Green Middleware

Hans-Arno Jacobsen and Vinod Muthusamy

1 Introduction

To foster innovation, address energy independence, react to global warming, and increase emergency resiliency, governments around the world are promoting efforts to modernize electricity networks by instigating smart grid, e-energy and e-mobility initiatives [5, 6, 3, 4]. A *smart grid* refers to efforts that enhance and extend today's electric power grids with information processing technology to more efficiently use the world's scarcest resource, *energy* [6].

A smart grid is envisioned to deliver electricity from suppliers to consumers guided by information communication technology (ICT) with two-way communications to monitor and control appliances and devices at consumers' homes, save energy and reduce emissions, increase reliability and transparency, and use distributed energy storage to level electricity supply and demand during peak and off-peak periods [6]. Smart grids also envision the integration of renewable and distributed energy resources, such as photovoltaic, wind, hydro, and tidal sources, in pursuit of the above outlined goals [2].

A smart grid overlays the electrical power grid with a massively distributed *information systems infrastructure* for large-scale monitoring and fine-grained control of all elements involved in the energy production and consumption pipeline. These elements can include energy generation, storage, transmission, distribution, and consumption. A smart grid ensures the secure, reliable and cost-effective flow of energy from distributed energy sources to energy consumers.

Hans-Arno Jacobsen
Middleware Systems Research Group (msrg.org), University of Toronto
e-mail: jacobsen@eecg.utoronto.ca

Vinod Muthusamy
Middleware Systems Research Group (msrg.org), University of Toronto
e-mail: vinod@eecg.utoronto.ca

This chapter is published in *Green IT: Technologies and Applications*, Springer, 2011.

It is the distributed information systems infrastructure that is the subject of this chapter. In particular, this chapter aims to highlight some of the research challenges and open problems faced by developers involved in building various information systems elements as well as applications enabled by smart grids. Furthermore, a potential approach to tackle some of the involved issues is presented.

Emerging smart grids will be ultra-large scale systems (ULS) architected as decentralized systems of systems. According to CMU's Software Engineering Institute [16], ULS are systems of unprecedented scale with respect to lines of code, number of interdependencies, number of interacting hardware, software and human elements, and amount of data stored and processed. ULS are characterized by operational and administrative independence, evolutionary development, emergent behaviour, geographic distribution, and heterogeneity.

The design, development and operation of smart grids require information systems abstractions that among others exhibit the following key properties and features:

1. *Decouple* involved systems and components to support the independent evolution and substitutability of sub-systems and components to reduce the impact on overall operational goals and metrics.
2. Similarly, system interactions should be *loosely coupled* to allow for continuous system evolution and support adaptive system behaviour.
3. Support *event processing* to detect and predict emergent behaviour.
4. Allow for *many-to-many* interactions to connect disparate systems.
5. Guarantee *interoperability* to mediate differences.
6. Finally, support *asynchronous operation* to enable performance and scale.

Traditionally, information system abstractions that facilitate the design, the development, the deployment, the operation and the integration of distributed applications in heterogeneous networked environments are referred to as *middleware*. We therefore refer to abstractions used in the development of the information communication technology elements of smart grids, in particular as pertaining to the distributed systems and application development aspects, as *green middleware*[1]. For example, *publish/subscribe* middleware offer many of the above postulated decoupling, loose coupling, many-to-many interaction, and asynchronous operation requirements and enable event processing, such as event detection and dissemination [26, 14, 17, 28, 9, 21, 22, 33, 23, 20].

However, the overall conception of an information systems infrastructure as information technology basis for smart grids is still an open problem. Furthermore, how to best address the above requirements is not know. While this chapter cannot answer these questions, we aim to raise a number of fundamental questions imposed by applications enabled by future e-energy initiatives for future middleware abstractions. From these questions, we will derive interesting research challenges.

This chapter is organized as follows. Section 2 reviews application scenarios that illustrate some of the potentials of smart grids and underline the features required

[1] We acknowledge that the potential scope of *green middleware* could be far larger, also including making the middleware abstractions themselves more energy-aware and -efficient, for example.

from green middleware systems. Section 3 lays out research questions and discusses open issues whose resolution would contribute to the successful, reliable and secure development and operation of smart grids. Section 4 presents the PADRES system, positioning it as one viable solution to address many of the requirements demanded of green middleware. While PADRES has not yet been evaluated in the context of smart grids, we believe it exhibits many characteristics that are of fundamental importance in the support of an information systems infrastructure and of applications for smart grids. Section 5 concludes this chapter with a few recommendations for next steps.

2 Motivating Application Scenarios

In this section, we first review projections for the evolution of energy consumption worldwide to underline the severity of the energy problem to be addressed by smart grids enabled through green middleware. Then, through forward-looking application scenarios, we further motivate the need for green middleware enabling these scenarios in future smart grids as well as illustrate the research challenges that need to be overcome in order to develop green middleware.

In terms of the energy consumption projections, the International Energy Agency (IEA) reports that in many western homes, energy use from Information and Communication Technology (ICT) exceeds that of traditional appliances [7]. This load-shift is due to the aggregate electrical power draw from devices such as flat screen TVs, digital VCRs, computers, home routers, modems, chargers etc. ICT now accounts for almost 42% of energy consumption in western homes and, given a forecasted annual growth rate in consumption of 15%, could consume 40% of the world's electricity by 2030 [32]. Even with the less optimistic growth rate projection of 6% per annum, power consumption would double every decade [7]. Similar figures are reported for commercial data centers that power today's information economy. Data centers are projected to consume 12% of all electricity in the U.S. by 2020 and data center electricity costs tallied up to about $5 billion dollars in 2006 according to findings from Lawrence Berkeley National Labs [31].

Energy consumption policies

The above projections are alarming at best. Of fundamental concern are the development of approaches to help reduce power consumption, to exploit and integrate renewable energy resources into power grids, to incentivise consumers to save energy, and to develop innovative solutions to save energy without much consumer input.

For instance, imagine a thermostat that allows its owner to opt for saving 15% on her electricity bill, either by letting the owner select the margin, or by having the thermostat offer discounts based on analysis of past usage patterns, and controlling

the home in appropriate ways. Owner consumption violations could be taxed at higher prices to account for the extra cost of provisioning surplus energy resources. A similar scenario applies at a larger scale, where a corporation decides to shave off a few percent of its monthly electricity spending by providing high-level policies to its power hungry equipment, office environment or HVAC operations management.

Across entire regions, utilities could make use of these signals to plan power generation schedules, to reduce operating cost of expensive stand-by generation resources, to lower emissions, and to comply with emission standards. Additionally, it is not inconceivable that consumers may wish to express resource preferences, possibly driven by regulatory policies, such as preferences over type or location of energy resources, degree of reliability, price and environment consciousness.

Appliance monitoring

With a smart grid infrastructure in place, applications can be developed to the customer experience beyond simply saving on energy costs.

For example, one often leaves the house, wondering after some time, if the oven had been shut off, or a person suffering from Alzheimer's may inadvertently forget to turn off the oven or other critical appliances. This is both wasteful and dangerous. Critical conditions of this kind could be detected either by monitoring for unusually long durations of oven use, or for certain correlated activities, such as the oven being on but the kitchen lights off (i.e., no one is in the kitchen) or the front door opening (i.e., someone is leaving the house), or no movement in the house for prolonged periods of time (i.e., someone either left the house or is resting.) Possible actions include automatically shutting off the appliance, alerting the person with a recorded warning, or calling a friend or relative.

Energy use pattern detection

A monitoring system could detect suspicious energy use. For example, lights coming on while the household is on vacation, or off at work and school, could indicate a break-in. Conversely, the absence of certain routine activities (such as an elderly person who watches TV or has a shower at the same time every day), may indicate a health issue that deserves attention. Again, the system can report this to the authorities or caregiver. Detecting these cases requires policies be written and/or the system learns about usage patterns and compares current usage with historical trends.

The system could recommend when appliances need maintenance or replacement. For example, based on the current temperature and thermostat settings, the air conditioner should operate within a range of duty cycles or energy use. Anything outside this range could indicate the air conditioner is not operating efficiently and needs to be replaced or repaired. Similarly, sensors placed around the house can detect irregular temperature readings. For example, large temperature differences

among rooms could indicate clogged or leaky vents; and on a more fine-grained level, temperature gradients within a room could suggest worn out window seals.

Aggregate pattern detection

A smart grid offers opportunities to mine the energy use patterns of many consumers. For example, health care professionals could detect disease outbreaks by monitoring deviations in aggregated energy use such as people going to bed earlier (all appliances and lights off), staying home (appliances used during the day), or using more hot water (longer showers).

Similarly, marketers can infer household lifestyles based on changes in energy use. For example, a sudden increase in the number of times lights are turned on for short periods late at night may indicate a new baby in the house. Similarly, certain devices may exhibit unique power fingerprints; for example, a certain brand of personal video recorder may turn itself on at 3 am every morning to update itself. Such knowledge can be used to deliver better targeted advertisements.

We are fully aware that these examples tread on very sensitive privacy issues that are out of the scope of this chapter. As we point out later, however, these concerns can potentially be addressed if the green middleware offered a way to detect such aggregate patterns without revealing personally identifiable information that may violate users' privacy expectations.

Middleware requirements

The examples above illustrate many of the green middleware properties listed in Section 1. For example, loosely coupled interactions among components are required for there to be any reasonable chance of managing such a large ecosystem of power generators, transmission lines, storage nodes, and appliances, all of which are owned, operated, administered, and adapted independently. Many of the above scenarios also rely on real-time monitoring of the status of individual components, a task that calls for some sort of event processing capabilities. Moreover, the above applications can largely be implemented in an asynchronous manner. For example, monitoring appliances for unusual usage only requires that these appliances report their status, perhaps by emitting events; the component that analyzes these event streams can consume and process these events at its own pace. Notably, there is no synchronous, step-by-step conversation among the components, as this would quickly become infeasible as the number of components increases.

These scenarios also reveal certain challenges for a green middleware. For example, it is not clear how to architect a middleware that can support the ultra large scale of a smart grid. Also there are seemingly conflicting requirements, such as the need for both efficiency and resilience in the middleware itself as well as in the generation and consumption of energy; the very redundancy in the system that improves resilience can also be seen as an inefficient use of resources. Furthermore, manag-

ing the components in a smart grid requires detailed knowledge of their behavior, but this knowledge may violate users' privacy regulations. The following section discusses these research challenges in more detail.

3 Research Challenges & Opportunities

The long-term goals of a green middleware roadmap are the development of advancements inspired from the demands of future smart grid information systems infrastructures. These advancements will lead to new standards that enable an open ecosystem for applications offered by third parties that exploit the capabilities offered by smart grids.

In this sense, the objectives are to contribute to the design of an information system infrastructure that supports a distributed energy grid enabling energy consumers and suppliers to negotiate supply and demand in real-time, to perform this negotiation at the device level driven by system-wide constraints, to reduce energy consumption and save cost, and to adhere to regulatory constraints.

The fundamental questions that must be answered are as follows:

1. *What is the architecture of the information systems infrastructure that supports the requirements of future smart grids, especially the ultra-large scale nature of these systems?*
   The emerging integration of distributed energy resources, the envisioned two-way flow of monitoring and control information, and the novel application scenarios present a paradigm shift away from the existing centrally managed and controlled, proprietary and closed, and one-way communication-based energy infrastructures of today.
2. *How to balance efficiency and resiliency to accommodate disturbances in ultra-large scale systems?*
   The lack of resiliency, for the benefit of efficiency and cost reduction, is often detrimental, especially when manifest as cascading failures such as blackouts.
3. *How to determine the aggregate system behaviour without needing to reveal individual behaviour?*
   For example, in order to prevent cascading failures, it is critically important to be able to detect and predict emergent behaviour and to react accordingly. In the context of smart grids, where transparency is paramount, this question becomes one of knowing how consumers act collectively without needing to know how they are acting individually. In other words, how can the system detect emergent global behaviour while preserving consumer privacy?

These questions lead to a number of interesting research challenges:

- *Designing the information systems infrastructure for the expected scale of operation.*

This scale amounts to millions of households, similar amount of electric cars as potential energy storage buffers, dozens of devices per household, and hundreds of thousand of distributed energy resources. The expected load amounts to millions of events per second, and the expected communication patterns constitute asymmetric event flows with lots of monitoring and a disproportionally smaller amount of control data.

- *Enabling real-time visibility into an ultra-large scale system architected as a system-of-systems.*
  In existing infrastructures, the lack of visibility is often used as an argument to fend off customer complaints about large charges. However, visibility and transparency in smart grids are essential to their success.
- *Ensuring fine-grained security constraints, including privacy and access control.*
  In ultra-large scale systems, different entities will inevitably need different views of information. In the context of smart grids, for instance, the billing department only needs to know the aggregate energy consumption per household over a large time window, and need not be concerned with how the energy was used. On the other hand, energy providers do not need to know who uses the energy but require fine-grained knowledge of usage patterns. Each consumer, however, should have immediate access to all information including per-device energy usage, ideally in real-time.
- *Designing an information system infrastructure that can accommodate growth and evolution over decades.*
  The scale of operation mandates provisions for gradual roll-out over time. Similarly, the use of open standards is needed to enable competition and choice.
- *Unifying information routing and storage, which are traditionally handled separately.*
  The content-based routing paradigm supports the routing and propagation of short-lived information, while content-centric networking addresses the efficient storing and querying of long-lived data in large-scale distributed environments. Both models are content-based, as opposed to the predominant addressed-based model of today's networks. Besides the benefits of increased privacy, logical addressing is a good abstraction for mobile entities. In both cases, there is a need to filter and aggregate information for privacy and efficiency.

4 Towards meeting the research challenges

Many of the challenges of a green middleware for smart grids arise from the need to support complex interactions among the many components in the system, including the enormous number of power generating sources, energy storage sites, transmission lines, consumer appliances, and plug-in electric vehicles.

The publish/subscribe interaction model offers the loose coupling and flexible many-to-many communication requirements of a green middleware. Moreover, implementations of the publish/subscribe model have been designed to provide scal-

able complex event processing that are also necessary for smart grids. This section provides an overview of the PADRES system, a open-source distributed publish/subscribe platform.

4.1 Publish/Subscribe

The publish/subscribe (pub/sub) paradigm provides a simple and effective method for disseminating data while maintaining a clean decoupling of data sources and sinks [14, 22]. This decoupling can enable the design of large, distributed, and loosely coupled systems that interoperate through simple publish and subscribe invocations. A large variety of emerging applications benefit from the expressiveness, the filtering, the distributed event correlation, and the complex event processing capabilities of content-based publish/subscribe.

These applications include RSS feed filtering [27, 29], system and network management, monitoring, and discovery [8, 15, 34], business process management and execution [30, 25], business activity monitoring [15], workflow management [13, 25], and automatic service composition [19, 18].

Typically, content-based publish/subscribe systems are built as application-level overlays of content-based publish/subscribe brokers, with publishing data sources and subscribing data sinks connecting to the broker overlay as clients. In content-based publish/subscribe, message routing decisions are based on evaluating subscriptions over the content of a message, and are not based on IP-address information. Events relevant to applications are conveyed as publications to the publish/subscribe system and routed based on their content to interested subscribers.

4.2 The **PADRES** *publish/subscribe system*

PADRES is a distributed content-based publish/subscribe system, developed by the Middleware Systems Research Group at the University of Toronto [1].

Figure 1 depicts the major layers in the PADRES system as they relate to a smart grid. At the bottom physical layer, around the core compute resources such as routers and compute servers, are entities such as households, consumer appliances, the electricity transmission infrastructure, and power generating and storage devices. The compute resources are used to build a distributed overlay network of publish/subscribe brokers, and the remaining entities participate as clients that interact with one another through the publish/subscribe broker network. Finally, the top layer includes the application logic that processes the event flows in the smart grid. The details of these layers will become clearer by the end of this section.

PADRES provides a number of novel features including composite subscriptions, composite event detection, historic query capability, load balancing, fault detection and repair, and monitoring support.

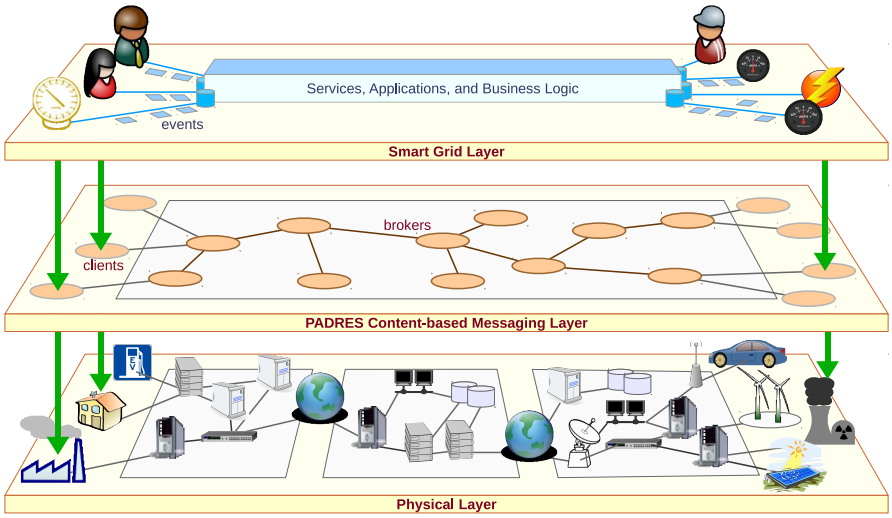

Fig. 1 PADRES as information systems infrastructure for smart grids

The historic data access module allows clients to subscribe to both future and historic publications [24]. This may be used to detect if the future energy consumption habits of a user deviate from their past. The load balancing and client relocation modules handle the scenarios in which a broker is overloaded by a large number of publishers or subscribers [11, 12, 19]. Such a feature would alleviate the need for a system administrator to manually reallocate resources as new energy producers or consumers enter the system. The fault tolerance module detects failures in the publish/subscribe layer and initiates failure recovery [20]. In the smart grid can continue to operate despite isolated faults in the system. A monitoring module, which is an administrative client in PADRES, can be used to visually display the broker network topology, trace messages, and measure the performance of the network. These facilities can be used to inspect, at a fine-grained level, the operation of the green middleware itself or the devices in the smart grid,

4.3 The PADRES *language and data model*

PADRES provides a SQL-like syntax PADRES SQL (PSQL) [24], which allows users to uniformly access data produced in the past and the future. The PSQL language includes the specification of notification semantics, and it can filter, aggregate, correlate and project any combination of historic and future data as described below. In PADRES, a message has a message header and a message body. The header includes the message identifier, which is unique throughout the system, the last hop and next hop information, which indicates where the message comes from and where it will be forwarded to, and the time stamp when the message is generated. There could be

four types of objects in the message body: publications, advertisements, subscriptions, and notifications.

Advertisements: Before each publisher issues its publications, it specifies a template that describes the publications it will publish. This is done by issuing an *advertisement* message. The advertisement is an indication of the data that the publisher is going to publish in the future. In this sense, an advertisement is like a database schema or a programming language type. An advertisement is said to *induce* publications. That means the attribute set of an *induced* publication is a subset of attributes defined in the advertisement, and values of each attribute in an induced publication must satisfy the predicate constraint defined in the advertisement. Note that only publications induced from an advertisement of a publisher are allowed to be published by the publisher. We adapt the equivalent SQL table creation statement to express advertisements.

```
CREATE TABLE (attr op val[, attr op val]*)
```

PSQL's `CREATE TABLE` differs slightly from the same statement in SQL. Tables are unnamed since they need not be referred to by subscriptions or publications. Also, the range of values of each attribute (or column) can be specified. Moreover, regardless of the attribute value constraint, each attribute can implicitly be a null value.

In the rest of this section, we use an energy consumption management example consisting of a thermostat that periodically emit events indicating the duty cycle of the air conditioners under its control, and an external weather service that reports on current weather conditions.

```
CREATE TABLE (class = thermostat, id = *, time = *
              ac_model = *, duty_cycle = *)
CREATE TABLE (class = weather, id = *,
              time = *, location = *,
              degrees = *, humidity = *)
```

In the above example, `*` is a wildcard that indicates that the corresponding attribute may have any value.

Publications: A publication is expressed using a construct similar to SQL's `INSERT` statement.

```
INSERT (attr[, attr]*) VALUES (val[, val]*)
```

The following publication is compatible with the advertisement schema defined above.

```
INSERT (class, time, duty_cycle)
  VALUES (thermostat, 9:00, 20%)
```

Notice that only a subset of attributes defined in the schema need to be specified. For example, the above publication does not include personally the identifiable attributes such as the user id and air conditioner model information.

A publication may also contain a *payload*, which is an optional data value delivered to subscribers. The payload cannot be referenced by a subscription's constraints.

In many applications, publications represent events. Often, both terms are used synonymously in the publish/subscribe literature.

Subscriptions: Subscribers express interests in receiving publications by issuing *subscriptions*. Subscriptions set constraints on matching publications. PADRES not only allows subscribers to subscribe to individual publications, but also allows correlations or joins across publications. In that sense, subscriptions can be classified into *atomic subscriptions* and *composite subscriptions*.

Subscribers issue `SELECT` statements to query both historic and future publications. With `SELECT`, a subscriber can specify a set of attributes or functions that she wants to receive once the subscription is matched. The `WHERE` clause indicates the predicate constraints applied to matching publications. The `FROM` and `HAVING` clauses are optional and are used to express joins and aggregations.

```
SELECT [ attr | function ], ...
 [FROM src, ...]
 WHERE attr op val, ...
[HAVING function, ...]
[GROUP BY attr, ...]
```

A traditional publish/subscribe subscription for future publications would look as follows in PSQL.

```
SELECT *
 WHERE class = thermostat, duty_cycle > 50%
```

Note that the above statement does not query a single table, so the results may have any number of attributes. The only guarantee is that all notifications will have the *class* and the *duty_cycle* attributes with values constrained as specified.

Reserved attributes *start_time* and *end_time* specify time constraints, and are used to query for publications from the past, the future, or both. For example, after replacing one of her air conditioners, a landlord may want monitor how it is performing. The following subscription queries data in a time window that begins two days before the time the query is issued and extends into the future.

```
SELECT *
 WHERE class = thermostat, ac_model = ACME3000,
       start_time = NOW - 2 days,
       end_time = NOW + 1 week
```

The system internally splits the above subscription: one purely historic subscription that is evaluated once, and one ongoing future subscription. A subscription for both historic and future data is a *hybrid* subscription.

Publish/Subscribe composite subscriptions [22] can be expressed with simple join conditions. The event correlation is supported using the `FROM` clause, where the event pattern can be specified using Boolean expressions. For instance, the subscription below monitors how air conditioners that have historically exhibited inefficient duty cycles perform during hot days in the future. In this case, the subscription is both a hybrid and composite subscription.

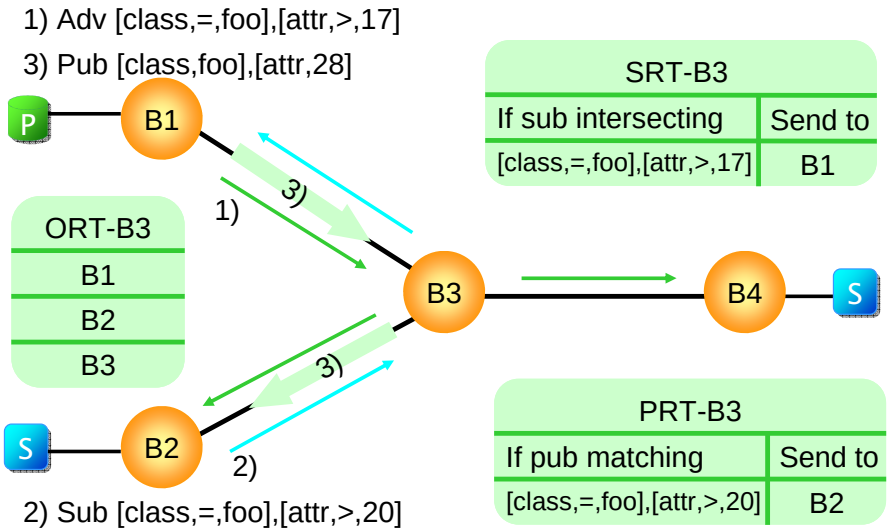

Fig. 2 PADRES Broker Network.

```
SELECT e2.time, e2.duty_cycle, e3.degrees
  FROM e1 AND e2 AND e3
 WHERE e1.class = thermostat,
       e1.duty_cycle > 90%,
       e1.start_time = NOW - 2 months
       e2.class = thermostat,
       e2.id = e1.id,
       e3.class = weather
       e3.location = Paris,
       e3.degrees > 30,
       e3.time = e2.time
```

The identifier in the `FROM` clause specifies that three different publications are required to satisfy this query, and each publication must match the `WHERE` constraints. The three publications may come from different publishers, and may conform to different schemas, as long as they match the specified constraints.

The subscription above queries air conditioner duty cycles over the past two months (event `e1`), correlates these with both duty cycle readings (event `e2`) and weather reports (event `e3`) in the future, and reports the performance of historically inefficient air conditioners during future warm weather situations. Manufacturers can use this information to study the behaviour of their poorly performing products.

Notice that a composite subscription can collect, correlate, and filter publications in the event processing network. Without this feature, a user must retrieve all future publications and then query databases for associated historic data. This would be expensive (both for the user and in terms of network traffic) in cases where the future publications are generated frequently.

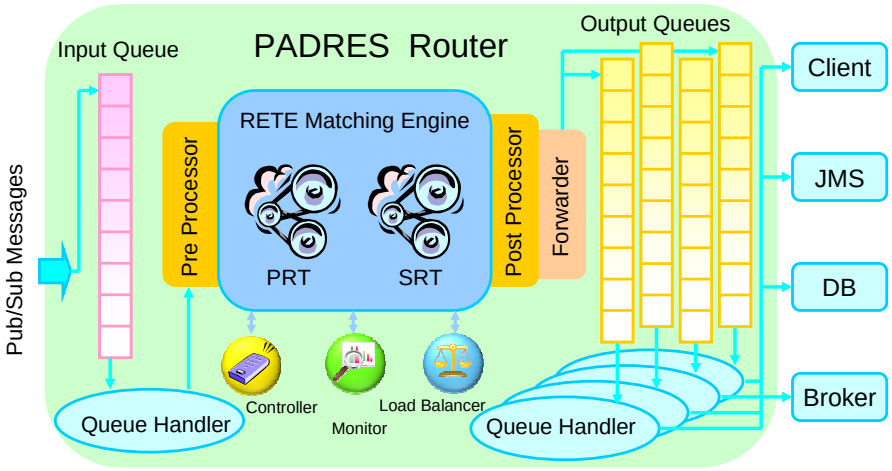

Fig. 3 PADRES Broker Architecture.

Event aggregation is supported in PSQL as well. The `HAVING` clause can specify constraints across a set of matching publications. The functions `AVG(`a_i`,N)`, `MAX(`a_i`,N)`, and `MIN(`a_i`,N)` compute the appropriate aggregation across attribute a_i in a window of N matching publications. The window may either slide over matching publications, or be reset when the `HAVING` constraints are satisfied. For example, the following subscription matches if the average duty cycle across 10 readings of a particular air conditioner exceeds maximum fault 80%.

```
SELECT *
 WHERE class = thermostat
HAVING AVG(duty_cycle, 10) > 80%
GROUP BY id
```

Any attributes specified by functions in the `HAVING` clause must appear in the publication. So, an implicit `duty_cycle = *` condition is added to the `WHERE` clause above. Also, the `GROUP BY` clause has the same semantics as in SQL and serves to constrain the set of publications over which the `HAVING` clause operates.

Although advertisements have the same format as atomic subscriptions, they have different semantics. Matching publications of the subscription must have all the attributes specified in the subscription, while publications induced from an advertisement may have only subset of the attributes defined in the advertisement. Another difference is a subscription may specify the notification semantics (e.g., in PADRES SQL). That is, what information(e.g., a subset of attributes) should be delivered to the subscriber if there is a match. A subscription *intersects* a advertisement if the sets of publications matching the advertisement and the subscription intersect.

Notifications: When a publication matches a subscription at a broker, a *notification* message is generated and further forwarded into the broker network until it is delivered to subscribers. Notification semantics do not constrain notifica-

tion results, but transform them. Notifications may include a subset of attributes in matching publications indicated in the `SELECT` clause in PSQL. Most existing publish/subscribe systems use matching publication messages as notifications. PSQL supports projections and aggregations over matching publications to simplify notifications delivered to subscribers and reduce overhead by eliminating unnecessary information.

4.4 The PADRES *broker overlay*

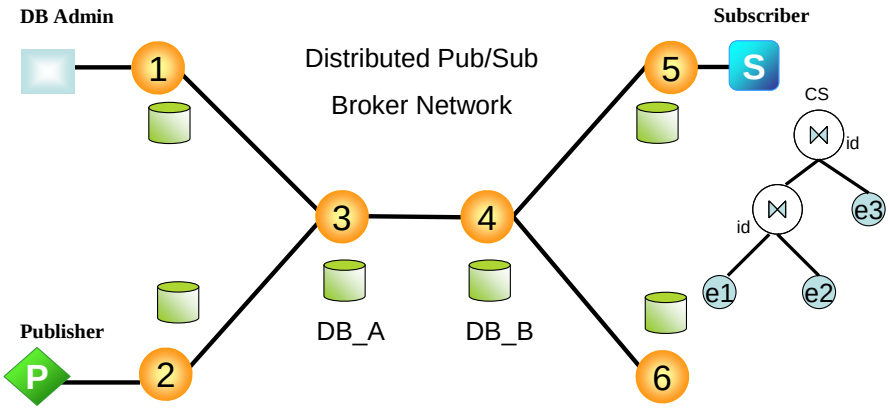

Fig. 4 Historic Data Access.

The PADRES system consists of a set of brokers connected in an overlay network, as shown in Fig. 2. The overlay network forms the basis for message routing and event processing. Each PADRES broker acts as a content-based message router that routes and matches publish/subscribe messages. Each PADRES broker is essentially an event processing engine. The PADRES overlay constitutes an event processing network that can filter events published by many sources, distribute events to many subscribers, correlate and aggregate events from multiple sources to detect composite events, match composite subscriptions and match subscriptions for future events, historic events, and combinations of future and historic events [22, 23, 24].

A PADRES broker only knows its direct neighbours. The overlay information is stored in the Overlay Routing Tables (ORT) at each broker. Clients connect to brokers using various binding interfaces such as Java Remote Method Invocation (RMI) and Java Messaging Service (JMS). Publishers and subscribers are clients to the overlay. A publisher issues an advertisement before it publishes. Advertisements are effectively flooded to all brokers along the overlay network. A subscriber may subscribe at any time. The subscriptions are processed according to the Subscription Routing Table (SRT), which is built based on the advertisements. The SRT

is essentially a list of [advertisement,last hop] tuples. If a subscription intersects an advertisement in the SRT, it will be forwarded to the last hop broker the advertisement came from. Subscriptions are routed hop by hop to the publisher, who advertises information of interest to the subscriber. Subscriptions are used to construct the Publication Routing Table (PRT). Like the SRT, the PRT is logically a list of [subscription,last hop] tuples, which is used to route publications. If a publication matches a subscription in the PRT, it is forwarded to the last hop broker of that subscription until it reaches the subscriber. A diagram showing the overlay network, SRT and PRT is provided in Fig. 2. In this figure in Step *1)* an advertisement is published at B_1. In Step *2)* a matching subscription enters from B_2. Since the subscription overlaps the advertisement at broker B_3, it is sent to B_1. In Step *3)* a publication is routed along the path established by the subscription to B_2.

Each broker consists of an input queue, a router, and a set of output queues, as shown in Fig. 3. A message arrives in the input queue. The router takes the message from the input queue, matches it against existing messages according to the message type, and puts it in the proper output queues representing different destinations. Other components are provided to support advanced features. For example, the controller component provides an interface for a system administrator to manipulate a broker (e.g., shut down a broker, inject a message into a broker etc.); the monitor component collects operational statistics (e.g., incoming message rate, average queueing time, and matching time etc.). If a broker is overloaded (e.g., the incoming message rate is above a certain threshold), a load balancer triggers offload algorithms [11] to balance the traffic among brokers. A failure detector is monitors the broker network. If a failure is detected, a recovery procedure is triggered in order to guarantee message delivery in presence of failures [20]. PADRES also proposes a novel policy model and framework for content-based publish/subscribe systems that benefits from the scalability and expressiveness of existing content-based publish/subscribe matching algorithms [33].

In PADRES, we explore optimized content-based routing protocols to provide more efficient and robust message delivery for the content-based publish/subscribe model, including covering-based routing [21] and adaptive content-based routing in cyclic overlays [23]. The goal of covering-based routing is to guarantee a compact routing table without information loss, so that the performance of a matching algorithm can be improved based on the concise routing table and no redundant information is forwarded into the network. The adaptive routing protocol takes advantage of multiple paths available in a cyclic network, balances publication traffic among alternative paths, and provide more robust message delivery.

4.5 Historic data access architecture

Subscriptions for future publications are routed and handled as usual [14, 10, 13, 26, 23]. To support historic subscriptions, databases are attached to a subset of brokers as shown in Fig. 4. The databases are provisioned to sink a specified subset

of publications, and to later respond to queries. The set of possible publications, as determined by the advertisements in the system, is partitioned and these partitions assigned to the databases. A partition may be assigned to multiple databases to achieve replication, and multiple partitions may be assigned to the same database if database consolidation is desired. Partition assignments can be modified at any time, and replicas will synchronize among themselves. The only constraint is that each partition be assigned to at least one database so no publications are lost. Partitioning algorithms, partition selection and partition assignment policies are described and evaluated in [24].

Each database subscribes to `DB_CONTROL` publications addressed to it, and the administrator assigns partitions to databases by sending publications with `STORE` commands to the appropriate database. For example, the following publications assigns to database *DB_A* a partition of weather reports for a particular city.

```
INSERT (class, command, db, partition_spec)
VALUES (DB_CONTROL, STORE, DB_A,
        'SELECT * WHERE class = weather, id = *,
         time = *, location = Paris,
         degrees = *, humidity = *')
```

The partition specification is itself a subscription with the selection formula expressed in the `WHERE` clause. A database that receives the `STORE` command will extract the partition specification and issue it as an ordinary future subscription. Matching publications will then be delivered to the database which will store them.

When the first broker receives a historic subscription issued by a subscriber, it assigns it a unique query identifier and then routes the subscription as usual towards publishers whose advertisements intersect the subscription. This ensures that the subscription will arrive at databases whose partitions intersect the subscription. The database(s) convert the subscription into a SQL query, retrieve matching publications from the database, and publish the results. These "historic" publications are annotated with the subscription's unique query identifier so they are only delivered to the requesting subscriber. After the result set has been published, the database will issue an `END` publication, which is used to *unsubscribe* the historic subscription.

The interaction with the databases fully leverages the content-based publish/subscribe model, and the databases are never addressed directly. In fact, it is impossible for publishers to discover where their publications are being stored, or for subscribers to know which databases process their queries. This simplifies management since databases can be moved, added or removed, and partitions reassigned at will.

To improve availability, fault-tolerance and query performance, a partition may be replicated. Partition assignment strategies include partitioning, partial replication and full replication.

With partitioning, a database may be assigned several partitions, but each partition is assigned to only one database. That is, there is only one replica per partition. With partial replication, a given partition may be replicated by assigning it to multiple databases. With full replication, every database maintains replicas of all partitions. That is, each database stores all publications.

The various strategies have tradeoffs and are appropriate under different circumstances. The partitioning policy is simple and avoids replica consistency issues, but is sensitive to failures. Partial replication can tolerate failures of all but one replica, but requires logic to ensure the historic subscription is answered by only one of the replicas. Full replication is even more robust, and historic subscriptions can always be answered fully by the nearest database, minimizing network traffic. However, the high degree of replication imposes greater overall traffic and storage costs, as well as larger synchronization overhead. The partition assignment policies allow an administrator to tradeoff storage space, routing complexity, query delay, network traffic, parallelism of queries, and robustness. Detailed discussion, algorithms, and evaluations of these strategies can be found in [24].

4.6 Subscription routing

Subscriptions in PADRES can be *atomic* expressing constraints on single publications, or *composite* expressing correlation constraints over multiple publications.

Atomic subscription routing: When a broker receives an atomic subscription, it checks the *start_time* and *end_time* attributes. A future subscription is forwarded to potential publishers using standard publish/subscribe routing [14, 10, 13, 26, 23]. A hybrid subscription is split into future and historic parts, with the historic subscription routed to potential databases as described next.

For historic subscriptions, a broker determines the set of advertisements that *overlap* the given subscription, and for each partition, selects the database with the minimum routing delay. The subscription is forwarded to only one database per partition to avoid duplicate results. When a database receives a historic subscription, it evaluates it as a database query, and publishes the results as publications to be routed back to the subscriber. Upon receiving an `END` publication, after the final result is published, the subscriber's host broker unsubscribes the historic subscription. This broker also unsubscribes future subscriptions whose *end_time* has expired.

Composite subscription routing: Topology-based composite subscription routing evaluates correlation constraints in the network where the paths from the publishers to the subscribers merge [22]. If a composite subscription correlates with a historic data source and with a publisher, where the former produces more publications, correlation detection would save network traffic if moved closer to the database, thereby filtering potentially unnecessary historic publications earlier in the network. We propose the adaptive composite subscription routing protocol in [24], which determines the locations of event correlation, refereed to as the *join points*, based on a routing cost model. The cost model minimizes the network traffic and the notification routing delay.

When network conditions change, join points may no longer be optimal and should be recomputed. A join point broker periodically evaluates the cost model, and upon finding a broker able to perform detection cheaper than itself, initiates a join point movement. The state transfer from the original join point to the new

one includes routing path information and partial matching states. Each part of the composite subscription should be routed to the proper destinations so routing information is consistent. Publications that partially match composite subscriptions stored at the join point broker must be delivered to the new join point.

5 Conclusions

To address concerns such as rapidly growing electricity demand, energy independence, and global warming, efforts are underway to develop a smart grid infrastructure to generate, transmit, deliver, and consume electricity. Among other benefits, a smart grid could use real-time knowledge of electricity providers and consumers to deliver cheap, reliable, and clean energy. Furthermore, preferences about household energy usage targets, fine-grained control over devices such as air conditioners, and weather pattern information can be used to manage aggregate energy demand, supply, and storage. Leveling the peaks and troughs of the power infrastructure in this way can potentially reduce costs for both energy producers and consumers, and result in a more stable and reliable system.

To carry out any kind of coordinated action, a smart grid requires an incredibly sophisticated information systems infrastructure, or *green middleware*, to monitor, manage, and control the huge number of energy generation, storage, transmission, and consumption devices present in the system. This green middleware architecture must not only handle the unprecedented size and volume of data in these ultra-large scale systems, but also be resilient to attacks, preserve privacy constraints, and support interoperability among existing components as well as those built with unanticipated requirements decades from now.

While we have presented the benefits of a publish/subscribe system for smart grids, the architecture of a green middleware is by no means settled. What is needed is an organization, say a Green Middleware Consortium, to agree on the architecture and standardize the core protocols and interfaces of this green middleware. This body will need to understand and accommodate the requirements and visions of all the key stakeholders in a smart grid, including but not limited to, public entities that manage power transmission and delivery infrastructure, operators of large scale power sources such as nuclear power plants, manufacturers of smaller scale power sources such as wind turbines, consumer electronics manufacturers, private energy brokers, and environmental and governmental organizations.

A green middleware designed according to some of the principles we have advocated in this chapter, such as the decoupling of components, will help manage the complexity of gradually migrating the current power grid to a more distributed smart grid that will continuously evolve over the decades to come. Moreover, standardization of this open information systems architecture would foster innovation at all levels, including the energy producers and consumers, the power generator and device manufacturers, and entirely new players in the smart grid landscape.

It is instructive to consider the example of the Internet. By agreeing on a set of core open standards governing the routing of packets among machines, the Internet has grown and evolved in ways never envisioned by the designers of these standards. It has allowed innovation at every layer from the underlying physical networks, such as fibre optic and wireless transmission, to the applications delivered over these networks, such as telephony and social networking services.

Similarly, in order to lay the foundations for a truly smart grid, it is crucial that we develop an open, standards-based green middleware; a middleware that will serve as the core information platform to connect the energy generation, storage, transmission, and consumption devices of today and of the future; a middleware that will support new business models using intelligent applications to monitor, reason about, and manage these devices; and ultimately a middleware that can deliver a better consumer experience and improved standard of living to everyone.

Acknowledgements

We would like to thank the members of the PADRES team for their insights into designing middleware platforms for large scale, enterprise applications. In particular, we further acknowledge Balasubramaneyam Maniymaran and Guoli Li for their efforts in preparing portions of this document, especially the sections about the PADRES system.

References

1. PADRES web site. http://padres.msrg.org
2. Distributed Generation Wikipedia Entry. http://en.wikipedia.org/wiki/Distributed_generation
3. E-Energy. http://www.e-energy.de/
4. MUTE. http://www.mute-automobile.de/
5. Smart Grid Newsletter IEEE. http://smartgrid.ieee.org/
6. Smart grid Wikipedia Entry. http://en.wikipedia.org/wiki/Smart_grid
7. The IEA – The International Energy Agency. http://www.iea.org/
8. B. Mukherjee L. T. Heberlein, K.L.: Network intrusion detection. IEEE Network **8**(3), 26–41 (1994)
9. Carzaniga, A., Rosenblum, D.S., Wolf, A.L.: Design and evaluation of a wide-area event notification service. ACM Transactions on Computer Systems **19**(3), 332–383 (2001)
10. Carzaniga, A., Wolf, A.L.: Forwarding in a content-based network. In: ACM Special Interest Group on Data Communication (SIGCOMM), pp. 163–174 (2003)
11. Cheung, A., Jacobsen, H.A.: Dynamic load balancing in distributed content-based publish/subscribe. In: ACM/IFIP/USENIX International Middleware Conference, pp. 249–269 (2006)
12. Cheung, A.K.Y., Jacobsen, H.A.: Load balancing content-based publish/subscribe systems. ACM Transactions on Computer Systems **28**, 9:1–9:55 (2010)
13. Cugola, G., Di Nitto, E., Fuggetta, A.: The JEDI event-based infrastructure and its application to the development of the OPSS WFMS. IEEE Transactions on Software Engineering **27**, 827–850 (2001)

14. Fabret, F., Jacobsen, H.A., Llirbat, F., Pereira, J., Ross, K.A., Shasha, D.: Filtering algorithms and implementation for very fast publish/subscribe systems. In: ACM Special Interest Group on Management of Data (SIGMOD), vol. 30, pp. 115–126 (2001)
15. Fawcett, T., Provost, F.: Activity monitoring: Noticing interesting changes in behavior. In: ACM Conference on Knowledge Discovery and Data Mining (SIGKDD), pp. 53–62 (1999)
16. Feiler, P., Gabriel, R.P., Goodenough, J., Linger, R., Longstaff, T., Kazman, R., Klein, M., Northrop, L., Schmidt, D., Sullivan, K., Wallnau, K.: Ultra-large-scale systems (2008)
17. Fiege, L., Mezini, M., Mühl, G., Buchmann, A.P.: Engineering event-based systems with scopes. In: European Conference on Object-Oriented Programming (ECOOP), pp. 309–333 (2002)
18. Hu, S., Muthusamy, V., Li, G., Jacobsen, H.A.: Distributed automatic service composition in large-scale systems. In: ACM/IEEE/IFIP/USENIX International Conference on Distributed Event-Based Systems (DEBS), pp. 233–244 (2008)
19. Hu, S., Muthusamy, V., Li, G., Jacobsen, H.A.: Transactional mobility in distributed content-based publish/subscribe systems. In: IEEE International Conference on Distributed Computing Systems (ICDCS), pp. 101–110 (2009)
20. Kazemzadeh, R.S., Jacobsen, H.A.: Reliable and highly available distributed publish/subscribe service. In: IEEE Symposium on Reliable Distributed Systems (SRDS), pp. 41–50 (2009)
21. Li, G., Hou, S., Jacobsen, H.A.: A unified approach to routing, covering and merging in publish/subscribe systems based on modified binary decision diagrams. In: IEEE International Conference on Distributed Computing Systems (ICDCS), pp. 447–457 (2005)
22. Li, G., Jacobsen, H.A.: Composite subscriptions in content-based publish/subscribe systems. In: ACM/IFIP/USENIX International Middleware Conference, pp. 249–269 (2005)
23. Li, G., Muthusamy, V., Jacobsen, H.A.: Adpative content-based routing in general overlay topologies. In: ACM/IFIP/USENIX International Middleware Conference, pp. 1–21 (2008)
24. Li, G., Muthusamy, V., Jacobsen, H.A.: Subscribing to the past in content-based publish/subscribe. Tech. Rep. CSRG-585, Middleware Systems Research Group, University of Toronto (2008)
25. Li, G., Muthusamy, V., Jacobsen, H.A.: A distributed service-oriented architecture for business process execution. ACM Transactions on the Web (TWEB) **4**, 2:1–2:33 (2010)
26. Opyrchal, L., Astley, M., Auerbach, J., Banavar, G., Strom, R., Sturman, D.: Exploiting IP multicast in content-based publish-subscribe systems. In: IFIP/ACM International Middleware Conference, pp. 185–207 (2000)
27. Petrovic, M., Liu, H., Jacobsen, H.A.: G-ToPSS: fast filtering of graph-based metadata. In: International World Wide Web Conference (WWW), pp. 539–547 (2005)
28. Pietzuch, P.R., Bacon, J.: Hermes: A distributed event-based middleware architecture. In: International Workshop on Distributed Event-Based Systems (DEBS) (2002)
29. Rose, I., Murty, R., Pietzuch, P., Ledlie, J., Roussopoulos, M., Welsh, M.: Cobra: Content-based Filtering and Aggregation of Blogs and RSS Feeds. In: USENIX Symposium on Networked Systems Design and Implementation (NSDI) (2007)
30. Schuler, C., Schuldt, H., Schek, H.J.: Supporting reliable transactional business processes by publish/subscribe techniques. In: International Workshop on Technologies for E-Services (TES), pp. 118–131 (2001)
31. U.S. Environmental Protection Agency ENERGY STAR Program: Report to congress on server and data center energy efficiency public law 109-431 (2007)
32. Webb, M.: Smart 2020: Enabling the low carbon economy in the information age. Tech. rep., The Climate Group on behalf of the Global eSustainability Initiative (GeSI) (2008)
33. Wun, A., Jacobsen, H.A.: A policy management framework for content-based publish/subscribe middleware. In: ACM/IFIP/USENIX International Middleware Conference, pp. 368–388 (2007)
34. Yan, W., Hu, S., Muthusamy, V., Jacobsen, H.A., Zha, L.: Efficient event-based resource discovery. In: ACM International Conference on Distributed Event-Based Systems (DEBS), pp. 19:1–19:12 (2009)