I_{ci}|$ is the total energy a consToken instance ci, and $P_{gt}.|I_{gt}|$ is the total energy of a genToken gt. The ILP has three constraints. The first constraint in Equation 2 requires that, for every point in the activation time of a genToken, the sum of the power levels of the packed consToken instances is less than the power level of the genToken. (In other words, the power level of a genToken is never exceeded.) The second constraint in Equation 3 requires that at most one instance of each consToken is activated. Finally, the constraint in Equation 4 requires that the decision variable is either 0 or 1, i.e., no splitting of a consToken is allowed.

We now show that the following decision version of the PACS problem is NP-complete: Given an average utilization bound, F, between 0 and 1 (both inclusive), does there exist a packing such that the average utilization over all genTokens is at least F? We show the NP-completeness by reduction from the subset-sum problem [14].

Theorem 1: The decision version of the PACS problem is NP-complete.

Proof: Given a packing of consTokens within genTokens, it is easy to verify whether the average utilization of genTokens is at least F. Thus the decision version of PACS is in NP.

In a subset-sum problem instance, given a positive integer F, and set S of P positive integers $\{x_1, \ldots, x_P\}$, we are asked to find a subset of S such that the sum of the elements in the subset is exactly F [14]. For this instance of the subset-sum problem, we construct an instance of PACS as follows.

There is a single genToken and a set CT of P consTokens $\{c_1, \ldots, c_P\}$. The power level of the genToken is F and the power level of each consToken c_i is x_i. The validity period of all tokens are identical, and their durations are also identical. The required bound on average utilization is 1. In other words, there is only one instance per consToken, and we are required to find a set of consTokens whose power levels add up to be exactly equal to the power level of the genToken. Suppose that $S' \subseteq S$ is a solution to the subset-sum problem instance. Thus, sum of all $x_i \in S'$ is exactly equal to F. Consider the set, CS', of consTokens $\{c_i \ : \ x_i \in S'\}$. Since, the power level of each consToken c_i is x_i, the sum of power levels of consTokens in CS' is F, and therefore CS' is a solution to the PACS problem instance. Similarly in the reverse direction, given a solution CS' for PACS problem instance, we construct a solution S' of the subset-sum instance as $\{x_i \ : \ c_i \in CS'\}$. This completes the reduction. ■

Thus, the primary admission control and scheduling problem is computationally hard to solve. Moreover, in a practical setting, a PHEV network may have a large number of consTokens, which will result in a large number of constraints for the ILP formulation. Therefore, the next section presents a system design that uses an effective heuristic for the PACS problem, and evaluates it through simulations. As described below, the system design and the heuristic is based on the ideas of token batching, prioritization and splitting.

B. System Description

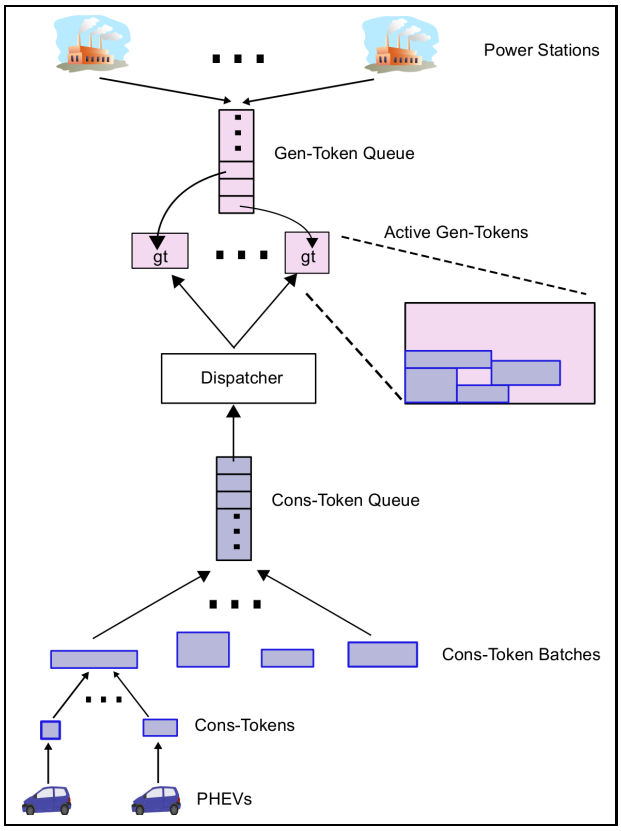

Fig. 2. The token admission control and scheduling system

We now describe the core of Token Management System that does the Admission Control and Scheduling (ACS) of tokens (Figure 2). (Appendix A and Figure 16 present the application-level distributed protocol among the various components of the PHEV network.)

consToken batching. In a real PHEV network, most genTokens, which are typically created by electricity companies on behalf of power stations, are very large compared to a consToken, which are typically created by charging PHEVs. Therefore, we group consTokens in batches, based on their start time and duration. We refer to a batch of consTokens as a $consBatch$. The maximum size of a batch is denoted by MAX_BATCH_SIZE. Creating such batches also reduces the computation load on the dispatcher. (A dispatcher is described below.) Our system can handle changes in the batch size during an execution. However, in

our experiments we use a fixed batch size for simplicity.[1]

Queues. The ACS maintains two incoming queues for genTokens and consBatchs, called genTokenPriorityQueue and consTokenFIFOQueue, respectively. At a high-level, the ACS selects some incoming genTokens and some incoming consBatch, and a sub-system of ACS, called *dispatcher*, packs the consTokens in genTokens. If a consToken is packed in a genToken, it implies that the genToken provides the required electrical power to the consToken.

genToken prioritization. To perform the online scheduling of tokens, the ACS first dequeues *a set* of genTokens from the genToken queue. The selection of these tokens from the genToken queue is dependent on the genToken prioritization scheme. (Hence, we use a priority token for genTokens.) At a given time, a genToken that is being packed by the dispatcher is said to be *active*. The list of active genTokens is denoted by agt, and the maximum number of active genTokens that is allowed is denoted MAX_GEN_ACTIVE. In our experiments, we evaluate the following genToken prioritization schemes: (1)First-in-First-Out (FIFO): tokens are dequeued according to increasing order of their arrival time at TMS, (2) round-robin (rr): tokens are dequeued in a round-robin manner across each power source (and within each source, tokens are dequeued in FIFO order), (3) earliest or latest: tokens are dequeued in the increasing or decreasing order of their expiration times, respectively, (4) smallest or largest: tokens are dequeued in the increasing or decreasing order of their power levels, respectively and (5) largest-latest: tokens are dequeued in the decreasing order of power level and expiration time, with ties being broken by the expiration time.
Dispatcher. The dispatcher (along with consBatch activation scheme described next) executes a heuristic for the PACS problem (where consTokens are replaced by consBatches). The heuristic, although suboptimal, simplifies the solving of the PACS problem by batching the consTokens, and by considering the consToken batches in FIFO order for packing into genTokens. In particular, to select the next consBatch, the dispatcher dequeues from the consToken queue in the First-In-First-Out (FIFO) order of their batching time. The order in which the genTokens are tried for packing a consBatch is determined by the scheduling schemes of the dispatcher. The scheduling scheme considers the active genTokens in one of the following ways: (1) endTime: in the increasing order of their expiration time, (2) freeEnergy: in the decreasing order of the amount of energy that is left in the token (3) random: selected using a uniform random distribution, and (4) util: in the increasing order of their current utilization, where the current utilization of the token is defined as 1 - (free energy/total energy).

consBatch Activation To pack a given consBatch token, cb, in a genToken, gt, we need to find an activation time t for cb in gt. (In terms of our PACS problem formulation, selecting an activation time for a consBatch is equivalent to selecting an instance for the consBatch.) Selecting an activation time t for cb implies that for each consToken $ct \in cb$, the consumer of ct can use ct to consume power from time t to $t + ct.duration$ (where $ct.duration$ is the duration of ct). Then, in genToken, gt, we say that consBatch cb is active from t to $t+cb.duration$, and consToken ct is active from t to $t + ct.duration$. For packing cb, the dispatcher searches for the lowest activation time of cb in gt such that, the following two conditions are satisfied: (1) At each time instant, the sum of the power levels of all the active consBatches that are packed in a genToken, gt, is less than the power level of gt, and (2) the activation period of cb satisfy the validity period requirements, i.e., for each $ct \in cb$, its activation period is contained in its validity period. If no such activation time is found in genToken, gt, then the next active genToken is selected using one of the scheduling schemes of the dispatcher. If the dispatcher is unable to pack a consBatch in gt, then it increments $gt.num\_rejects$ by 1.

Note that, the first condition of consBatch Activation corresponds to Equation 2 in the PACS problem formulation, and the second condition corresponds to selecting a valid instance of consBatch. Since the heuristic selects only one activation time for a consBatch, and it does not split the consBatches, the heuristics also satisfies Equations 3 and 4.

If the allowed activation time of a consBatch can be any time within the validity period of genToken, then we call it an *unslotted* activation. If there are only some pre-defined time instants (at regular intervals) where a consBatch can be activated, then we call it a *slotted* activation.

If a consBatch can be packed in one active genToken while satisfying all the conditions, then the ACS admits all consTokens in the batch, and sends $CONS\_ACK$ messages to the corresponding consumers along with the activation time. Otherwise, the ACS rejects the consBatch, and it sends $CONS\_NACK$ messages.

genToken Replacement An active genToken is replaced when (1) its utilization is above a certain fixed threshold, which is denoted by MAX_UTIL or (2) if the number of rejected consTokens ($num\_rejects$), is more than a fixed threshold, which is denoted by $NUM\_MAX\_REJECTS$. Such a genToken is replaced by the token with the highest priority in the genToken queue.

C. Splitting of consBatch

In the above system, if the consBatch sizes are comparable to the genToken sizes, it is difficult to achieve an average utilization that is close to 1. This problem occurs because the system does not allow a consBatch to be split across multiple genTokens. This section proposes schemes to split a consBatch into multiple parts, and pack each part in possibly different genTokens. To this end, a consBatch is viewed as a rectangle on *power* and *time* axes. This section consider three schemes for splitting the consBatches. All the three schemes first try to schedule the original consBatch. If a scheme is

[1] A discharging PHEV can also create a genToken, which can be of a similar size as a consToken from a charging PHEV. As in the case of consTokens, we can handle such small genTokens by batching them. However, we omit the details for the sake of brevity.

unable to do so, then it tries to split the set of consBatch, and schedule the smaller consBatches individually. It is possible that some of these batches might get scheduled successfully, and it might not be possible to schedule the rest. For the consBatches that could not be scheduled, the system extracts the individual consTokens, and sends *Reject* messages to the associated PHEVs.

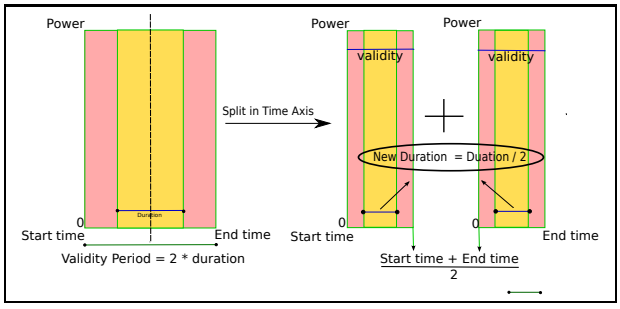

Fig. 3. One dimensional splitting on Time axis

1) One-dimensional split on time axis: In this case, the consBatch is split on the time axis into two smaller batches, i.e. the power required by the split batches remains the same as the original consBatch but the duration and the validity period are halved (see Figure 3). Other attributes such as the startTime and endTime remain the same.

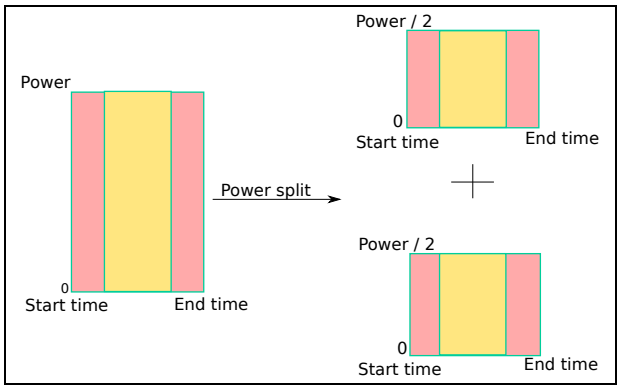

Fig. 4. One dimensional splitting on Power axis

2) One-dimensional split on power axis: In power axis based split, a consBatch split into two smaller batches along the power axis, i.e., the power required by the split batches is half of the power of the original. However, other attributes like the startTime, endTime, and duration remain the same. This is shown in Figure 4.

3) Two-dimensional split: This scheme combines both the one dimensional schemes. A single consBatch is split along both the power and time axes to create four smaller consBatches. The split batches have half the power and duration of the original (see Figure 5).

4) Token splitting in the PACS problem: To show that a splitting scheme cannot adversely impact the genToken utilization, the next theorem shows that the optimal value of a PACS problem instance cannot decrease due to splitting.

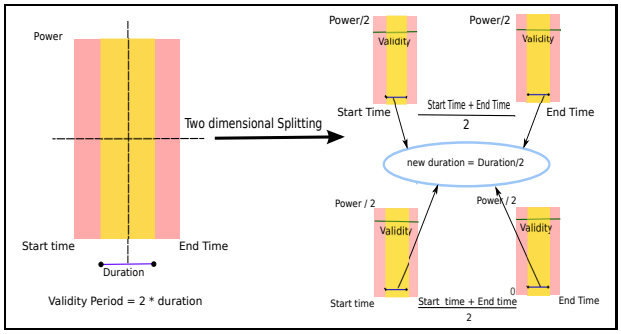

Fig. 5. Two dimensional splitting of a consBatch

First, consider the ILPs for variants of PACS that allow splitting by considering more consBatches and instances. (Recall that, due to consToken batching in the system, consToken is substituted by consBatch in the original PACS formulation.) A PACS problem instance with Time Split (PACS^{TS}) can be obtained by creating a new set CS^{TS} from the set of available consBatches, and another new set S^{TS}_{ins} of their instances, as follows. For every consBatch cb in CS, introduce two batches in CS^{TS}, each with validity period that partitions the validity period of cb into two equal parts, and with duration that is half the duration of cb. Also, following the new validity period and duration, introduce new instances in S^{TS}_{ins}.

Now, the ILP for PACS^{TS} can be obtained by replacing CS and S_{ins} with CS^{TS} and S^{TS}_{ins}, respectively, in Equations 1 to 4. PACS with Power Split (PACS^{PS}) and PACS with 2D Split (PACS^{2DS}) are obtained similarly by introducing two and four batches, respectively, for each original consBatch, and modifying their instances appropriately.

For the theorem below, consider a PACS problem instance pi, and the instances pi^{TS}, pi^{PS}, and pi^{2DS} derived from pi for problems PACS^{TS}, PACS^{PS} and PACS^{2DS}, respectively. Let $opt(x)$ denote the optimal value of the objective (average genToken utilization) for a problem instance x. The following theorem shows that splitting does not degrade the optimal value of the problem.

Theorem 2: (**a**) $opt(pi) \leq opt(pi^{TS}) \leq opt(pi^{2DS})$ and (**b**) $opt(pi) \leq opt(pi^{PS}) \leq opt(pi^{2DS})$.

Proof: Observe that for each consBatch instance in pi, (1) there are two corresponding batches in pi^{TS} with half the validity period and duration of the original, and (2) there are two corresponding batches in pi^{PS} with half the power level of the original. For any feasible solution of pi (including the optimal solution of pi), construct a corresponding feasible solution of pi^{TS} with the same value of the objective as follows: for each consBatch instance ci selected for packing in a genToken, pack the two corresponding batches in pi^{TS} in the same genToken. One can similarly construct a feasible solution for pi^{PS} from any feasible solution for pi with the same value of the objective. Therefore, $opt(pi) \leq opt(pi^{TS})$ and $opt(pi) \leq opt(pi^{PS})$. Next notice that, each batch in pi^{TS} has two corresponding batches with half the power level in pi^{2DS} because each batch in pi^{TS} is further split along the power axis to obtain batches for pi^{2DS}. Thus, one can construct a feasible solution for pi^{2DS} from any feasible solution for pi^{TS}

(including the optimal solution of pi^{TS}) with the same value of the objective by packing the split batches into the same genToken as the original. Similarly, noticing that each batch in pi^{PS} has two corresponding batches with half the validity period and duration in pi^{2DS}, one can construct a feasible solution for pi^{2DS} from any feasible solution for pi^{PS}. Therefore, $opt(pi^{PS}) \leq opt(pi^{2DS})$ and $opt(pi^{TS}) \leq opt(pi^{2DS})$. ■

IV. Evaluation

A. Experimental Setup

The implementation consists of a multi-threaded PHEV simulator in Java 1.6. The token system, and the token accounting system were implemented fully. Instead of simulating individual of PHEVs, the simulation considers consBatches of 100 (MAX_BATCH_SIZE) PHEVs as explained in Section III. The implementation also contained the protocol for requesting consTokens and genTokens including the ACK and $NACK$ messages.

To simulate realistic power demands, the experiments use the power traces for the last five years from the Australian Power Grid [15]. Figure 6 shows the power supply for the month of december in 2009: it presents the data for five Australian states: South Australia (SA), Tasmania (TAS), Queensland (QLD), Victoria (VIC), and New South Wales(NSW). We observe daily cycles and weekly cycles. We assume that the power available for charging PHEVs is roughly 10% of the overall power.

We simulate the Token Management System (TMS) based on data from [10], [11], [16]. A typical PHEV has a battery capacity of 10-15kWh [10], [11], and the typical power usage of a PHEV while charging is 25kW [16]. Hence, a PHEV will typically take 20-30 minutes to charge (which is set as the duration of a typical consToken). A power station should typically produce long genTokens [10]. Currently, power stations are given contracts on a daily basis. However, in the near future, power stations will require about eight hours of continuous power generation to remain viable. They cannot be frequently switched on and off based on demand. Hence, we take the duration of a genToken to be eight hours, and assume that the duration of a consToken is 5% of this value (i.e. 24 mins). It is expected that as we integrate sources of renewable energy that are more intermittent in nature, this duration will further reduce. We choose the *util* scheduling scheme for the dispatcher, the *largest* first prioritization scheme for genTokens and the *two-dimensional splitting* algorithm as the default. Unless specified otherwise, we use 5 active power stations (MAX_GEN_ACTIVE), one for each state, and do not use slots. The default consToken validity time is twice its duration. The values of MAX_UTIL and $NUM\_MAX\_REJECTS$ are set to 1 and 5, respectively (see Figure 16).

B. Throughput

We plot the utilization of genTokens as a function of the number of PHEVs that we desire to service per hour in Figure 7. We observe that for any number of power stations(in the current case, up till 5) the utilization reaches very close to 98%. We further observe that with just 1 power station the saturation level, i.e., the maximum utilization is attained with 500k PHEVs per hour. As we increase the number of power stations, the saturation level increases. It reaches about 3.2 million for 5 power stations.

Figure 8 plots the actual number of PHEVs serviced for different power stations. We observe that beyond a point, we are not able to get any further improvement since we attain the peak utilization. Secondly, for different power stations the curves saturate at different points as expected. We would ideally like to operate at the point at which the curves saturate to maximize the utilization, and ensure that the probability of a consToken finding a free slot is high. Figure 9 shows scenarios in which we consider smaller genTokens. We vary the relative ratio of the duration of the consToken and that of the genToken from 2% to 100%. We observe that for different duration ratios, the graphs saturate at different utilizations. For 2%, the maximum utilization level achievable is approximately 99.5% whereas for 70%, it drops down to around 84%, and further for 100%, it drops down even more to 67%. The realization is that the saturation level decreases as the duration ratio of the consToken This happens because there is a lot of internal fragmentation as the relative size of the consToken is increased. The ideal range is from 2% to 35% where the utilization remains above 90%.

C. Splitting

In this section, we plot the utilization of genTokens by using different algorithms for the splitting consTokens. For consTokens with a very small *consToken duration to genToken duration ratio (5%)*, we do not find a significant difference between the different algorithms (see Figure 10). The two-dimensional split and time-split algorithms perform slightly better as they reach the peak earlier. The utilization varies between 98% to 99%. But for consTokens with higher *consToken duration to genToken duration ratio (30%)*, we can see considerable improvement in utilization in case of *two-dimensional* and *time* split algorithms as shown in Figure 11. At saturation, the two-dimensional split algorithm shows 95% utilization, the time split algorithm shows 94.5% utilization whereas the power split and no split algorithms show around 84-85% utilization.

We can conclude from Figures 10 and 11 that internal fragmentation becomes a problem for packing consTokens in genTokens as their relative size increases. Consequently, splitting a token leads to higher utilizations. Furthermore, we empirically observe that splitting along the time axis, is better than splitting along the power axis.

D. Sensitivity

In this section, we vary the different parameters in TMS and measure their effect on the experimental results. In Figure 12, we vary the different scheduling schemes in the dispatcher and measure the utilization. We observe that the *endTime* algorithm shows better performance than any other scheme. This is because we should try to make the maximize the utilization of any genToken before its expiration.

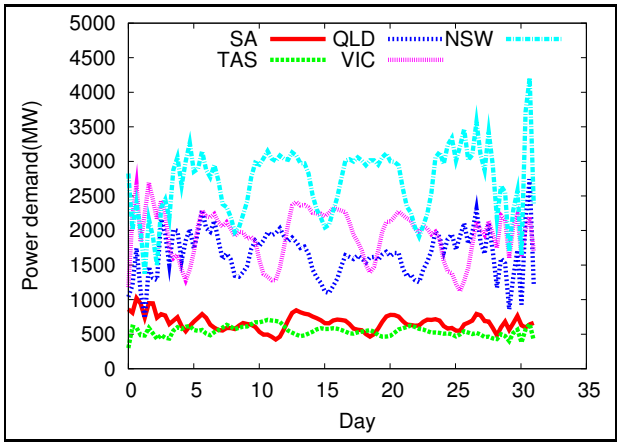

Fig. 6. Power demand in Australia for the last 5 years

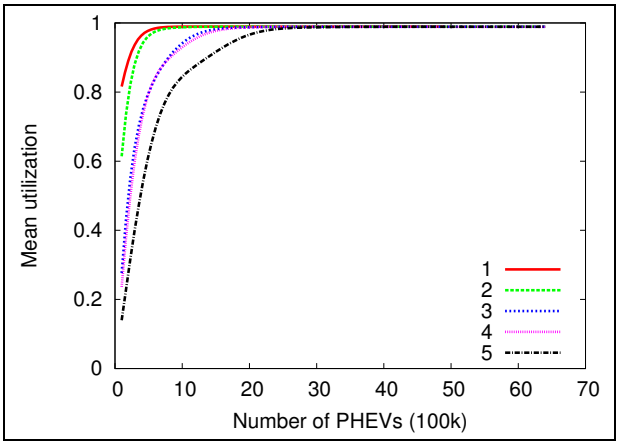

Fig. 7. Utilization for different number of power stations

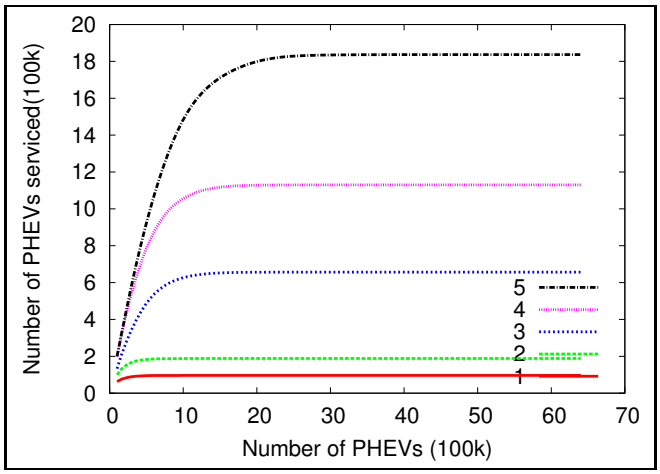

Fig. 8. Number of PHEVs actually serviced for different number of power stations

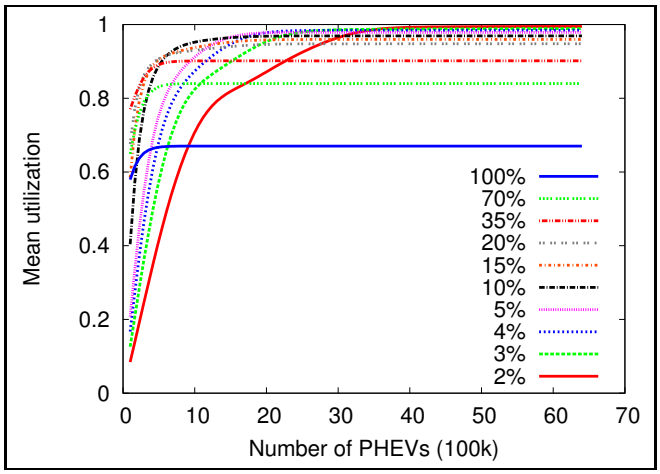

Fig. 9. Utilization vs. consToken duration

Thus giving priority to the genTokens whose end times would soon approach, increases the utilization considerably. When the number of PHEVs is very large, the scheduling schemes for the genTokens do not affect utilization, since some or the other consToken will get scheduled into each of the genTokens. Figure 13 shows the utilization as a function of the prioritization scheme for selecting genTokens: rr, $fifo$, $largest$, $smallest$, $earliest$, and $latest$. We observe that for small number of PHEVs (uptil 700k PHEVs), the *latest* scheme performs better. As the number of PHEVs increases, the *largest* scheme performs the best. The superior performance of the *latest* scheme owes to giving priority to genTokens which would expire sooner. With an increase in the number of PHEVs, there are enough consTokens to be packed into all genTokens. There is contention among the genTokens to get into the active genToken space(*agt*) because some genTokens may have to expire without getting a chance to enter *agt*. Thus preferring the ones with larger power helps because more conTokens can be packed. The largest-latest scheme, which is a combination of the two schemes, yields considerably better throughout for any number of PHEVs.

Figure 14 shows the number of PHEVs serviced as a function of the validity time of a consToken. The validity time is a multiple of the duration. We vary this factor from 1 to 10. As expected, we observe an increase in the number of PHEVs that we can service with increasing validity periods because it gives us a larger window to schedule the consTokens. Lastly Figure 15 shows a slotted activation, where we vary the number of slots from 50 to 300. As expected, all the schemes are worse than the unslotted case. Even with 700 slots we are pretty far from the unslotted case. The utilization for 300 and above slots is above 90%. Consequently, we can conclude that to get acceptable performance, we need to have a minimum of 300 slots in a genToken.

V. Related Work

To the best of our knowledge, this paper presents the first work that proposes an IT infrastructure for implementing energy-as-a-service for PHEVs. This section discusses the related work on characterizing the power demands for PHEVs, and the application of tokens in networked and distributed systems.

1) Power demands for PHEVs: The seminal papers by Kempton and Tomic [10], [11] provide an overview of the electricity markets in the future. They describe the charging requirements of PHEVs along with their economic consequences. They observe that PHEVs have a massive potential

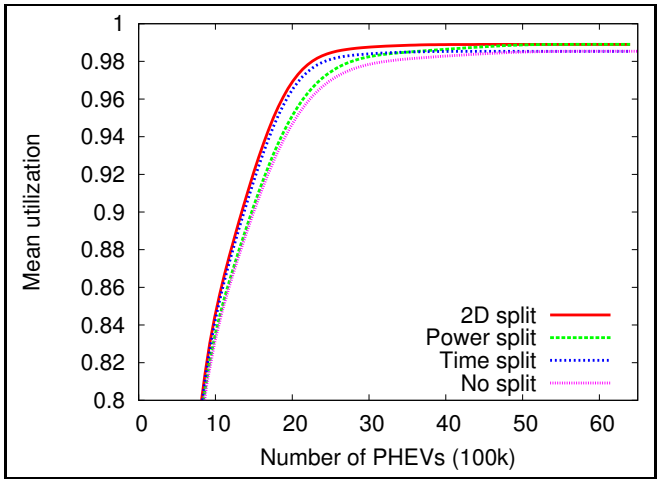

Fig. 10. Utilization vs. splitting algorithm (small consTokens)

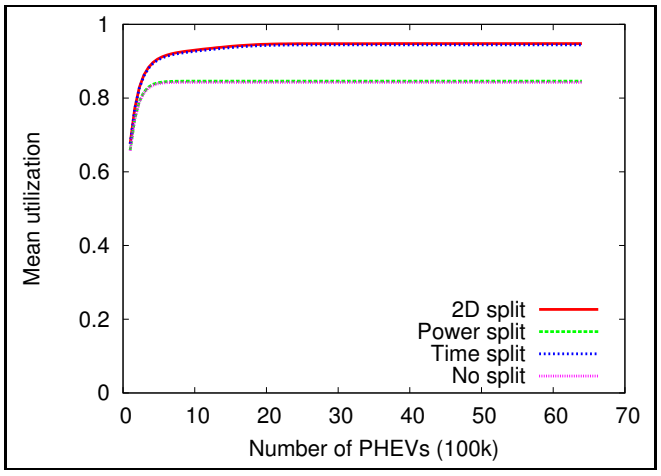

Fig. 11. Utilization vs. splitting algorithm (large consTokens)

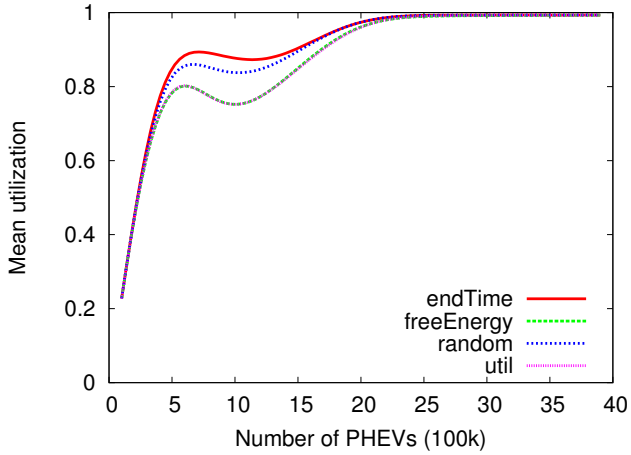

Fig. 12. Different scheduling schemes in dispatcher

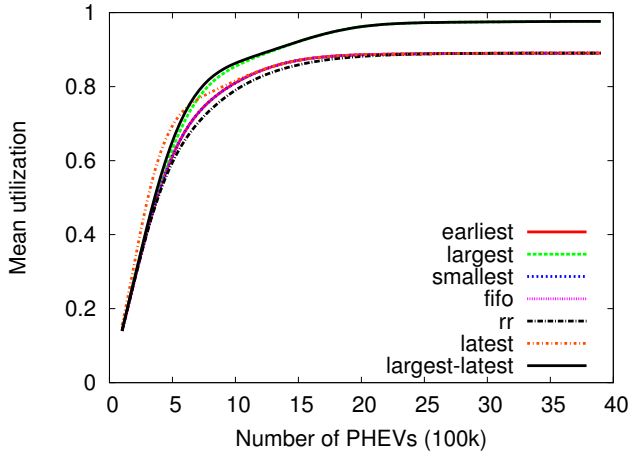

Fig. 13. Different prioritization schemes for selecting genTokens

for supplying power back to the grid, called Vehicle-to-Grid (V2G), and estimate the economic impact of V2G to be about 12 billions dollars per year in the US. Several other studies [3], [2] have looked at the potential for reduction in emissions, and the additional infrastructure that is required in the power grid. They observe that in developed countries, the power grid will be able to manage the peak requirement for less than 30% PHEV market penetration. Beyond that, there is need to schedule the PHEV charging appropriately using IT infrastructure and communication hardware. Putrus et. al. [7] consider the problem of supply-demand matching in power distribution networks. They observe that the problem of supply-demand matching gets exacerbated with increasing number of PHEVs.

Cirrincione et. al. [17] propose an intelligent energy management system, where there is a producer, consumer, and a broker. The broker provides the list of producers to the consumer. Unlike the Token Management System (TMS) presented in this paper, the system in [17] does not schedule on demand. Kulshreshtha et. al. [18] present the architecture of a smart parking lot for charging PHEVs. Here PHEVs submit their requests to a centralized agent, which has a fixed power budget. It then schedules the PHEVs in ascending order of their charging times. In comparison, TMS implements a much larger distributed system, pays much more attention to the communication aspects, and considers many different scheduling schemes.

2) Token-based systems: Tokens have traditionally been used for granting exclusive access to resources. The rest of this section discusses some important applications of tokens in networked and distributed systems, and points out how they differ from TMS.

Token passing is used as a general communication channel access mechanism in which a token is passed between different nodes that share a common channel [19]. A node is authorized to transmit on the channel only when it holds the token, thus avoiding collisions during transmission. Token passing is used in multiple communication protocol, e.g., IEEE 802.5 Token Ring, IEEE 802.4 Token Bus, and Fiber Distributed Data Interface (FDDI). TMS is different from token passing protocols in the following ways: (1) it allows both the generators and consumers of the resource (power) to interact using generator and consumer tokens, whereas in earlier token passing protocol the generator of the resource (the communication channel) is a passive entity, (2) it handles production, scheduling and

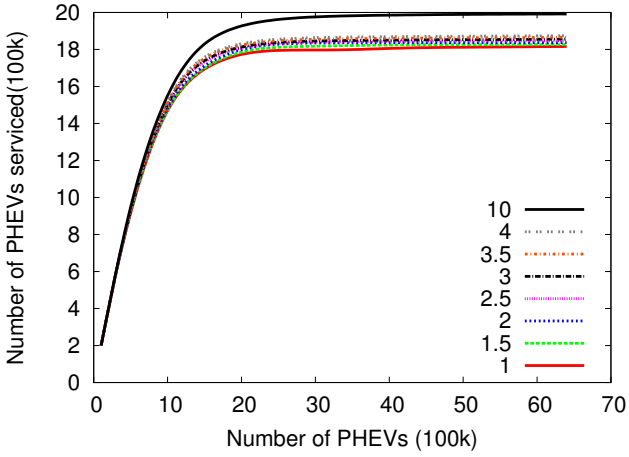

Fig. 14. Throughput as a function of the consToken validity period

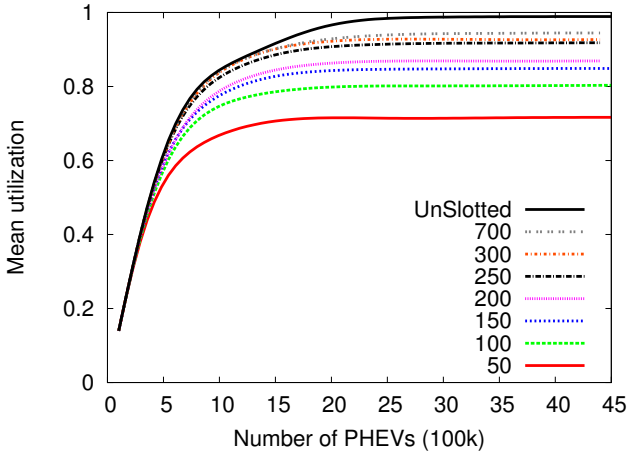

Fig. 15. Utilization vs number of slots in a genToken

distribution of multiple concurrent tokens, and (3) it can handle variability in the amount of resource over time.

Tokens are also used for traffic shaping in a communication network. In the token bucket scheme [19], tokens are generated periodically in a bucket at a given rate, and a node uses up certain number of tokens in the bucket to transmit a packet. Token bucket schemes can shape the traffic from the node so as to maintain an average bit rate, while allowing for some level of burstiness. TMS is orthogonal to the token bucket scheme since it does not shape the token generation and consumption rate. However, token buckets can be used in TMS to impose limits on these rates.

In peer-to-peer systems, [20], [21] present tokens based schemes for sharing resources between peers. They use tokens as a proof of service usage. Tokens are also tagged with the name of the original owner. The dynamic distribution of tokens determines the resource usage. [22] uses generalized tokens to evaluate the trustworthiness of nodes in a peer to peer system, and to also ensure fair resource usage. [23] uses a token based scheme to provide incentives to users to collaborate more in a peer to peer setting. In all these schemes, tokens are viewed as a proof of usage and are discarded after they are received. In TMS, tokens contain much more information, and are viewed as active entities containing state.

Tokens are also used in network security protocols like Kerberos [24] to setup a mutually agreed private key, and are also used for authentication and trust management. In these network security protocols, a token is simply used to encapsulate a message. However, in TMS, tokens maintain state information, and they have a wide range of attributes to capture various access policies. Furthermore, in contrast to all the above token based schemes in communication networks, the tokens in TMS can be merged and split to provide a fine-grained access control.

Token based algorithms are also used for one of the fundamental problems in distributed computing systems: the mutual exclusion problem. Mutual exclusion algorithms are used to prevent multiple processes to simultaneously access common resources, i.e., to ensure that at most one process uses the common resources at every time instant. In token based mutual exclusion algorithms, a process can access a common resource only when it holds the token, but the algorithms differ in how the tokens are exchanged between the processes [25]. One of the requirements of TMS is indeed similar to the mutual exclusion problem: At every time instant, the consumption of (the common resource) power by PHEVs should be at most equal to the generation of power allocated to PHEVs. However, TMS circumvents the problem of solving mutual exclusion in a distributed setting by controlling the access from a central Admission Control and Scheduling System (ACS). Also, since the amount of power varies with time and the system needs to satisfy various other access policies (in addition to the power constraint), mutual exclusion algorithms cannot be applied directly to TMS.

VI. Conclusion

This paper presented the generalized architecture of an IT system to provide energy-as-a-service for PHEVs and power producers using the notion of a *token* as its internal currency. The paper presented a token management system (TMS) to manage tokens generated by millions of PHEVs and many different kinds of power stations. The TMS contains the Admission Control System (ACS) that takes all the vital scheduling and policy decisions in this system. The paper presented several scheduling policies for different modules in the ACS. Section A presented the communication protocol between different entities in the system. Finally, simulation results are obtained by implementing the infrastructure and simulating millions of PHEVs using traces from the Australian power grid for the last five years.

The primary observation from the simulations is that we are able to get a fairly high level of utilization($>$95%) for a multitude of power stations. Secondly, the performance of scheduling expressed in terms of the utilization of genTokens and the admission rate of consTokens, is very sensitive to the choice of the number of slots, the durations of the tokens, and its validity period. In comparison, the service quality is moderately sensitive to the scheduling and splitting policies. We would like to conclude by observing that the system presented in this paper is not limited to PHEV charging.

As a part of future work, we plan to generalize it to much larger utility networks, and incorporate many more sources of renewable energy.

APPENDIX

A. The Communication Protocol

at a generator g
upon power generation request **do**
  create genToken gt
  send $<GEN\_REQ, gt>$ to ACS
upon receiving $<GEN\_ACK, gt>$ from ACS **do**
  start generation for gt from its start time
upon receiving $<GEN\_NACK, gt>$ from ACS **do**
  remove gt from local token list

at a consumer c
upon power consumption request **do**
  create consToken ct
  send $<CONS\_REQ, ct>$ to ACS
upon receiving $<CONS\_ACK, ct, activation\_time>$ from ACS **do**
  start generation for gt from $activation\_time$
upon receiving $<CONS\_NACK, ct>$ from ACS **do**
  remove ct from local token list

at TMS
upon receiving $<GEN\_REQ, gt>$ from generator g **do**
  genTokenPriorityQueue.enqueue(gt)
upon receiving $<CONS\_REQ, ct>$ from consumer c **do**
  {*batch consToken*}
  select a $consBatch$ cb with with similar start time and duration as ct; add ct to cb
  if size of $cb \geq MAX\_BATCH\_SIZE$ **then**
    consTokenFIFOQueue.enqueue(cb); $cb := \emptyset$

at ACS {*Admission Control and Scheduling*}
while true **do**
  {*remove almost filled active genTokens*}
  {*agt is the list of active genTokens*}
  for each genToken $gt \in agt$ **do**
    if $(gt.utilization \geq MAX\_UTIL)$ **and** $(gt.num\_rejects \geq NUM\_MAX\_REJECTS)$ **then**
      remove gt from agt
  {*add new active genToken*}
  while size of $agt < MAX\_GEN\_ACTIVE$ **do**
    gt := genTokenPriorityQueue.dequeue()
    if gt has not expired **then**
      add gt to agt
      send $<GEN\_ACK, gt>$ to the generator of gt
    else
      send $<GEN\_NACK, gt>$ to the generator of gt
  cb := consTokenFIFOQueue.dequeue()
  dispatcher schedules consBatch cb by assigning an $activation\_time$ in a genToken
  for each genToken $gt \in agt$ **do** update $gt.utilization$ and $gt.num\_rejects$
  if cb is admitted **then**
    for each $ct \in cb$ **do** send $<CONS\_ACK, ct, activation\_time>$ to the consumer of ct
  else
    for each $ct \in consBatch$ **do** send $<CONS\_NACK, ct>$ to the consumer of ct

Fig. 16. Application-level communication protocol

The PHEV network can be viewed as a distributed system composed of the LMs at various nodes and the TMS. In Figure 16, we present the application level distributed protocol among the various components of the PHEV network. Request for a genToken is sent using a GEN_REQ message. The ACS replies with a GEN_ACK or a GEN_NACK depending on whether the genToken is used for scheduling or not. Similarly, request for a consToken is sent using a $CONS\_REQ$ message, and the ACS replies with a $CONS\_ACK$ or a $CONS\_NACK$ depending on whether the consToken is scheduled or not.

REFERENCES

[1] Y. Christopher and R. W. McCarthy, "Electricity grid: Impacts of plug-in electric vehicle charging," U. C. Davis, Tech. Rep. UCD-ITS-RP-09-17, June 2009.

[2] N. DeForest, J. Funk, A. Lorimer, B. Ur, I. Sidhu, P. Kaminsky, and B. Tenderich, "Impact of widespread electric vehicle adoption on the electrical utility business - threats and opportunities," Center for Entrepreneurship and Technology (CET), University of California, Berkeley, Tech. Rep., 2009.

[3] M. Tran and J. Bishop, "Exploring the potential impact of electric vehicles: Heroic assumptions, key uncertainties and the long road ahead," University of Oxford, Tech. Rep. Working Paper 1042, December 2009.

[4] M. Kintner-Meyer, K. Schneider, and R. Pratt, "Impacts assessment of plug-in hybrid vehicles on electric utilities and regional U.S. power grids," Pacific Northwest Laboratory, Tech. Rep., January 2008.

[5] SENTECH Inc., "Plug-in hybrid electric vehicle value proposition study," Oak Ridge National Laboratory, Tech. Rep., June 2008.

[6] "The global outlook for PHEVs: Business issues, technology issues, key players, and market forecasts," Argonne National Laboratory, Tech. Rep., June 2009.

[7] G. A. Putrus, P. Suwanapingkarl, D. Johnston, E. C. Bentley, and M. Narayana, "Impact of electric vehicles on power distribution networks," in *IEEE Vehicle Power and Propulsion Conference*, September 2009.

[8] J. Gartner, "Balancing phev power demand," http://featured.matternetwork.com/2007/2/controlling-demand.cfm, Accessed on Feb 25th 2010.

[9] W. S. Hadley and A. Tsvetkova, "Potential impacts of plug-in hybrid electric vehicles on regional power generation," Oak Ridge National Laboratory, Tech. Rep., January 2008.

[10] W. Kempton and J. Tomic, "Vehicle-to-grid power fundamentals: Calculating capacity and net revenue," *Journal of Power Sources*, vol. 144, no. 1, pp. 268–279, 2005.

[11] ——, "Vehicle-to-grid power implementation: From stabilizing the grid to supporting large-scale renewable energy," *Journal of Power Sources*, vol. 144, no. 1, pp. 280–294, 2005.

[12] V. T. Chakaravarthy, V. Pandit, Y. Sabharwal, and D. P. Seetharam, "Varying bandwidth resource allocation problem with bag constraints," in *IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*, 2010.

[13] G. Calinescu, A. Chakrabarti, H. J. Karloff, and Y. Rabani, "Improved approximation algorithms for resource allocation," in *International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, 2002.

[14] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Second Edition*. The MIT Press and McGraw-Hill Book Company, 2001.

[15] AEMO, "Australian energy management operator," http://www.aemo.com.au/, Accessed on Feb 25th 2010.

[16] P. Sharer, "Vehicle simulation results for plugin HEV battery requirements," Argonne National Laboratory, Tech. Rep., October 2006.

[17] M. Cirrincione, M. Cossentino, S. Gaglio, V. Hilaire, A. Koukam, M. Pucci, L. Sabatucci, and G. Vitale, "Intelligent energy management system," in *IEEE International Conference on Industrial Informatics*, June 2009.

[18] P. Kulshreshtha, L. Wang, M. Chow, and S. Lukic, "Intelligent energy management system simulator for PHEVs at municipal parking deck in a smart grid environment," in *Power and Energy Society General Meeting (PES)*, 2009.

[19] A. S. Tanenbaum, *Computer Networks*. Prentice Hall, 2010.

[20] N. Liebau, V. Darlagiannis, M. Mauthe, and R. Steinmetz, "Token-based accounting for p2p-systems," *Kommunikation in Verteilten Systemen (KiVS)*, 2005.
[21] D. Hausheer, N. C. Liebau, A. Mauthe, R. Steinmetz, and B. Stiller, "Token-based accounting and distributed pricing to introduce market mechanisms in a peer-to-peer file sharing scenario," in *International Conference on Peer-to-Peer Computing*, 2003.
[22] T. Moreton and A. Twigg, "Trading in trust, tokens, and stamps," in *First Workshop on Economics of Peer-to-Peer Systems*, 2003.
[23] K. Ranganathan, M. Ripeanu, A. Sarin, and I. Foster, "Incentive mechanisms for large collaborative resource sharing," in *IEEE International Symposium on Cluster Computing and the Grid*, 2004.
[24] B. C. Neuman and T. Ts'o, "Kerberos : An authentication service for computer networks," USC/ISI, Tech. Rep. ISI/RS-94-399, 1994.
[25] I. Suzuki and T. Kasami, "A distributed mutual exclusion algorithm," *ACM Trans. Comput. Systems*, vol. 3, no. 4, pp. 344–349, 1985.

Smruti R. Sarangi is an Assistant Professor in the Department of Computer Science and Engineering, IIT Delhi, India. He has spent four years in industry working in IBM India Research Labs, and Synopsys. He graduated with a M.S and Ph.D in computer architecture from the University of Illinois at Urbana-Champaign in 2007, and a B.Tech in computer science from IIT Kharagpur, India, in 2002. He works in the areas of computer architecture and operating systems. Prof. Sarangi is a member of the ACM.

Partha Dutta is a research staff member in the IBM Research Labs, Bangalore, India. Dr. Dutta received a B.Tech. in electrical engineering from the Indian Institute of Technology, Kanpur, India, in 1999 and a Ph.D. in computer science from Swiss Federal Institute of Technology, Lausanne, Switzerland, in 2005. His research interests are in distributed systems and computer networking. Dr. Dutta is a member of IEEE and ACM.

Komal Jalan is currently working in Google, Bangalore, as a Software Engineer. She received her B.Tech degree degree in Computer Science and Engineering from the Indian Institute of Technology, Guwahati, India, in 2011. She has also been associated with IBM Research Labs, Bangalore, as a research intern. She is works in the field of networks and cloud computing.