# Table of Contents

1 Introduction ............................................................. 1

2 Objectives ............................................................. 2

3 Concepts .................................................................. 2

3.1 Infrastructure as a Service .............................................. 2

3.2 Virtual Organization ................................................... 2

3.3 Virtual Infrastructure ............................................... 2

3.4 Infrastructure Provider .............................................. 2

3.5 Service Provider ...................................................... 3

3.6 Follow the sun /Follow the wind .................................. 3

3.7 IaaS Framework 2.0 .................................................. 3

4 Architecture and design ............................................ 4

4.1 High Level Architecture .............................................. 4

4.2 Middleware Resources ............................................... 4

4.2.1 Cloud Resource ..................................................... 4

4.2.2 Facility Resource .................................................... 5

4.2.3 Network Resource .................................................. 5

4.3 Middleware Engines ............................................... 5

4.3.1 Libvirt Engine ...................................................... 5

4.3.2 Power Source Engine .............................................. 5

4.3.3 PDU Engine ........................................................ 5

4.3.4 Climate Engine ..................................................... 5

4.3.5 Power Distribution Unit ......................................... 5

4.4 Middleware Services ............................................... 5

4.4.1 Cloud Manager .................................................... 6

4.4.2 Facility Manager ................................................... 6

4.4.3 Network Manager ................................................. 6

4.4.4 Domain Manager .................................................. 6

4.4.5 GSN Controller .................................................... 6

4.5 GUIs ...................................................................... 7

4.5.1 GSN Client ........................................................ 7

4.5.2 Facility Management ............................................. 7

4.5.3 Power Source Management .................................. 7

4.5.4 Climate Management ............................................. 7

4.5.5 Network Management .......................................... 7

4.5.6 GSN Client......................................................... 7

5 Implementation ....................................................... 8

5.1 Cloud Computing .................................................... 8

5.1.1 Cloud Manager ................................................... 10

5.1.2 Host (Libvirt) Resource ........................................ 10

5.1.3 Shared Storage .................................................... 10

5.1.4 High Availability and Host Daemon ....................... 10

5.2 Facility Management ............................................... 11

5.2.1 Facility Manager .................................................. 11

5.2.2 Facility Resource .................................................. 12

5.2.3 Power Distribution Unit ........................................ 14

5.2.4 Climate Sensor ..................................................... 14

5.2.5 Power Source ...................................................... 15

5.3 Networking .......................................................... 15

5.4 GSN Controller ..................................................... 16

5.5 Domain, Identity and Access Control

Management ............................................................... 17

5.5.1 Domain management ........................................... 17

5.5.2 Identity Management and Access Control ............ 19

5.6 GSN Client ........................................................... 21

6 References ........................................................... 21

# Table of Figures

Figure 1: The Green Star Network: Canadian and international nodes ............................................ 1

Figure 2: GSN Components over the IaaS Framework ................................................................. 3

Figure 3: GSN Middleware High Level Architecture ................................................................. 4

Figure 4: GSN Controller architecture ....................................................................................... 6

Figure 5: GSN Client Architecture ............................................................................................ 7

Figure 6: GSN Components .................................................................................................. 8

Figure 7: Cloud Manager Component Diagram ................................................................. 9

Figure 8: Host Resource Class Diagram ................................................................. 9

Figure 9: GSN Shared Storage .................................................................................................... 10

Figure 10: Leasing System ........................................................................................................... 10

Figure 11: HA Failover Process .................................................................................................. 10

Figure 12: Facility Manager Component Diagram ............................................................... 11

Figure 13: Facility Resource Component Diagram ............................................................. 12

Figure 14: Facility Resource Class Diagram ............................................................................ 13

Figure 15: PDU Model Class Diagram ...................................................................................... 14

Figure 16: Climate Model Class Diagram ............................................................................... 14

Figure 17: Power Source Model Class Diagram ............................................................... 14

Figure 18: Network Manager Component Diagram ........................................................ 15

Figure 19: GSN Controller Component Diagram ............................................................ 16

Figure 20: GSN Controller Class Diagram ....................................................................... 16

Figure 21: Controller Execution Algorithm ............................................................................. 18

Figure 22: Domain Manager .................................................................................................. 18

Figure 23: Federated Identity Management .......................................................................... 19

Figure 24: GSN Client Dashboard ......................................................................................... 20

Figure 25: GSN Client VM View ............................................................................................ 20
Introduction

The CANARIE Green Star Network (GSN) is a pilot project funded by CANARIE and led by Dr. Mohamed Cheriet, at École de technologie supérieure in Montreal throughout 2010 – 2011 as part of the CANARIE $2.4 million Green IT Program.

The pilot was initiated by a Canadian consortium of industry, universities and government agencies with the common goal of reducing greenhouse gas (GHG) emissions arising from Information and Communication Technologies (ICT) services. The idea behind the GSN project is to build datacenters near clean power sources and use high-speed networks to transfer application data instead of the generated power to the city centers and user locations. Because many sources of green power are not always available, Virtual Machines (VMs) in the datacenters must be able to be moved on the fly to other datacenters that have sufficient power to run them.

The current Canadian GSN test bed, as shown in Figure 1, includes six core nodes built on top of the CANARIE high-speed optical network. Connected to the Canadian GSN hub located at École de technologie supérieure (ETS) in Montreal are participating international nodes located in the US, Europe, and China. Each node has a small data center powered by solar, wind, hydroelectricity or geothermal renewable energy.

Figure 1. The Green Star Network: Canadian and international nodes

Acknowledgements

Appreciation is extended to the following for the development of the middleware software and the creation this document:

• Scott Campbell, Communications Research Centre Canada
• Mathieu Lemay, Inocybe Technologies Inc
• Samir Hadjout, Kim Khoa Nguyen, Ali Lahlou, Jonathan Tellier, Fereydoun Farrahi Moghaddam, Abdelhamid Daouadj, Tara Nath Subedi, École de technologie supérieure of University of Québec
2 Objectives

The objective of the GSN Project is to develop a practical carbon footprint exchange standard for ICT services, and from the standard build a growing GSN network of carbon-centric ICT service providers. The GreenStar Network will be applied to two Green ICT service provision scenarios: (1) The quantification of the reduction of carbon emissions through the utilization of a green data center; and (2) development of management & technical policies that leverage virtualization mobility to facilitate the use of renewable hydroelectric, solar, wind and geothermal energy within the GreenStar Network.

The expected result is the creation of tools, protocols, procedures and use cases for a growing network of ICT service providers that offer customers the lowest price and greenest services.

3 Concepts

3.1 Infrastructure as a Service

The Infrastructure as a Service (IaaS) concept is considered a new concept dealing with the delivery of computing infrastructure [1]. IaaS allows clients to fully outsource services such as servers, software, data center space, and network equipment without the need to purchase these resources or having to deal with some of the difficulties the equipment owners face. Inspired from the original term Hardware as a Service (HaaS), IaaS is now provided by a number of companies, such as Savvis, BlueLock, ZDNet, Tier 3, and could have been taken into account in business plans of major world-class providers [2].

The main notion of IaaS is Cloud Architecture, which addresses key difficulties of large-scale data processing. Applications built on cloud architectures run in the cloud regardless of the physical location of the infrastructure which is determined by the provider. This allows complex underlying services to remain hidden inside the cloud. Resources are used in cloud architectures according to the user’s needs and so the highest utilization and optimization levels can be achieved. Widely-used cloud architectures in the market include Amazon EC2, Microsoft Window Azure and Google App Engine.

3.2 Virtual Organization

A virtual organization is a group of people from many different organizations forming a single organizational entity. Virtual Organizations (VOs) can acquire infrastructure from different domains to create one or many Virtual Infrastructures that the users in the VO can use. Simply, a VO represents a federation of users.

3.3 Virtual Infrastructure

Virtual Infrastructure is a virtual network that interconnects infrastructure pieces such as hosts, storage, networking, etc. from different physical infrastructure providers into a single entity.

3.4 Infrastructure Provider

An Infrastructure Provider is the entity or organization that owns physical resources (e.g. a server), virtual resources (e.g. a logical router) or both and makes them accessible to their clients. In GSN there are many infrastructure providers, (ETS, CRC, Cybera, etc.), that provide resources (Host/Server, Storage, Facility, Network, etc.) and/or services (Cloud services, Network services, Facility services) within their GSN node.
3.5 Service Provider

A Service Provider is an organization that can gather infrastructure instances from one or more Infrastructure Providers and integrate them into its management domain (e.g. integrate several hosts to create a distributed Cloud to offer Cloud services). It can also act as an Infrastructure Provider and reassign the permissions on its infrastructure instances so that other Service Providers can control them (it is a recursive process). It normally uses the infrastructure instances of its domain to provide some kind of service to end users (e.g. Cloud Services). In GSN we have created one VO (GSN VO) that is acting as a Service Provider. There are 3 kinds of services that are provided for GSN end-user (client): Cloud Services, Facility Services and Network Services.

3.6 Follow the Sun / Follow the Wind

The mobility principle known as Follow the sun / Follow the wind is referring to the migration of applications and data over the network to geographically distributed datacenters with intermittently available renewable power sources, thereby, optimizing energy utilization by following the green power availability. The applications and services are moved where computing or networking resources are available (due to the presence of sun or wind power). This may be within a single domain of operation or across multiple domains. This is done by using virtualization technologies that allow Virtual Machines to migrate between nodes in a network based on the availability of renewable energy (solar, wind and hydro generated electricity).

3.7 IaaS Framework 2.0

The IaaS framework 2.0 is an open source framework for resource description and exchange based on the akka messaging platform [3]. It is used to wrap the different cloud solutions, tools and service offerings at the GSN nodes. In the IaaS Framework, the Engine component is positioned at the lowest level of the architecture and communicates with the physical devices. Engines will receive commands from the upper layers, send them to the device and parse the response to populate the model. The Resource component serves as a layer of abstraction between the Engines and Services and provides (stateful) behaviour. The Services component sits at the top of the architecture and provides a higher level of functionality to a collection of resources. Figure 2 shows the GSN components with respect to the IaaS Framework.
4 Architecture and design

4.1 High Level Architecture

As shown in Figure 3, the GreenStar Network is controlled by an energy management middleware, including two layers: Services and Resources/Engines.

- **Middleware Resources (Resource Layer):** Includes a collection of resources and engines for physical devices such as Servers, PDUs, Power Source, etc.
- **Middleware Services (Management layer):** Includes the Controller and a set of managers. Each manager is responsible for a category of resource. The services are accessible by external clients, agents and UI elements.

The GSN Middleware (Resources and Services) is based on the IaaS Framework.

4.2 Middleware Resources

As mentioned above, a resource provides a level of abstraction for a device and exposes the functionality of the device to other resources and services. Each resource has an engine which communicates with the device to send commands and get information back.

There are three different categories of resources in GSN; Cloud Resources and Facility Resources and, Network Resources. Each type of resource is managed by a manager with the same name. The job of each manager is to manage the different resources in its category and to provide access to them by the controller and external clients such as the GSN Web Client. Complete details about all of the services and resources are described below.

4.2.1 Cloud Resource

Cloud Resources abstract computing resources (servers and virtual machines) within the GSN nodes. Currently there is one type of cloud resource implemented – the Host Resource. Through these resources, users and operators can create, delete, start, stop, and restart VMs as well as migrate them to other GSN nodes. Host availability and health is also monitored and if a host crashes, or communication to it is lost, an alarm will be triggered to attempt to recover the VMs by having them restarted on another host. The Cloud Manager interacts with these resources to provide the overall control and management of the VMs.
The Host Resources use an associated Libvirt Engine to communicate with the physical device.

### 4.2.2 Facility Resource

Facility Resources are an abstraction of a GSN node. It collects information about the overall power being consumed and generated at the node, climate/environmental information, and maintains a topology of how the local devices (hosts, PDUs, power equipment) are interconnected to each other. In real time, it performs the calculations necessary to determine the amount of remaining operating hours that are available at the node based on the amount of green power being generated and the current and max load that the system can handle. This Resource communicates the PDU Engine, Power Source Engine and Climate Engine to get the necessary information.

### 4.2.3 Network Resource

Network Resources are an abstraction of the underlying network pieces on the GSN testbed. These can be anything from individual channels or VLANs to switches to full networks. They can be used to dynamically configure sub domains with the cloud, or to setup dynamic connections to external hosts and clouds. In GSN, Network Resources are managed by third party Network Resource Provisioning Systems (NRPS) such as Argia or Mantychore and are only available through those respective APIs. In the currently deployment Mantychore is used to interconnect to other clouds over the IP Layer when exporting and importing VMs to/from other clouds.

### 4.3 Middleware Engines

Engines are components that communicate directly with the devices that are being managed. Communication can be via Telnet or SSH and can use CLI or SNMP communication protocols.

#### 4.3.1 Libvirt Engine

The Libvirt Engine interacts with the servers deployed at the GSN nodes and carries out the actions and commands initiated by Host Resources. The engine uses SSH to connect to the host and executes commands from the Libvirt toolkit.

#### 4.3.2 Power Source Engine

Power Source Engines collect information about the power being generated at GSN nodes that have green power systems such as sun or wind. The data that is collected is used in the calculations to determine the availability of green power that the node will have in real time. It will also trigger alarms if the facility is running out of power to alert the controller that the compute resource should be migrated to other facilities.

The Engine communicates with the power generation facilities via a serial to Ethernet (iOLAN) adapter to collect a continuous stream of data from the system.

#### 4.3.3 PDU Engine

PDU Engines collect information about the power being consumed by a GSN facility and maintain the topology of how the devices (servers, switches, sensors) are connected showing the power distribution for the facility.

PDU Engines communicate with the various PDU devices on GSN via Telnet, SSH and SNMP.

#### 4.3.4 Climate Engine

Climate engines abstract information about the environment of a GSN facility. Each facility has one or more environment sensor that collects information about temperature and humidity. The Engine will keep track of this information and trigger alarms if the temperature or humidity is too high or too low.

In the GSN facility deployments, the climate sensors are integrated with the Raritan PDU devices, and so the engine for that PDU is used to gather the data.

### 4.4 Middleware Services

The Middleware Services provide a management layer on top of the GSN Resources and Engines to coordinate communication to the specific types of resources and engines and to expose
the middleware functionality to higher level services and clients. The middleware services include the Cloud Manager, Facility Manager, Network Manager and the GSN Controller.

4.4.1 Cloud Manager

The Cloud Manager is responsible for control and management of compute resources such as clouds, hosts and virtual machines. The Cloud Manager coordinates communication from the higher level services and user facing applications to the compute resources at the various GSN facilities. It is responsible for handling the migration of VMs from server to server and ensuring that VM integrity is maintained across all of the hosts.

4.4.2 Facility Manager

The Facility Manager is responsible for the coordination and management of Facility resources. It provides the information required by the higher level services and administrative interfaces to display what is happening at the nodes with respect to power (being generated and consumed), as well as the climate.

4.4.3 Network Manager

The Network Manager provides network management function to Cloud Manager. It is responsible for maintaining and dispatching IP Addresses to virtual machines when they are created. It will also communicate with third party network provisioning systems to setup dynamic connections to import and export VM images from/to other clouds.

4.4.4 Domain Manager

The Domain Manager manages the identification of the various infrastructure provider domains that are part of the main GSN Virtual Organization. This information includes the name, address of each GSN Node and also the URLs to the resources that the domain is exporting so they can communicated with by the other GSN resources and services.

4.4.5 GSN Controller

The GSN Controller is an intelligent component that manages real time data distribution and performs optimization techniques on virtualized GSN resources. The overall

![Figure 4. GSN Controller Architecture](image-url)
The objective of the controller is to maximize the utilization of sun & wind-powered facilities. Its performance can be measured in terms of “intermittent power utilization”. Its role is to:

• Determine required migrations: Identifies virtual machines that must be migrated and the facilities they might be moved to. It gives preference to hosts running in other sun or wind powered facilities having long expected number of operation hours. It also considers hosting capacity on new hosts such as available cores and memory.

• Generate migration plans: Specifies pairs of hosts, together with a list of VMs to be migrated between them.

• Augment migration plans: If the plan generated in the previous step does not include moving VMs to hydro-powered facilities, this step takes the opportunity to augment the plan to move VMs currently residing at hydro-powered facilities to sun & wind-powered facilities.

• Execute migration plans: Migration plans are executed in several parallel streams of serial migration. Migration is organized so as to not overload any link in the network.

The overall architecture of the GSN Controller is shown in Figure 4. The text in the remaining sections provides detail and explanation.

The controller is the brain of the network, which is responsible for determining the optimal location of each VM. It computes the action plans to be executed on resources, and then orders the managers to perform them. Based on information provided by the controller, the managers execute relocation and migration tasks. The relationship between the controller and managers can be regarded as the Controller/Forwarder connection in an IP router. The controller keeps an overall view of the entire network; it computes the best location for each VM and updates a Resource Location Table (RLT). A “snapshot” of the RLT is given to the cloud manager. When there is a request for the creation of a new VM, the manager will look up the table to determine the best location of the new VM. If there is a change in the network, e.g., when the power source of a data center dwindles, the controller re-computes best locations for VMs and updates the cloud manager with the new RLT.

4.5 GUIs

4.5.1 GSN Client

GSN Client GUI is a simplified web application built with Adobe Flex to enable infrastructure administrators and end-users to manage the resources in their virtual organization. Through the client, they can create and manage virtual machines, see real time information about power usage and generation, see the location of VMs on the testbed, and also look at controller logs. Figure 5 shows the architecture of the GUI.

![Figure 5. GSN Client Architecture](image-url)
The implementation of GSN tools and services are based on IaaS Framework 2.0. The IaaS Framework is a set of resources, libraries and tools licensed under the Apache Software License version 2 that enable developers to quickly create new Infrastructure as a Service solutions based on the Framework programming model. The functionality provided by these tools allows developers to choose which web service stack will be used to expose the physical infrastructure as a service, and provide a series of modules to plug-in capabilities like security, reservation management and data persistence to the infrastructure service. The Framework also provides libraries to aid in the development of engines to communicate with the physical devices such as protocol handlers (CLI), transport handlers (TCP, SSL, SSH, Telnet) and an engine architecture.

The GSN Middleware is a set of components and IaaS plugins that work in a coordinated effort to provide the real time information that is needed to control and manage the GSN cloud. Figure 6 shows all of these components and how they are interconnected.

The communication between components uses messaging technologies and web services:
- AMQP messaging (using Rabbit MQ server) for all communication between GSN components. (Services, Resources, Engines)
- CXF web services for exposing GSN services to GUI or other services.

All of the services have two primary interfaces; IXxxManagerService which is the service interface used by GUIs and other third party clients via SOAP, and IXxxManager which is the internal implementation of the manager. All of the internal manager interfaces are implemented with an AmqpXxxManager class which uses AMQP Messaging to communicate with the other components.

5.1 Cloud Computing

The Cloud Computing component is implemented with two principal parts: The Cloud Manager and Host (Libvirt) Resource. As mentioned above, Libvirt is used as the Virtualization API and is deployed on Ubuntu servers using the KVM hypervisor.
Figure 7. Cloud Manager Component Diagram

Figure 8. Host Resource Class Diagram
5.1.1 Cloud Manager

The cloud manager enables users and service providers to control and manage virtual machines and gather information about hosts and VMs. A component diagram of the Cloud Manager is shown below in Figure 7 showing the service interface and implementation as well as the internal interface and AMQP implementations.

The Cloud Manager communicates with the Host Resource for compute related operations, the Network Manager to get IP addresses to assign to the VMs and the Controller to provide it with the latest information about the overall state of the compute resources so it can make its decisions about migrations.

5.1.2 Host (Libvirt) Resource

The Host Resource communicates with the hosts using SSH and CLI. Operations that can be performed are: create, delete, start, stop, restart, suspend, resume, and migrate virtual machines. There are also operations to get status information from the hosts such as its memory and compute resources. The Host resource uses the Host model to abstract the Host and Virtual Machines. Figure 8 below shows the Host Model.

5.1.3 Shared Storage

A shared storage system is used to store GSN virtual machine Images (running images and VM templates). SSHFS (SSH File System or Secure SHell File System) is the protocol used for the shared file system. Since GSN is deployed in a flat network, all KVM Hosts use the same mounting point. Figure 9 shows the shared storage deployment.

5.1.4 High Availability and Host Daemon

In the cloud computing environment High Availability (HA) of virtual machines and compute resources is extremely important. However in a geographically disperse environment such as the GSN testbed, it is expected that failures will happen causing compute resources (Hosts) to become disconnected from the network. Failures such as network outages and hardware failures are beyond our control, but we need to have mechanisms in place to minimize the downtime. To handle these types of situations we have developed some tools and protocols to ensure the High Availability of the VMs and compute resources on the testbed. A Host Daemon has been developed and deployed on all of the servers which monitors key systems on the Host, and implements a lease renewal system to ensure the host stays in contact with the Host Resource.
The leasing system works between the Host Resource and the Host that the resource is managing. When the Host Resource is first deployed, it will contact the Host and give it a lease along with a call-back address for lease renewals. At periodic intervals, the Host Daemon must send lease renewals to this address so that the Host Resource will know that the Host is still healthy and stable. If the lease on the Host Resource expires, the Resource will assume that the Host has failed and can trigger an HA alarm to handle the situation. The HA alarm will be sent to the Cloud Manager, who will be able to restart all of the VMs that were on the failed host on other hosts in the testbed. This is equivalent to a cold reboot of the VMs, and it’s possible that work could be lost, but is an acceptable reality in cloud computing. Figure 10 illustrates the lease renewal process and Figure 11 illustrates the failover event.

Another potential failing point is the Host’s connect to the shared storage. If this connection is lost, the VMs will not be able to save any data, and the state of the VMs will lose their state. To try and minimize this problem, the Host Daemon will continuously check to make sure the connection to the shared storage is up. If it’s not, the Host Daemon will attempt to reconnect it. If it can’t, it will send a notification to the Host Resource to trigger an HA event and the host will be put in a failure state until the connection to the storage can be re-established.

The Host Daemon is written in Python and running as a stand-alone process on each host. It uses messaging to communicate with the Host Resource.

5.2 Facility Management

The Facility Management implementation is based on two principal components: The Facility Resource, which is composed of the Power Source, PDU and Climate engines, and the Facility Manager which manages the Facility Resources. The PDU devices are used to get power consumption and environmental data from each GSN Facility. The Power Generation System (Outback Mate for Solar Panels) provides data about Green Power that is generated.

5.2.1 Facility Manager

The Facility Manager is responsible for managing the many Facility Resources in the GSN VO. It fetches the required information for the controller, GSN GUI and other third party clients. A component diagram of the Facility Manager is shown below in Figure 12 showing the service interface and implementation as well as the internal interface and AMQP implementation.

![Facility Manager Component Diagram](image)
5.2.2 Facility Resource

The Facility Resource is responsible for abstracting the entire facility into a single entity. It collects the information from the device engines in the facility and presents them as a single manageable entity. These devices are the Power Source Engine, PDU Engines, and Climate Engines. The Facility Resource is also responsible for verifying thresholds and receiving alarms triggered by the sub resources. It will pass any alarms to the necessary components that have to deal with them and will also send email alerts to facility administrators.

The Figure 13 illustrates the principal components and interactions for the Facility Resources.

With the information it collects from the engines, the Facility Resource does a number of important calculations to determine the overall state of the facility. These calculations are used by the controller while building migration plans. They are:

- batteryChargePercentage: calculates the capacity remaining in the battery bank. The value is taken from the battery voltage which is read from power source model.
- onGrid: Tells if the facility is being powered by solar power or by grid power. Depending on the status of the inverter within the solar power system, if there is not enough solar power being generated to power the equipment, this field will be true indicating that the equipment is using grid power. If false, the facility is running on solar power.
- domainGreenPercentage: According to the onGrid parameter, this value will set to the green percentage of the grid or green percentage of the power source. If the facility is running on grid power, it is said to have 0% green power. If it is running on solar power, it is said to be 95% green. This is because the climate control systems (heating and cooling) are running on grid power continuously and is taken into account with the percentage.
- opHourUnderCurrentLoad: From the batteryChargePercentage value, a value will be calculated to determine the remaining operational hours under the current load according to the power load which is gathered from the PDU. Total stored energy in the battery bank is calculated with:

\[ E_{\text{stored}} = E_{\text{max}} \times \text{Battery Charge State} \]

And then, the remaining operating hours under current load are calculated with:
• opHourUnderMaximumLoad: The remaining operating hours under maximum load is the amount of time that the facility can remain on green power while running at maximum capacity. It is calculated with the following formula:

\[ \text{opHour}_{\text{currentLoad}} = \frac{E_{\text{stored}}}{P_{\text{out}}} \times I_{\text{out}} \]

- totalConsumingPower: The total power that is being consumed.

\[ P_{\text{consuming}} = V_{\text{out}} \times I_{\text{out}} \]

- totalGeneratingPower: Total power being generated. It is calculated with:

\[ P_{\text{gen}} = V_{\text{out}} \times I_{\text{out}} \]

The Facility Resource also maintains a power distribution table that shows power sinks (devices consuming Power) and power sources. This table is used to form the topology of the facility.

The full facility model is too large to show in one diagram and so it is broken up into its various roles and described in the sections below. Figure 14 shows the core Facility Model with the power distribution and the operational specs.

The Facility Resource plays a critical role in the system and without it the controller would not have any information about green power availability and could not implement the follow the sun / follow the wind scenario. In the current deployment of the GSN testbed, there are only two nodes (CRC and Cybera) that have real solar power facilities which is not enough to fully realize a global scenario where VMs can be migrated around the glob. The other GSN nodes are still powered by green power, and some of them even with solar and/or wind power but they do not have the equipment such as the Outback Mate system that allows us to gather the real time information about the power being generated. To enable this, and so we can have more nodes for the controller to consider as having "intermittently available green power", we have developed a virtual facility mode where a GSN node will have real hosts, and real PDUs collecting data about the power being consumed, but have simulated data about the power being generated. The data for the simulated power source has been taken from the CRC solar panel system over the period of two weeks and organized in 24 hour period datasets. For each 24 hour period in the simulated facility’s location, the virtual power source will randomly select a dataset to use and represent the power source data at that node for that day. By using this hybrid solution we can gather more useful data about the controller’s performance and cover a much larger geographic area.
5.2.3 Power Distribution Unit

There are currently three types of PDU devices being controlled by PDU Engines: Raritan PCR8, ServerTech CWG and Eaton PW. Communication to each of the devices varies based on the capabilities of the device. The Raritan can use Telnet or SSH. SSH is preferred over telnet but the device does not allow multiple SSH sessions at the same time and so sessions by the Raritan Engine would conflict with sessions by administrators and could potentially cause some failures so we were forced to use Telnet instead. The Servertech CWG does not support SSH and so we have no choice but to use Telnet. For the Eaton PW we decided to use SNMP. This is because there are many other different types of PDUs owned by other associate partners, each with their own proprietary CLI, having the ability to communicate with them using SNMP will make it much easier to port the different models. Figure 15 illustrates the principal components and interactions for PDU resource implementation.

5.2.4 Climate Sensor

As mentioned earlier in the document, the climate sensors for a facility are part of the Raritan PDU. Because of this the climate model and commands to get the information are integrated with the Raritan PDU engine and abstracted at the
resource level. Figure 16 below shows the class diagram for the Climate piece of the Facility model.

Environmental parameters that are considered are temperatures and humidify. The climate model also has thresholds for minimum and maximum levels for temperature and humidity. If the readings are outside these thresholds an alarm will be triggered and processed by the Facility Resource.

5.2.5 Power Source

GSN supports one type of Power Source, called OutbackMate which is the type of power generation system for the solar panel systems. The Power Source is responsible for collecting information about the power being generated by the solar panels, which includes an inverter and a charger. The Outback Mate sends a continuous stream of status data to a serial port which the Outback Mate Engine listens to (via the IOLAN Serial to IP Converter) to collect the real time information about the power being generated. The data that is collected is used by the Facility Resource to do the required calculations to determine the available number of operational hours based on the current and maximum load at the site. If there is not enough power being generated to power the facility, an alarm will be sent to the Facility Resource so it can notify other components in the system that the VMs at the facility should be migrated elsewhere. Figure 17 shows the Power Source Model.

5.3 Networking

The Network Manager provides network related functionality to the Cloud Manager. It has the following functions:

- Generate a DHCP table which is then imported to the DHCP server. The table defines MAC-IP bindings based on which the DHCP server will assign IP address to VMs in the GSN.
- Assign an available MAC address to a new VM, release the MAC address when a VM is removed, and determine the IP address based on MAC.
- Communicate with third party network provisioning systems to setup dynamic connections to import and export VM images from/to other clouds.

Figure 18 shows the Network Manager components.
5.4 GSN Controller

The GSN controller is the central brain that implements Follow the Sun/ Follow the Wind algorithm and is responsible for optimizing the green power usage on the GSN testbed. It has four main components shown as in Figure 19.

ControllerImpl is the main class of the controller and contains a host list, a link table which shows the connectivity map of the hosts to each other, and a migration plan. It orchestrates the generation and execution of migration plans.

The Executor class maintains a list of migration actions. Each action could contain single or parallel migrations for a specific link. Three steps are taken for each new migration:

1. Check if there is a link with the same source and destination in the action list.
2. Check if a parallel migration can be added to an existing action or create a new action and add this migration to the parallel migration.
3. Execute the list of actions by sending messages to the Cloud Manager. Each action message contains a migration or a set of parallel migrations.
The Plan Generator class will receive a list of Hosts and VMs as inputs and will calculate a migration plan for hosting those VMs on those Hosts.

The Aggregated Plan Generator will look at the migration plans that have been generated for each node and attempt to optimize the distribution of the resources based on the availability of intermittently available green power. It will favour sites that have solar and wind power over ones with Hydro or other always available green power sources. It will also look to see if there are VMs already located at sites with always available green power and will move them to the favoured solar and wind sites if there is available power to do so.

The Controller’s model is shown in the class diagram in Figure 20.

The controller components use data models which are a combination of the host model components plus some energy information. These models are exposed to the controller client.

- ControllerImpl is the parent class containing the model
- MigrationPlan contains the list of migrations to be executed
- Migration contains information about the migration of virtual machines from a source host to a destination host.
- MigrationPlan is a list of Migrations
- Host stores the host model plus additional energy information such as energy-priority and lifetime.
- HostList is used to keep a list of Hosts
- VirtualMachine is a store for the VM model and additional information such as the priority used when sorting VMs.
- VMList is a list of VirtualMachines

Controller Algorithm

Once all of the information has been collected from the managers, the controller can begin to generation the migration plans. The process for doing this is described below.

1. First, the hosts are sorted into groups based on their type of power source and then amount of power available. As mentioned above, sites with intermittently available green power such as solar and wind are favoured over sites with always available power such as hydro.
2. It will then make a list of VMs that have to be migrated away from the site(s) that don’t have enough available power to sustain them.
3. A second list will be created containing the hosts that have available green power, ranking them so the favoured power sources with the most available power will be on the top of the list.
4. The controller will then assign virtual machines from hosts that don’t have enough power, to hosts that have available power, favouring the solar and wind sites. The amount of compute resources on these hosts is also considered.
5. If there aren’t enough hosts with enough available green power or compute resources at the favoured sites, the controller will look at hosts with less favoured power sources such as hydro to migrate VMs to. If all of the VMs have been assigned to a new host and the favoured sites still have capacity for more VMs, the controller will take VMs from less favoured sites and move them to the ones with the intermittently available green power.
6. Now that the migration plan has been completed, it will be executed.

Figure 21 below shows the algorithm.

5.5 Domain, Identity and Access Control Management

5.5.1 Domain management

The Domain Manager manages the identification of infrastructure provider’s domains including their names and IP addresses. Other managers use the domain manager to lookup the addresses of the infrastructure provider domains to know how to communicate with them. When domains are added or removed, the managers are notified with the change and can update their own address books.

Figure 22 shows the interactions of the Domain Manager with the Cloud and Facility Managers.
Figure 21. Controller Execution Algorithm

Figure 22. Domain Manager
5.5.2 Identity Management and Access Control

GSN relies entirely on the IaaS Framework in order to perform identity management and access control.

The IaaS Framework enables the creation of federated clouds and manages identity in a federated way. Federated identity management is achieved by establishing trust relationships that allow a subject of a domain to access resources and services managed by other domains. The attributes of a subject as well as the authentication process are handled by a given Identity Provider (IdP), usually located at his home domain. Once authenticated, the subject receives some credentials that contain a subset of his attributes or access rights, and a reference to his home domain or IdP. The credentials can be presented to any Service Provider (SP) software component (not to be confused with the Service Provider described in Section 3.5) which trusts the given IdP in order to access its assets [4, 5, 6]. This process is illustrated in the following Figure 23.

1. The subject authenticates with his IdP using one of the methods supported by its policies (user name and password, X.509 certificates, biometrics, etc.). Upon a successful authentication, the IdP fetches the relevant identity information from its identity repository.
2. The IdP returns a proof of authentication (i.e., a credential) to the subject.
3. The subject makes a request to the SP in order to access a protected resource. The credential issued in Step 2 is included in the request.
4. The SP analyzes the credential along with any other relevant information in order to authorize the request according to its local policies.
5. If the request is granted, the subject can access the resources from different security domains.

The federation-enabling standard that is used by the IaaS Framework is the OASIS Security Assertion Mark-up Language (SAML) [7]. It is an XML-based framework that
can be used to exchange information about security policies or assertions on a subject between systems from different security domains.

The first interaction in the IaaS Framework access control flow occurs when a subject wants to authenticate to the IdP managing his identity in order to retrieve SAML credentials. This is represented by the first step in Figure 23. For this component, the Shibboleth [8] Identity Provider has been used. The Shibboleth project is a SAML implementation for which most core developers also work within OASIS on the SAML specification. As a result, it is highly standard compliant and has proven itself as the most appropriate solution for many organizations around the world [8]. Once a subject as been authenticated by his IdP and is in

Figure 24. GSN Client Dashboard

Figure 25. GSN Client VM View
possession of a SAML token, he presents it to an SP in order to have access to protected resources. This step is the third in Figure 23. Because of the ease with which it can be integrated into the IaaS Framework, the Spring Security Extensions Project [10] is the SP used within the platform.

In the IaaS Framework, the SP has the responsibility of extracting information from the SAML token issued by the IdP. This information is then handed to the modules in charge of policy decisions and enforcement.

Those modules perform access control by using the XACML [11] standard. This allows great flexibility regarding security policies. Access control decisions can be made on the basis of any attributes of either the subject wanting to perform an action, the resource on which the action is to be taken and the environment where the resource is located.

## 5.6 GSN Client

As described in section 4.5.1 above, the GSN Client GUI is a web application build using Adobe Flex. It allows users and administrators to manage the virtual resources and see the overall state of the GSN testbed. Below are some screenshots of the GUI.

## 6 References