From Kansei to KanseiGenie: Architecture of Federated, Programmable Wireless Sensor Fabrics

Mukundan Sridharan[1], Wenjie Zeng[1], William Leal[1], Xi Ju[2], Rajiv Ramnath[1], Hongwei Zhang[2] and Anish Arora[1]

[1] Dept. of Computer Science and Engg.,
The Ohio State University,
Columbus, OH, 43210, USA
{sridhara,zengw,leal,ramnath,anish}@cse.ohio-state.edu

[2] Dept. of Computer Science,
Wayne State University,
Detroit, MI, 48202, USA
{xiju,hongwei}@wayne.edu

Abstract. This paper deals with challenges in federating wireless sensing fabrics. Federations of this sort are currently being developed in next generation global end-to-end experimentation infrastructures, such as GENI, to support rapid prototyping and hi-fidelity validation of protocols and applications. On one hand, federation should support access to diverse (and potentially provider-specific) wireless sensor resources and, on the other, it should enable users to uniformly task these resources. Instead of more simply basing federation upon a standard description of resources, we propose an architecture where the ontology of resource description can vary across providers, and a mapping of user needs to resources is performed to achieve uniform tasking. We illustrate one realization of this architecture, in terms of our refactoring the Kansei testbed to become the KanseiGenie federated fabric manager, which has full support for programmability, sliceability and federated experimentation over heterogeneous sensing fabrics.

Key words: wireless sensor network, federation, fabrics, resource specification, ontology, experiment specification, GENI, KanseiGenie

1 Introduction

Several edge networking testbeds have been realized during this decade, in part due to the recognition within the networking community that testbeds enable at-scale development and validation of next generation networks. The role of edge networks —and edge networking testbeds— is likely to only increase, given the growth of wireless networks of sensors, vehicles, mobile communicators, and the like. In this paper, we focus our attention on an emergent architecture for next generation Wireless Sensor Network (WSN) testbeds.

WSN Testbeds. Many WSN testbeds are in use today, of which Kansei [5], Motelab [6], Orbit [10], NetEye [7], and PeopleNet [12] are but a few. Experiments are often used to understand and to deal with the complex dynamics and uncertainties of wireless communication and sensing. Two recent usage trends are worth noting: One, experiments are being repeated in multiple testbeds, to learn about (potentially substantial) variability of performance in different backgrounds, radio types, and size scales. And two, a number of experiments involve long running deployments—they often yield long lived sensing services—which in turn implies that testbeds are increasingly hosting concurrent experiments. These trends motivate the emergent requirement that testbeds need to be *federations of programmable fabrics.*

By programmable WSN fabrics, we mean that individual sensor arrays offer not just resources on which programs can be executed, they also provide network abstractions for simplifying WSN application development and operation. Examples include APIs for scheduling tasks, monitoring system health, and in-the-field programming and upgrade of applications, network components, and sensing components. Fabrics can also support and manage the concurrent operation of multiple applications. Figure 1 compares the traditional WSN model with the emerging fabric model of WSNs.

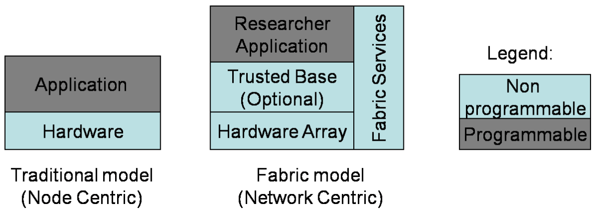

Fig. 1: Traditional model and the fabric model

By federated WSN testbeds, we mean multiple WSN testbeds that are loosely coordinated to support geographically and logically distinct resource sharing. A federation provides users with a convenient, uniform way of discovering and tasking desired WSN resources. Experiments can simultaneously use resources in multiple testbeds, for applications ranging from regression testing, producer-consumer, parallel processing, to enterprise-edge co-operation.

GENI. The Global Environment for Network Innovation project [2] concretely illustrates an architecture where WSN fabrics are a key component. GENI is a next-generation experimental network research infrastructure currently in its prototyping phase. It includes support for control and programming of resources that span facilities with next-generation fiber optics and switches, high-speed routers, city-wide experimental urban radio networks, high-end computational clusters, and sensor grids. It intends to support large numbers of users and large and simultaneous experiments with extensive instrumentation designed to make it easy to collect, analyze, and share real measurements and to test load conditions that match those of current or projected internet usage.

Figure 2 depicts the GENI architecture from a usage perspective. In a nutshell, GENI consists of three entities: Researchers, Clearinghouses and Sites (aka resource aggregates). A Researcher (interacting typically via a specially designed Portal) queries a Clearinghouse for the set of available resources at one or more Sites and requests reservations for those resources that she requires. To run an experiment, she configures the resources allocated to her slice, which is a virtual container for the reserved resource, and controls her slice through well-defined interfaces.

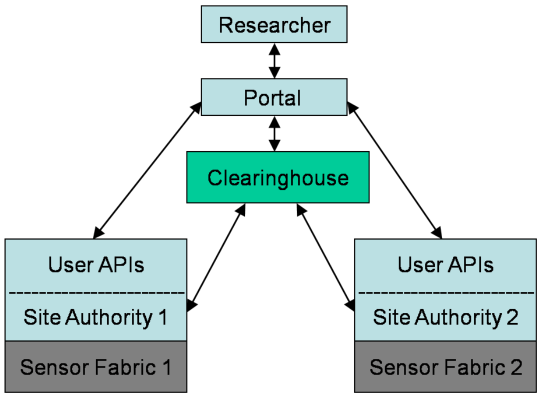

Fig. 2: Federated fabric/GENI model

Researchers and Sites in GENI establish trust relationships and authenticate each other via GENI Clearinghouses. The Clearinghouse keeps track of the authenticated users, resource aggregates, slices, and reservations. Each resource provider may be associated with its own clearinghouse but there are also central GENI Clearinghouses for federated discovery and management of all resources owned by participating organizations. GENI also relies on all entities to describe their underlying resource. Resource descriptions serve as the glue for the three entities because all interactions involve some description of resource, be it a physical resource, such as a router and a cluster, or a logical resource, such as CPU time or wireless frequency.

Overview of the paper. A federation of WSN fabric testbeds needs to address two core issues: One, an efficient and flexible method for resource description, discovery and reservation. And two, a convenient, uniform way of tasking and utilizing federation resources. While standardizing resource descriptions would simplify addressing these two issues, we find that the diversity of sensor characteristics and the lack of a compelling standard model for describing wireless networks complicate the federation of WSN fabrics.

In this paper, we propose a software architecture that we call KanseiGenie for federating WSN fabrics that is compatible with GENI. KanseiGenie is based on the position that, on one hand, different WSN fabric aggregates can advertise resources based on different resource ontologies. On the other hand, users can obtain uniform experimentation support from a portal supporting a given federation. Central to the KanseiGenie architecture is a mapping between a uniform experiment specification and a non-uniform resource specification, which is handled by the portal and/or clearinghouse. Also, in keeping with the fabric model, KanseiGenie offers network abstractions that simplify the programming task of users by letting low-level fabric-specific implementation details to be handled by user services, which are realized by the Sites and/or the Portal.

The rest of the paper is organized as follows: In Section 2, we detail the requirements of each actor in a WSN federation. In Section 3, we discuss Resource Specifications, their role in federation, and the need for and challenges in using multiple resource ontologies in federation and our solution to the challenges. We also outline the need for an Experiment Specification language in federations. Then, we present the KanseiGenie architecture and its implementation in Section 4. We discuss various issues related to the KanseiGenie architecture and alternative designs in Section 5, and make concluding remarks in Section 6.

2 Requirements of Federated WSN Fabrics

As explained in Section 1, the federated WSN fabric model distinguishes three actors: the Site that owns and maintains WSN aggregate resources, the Researcher who deploys/tests applications via a Portal and who need not be a WSN expert, and the Clearinghouse (CH) that enables discovery and manages resource inventory and allocation. In this section, we analyze the requirements of each of these actors. Broadly speaking, these requirements aim to make user experimentation easy, repeatable, verifiable and secure, while maximizing resource utilization.

2.1 Clearinghouse Requirements

A Clearinghouse has two broad functions: One, identification and authentication of various actors in the system (details of this function are beyond the scope of this paper and will not be discussed here). And two, resource management including resource representation, resource discovery and allocation.

Resource Representation. A basic issue for federated WSN fabrics is how to represent a resource in a fashion that will allow multiple Sites with different types of fabrics to publish resources to the same CH, while allowing Sites, Portals, and CHs to evolve over time. The choice of this representation potentially affects all actors: Sites need to advertise the resource, Portals needs to request the resources, and CHs need to match Portal requests to resources available at Sites. Also, CHs may need to communicate with each other for federated resource discovery and allocation. All of the above call for a language that can be used to precisely specify information about the resource.

Note that the need for a resource description language does not mean that the same type of device/network must be defined by all fabrics in a globally unique way. Given the vast heterogeneity of sensor devices, aggregate architectures, fabric service abstractions/semantics, and administrative domains, each fabric may define its resources in a locally unique way. By way of illustration, even the use of IP addressing for WSN devices remains a controversial issue: some fabrics may use this choice while other may not. Likewise, each fabric may choose to associate only locally unique identifiers with devices in its namespace while others might insist on globally unique identifiers.

As is common practice, we refer to a WSN device/network description as a Resource Specification or RSpec. RSpecs tend to be declarative rather than descriptive. In other words, they concisely define what the resource is and eschew details about how the resource is used (that is described in Experiment Specifications, which we discuss later). RSpecs need not necessarily be human-readable because most Researchers are expected to interact with Portals in convenient ways, e.g. with graphical interfaces or library support to help and automate the composition of resource requests.

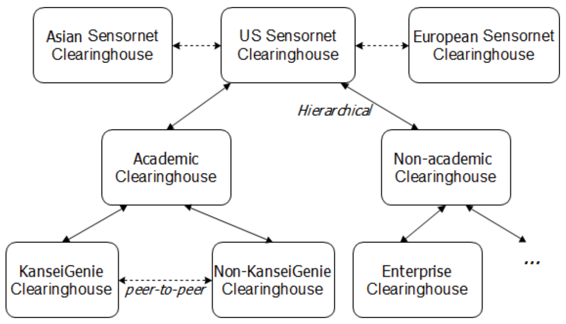
Asian Sensornet Clearinghouse
US Sensornet Clearinghouse
European Sensornet Clearinghouse
Hierarchical
Academic Clearinghouse
Non-academic Clearinghouse
KanseiGenie Clearinghouse
peer-to-peer
Non-KanseiGenie Clearinghouse
Enterprise Clearinghouse
...

Fig. 3: Clearinghouse to Clearinghouse interaction architecture

Resource Discovery. Sites advertise their resources to well-known CHs, so that researchers can discover their resources. CHs of different levels may discover resources directly from the fabric provider through their advertisements or indirectly through other CHs. We envisage a hybrid CH architecture where hierarchical as well as peer-to-peer communication is possible (Figure 3 shows the communications in such a (hypothical) federation). Push and pull models of resource discovery are readily conceived: In the push model, a CH periodically announces to its peers or upper-level CHs the available resources at its associated fabrics that can be shared. In the pull model, a CH requests from its peers or upper-level CHs their latest resource availability. The pull model is likely to be used in an on-demand manner when a CH cannot find enough resources to satisfy a user request.

Resource Allocation. In a federated experiment, a researcher might want to request, via one or more CHs, resources from multiple sites into a slice. It is possible that not all requested resources are available at the same CH, so Portals may have to coordinate request. Broadly speaking, there are two approaches for federated resource allocation depending on whether the CH or the Portal will own the responsibility of getting all of the requested resources: In one, the Portal will directly request the resource from multiple CHs; this approach lacks scalability. In the other, the Portal communicates with a single

CH which in turn communicates with other CHs to get the requested resources. The second approach requires CH-to-CH resource delegation, as shown in Figure 3. KanseiGenie currently implements the first approach, although we envision supporting both approaches in future.

2.2 Site Requirements

Sliceability. In the GENI model of experimentation, each Researcher owns a virtual container, aka a Slice, to which it can deploy/execute experiments and add/remove resources. This view fundamentally decouples the physical location of the resource from its reuse. It follows that all resources leased to a Researcher should be able to communicate with each other and only with each other. Sliceability is also fundamental for federated experimentation. In federated experimentation, a Researcher selects resources from multiple sites and adds them to a federated slice and runs experiments on this federated slice.

Sliceability may be fine-grained. To share memory, processing, or links between slices in a transparent manner, it is necessary to achieve node/network virtualization of resources, which we will describe next. We note that fabric model suits virtualization since it allows users to interact with resources only through well-defines APIs.

Virtualization. The requirement that Sites allow WSN resources to be sliced finely enables multiple slices to co-exist. The challenge in virtualization is to provide as much control to the users (as low in the network stack as is possible) while retaining the ability to share and safely recover the resource.

Virtualization in WSN fabrics is nontrivial. Not only do Sites have to virtualize the hardware, but also the network. Recall that WSN fabrics may span multiple arrays of sensors, and multiple researchers may run their experiments concurrently on subsets of one or more arrays. Usually sensors are densely deployed over space, if only to let sensors share the same geographical space and be subject to similar, if not statistically identical, physical phenomena/environments. Wireless interference between slices is thus an inherent problem due to the broadcast nature of the wireless communications. Virtualization has to thus isolate the communications of an experiment running on a slice, to enable repeatable performance. For instance, channel properties such as signal to noise ratios among wireless nodes may need to be (statistically) similar across repeated experiments.

Programmability. WSN fabrics are expected to provide the hardware and software infrastructure for an end-to-end reprogramming service, which reliably deploys the sensing applications composed by Researchers on the corresponding slice. Sites should also provide monitoring and logging services. In particular, they should also provide feedback to the Researcher about the environment and any failures that occur during programming or execution of an experiment. When Sites are situated in environments that are not representative of sensing phenomena, it is desirable that they provide services for external sensor data injection. Finally, sites and/or Portals should support workflow services, that will allow staging and complex experimentation.

2.3 Portal Requirements

Resource Utilization. To simplify the Researchers task in using federated resources, a Portal needs to provide a uniform resource utilization or experimentation framework. This is challenging since the federation may consist of fabrics with a great variety of available platforms, sensors, radios, operating systems and libraries. For instance, while for some platforms such as XSM [18] and TelosB [17] the default is to program on bare metal, others such as iMote2 [3], Sunspots [16] and Stargates [15] host their own operating system. And the execution environments in these platforms vary from a simple file download and programming the flash, to command line interfaces and virtual machines.

All of above call for an Experiment Specification language that enables Researchers to configure slices in a generic manner. Intuitively, an Experiment Specification should include the resource description that the experiment is to be run on. It also includes a selection of user services that is relevant to the experiment. In addition to these declarative elements, the experiment specification language includes procedural descriptions (or workflow elements). Unlike Resource Specifications where readability is not important, Experiment Specifications should provide good readability because a Researcher might want to script their experiments to iterate through a bunch of test parameters. Also, the Researcher would like to reuse the same experiment specification on different slices which makes the experiments repeatable.

Resource Translation. It is often more convenient for a Researcher to request a networked resource in an abstract manner. For instance, requesting a 5-by-5 connected grid or a linear array of 10 nodes with 90% link delivery radio is much easier than identifying specific sensor devices which match the required topology. Since the resources published at the CHs are specified concretely, a Portal needs to translate the abstract spec to embed it into site resources, although it is possible that this be realized at the CH as well.

In a federated setting where resources are variously represented by different Sites, a service is required that provides a mapping between the Researchers resource need and a resource request that can be processed by different CHs. This service is likely to be implemented at the Portal, if not in a CH. We discuss the motivation for this sort of mapping in more detail in the next section.

3 Specification Languages

Researchers wishing to use WSN fabrics first query the system to discover where/which resources are available. They then select a set of resources and obtain a lease for them. Finally they configure the resources and user services needed to carry out their intended experiment(s). Each of these steps needs a flexible, feature-rich and extensible language to convey the desired goals of the researcher to the system. We refer to the language used to publish, query, request and allocate resources as the resource description language, and the language used to configure resources and script workflow as the experiment specification language.

3.1 Resource Ontologies for Resource Description

As mentioned in Section 2, resources from multiple Sites may well be similar to each other and just as well be different from each other. Resource description in terms of taxonomical flat-schema could hide minor differences by shoe-horning similar resources into the same category. However, building an exhaustive schema for WSNs, which will be conformed to by every Site, Clearinghouse and Portal is both difficult and undesirable. The current GENI proposal [1] mitigates this problem somewhat by proposing that the community agree only on the core of resource description, it leaves the details to domain-specific extensions (to which a number of actors would still need to agree to).

A major drawback of this approach is that it does not capture the relationships or constraints between resources. For example, in a wireless network, while a node and channel might be two separate resources, it might not be possible to allocated just the node and not allocate any part of the channel, or vice versa. It might be better to explicitly capture such dependencies and relationships in the resource specification which would lead to a better allocation of resources. Another example is that in a long running experiment you might want to add new resources to your slice which are *compatible* with your current experiment and resource specifications. Such querying will be extremely difficult to perform if constraints on resources (like software supported) and experiments (like libraries/services used) can be specified .

One approach that redresses this drawback is to represent resources in terms of ontologies, using a resource description language such as OWL/RDF [11] or NDL [8]. Note that with this approach in order for two entities to communicate they only need to agree on the language in which resources are specified and not on the ontology.

Using Multiple Resource Ontologies for WSN Resources. We offer two example arguments (node addressing and network links) for using multiple ontologies in WSN resource description. The first deals with the lack of agreement in specifying, or addressing, sensor nodes. Unlike the core internet, WSNs have not adopted IP as their standard for addressing. Now, there is controversy on this point: 6lowpan [4] uses an IP stack and others [20] have proposed solutions to bridge WSNs to the enterprise IP network either by delegating the address mapping to the gateway. Nevertheless, one cannot assume that such universal addressing scheme can be realized in every WSN fabric; some sensor devices simply do not have enough ROM and RAM to hold the IP stack, even if we can tolerate the communication and computation overhead in these proposed schemes [25].

Along these lines, while a Researcher might have some notion of a “sensor mote” and a “gateway device”, both of these classes could be represented by some Site using a common name such as “node” with different attributes. Conversely, a “sensor mote could be represented by another Site as the composition of a “sensor and a “mote. Multiple ontologies would support these different world views.

Our second argument deals with the difficulty of specifying wireless network topologies. The traditional model of graphs (with nodes and links) is insufficient given the time-and-space variant background noise and interference. A plethora of models, namely the Unit Disk Graph, Dual Disk Graphs, Physical Model of Links, Topology (Ball) Model exist for defining the notion of links, and the choice of which one to offer is subjective.

As a concrete example, consider a Researcher request for a 5-by-5 fully connected sensor network. Figure 4 gives two possible ways of specifying this request for different WSNs. In a geometric model, network connectivity would defined by specifying the transmission power and antenna direction for each node. However, since some sensing devices do not support the control of transmission power and as a result, link sets could be alternatively specified for instance using received signal strength indicator induced by each node at its neighboring receivers.

```
<node id=1, power=10mw, direction=270>
```

(a) Geometric model

```
<node>
  <linkSet>
    <neighbor nodeID=2, rssi=0dbm/>
    ...
  </linkSet>
</node>
```

(b) Link set model

Fig. 4: Two ways of specifying links in sensor networks

Implication of Using Multiple Ontologies. Given the intense debate in the WSN community, forcing Sites to use a single ontology is likely to throttle innovation and will result in a needlessly bulky ontology, which is not easily extensible. Since most Researchers are expected to only interact through a Portal (or two) of their choice, we envision that Portals will serve as the unifying agent for resource specifications. Since all Sites will use the same language for their descriptions, the Portal can combine the different ontologies used by the Site to provide a single ontology to the researchers and perform the necessary translation of Researcher requests. We note that there are several extant techniques and tools to map and align ontologies [13, 23].

3.2 Experiment Specification and Work Flow Control

WSN applications typically run in multiple well defined phases, with each phase involving a possibly different configuration. Another use case for Experiment Specifications is iterative experimentation, where a researcher programs repeated experiments, where the configuration of each depending on the outcome of the previous ones. Moreover, it is also not uncommon for Researchers to selectively specify which libraries/protocols (and sometimes even implementation versions) to use with a particular platform.

Experiment Specifications thus provide Researchers with a flexible and feature-rich way of interacting with resources, rather than just a GUI or a command line interface. They become particularly relevant for future scenarios where applications will primarily involve machine-to-machine, as opposed to human-to-machine, interaction. That they provide a uniform way of configuring experiments in a heterogenous setting does however imply that each Site has to implement the necessary logic. The idea then is to standardize the Experiment Specification language and not the format of interaction, as has been suggested in [24].

4 KanseiGenie

KanseiGenie is a refactoring of the Kansei testbed, to support a GENI federation of geographically separated Sites, each hosting one or more WSN fabric arrays. Its software architecture comprises components and aggregates. Each sensor device is represented as a component that defines a uniform set of interfaces for managing that sensor device. An aggregate contains a set of components of the same type and provides control over the set. (In WSN experiments, Researchers normally interact with a fabric arrays through the aggregate interface rather than individual component interfaces.)

An aggregate also provides other internal APIs needed for inter-component interactions, as called for in the fabric model.

4.1 Architecture and Implementation

In keeping with the GENI architecture, KanseiGenie consists of actors for a Site, a Clearinghouse, and a Portal. The current implementation of federation consists of a Site at The Ohio State University, which has four different sensor fabric arrays, and a Site at Wayne State University, which has two different sensor fabric arrays. The Sites and the Research Portal (which is hosted at Ohio State) run the KanseiGenie software developed at Ohio State. One of the Clearinghouse functions, namely resource management, is implemented using ORCA [9].

KanseiGenie Site. A KanseiGenie Site has four components: Aggregate of Aggregate Manager (AAM), the Web Service Layer (WSL), the individual Component Managers (CM) for each device type, and the Orca Site Authority module.

Aggregate of Aggregate Manager. Given that each fabric array is an aggregate, the KanseiGenie Site Authority (SA) is conceptually an Aggregate of Aggregate Managers that provides access to all the arrays. AAM is responsible for implementing the fabric APIs. AAM provides an AM interface for each sensor array through parameterization. Externally, AAM (i) administers usage of the resource provided by the Site according to local resource management policies, (ii) provides the interface through which the SA advertises its shared resource to one or more authenticated CHs and, (iii) provides a programming interface through which Researcher (via the Portal) can schedule, configure, deploy, monitor and analyze their experiments. Internally, the AAM provides mechanisms for inter-aggregate communications and coordination.

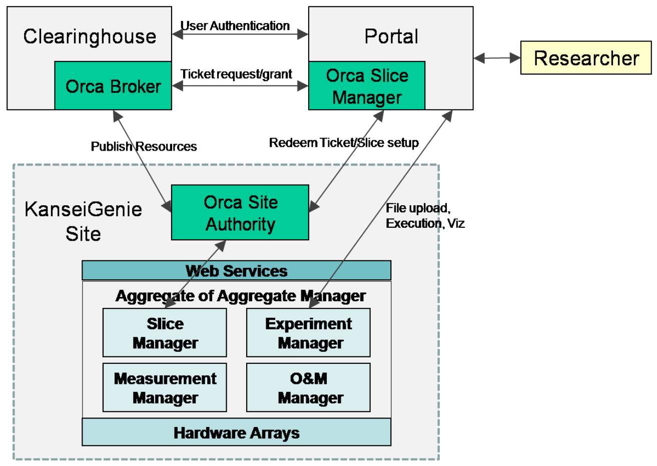

Fig. 5: KanseiGenie Architecture

The fabric APIs provided by an AM are organized into the four functional planes:

– Resource/Slice services: help researchers discover, reserve, and configure resources for experiments.
– Experiment services: provide basic data communication and control between experiments.
– Operation and management services: enable administrators to manage the resources.
– Instrumentation and measurement services: enable the fabric to make measurements of physical phenomena, store the measurements and make them securely available.

Web Service Layer. WSL provides a wrapper for AAM and acts as a single-point external interface for the KanseiGenie SA. The WSL layer provides a programmatic, standards-based interface to the AAM. We utilize the Enterprise Java Bean framework to wrap the four functional GENI planes. Each of the manager interfaces are implemented as a SessionBean. We utilize the JBoss application server as the container for the EJBs partly because JBoss provides mechanisms for users to conveniently expose the interface of Session Beans as web services. Other reasons for choosing JBoss include its community support, wide adoption, open-source licence, and stability.

Component Manager. Each sensor device in KanseiGenie has its own Manager (although for some primitive devices such as motes the Manager is itself implemented on other more capable devices). The Component Manager implements the same APIs as that of AAM and is responsible for executing the APIs on the individual devices. The logical collection of all the Managers of devices belonging to the same device array form the Aggregate Manager of that device. Currently KanseiGenie supports Linux-based PCs/Laptops (Redhat and Ubuntu), Stargates [15], TelosB [17], and XSMs [18]. CMs for Imote2 [3] and SunSpots are under development. All of the CMs are implemented using Perl. A number of tools for user programming of and interaction with motes are written in the Python and C programming languages.

KanseiGenie Portal. The Portal contains a suite of tools for the life cycle of an experiment, ranging from resource reservation to experiment cleanup. It provides an easy interface for experiment specification; at present, this is a user-friendly GUI; a user programmable interface under planning.

The Portal is implemented using the PHP programming language. It also automates tasks for resource specification creation, requesting, and subsequent experiment download. For a federated setting, it serves as an unifying point by mapping dissimilar resource ontologies and by automatically stitching the slices from multiple sites into a single federated slice. Specifically, it uses the Orca Slice Manager, explained below, to reserve resources requested by the Researcher. Once the reservation is done, it interacts with the AAM web interface to configure and run experiments. Of course, a Researcher could directly program against the AAM web interfaces to gain more fine-grained control of experiments, i.e., write his own portal as need be.

KanseiGenie Clearinghouse CH has two main functions:

1. Resource Management: Each Site choose a CH to which it delegates its resources, the CH manages its resources on behalf of the Site and leases these resources to the Researchers. To this end, CH maintains a repository of resources available at each Site and the state of the resources and leases. Even though CH creates the leases, it is up to the Site to honor these leases. Researchers also use CH as central point to discover resources available at the site. In KanseiGenie, this task is delegated to a sub-entity called the Resource Broker implemented by ORCA.
2. Identity, Authentication and Trust: CH is also responsible for the overall security of the system. It authenticates Users, Sites and Portals. A new Site, Portal or Researcher should first contact the CH and get credentials, using which it can communicate with the other system entities. In case of a federation, the CH could implement trust-chaining to authenticate Researchers and Brokers from other domains. KanseiGenie, consistent with the GENI/ORCA effort, plans to use Shibboleth [14] as the Identity and Authentication management software.

ORCA-based Resource Management System. ORCA consists of 3 entities, each one is correspondingly embedded into the three KanseiGenie actors (Portal, Site and Clearinghouse) rExperiment Specificationtively. Collectively, they implement the resource management function.

– ORCA Slice Manager. The Slice Manager interacts with the Researcher and gets the resource request, forwards it to the Broker and gets the lease for the

resources. Once a lease is received, the Slice Manager, forwards it to the Site Authority to redeem the lease.
- ORCA Site Authority. The Site Authority keeps inventory of all the resources that need to be managed. It delegates these resources to one or more Brokers, which in turn lease the resources to Researchers through the Slice Manager.
- ORCA Broker: The Broker keeps track of the resources delegated by various Site Authorities. It receives the resource request from the Slice Manager and if the resource is free, it leases the resource to the Slice Manager. A number of different allocation policies are implemented using a policy plug-in.

To integrate ORCA for resource management, we modified the ORCA Slice Manager to include an XML-RPC server that receives the resource requests from the Portal. Similarly the ORCA SA was suitably modified to make web service calls to the KanseiGenie AAM for experiment setup and tear down. Figure 5 shows this integration architecture.

4.2 Portal-based Federation in KanseiGenie.

Apart from being the single point of access for a Researcher, the Portal plays an important role in KanseiGenie federation architecture. The Portal has three important functions in federation, namely Resource Specification Mapping, Experiment Specification Mapping, and Federated Slice Stitching.

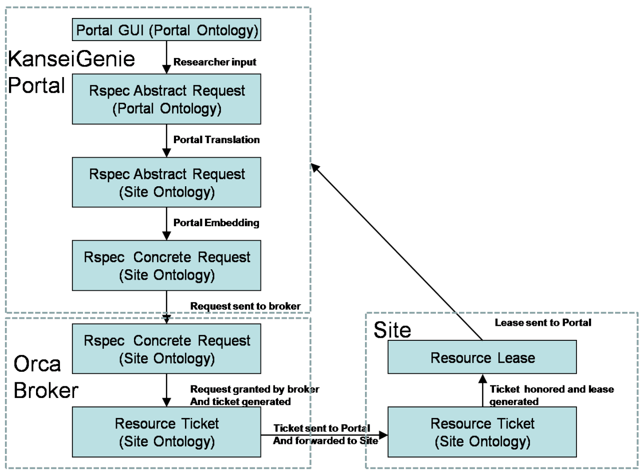
KanseiGenie Portal
Portal GUI (Portal Ontology)
Researcher input
Rspec Abstract Request (Portal Ontology)
Portal Translation
Rspec Abstract Request (Site Ontology)
Portal Embedding
Rspec Concrete Request (Site Ontology)
Request sent to broker
Orca Broker
Rspec Concrete Request (Site Ontology)
Request granted by broker And ticket generated
Resource Ticket (Site Ontology)
Ticket sent to Portal And forwarded to Site
Site
Lease sent to Portal
Resource Lease
Ticket honored and lease generated
Resource Ticket (Site Ontology)

Fig. 6: Resource Specification mapping, translation and lease generation

Resource Specification Mapping. As explained in Section 3 our position is that Sites may use their own Resource Specification dialects (ontologies). To provide a unified experience to the Researchers, we put the onus of interpreting multiple ontologies on the Portal. The Portal discovers resources from multiple sites, understands the Resource Specification ontologies and remaps the different ontologies into a unified ontology at the Portal. The Current research [23, 21] suggest that this remapping of ontologies can be done automatically with very high probabilities for related ontologies. However, we do the mapping manually since it is a one-time operation per Site. In the current implementation, the Portal is also responsible for doing the embedding an abstract Resource Specification into a concrete Resource Specification which we call *Resource Specification Embedding*), although it is possible that this feature could be moved to the Orca Broker in future. The Resource Specification generation, translation, embedding, ticketing and lease generation process is shown in Figure 6.

Experiment Specification Mapping. So that the Researcher can configure experiments in a uniform way, using Experiment Specifications, the Portal maps the experiment configuration onto Site specific experiment manager APIs. Thus, a Researcher may "stitch $fabric_1$ *slice* and $fabric_2$ *slice*", "inject data on $fabric_1$ *slice* from $file_1$", etc., without worrying about the details of $fabric_1$ *slice* and $fabric_2$ *slice*; he may likewise repeat the same experiment on different fabric slices easily. In KanseiGenie, we attempt to use the same APIs for different aggregates whenever possible; nevertheless, when the notion of a service (say for logging) on a Virtual Machine fabric is different from that of a Mote fabric, the complexity of mapping the configuration onto the corresponding APIs is handled by the Portal.

Federated Slice Stitching. When conducting a federated experiment, a Researcher requests a federated slice. She expects seamless communication between the resources in the federated slice, which means that the slices from different Sites needs to be stitched together. We again put the onus of stitching the slices on the Portal. Even though the individual Sites need to provide the services that will allow for the stitching, the Portal possess the knowledge for implementing stitching (such as VLAN numbers, IP addresses, ports, web URLs, etc.). Note that multiple types of stitching might be needed (and possible) depending on the sensing fabrics involved and their capabilities, e.g., it is easy to stitch a federated slice consisting exclusively of virtual machines connected by wired virtual LANs, while it is much harder to stitch two wireless network slices to create a single federated network slices.

5 Discussion

Here, we discuss some of our design decisions taken in realizing KanseiGenie.

What to Federate at Portals? KanseiGenie has thus far chosen the Portal as the main federating agent in the system, including roles for unifying ontologies, embedding Resource Specifications, providing a uniform Experiment Specification, and federated slice stitching. This design suits the view that a Portal realizes *application domain specific support*, and that for different domains, different

Portals may suit. In other words, we view the KanseiGenie Portal as suitable for WSN experiments (and perhaps only some classes of WSN experiments) as opposed to sufficing for all GENI-Researcher needs.

In this view, Clearinghouses are treated as being generic rather than domain specific. Roles which are less domain specific, e.g. embedding Resource Specifications or slice stitching, can be moved from Portals to CHs (or even to Sites) assuming the method of stitching desired is communicated to them. Now, should CHs evolve to become domain specific, they may import more roles from Portals. Taken to the extreme, this would suggest that a top level CH be directly or indirectly capable of unifying all resources in GENI.

Fabric-specific APIs? KanseiGenie leaves the choice and implementation of Aggregate and Component APIs up to the Site. On one hand, each Site Experiment Manager is expected to support standard/generic APIs, on the other, there are cases where Sites need to provide specialized domain-specific interfaces [22] to support certain Portals.

Why not standardize resource specifications? GENI [1] partially standardizes resource specifications and proposes the adoption of a uniform core RSpec for all federated sites. Although this solution might suit a traditional core network that consists of homogeneous virtual machines, it is not equally applicable to WSN fabrics. Roscoe in [24] lays out the drawbacks of going down the standardization path. A key challenge in using a standardized specification is anticipating in advance all of the possible things that must be expressible. He argues, citing the experience with the erstwhile ANSA trading model [19], that the resource request instead should be viewed as a *constraint satisfaction problem* and not a simple database query. This calls for a language in which we can specify constraints, the community need only standardize the specification language and not the format. Thus a decentralized resource specification scheme better supports the inherent heterogeneity in resource and experiment specification in future federated fabrics.

6 Conclusion

In this paper, we described the KanseiGenie software architecture for federated WSN fabrics. We argued a case for letting WSN fabrics choose there own Resource Specification ontologies and showed how a WSN federation can accommodate heterogenous resources. Our implementation resolves the diversity in resource specifications by letting the Portal map the Site specific description to a local ontology with which the Researchers can interact. We also illustrated the need for a Experiment Specification language to enable Researchers to uniformly interact with multiple WSN fabrics in the federation; Experiment Specifications further enable scripted experimentation and complex staging between fabrics. As KanseiGenie grows to accommodate other sites, it remains to be seen whether a need to develop other Portals will emerge or some of the mapping functionality in the Portal will migrate to Clearinghouses.

References

1. GENI Draft: RSpec. http://groups.geni.net/geni/attachment/wiki/GEC2/TF_RSpec.pdf.
2. GENI: Global environment for network innovation. http://www.geni.net.
3. Imote2: High-performance sensor node. http://docs.tinyos.net/index.php/Imote2.
4. IPv6 over Low power WPAN.
5. Kansei wireless sensor testbed. http://kansei.cse.ohio-state.edu.
6. Motelab: Harvard sensor setwork testbed. http://motelab.eecs.harvard.edu/.
7. NetEye wireless sensor testbed. http://neteye.cs.wayne.edu.
8. Network description language.
9. Open resource control architecture. https://geni-orca.renci.org/.
10. Orbit network testbed. http://www.orbit-lab.org/.
11. OWL: Web ontology language. http://www.w3.org/2004/OWL/.
12. PeopleNet mobility testbed. http://peoplenet.cse.ohio-state.edu.
13. Protege: Ontology editor-knowledge framework. http://protege.stanford.edu/.
14. Shibboleth: Software package for identity and authorization management. http://shibboleth.internet2.edu/.
15. Stargate gateway devices. http://blog.xbow.com/xblog/stargate_xscale_platform/.
16. SunSpots: A java based sensor mote. http://www.sunspotworld.com/.
17. TelosB sensor motes. http://blog.xbow.com/xblog/telosb/.
18. XSM: ExScale sensor motes. http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MSP410CA_Datasheet.pdf.
19. J.P. Deschrevel. The ANSA Model for Trading and Federation, ANSA Architecture Report APM.1005.01, Architecture Projects Management Limited, July 1993.
20. A. Dunkels, T. Voigt, and J. Alonso. Making TCP/IP viable for wireless sensor networks. In *Proceedings of the First European Workshop on Wireless Sensor Networks (EWSN 2004), work-in-progress session, Berlin, Germany*, 2004.
21. H. L. Johnson, K. B. Cohen, and L. Hunter. A fault model for ontology mapping, alignment, and linking systems. In *In Pacific Symposium on Biocomputing*, 2007.
22. V. Kulathamani, M. Sridharan, A. Arora, and R. Ramnath. Weave: An architecture for tailoring urban sensing applications across multiple sensor fabrics. MODUS, International Workshop on Mobile Devices and Urban Sensing, 2008.
23. N.F. Noy and M.A. Musen. Prompt: Algorithm and tool for automated ontology merging and alignment. In *Proc. of the 7th National Conference on Artificial Intelligence*, pages 450–455. AAAI Press / The MIT Press, 2000.
24. T. Roscoe. Languages not Formats: Tackling network heterogeneity head-on. In *Proceedings of the 3rd Workshop on Future Directions in Distributed Computing (FuDiCo III), Bertinoro, Italy, June 2007.*
25. R. Silva, J.S. Silva, and F. Boavida. Evaluating 6lowPAN implementations in WSNs. In *Proceedings of 9th Conferncia sobre Redes de Computadores Oeiras, Portugal*, October 2009.