A Distributed In-Transit Processing Infrastructure for Forecasting Electric Vehicle Charging Demand

Rafael Tolosana-Calasanz[1], José Ángel Bañares[1], Liana Cipcigan[2],
Omer Rana[4], Panagiotis Papadopoulos[5], Congduc Pham[3]
[1]Aragón Institute of Engineering Research (I3A), University of Zaragoza, Spain
[2]Institute of Energy, School of Engineering, Cardiff University, UK
[3]LIUPPA Laboratory, University of Pau, France
[4]School of Computer Science & Informatics, Cardiff University, UK
[5]EDF Energy, R&D Centre, London, UK

***Abstract*—With an increasing interest in Electric Vehicles (EVs), it is essential to understand how EV charging could impact demand on the Electricity Grid. Existing approaches used to achieve this make use of a centralised data collection mechanism – which often is agnostic of demand variation in a given geographical area. We present an in-transit data processing architecture that is more efficient and can aggregate a variety of different types of data. A model using Reference nets has been developed and evaluated. Our focus in this paper is primarily to introduce requirements for such an architecture.**

I. Introduction & Motivation

The introduction of government legislation and the associated penalties for exceeding certain CO_2 emission levels has encouraged major automotive manufacturers to announce production of Electric Vehicles (EVs). Virtually all major car manufacturers (such as European (BMW, Volvo, Volkswagen, Fiat, Peugeot, Renault), Japanese (Suzuki, Toyota, Nissan, Honda, Mitsubhishi) and American (GM/Chevrolet, Ford)) have announced the inclusion of EVs in their production facilities and marketing plans. These vehicles are anticipated to gain an important market share over conventional Internal Combustion Engine (ICE) powered vehicles. Analysis on lifecycle CO_2 emissions [1] that was published for the British government concludes that in the year 2030, EVs may be able to produce less than 50g/km CO_2 emissions, approximately one third of petrol based vehicles. In the same context, the Committee on Climate Change recommends that the Government should aim for 1.7 million EVs on the road by 2025 [2]. In order to re-charge their batteries and provide traction, EVs will have to be connected to power networks [3].

From a power system's (electricity grid) perspective, EVs may be considered as: (i) simple loads drawing a continuous current from the electricity network; (ii) flexible loads that may allow an *aggregator* company to interrupt or coordinate their charging procedure; (iii) storage devices that may allow an aggregator company to request power injections from their batteries back to the electricity grid (the latter is known as Vehicle to Grid (V2G)). A unique characteristic of EVs in terms of power systems is that they are mobile devices expected to connect to various locations of electricity networks at different times of a day. Recent studies estimate that by 2030, if the re-charging processes of EV batteries are left uncontrolled, a significant increase in the electricity demand peaks is to be expected [4]. Moreover, the impact of EVs is expected to be at the local level where hotspots will be created that depend on how EVs will cluster within a particular geographical location, creating a possible overload on the low voltage distribution networks. Even a small number of uncontrolled vehicles charging at peak periods could significantly stress the distribution system, slowing EV adoption and requiring major electricity (generation and distribution) infrastructure investments. Our focus here is on EVs that are used by individuals in a residential or city context, and not on vehicle fleets where charging models can be different.

A. Approach

A key challenge in supporting EV demand forecasting is understanding: (i) over what period of the day an EV owner is likely to request charging; (ii) characteristics of the EV (such as battery being used, distance travelled, etc); (iii) driving behaviour of the vehicle user; and (iv) environment factors that impact (ii) and (iii). Such information is generally not available at a single location, for instance some of these factors reside on the EV while others need to be derived from independently managed data sources (operated by the weather or traffic agencies). The information needed to support EV demand forecasting can therefore consist of (i) heterogeneous data formats & varying transmission rates; (ii) irregular or bursty data streams; and (iii) variable, difficult to predict processing requirements per data stream. The processing of these data streams may involve filtering, data correlation and trends analysis. Processing is also typically *stateful*, i.e. after the processing of a data element, a state must be kept in memory for processing subsequent data elements. The outcome of stream analysis must be exploited and compared with historical load demands (stateless), so that a forecast can be obtained. Identifying how limited network and computational resources (subject to congestion, failure and performance degradation) can be applied to stream analysis is an important challenge and the focus of the work reported here. Demand forecasting for EV charging is also a time critical process, as vehicle owners may need their battery to be charged within

a few hours. Additionally, emergency and unforeseen traffic or weather events may render previous forecasts inaccurate or unusable, thereby requiring a new forecast to be developed. The underlying network and computational architecture must therefore support Quality of Service (QoS) constraints to be observed to support the generation of timely forecasts.

We propose a distributed computing architecture and a model for supporting EV demand forecasting, which consists of a number of autonomous stages, where each stage consists of a combination of data access and regulation, computation, data transfer capability, and a rule-based controller component. The computation stage makes use of a dynamically adaptable resource pool, enabling multiple computational resources to be used (and released) on-demand. To support QoS constraints, we consider: (i) the average throughput per stream; (ii) the maximum allowed burst per stream, and (iii) the need for data dropping – i.e. dropping some data elements if they do not directly contribute to demand forecasting. We extend previous work [5], [6], [7] by integrating an admission control policy in the incoming traffic regulation component, support for both stateful and stateless stream processing. We have developed a proof of concept prototype of the system with Reference nets (a type of executable Petri nets that support Java actions) [8].

The rest of the paper is organised as follows. In Section II, EV charging requirements are provided. Requirements for data handling and distributed processing are identified in this section. In Section III, we present the system architecture for making the EV demand forecasting application deployable on a shared computing infrastructure. Two experiments are conducted to show the key aspects of the model, with results described in Section IV. Previous related work is compared to this proposal in Section V. Conclusions and future work are discussed in Section VI.

II. Electric Vehicle Demand Forecasting

As in figure 1 there are a number of possible sources of data that contribute to the forecasting process. This includes: (i) data obtained from vehicles (ii) data obtained from a charging station – such as duration of charging, energy consumed/used up during charging, type of charger used, desired time of disconnection etc; (iii) data obtained from external data agencies that have a bearing on the charging requirement of vehicles. This data is streamed to a number of nodes, indicated as node Ni in the figure. The architecture for each node in the system is discussed further in Section III. A key requirement in this approach is to forecast the total energy demand at a charging point – and subsequently aggregate this demand within a particular region. In our approach, we do not collect the data relating to vehicles, charging points, weather/traffic (identified above) at a central point. Instead, we develop a model for each charging point and then combine the outcome of these models using our distributed data management and computation infrastructure. This enables different types of models to be used concurrently, where the complexity of a model depends on the likely rate of change at a particular charging point. A residential charging point is likely to have less demand variation than a charging point within a city. The computational requirement for each model can therefore change based on its location within a particular geographical area.

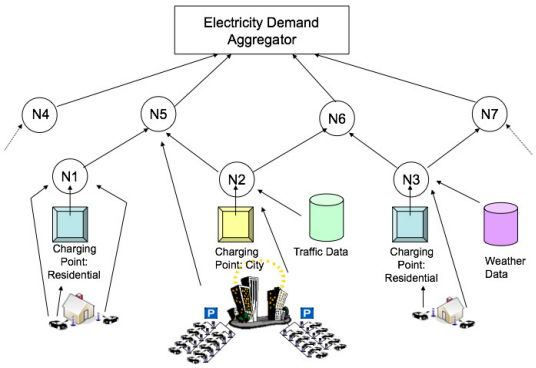

Fig. 1. Data management architecture within a particular region. The "aggregator" is responsible for combining electricity demand within a particular geographical area.

A. EV Demand Forecasting Model Requirements

We identify various scenarios that illustrate the requirements on the data management and computational infrastructure required for EV demand forecasting in Table I. These scenarios are used to derive the requirements for our system architecture and model. Sensors in this instance may be associated with an EV (such as monitoring current battery charge) or those associated with a charging point (which may be located at homes, commercial buildings or service stations within the city or along motorways). The simultaneous charging of a number of EVs will generate a significant load on the electricity grid and thus some coordination is necessary to distribute this charging requirement over time. Various approaches are possible to achieve this – such as synchronising charging with tariffs (including the availability of real time pricing signals) identified by the utility company or with the load currently on the electricity grid. Demand forecasting is therefore essential to enable the utility company to enable such coordination to take place.

III. System Architecture

In this section, we will present our architecture for enabling an elastic, QoS-aware, and scalable distributed streaming system for EV demand forecasting. The proposed architecture supports the processing of multiple sources of information over a shared infrastructure, and consists of a sequence of nodes that accomplish the computations. Our objective is to maintaining an end-to-end QoS for each source of information, by enforcing QoS at each node when data is streamed through them. Each node is independent of another, and we assume that (i) data transmissions required for meeting QoS, on average, do not exceed the network bandwidth available, (ii) the required processing capability on average does not exceed the computational power of the available resources.

EV scenario	Data management & computation requirements
Forecasting demand at a charging point requires handling a number of concurrent data streams. Each stream can have different characteristics: rate, frequency and sample size. The influence of data within a particular stream on the forecast may also vary – i.e. some data streams may be of greater importance in determining the outcome. Some streams may come directly from the vehicle or the charging point, whereas others may need to be derived from data maintained by third parties (such as weather and traffic agencies) and filtered to determine regional characteristics.	**Requirement #1**. It is necessary to allocate priority to different data streams, based on their impact on the prediction outcome. A resource management strategy needs to be used that allocates a greater number of computational resources to process high priority streams and alter this dynamically based on change in stream behaviour. The underlying infrastructure must be able to support admission control and enable a variable processing rate per stream.
Current demand aggregation is focused on a regional context. Hence, demand identified at charging points within a residential or city area needs to be combined. It may be necessary to change the granularity of the region being considered to more accurately reflect a variation in demand and identify the likely sub-region where such variation occurs.	**Requirement #2**. The data streams needed to support demand prediction can change – requiring the addition or removal of particular streams to enable analysis to take place for different geographical areas. A key requirement in this context is the ability to dynamically choose the data streams of interest.
Different charging points may exhibit behaviour of varying complexity. For instance, charging points which exist within residential areas may have a more regular demand pattern, compared to those that occur within a city where demand can be influenced by events such as road congestion (due to sporting events, accidents), weather patterns, etc.	**Requirement #3**. The computational resources needed to forecast demand will depend on the likely rate of change in demand at a particular charging point. It is therefore likely that a different forecast model would be appropriate for residential areas compared to a city-based charging point. The infrastructure should therefore provide capability to consider different forecast modelling capabilities (such as linear regression, neural networks, bayesian models, etc) to co-exist. In the same context, EV usage patterns in residential areas are likely to be regular, implying that data rates are likely to be well defined. Conversely, charging points at shopping centres or public places at city centres are likely to see varying demand, leading to data capture with periods of burstiness.
The data used to generate forecast models may have interdependencies. For instance, weather data may influence traffic congestion within a particular area. Similarly, the state of charge of a vehicle battery would depend on the type of battery being used, the traffic flow within a given area, etc. Therefore, although different data streams are needed, the dependencies between the data streams may influence the forecast outcome.	**Requirement #4**. The data management system should enable mechanisms to provide isolation between streams and where necessary, to also enable dependencies between streams to be identified. Data streams will also have errors and involve missing data items. It is therefore necessary when processing these streams to be aware of how one stream influences another and how data quality within one stream influences another.
Developing a model to forecast EV demand is a time critical process, as the aggregator identified in figure 1 needs to request electricity from a power generating company only a limited time prior to its use. To facilitate this, there is a time window within which data from the EV and the charging point needs to propagate to the computational node responsible for developing the forecast.	**Requirement #5**. With multiple data flows present within the system and the need to observe particular time thresholds, it is necessary to develop Quality of Service (QoS) constraints to limit data loss and latency. Such QoS constraints should also identify bounds on the computational time necessary to develop the forecast.

TABLE I: Data management and computation requirements for various EV demand forecast scenarios – with reference to the regional demand aggregation illustrated in Figure 1

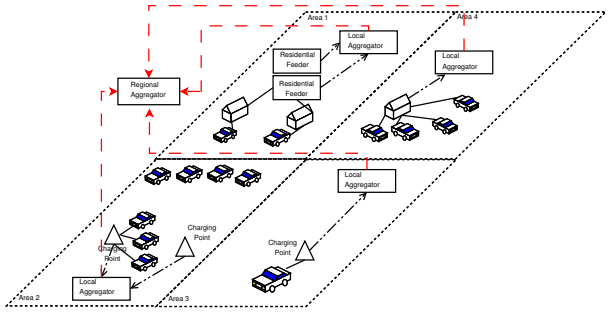

Fig. 2. Hierarchical division into sub-areas with local and regional aggregation of demand

Our architecture relies on the current hierarchical decomposition of an electricity grid into logical zones to support monitoring and control, as identified by Serizawa et al. [9]. Such a decomposition enables a more scalable means to control and manage a particular part of the power network. We take two perspectives to support load forecasting: (i) an EV and charging point perspective; (ii) a zonal perspective – which enables aggregation across all EV charging in that zone. The first of these corresponds to determining likely charging regimes for an EV at different times of day, the accuracy of which can vary with the behaviour of the driver, the pricing tariffs for charging currently enforced by the utility company and other external factors which impact the battery charge (such as weather and traffic congestion events, etc). Figure 2 illustrates the sub-division into areas and local aggregation of demand within each area. These areas can include residential feeders that also aggregate demand from homes and EV charging points associated with a home (Area 1), to office blocks (Area 4), city areas (Areas 2 and 3 in the figure) which can include multiple charging points and which can have variable density of EVs at any given time. A Regional Aggregator then combines demand from each of these local aggregators (as also illustrated in the figure). From an electrical power grid perspective, charging points within each of these areas are at the Low Voltage substation level, local aggregators at Low to Medium Voltage substation level, and the regional aggregator at the High Voltage Level. EV charging regimes can range from: Uncontrolled Regime: EV charging begins as soon as a commuter connects to the charging point. The daily traffic pattern of commuters determines when EV charging will take place. Dual Tariff Regime: The utility company introduces two tariffs – a higher price tariff in the morning when there is a greater demand for power and a low price tariff at night. Variable Price Regime: This assumes that the utility company provides real time pricing signals, based on the existence of smart meters at consumer premises. Hence, a wide variety of possible pricing signals can be produced, leading to different charging regimes based on demand from the commuter and pricing information from the utility company. Mixed Charging Regime: In which the above three regimes were combined in different ways – with different proportions of commuters charging their vehicles using one of the three regimes outlined above. In the same way, when considering EV charging outside residential areas, traffic patterns can be used to infer possible charging regimes. Whereas dual tariff regimes would be appropriate for commuters charging at home – i.e. preference for over night charging when price is low, such a regime would be inappropriate for city charging, where a spot price would need to be determined when a charging request is made (in the absence of energy storage capability). Our architecture consists of a number of nodes that can receive data from charging points and local aggregators. We estimate the total data sizes we need to consider as follows:

- Each EV profile is approximately 1KB – and we assume approximately 400 EVs in each area (as illustrated in figure 2). This value is based on the number of customers connected to a secondary transformer in a residential area, which for the UK generic distribution network is 384 [10]. The EV profile generally consists of the following parameters: (a) EV battery State of Charge (SoC); (b) EV battery characteristics – which are dependent on the manufacturer and the current age of the battery; (c) Inverter/Charging point efficiency – i.e. how efficient is the overall charging process; (d) The EV battery utilisation cost; (e) Preferred charging mode; (f) Time of disconnection and SoC; (g) Electricity price schedule.
- We assume data may be communicated bi-directionally – from a charging point to a local aggregator and from a utility company to a charging point (such as price signals).
- Depending on the charging regime being used, it is necessary to estimate the number of EVs at any particular charging interval (i.e. the time over which a demand for energy is made by an EV owner). We assume that residential areas have more predictable demand than city areas (which can change based on time of day or other events (weather or congestion related, for instance)).
- Data from an EV can become available when: (i) the EV connects to a charging point within a residential or city area; (ii) when an EV is within some geographical proximity to a charging point – which advertises the current tariffs to the EV owner.

A. Architecture

Our architecture consists of a number of nodes located within each area illustrated in figure 2. These nodes are connected by networks that can support a variety of different Quality of Service characteristics (ranging from wired to wireless connectivity). A node can be physically hosted at a charging point or may be a data collection resource operated by a mobile phone network operator. A number of such nodes form our demand forecasting architecture, a key feature of which is that forecasting should not be carried out at a central point in the network (i.e. at local or regional aggregators only). Instead, demand forecasting may be distributed across multiple nodes within a data capture and processing network, with some of the nodes being hosted at different substation levels of the electricity grid (low, medium and high voltage).

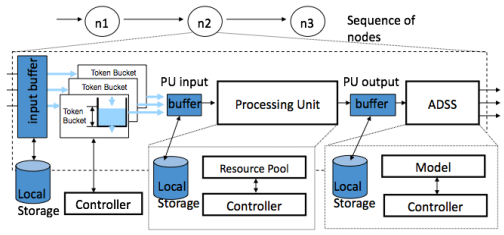

Fig. 3. System Architecture, and the elements of a node. ADSS is the Autonomic Data Streaming Service – details of which can be found in [5]

All nodes have a similar architecture, consisting of three key components: (i) a token bucket per data stream – for each type of data that contributes towards forecasting demand. A data stream can include: EV profile, pricing signals, weather or congestion events, etc; (ii) a processing unit (PU) component that assigns data to computational resources. The number of processing units allocated to each stream can vary, and are dependent on the urgency needed to process a given data stream at any given time. We assume that the number of processing units allocated to a stream are dependent on the build up of tokens within a particular stream within the token bucket associated with the data stream; and (iii) a data transmission service that subsequently forwards partially processed outcomes to the following node in the network. Figure 3 illustrates this architecture where each stage contains its own controller component. Each node corresponds to Ni identified in figure 1.

B. Handling multiple, heterogeneous data streams

Within a data stream, it is often useful to identify a "data acceptance rate", which is often different from the physical link capacity connecting nodes and which identifies the rate at which a stage can receive and process data. Each node tries to maintain this acceptance rate as the output rate. We characterise it for each flow by means of three QoS parameters: (i) average throughput (average number of data elements processed per second), (ii) maximum allowed burst, and (iii) an optional load shedding (data dropping) rate. We make the first two parameters match R and b of the token bucket respectively. For each data stream, its associated token bucket will allow data elements to enter into the PU stage according to the R parameter. The token bucket can also accept a burst of b data elements. Subsequently, a data element is forwarded to a First Come First Serve (FCFS) queue buffer at a PU. In addition to regulating access to the PU and enforcing QoS per data stream, the token bucket also achieves stream isolation, i.e. a data burst in one stream does not interfere with another. The load shedding mechanism acts at the input buffer of the node by discarding the older data elements of a flow at a specified rate. It is only active, however, when triggered by the PU stage controller component.

IV. Evaluation

We evaluate the effectiveness of our architecture for the scenarios described in table I. In particular, we demonstrate how adaption of token bucket parameters and the number of processing elements allocated to each stream can help address some of the data management and computation requirements identified in table I.

A. Experiment 1: Self-adaptation with different performance constraints

This experiment focuses on requirements 1 and 2, i.e. how priority can be allocated to different data streams (originating at a charging point or a local aggregator) based on the impact the data will have on the prediction outcome. In particular, the objective is to demonstrate how resources can be dynamically allocated to process a stream from a more congested area (simulated by a burst in data traffic received from the area), compared to data traffic received from an area with a more EV charging requirement. It is therefore possible to modify the rate at which a stream enters a node – by altering token bucket parameters, or the subsequent allocation of processing units to higher priority streams.

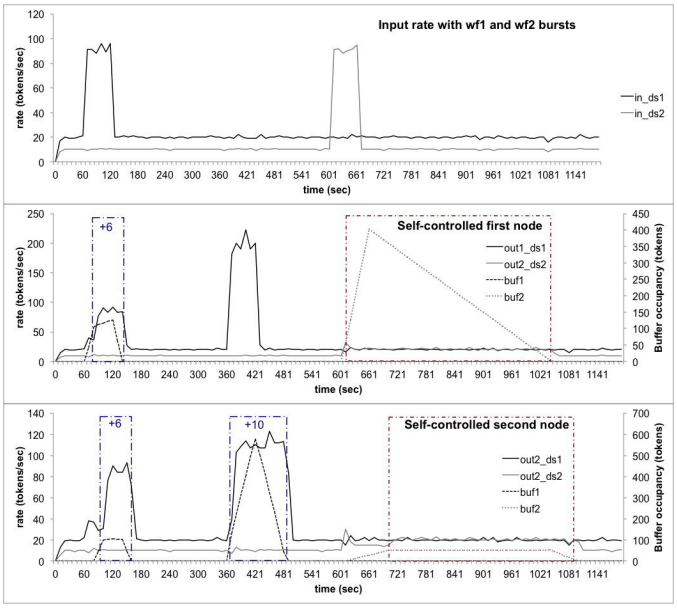

Fig. 4. Throughput with self-adaptive control and token bucket at each stage.

We consider two streams – ds_1 and ds_2, which have been generated by two local aggregators – and are received by a node at the regional aggregator, as illustrated in figure 2. ds_1 is of a higher priority and therefore must be allocated additional resources to achieve a processing time target, compared to ds_2, which only makes use of resources that are free.

The top graph in figure 4 shows the input rate for ds_1 and ds_2. A burst is produced during interval 60s-120s for ds_1 and during interval 600s-660s for ds_2. This burst simulates an increase in demand for charging over this interval within the region associated with ds_1. The graph in the middle of Figure 4 shows the throughput for ds_1 and ds_2 at the first node. It shows ds_1's throughput at the first stage, regulated by the token bucket throttling mechanism: the first small peak at time 60s is caused by the allowed burst. As the violation persists,

the second peak shows how more flexible constraints allow the addition of 6 resources and the throttling rate is increased until 90 tokens/s during the considered interval. The 3^{rd} peak of 200 tokens/s is produced to simulate a *data inflation* – i.e. there is a significant increase in the amount of data produced (compared to the size of the input data). The bottom graph shows how the adaptation to the burst at ds_1's input is propagated at the second stage, which reacts by introducing 6 additional resources for ds_1 to the second node. Data inflation in ds_1 at the first stage triggers the use of 10 additional resources at the second stage of ds_1 until the input buffer of the second stage is emptied. These graphs also show buffer occupancy at each node on the right y-axis. In this example, the buffer occupancy threshold to trigger the addition of new resources has been set in ds_1's configuration to be a value of 50 tokens with the resources being returned when the buffer is emptied. The peaks of ds_1's buffer occupancy due to the input burst is controlled in the two stages as additional resources can be introduced to correct the difference between input and output rates. However, there are not sufficient resources to absorb ds_1's data inflation introduced in the first node. These graphs show that during the interval where burst and data inflation are produced in ds_1 there is no noticeable effects on ds_2.

The top graph in figure 4 also depicts an input rate burst in ds_2. ds_2's requirements only supports the use of free resources that may be used when the $wf2$ buffer occupancy reaches 50 tokens. The middle graph shows how the first node increases the token bucket rate to 20 tokens/s (5 tokens/s over the initial R of 15 tokens/s). Looking at ds_2's buffer occupancy at each node, when the buffer is emptied R returns to the initial value and the throughput returns to 10 tokens/s provided by the input rate. ds_2 buffer occupancy at node 1 is over the threshold identified in ds_2's configuration, because there insufficient free resources to absorb the burst and therefore data are buffered. The bottom graph in figure 4 illustrates the same adaptation for ds_2 at the second node. Until ds_2's buffer occupancy reaches 50 tokens, ds_2's token bucket rate is 15 tokens/s. Once this threshold is reached,ds_2's token bucket rate is increased until 20 tokens/s to use free resources. When the buffer is emptied, R returns to the initial value and the throughput returns to 10 tokens/s provided by the input rate. In this case, ds_2 buffer occupancy is under the predefined threshold because the free resources are sufficient to absorb ds_2's rate adaptation at the first node. Again, these graphs show that during the interval where the burst is produced in ds_2 there is not noticeable effects on ds_1.

An additional control action is load-shedding, which is primarily used when it is not necessary to process all data received from a charging point – as the demand pattern may be regular and not all data samples are needed to support forecasting. Alternatively, a sample strategy (considering every n^{th} sample from the data stream) may be used to approximate charging demand patterns. In this case, data in the buffer can be dropped based on the chosen sampling strategy. Figure 5 reproduces the previous scenario, but in this case ds_1 and ds_2's configuration parameters state a threshold of 50 tokens for their buffer occupancy before the control logic drops data. Depending on the buffer strategy (e.g FIFO, LIFO, etc) oldest or newest data may be dropped. The graphs in figure 5 show how buffer occupancy time is shortened and how the number of resources in use can be reduced.

These previous scenarios show the behaviour of the token bucket and control actions to enforce the QoS of each data stream simulating punctual but strong burst to show the way the system adapt resources and data injection rates. However data streams may have very irregular data injection rates, as illustrated in Figure 6.

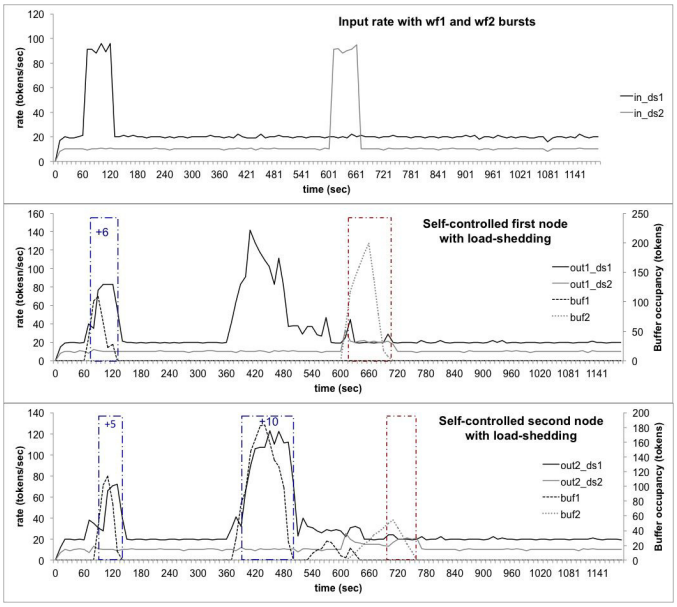

Fig. 5. Throughput with self-configuring control and token bucket at each stage using load-shedding during congestion.

We use the Poisson distribution to control both the probability of having bursts (by defining burst inter-arrival time) and a burst's duration. We set both the burst inter-arrival time and the burst's duration between 1s and 6s. The top graph shows a sample of irregular input rates distribution in time. The graph in the middle shows how ds_2's token bucket is adapting to 20 tokens/s during the intervals shown by the red boxes. As a consequence of this adaptation at the first stage, there are few impacts on the second stage as shown in the bottom figure where the load-shedding mechanism in that stage was not triggered.

B. Experiment 2: Processing Demand Surge

In this scenario, we evaluate the influence of varying the data processing rate on each stream arriving at an aggregator node. We consider the previous scenario in which two data streams ds_1 and ds_2 are executed simultaneously over two shared nodes. The left part of Figure 7 shows the output rates of ds_1 and ds_2 with no addition of resources when data can require more processing time (this is typically the case when fault tolerance mechanism are introduced which can generate extra processing overheads). In this simulation, we can observe

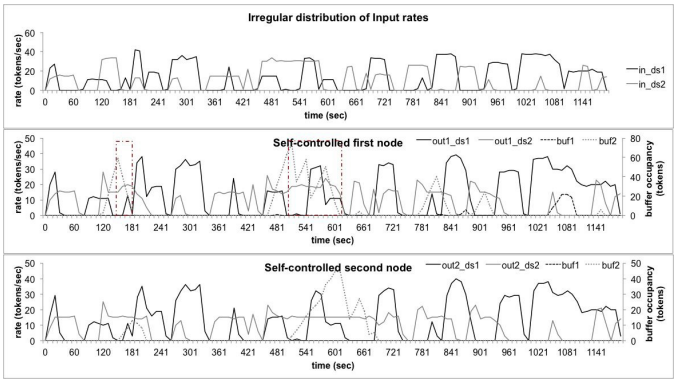

Fig. 6. Throughput with self-configuring control and token bucket at each stage with irregular input rate scenario.

that the processing rates for ds_1 at the first stage is reduced to 5 tokens/s between time 120s and 240s. Resources are provided to injected tokens assuming all tokens require the same processing time. However, in this new scenario, tokens coming from ds_1 require more processing time. The graph shows how the throughput of ds_1 falls to 20 token/s at time 120s, and how this variation affects the other applications sharing the resources.

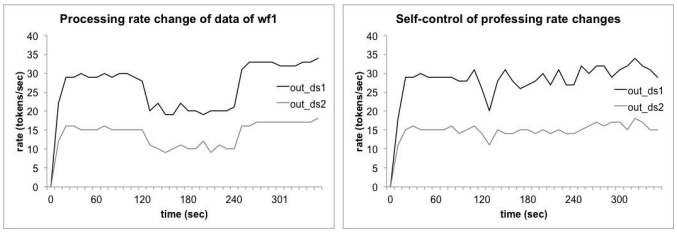

Fig. 7. Throughput with self-configuring control and a token bucket at each stage.

The right part of Figure 7 now shows the output rates of ds_1 and ds_2 with the a resource provisioning mechanism that takes into account the variation of data processing rates. The control loop triggers the addition of new resources when the difference of input and output rates of ds_1 in the first stage exceeds a threshold. Without loss of generality, we assume that the penalty is larger than the additional resource cost. In this case, the control loop adds two additional resources to the first stage. At time 240s ds_1 recovers its initial data processing rate of 10 tokens/s. The control loop returns resources when the number of resources show more capacity than the input rate. We can also see that although the fault tolerance mechanism does affects the throughput of the other workflows, the impact here is much smaller and the mean throughput is maintained. The figures also show that the processing rate after this variation is a little over the input rate because of the tokens that were built up in the buffer of the PU.

V. Related Work

Research in Data Stream Management Systems (DSMS) and Complex Event Processing (CEP) has evolved separately, despite the fact that both communities share a number of important similarities and challenges such as scalability, fault tolerance and performance. Data Stream Management Systems (DSMS) shifted the paradigm of Database Management Systems as the need for efficiently processing streamed datasets in real-time or nearly in real-time arose. In general, DSMSs focus on performance by restricting the language in which they can be programmed to graphs of operators with well-defined semantics [11]. This allows these systems to automatically rewrite or compile the specified stream pipelines to a more efficient version. Scalability and query distribution were considered in Aurora [12], Borealis [13] and Stream Cloud [14]. Additionally, QoS support in DSMS has been a critical requirement [11] and various scheduling strategies and heuristics have been developed. When data streams arrives at an expected rate with low variability, near optimal scheduling strategies have shown to enforce QoS for multiple data streams successfully. However, if the data streams arrive with unpredictable and variable bursts, the scheduling heuristics comprise a combination of strategies that may not always show satisfactory QoS enforcement [11]. These systems therefore employ load shedding strategies – i.e. discarding of data elements from a stream when the loss of some data elements is acceptable. In our approach, we rely on estimations of average input rates for data streams, a token bucket model for regulating data access to the computational resources, on the elastic Processing Unit Component (increase / decrease of the number of computational resources) and on the autonomous behavior of each node.

DSMSs have little or no support to express events as the outcome of continuous queries nor further support to form complex events. CEP has seen a resurgence in the last few years, though the need for events, rules, and triggers was accomplished more than two decades ago. Examples of CEP are SPADE/IBM InfoSphere Streams, Esper and DROOLS Fusion [15]. The main difference between existing CEP and stream processing systems is that in the former each event is processed at arrival time, while stream processing systems involve accumulating a data set over a time (or count – i.e. when a certain number of events have arrived) window and processing it at once. Event processing systems assume that the incoming events are not bursty and do not generally consider the presence of queues/buffers between event operators. Additionally, most CEP have little support for QoS requirements, except for the MavEStream system [11] that integrates CEP into a QoS-driven DSMS system. Again, MavEStream system enforces QoS by complex scheduling heuristics that may not always perform suitably under bursty conditions. Additionally, this lack of QoS support in CEP has also been recently considered in the literature: in [16] several micro-benchmarks were proposed to compare different CEP engines and assess their scalability with respect to a number of queries. Various ways of evaluating their ability to changes in load conditions were also discussed. Regarding scalability, which has not received much attention in CEP, some event processing languages extend production rules to support event processing

and provide run-time optimizations by extending the Rete [17] or Treat [18] algorithms to scale with the number of queries. In [19] the importance of scalability for CEP is recognised and event processing is partitioned into a number of stages. At each stage resources can process all of the incoming events in parallel under peak input event traffic conditions. However this approach does not provide a run-time adaptation, does not involve queues and buffers and assumes the best-effort strategy (as is usual in CEP).

VI. Conclusions

EV charging imposes a load on electricity distribution networks which are already operated close to their loading limit. Hence, a more efficient management of the energy will be required and electricity distribution and generation infrastructures will be turning into *Smart Grids*. This involves demand forecast methods, state estimation techniques and real-time monitoring, leading to communication and control of residential and city areas taking place at a fine granularity. Identifying charging schedules for EVs that take into account user demand, electricity grid capacity and cost considerations can often require dealing with large-scale data in real-time, from multiple distributed third-party sources. When consider finer grained analysis of the sources of energy consumption in the EV context, it is necessary to obtain data from the vehicles along with data from charging points.

We analyse the computational requirements for EV electricity demand forecasting and propose a system model and an architecture. Our model can support different sources of information, such as data streams coming from sensors or historical data (i.e. past load demand). Our focus is on EVs used by individuals in a residential/city context, and not on vehicle fleets. We envision that information needed to support EV demand forecasting will consist of (i) heterogeneous data formats & transmission rates; (ii) irregular/bursty data streams; and (iii) variable, difficult to predict processing requirements per data stream. The processing of these data streams may involve filtering, data correlation and trends analysis. The underlying network and computational architecture must therefore support Quality of Service (QoS) constraints to be observed to support the generation of timely forecasts. We propose a distributed computing architecture and a model for supporting EV demand forecasting, consisting of autonomous stages, comprising a combination of data access and regulation, computation, data transfer capability and a rule-based controller component. We envisage that these kinds of applications will certainly grow given the trend on smart cities and other near-real time surveillance applications, especially involving stream analysis and correlation from a variety of different data sources. The proposed distributed computing architecture, along with the model that consists of a number of autonomous stages, are fully compatible with a multi-tenancy, shared Cloud environment – enabling multiple data streams to be processed using the same elastic infrastructure.

References

[1] BERR and DfT, "Investigation into the scope for the transport sector to switch to electric vehicles and plug-in hybrid vehicles," Available at: http://www.bis.gov.uk/files/file48653.pdf, Tech. Rep., February 2008.

[2] F. McMorrin, R. Anderson, I. Featherstone, and C. Watson, "Plugged-in fleets: A guide to deploying electric vehicles (EVs) in fleets," The Climate Group, Tech. Rep., February 2012.

[3] P. Papadopoulos, "Integration of electric vehicles intro distribution networks," Ph.D. dissertation, Cardiff University, 2012.

[4] P. Papadopoulos, O. Akizu, L. Cipcigan, N. Jenkins, and E. Zabala, "Electricity demand with electric cars in 2030: comparing Great Britain and Spain," *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy*, vol. 225, no. 5, pp. 551–566, 2011.

[5] R. Tolosana-Calasanz, J. A. Bañares, and O. F. Rana, "Autonomic streaming pipeline for scientific workflows," *Concurr. Comput. : Pract. Exper.*, vol. 23, no. 16, pp. 1868–1892, 2011.

[6] R. Tolosana-Calasanz, J. A. Bañares, C. Pham, and O. F. Rana, "End-to-end qos on shared clouds for highly dynamic, large-scale sensing data streams," in *To appear in Proc. of 1st International Workshop on Data-intensive Process Management in Large-Scale Sensor Systems (DPMSS 2012): From Sensor Networks to Sensor Clouds*, 2012.

[7] ——, "Enforcing qos in scientific workflow systems enacted over cloud infrastructures," *Journal of Computer and System Sciences*, 2012.

[8] O. Kummer, *Referenznetze*. Berlin: Logos Verlag, 2002.

[9] Y. Serizawa, E. Ohba, and M. Kurono, "Present and future ict infrastructures for a smarter grid in japan," in *Innovative Smart Grid Technologies (ISGT), 2010*, jan. 2010, pp. 1 –5.

[10] S. Ingram, S. Probert, and K. Jackson, "The impact of small scale embedded generation on the operating parameters of distribution networks," UK Department of Trade and Industry (DTI), Tech. Rep., 2003.

[11] S. Chakravarthy and Q. Jiang, *Stream Data Processing: A Quality of Service Perspective Modeling, Scheduling, Load Shedding, and Complex Event Processing*, 1st ed. Springer Publishing Company, Incorporated, 2009.

[12] M. Cherniack, H. Balakrishnan, M. Balazinska, D. Carney, U. Cetintemel, Y. Xing, and S. Zdonik, "Scalable Distributed Stream Processing," in *CIDR 2003 - First Biennial Conference on Innovative Data Systems Research*, Asilomar, CA, January 2003.

[13] D. J. Abadi, Y. Ahmad, M. Balazinska, U. Cetintemel, M. Cherniack, J.-H. Hwang, W. Lindner, A. S. Maskey, A. Rasin, E. Ryvkina, N. Tatbul, Y. Xing, and S. Zdonik, "The Design of the Borealis Stream Processing Engine," in *Second Biennial Conference on Innovative Data Systems Research (CIDR 2005)*, Asilomar, CA, January 2005.

[14] V. Gulisano, R. Jimenez-Peris, M. Patino-Martinez, and P. Valduriez, "Streamcloud: A large scale data streaming system," in *Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on*, june 2010, pp. 126 –137.

[15] O. Etzion and P. Niblett, *Event Processing in Action*, 1st ed. Greenwich, CT, USA: Manning Publications Co., 2010.

[16] M. R. N. Mendes, P. Bizarro, and P. Marques, "A performance study of event processing systems," in *TPCTC*, ser. Lecture Notes in Computer Science, R. O. Nambiar and M. Poess, Eds., vol. 5895. Springer, 2009, pp. 221–236.

[17] C. L. Forgy, "Rete: A fast algorithm for the many pattern/many object pattern match problem," *Artificial Intelligence*, vol. 19, no. 1, pp. 17–37, 1982.

[18] D. P. Miranker, "Treat: A better match algorithm for ai production system matching," in *AAAI*, K. D. Forbus and H. E. Shrobe, Eds. Morgan Kaufmann, 1987, pp. 42–47.

[19] G. T. Lakshmanan, Y. G. Rabinovich, and O. Etzion, "A stratified approach for supporting high throughput event processing applications," in *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*, ser. DEBS '09. New York, NY, USA: ACM, 2009, pp. 5:1–5:12. [Online]. Available: http://doi.acm.org/10.1145/1619258.1619265