Requirements for Wireless GENI Management and Control

GDD-06-15

GENI: Global Environment for Network Innovations

September 15, 2006

Status: Draft (Version 1.0)

Note to the reader: this document is a work in progress and continues to evolve rapidly. Certain aspects of the GENI architecture are not yet addressed at all, and, for those aspects that are addressed here, a number of unresolved issues are identified in the text. Further, due to the active development and editing process, some portions of the document may be logically inconsistent with others.

This document is prepared by the Wireless Working Group.

Editor:

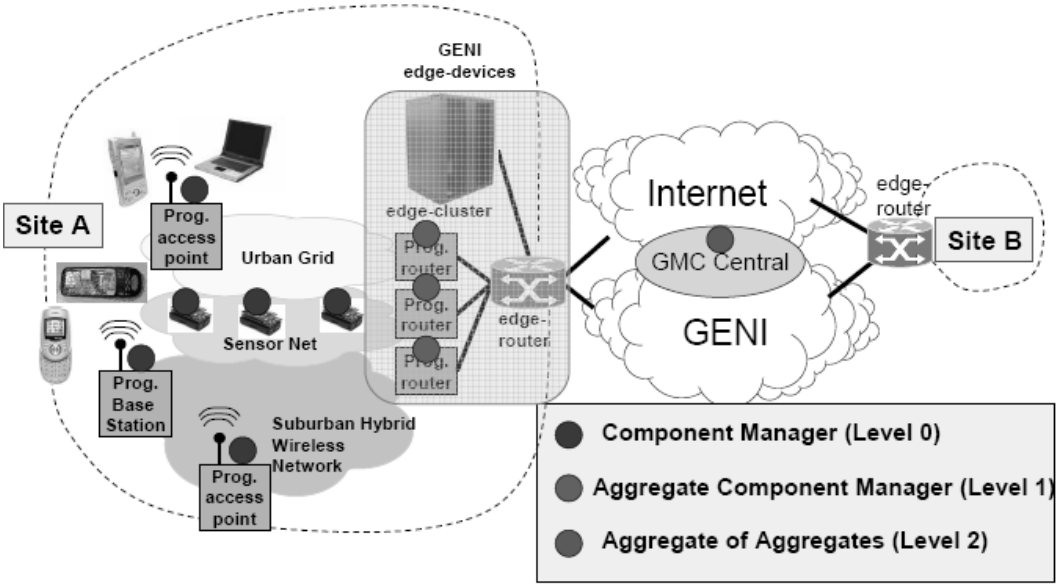
Sanjoy Paul, *Rutgers University*

1 Introduction

This document is used to describe the collective thinking of the wireless sub-group of GENI to describe the requirements for management and control of wireless networks. The goal is to describe the requirements from both top-down as well as bottom-up perspectives. In addition, the requirements stated in this document cover both long-running services as well as short-term experiments running on the wireless networks.

2 GENI Management and Control (GMC)

Figure-1 captures the architecture of GENI Management and Control (GMC). GMC is organized as a hierarchical architecture where each component is managed by a Component Manager (CM), a group of components is managed by an "Aggregate" Component Manager and entire GENI is managed by GMC Central which is an "Aggregate of Aggregates".



3 Definitions

3.1 Component:

Components are the primary building blocks of GENI. For example, Edge computer, Programmable Router, Programmable Access Point, Programmable IP Base Stations are examples of components pertaining to a wireless edge network. Each component consists of a collection of (1) Physical resources (CPU, Memory, Disk, and Bandwidth), (2) Logical resources (file descriptor, port #) and (3) Privileged operations. It must be possible to partition the resources belonging to a component among multiple users – each partition is called a "sliver". Each sliver will have two interfaces: (1) Signal interface which is

needed for a component to signal an event to the user program running on the sliver, and (2) Remote Access interface which is needed for enabling users to remotely execute a command on the sliver and/or log in to the sliver. Resource allocation policies on a component are dictated by the component owner. Each component has a unique id.

3.2 Aggregate Component:

Aggregate components represent a collection of components. The goal is to be able to use a common language to represent resources whether they belong to a single component or they belong to a collection of components.

Two types of aggregate components are being envisioned:

(1) Coordination Aggregate (CA): CA represents a collection of related components which will be managed in a coordinated way, and will be responsible for resource allocation across these components. For example, an urban wireless grid CA would maintain state in terms of the resources allocated/available on the nodes in the grid, and would coordinate spectrum allocation among the nodes (Access Points).

(3) Portal Aggregate (PA): PA represents a front-end to a collection of experiment management services, and will be responsible for creating and scheduling slices on behalf of researchers running short-term controlled experiments, and acquiring resources for slices running long-term services. For example, a wireless testbed like ORBIT with its web-based front-end for the experimenters to schedule their experiments would fall under this category.

GMC Central (aka "Root") would be an aggregate of aggregates.

3.3 Component Manager (CM):

Component Manager's responsibility is to control and manage a component. Specifically, CM exports a well-defined interface for:

(1) Creating and destroying slivers, binding resources to and reclaiming resources from slivers

(2) Mechanisms for isolating slivers from each other

a. Resources consumed by one sliver should not unduly affect performance other slivers

b. One sliver cannot eavesdrop network traffic to and from another sliver

c. One sliver should not be able to access objects (files, ports, processes) belonging to another sliver

d. Users are allowed to install their own software packages regardless of what is installed on other slivers on the same component

(3) Mechanism to allow secure log in to respective slivers

(4) Mechanism to deliver signals (such as reboot) to respective slivers

(5) Mechanism to audit all packets transmitted and received by a sliver.

Note that there would be "Aggregate Component Managers" which would provide similar interfaces for "slices" running on a set of components.

3.4 Resource Specification (Rspec):

Rspec is the resource specification language for GENI. Specifically, Rspec is a set of attributes that describe properties of a component such as, (1) Resources (e.g., a request for 10Mbps of link bandwidth) and (2) Privileged operations that the slice hopes to be able to invoke on the component (e.g., right to create a slice/read a particular instrumentation variable).

A key attribute of Rspec is "type" of sliver and some base types are defined which can be extended to support more complex forms of slivers. The base sliver types at the time of writing this document are virtual server, virtual router, virtual switch, and virtual AP. The base sliver types can be extended to cover other interesting classes.

Rspec is used for three different purposes: (1) Advertising a set of potential resources, (2) Requesting a set of desired resources, and (3) Promising a set of available resources.

Rspec will be general enough to specify slices at different levels of abstraction: (1) Low-level spec for a sliver to be instantiated on a specific component, and (2) GENI-wide spec for a slice across a possibly heterogeneous set of components.

Rspec is like the "machine language" for a component. While some users may choose to use this language directly, most users will choose to use some higher level front-end, such as GUI, SWORD, or Emulab's extensions to the ns-2 language for specifying their resource needs from GENI. This higher-level specification will be "compiled" to Rspec.

Rspec abstraction can also be used to specify that a component can process packets. In this context, there are the following options:

- Modify packets - can do things like update packet headers (ie. decrement TTL, set 'next hop' address), possibly packet contents (in the case of transcoding audio or video).

- Make forwarding decisions - an object which, having received a packet, can make decisions about which next hop(s) to forward the packet to. A switch would be an example of this, as would be an IP router (which would also have the 'modify packets' attribute).

- Classify packets - match packets based on some criteria, with a firewall as an example.

- Generate packets - source packets, likely with user-defined contents.
- Receive packets - be a sink for packets

All of the operations above can be done at different network layers by different devices. For example,

- A software-defined radio (aka 'cognitive radio') can generate packets and receive packets at the 'PHYSICAL' layer.
- A traditional IP router can modify packets at the 'IP' layer and make forwarding decisions at the 'IP' layer.
- A server can generate and receive packets at the 'IP' and 'TCP' layers. If the user has root on the box, they might also be able generate packets at the 'ETHERNET' layer, and maybe classify packets at the 'IP', 'TCP', and 'UDP' layers (ie. iptables).
- A sensor network node can generate packets at the link layer (for example 'ZIGBEE'). It can also generate, receive, and forward packets at this layer.
- A traditional firewall box can classify packets in layers 'IP', 'TCP', and 'UDP', and can make forwarding decisions (i.e. forward or don't forward) at the 'IP' layer. Fancier firewalls might be able to classify at other layers, such as 'HTTP', and might be able to modify packets.
- A network processor such as an Intel IXP can generate, receive, forward, classify, and modify packets at the 'ETHERNET', 'IP', 'TCP', and 'UDP' layers.

4 Wireless-specific Requirements for GENI

4.1 Specification of Requirements

While Rspec can be used to specify requirements for individual wireless components at very low level of details, it may be very difficult and cumbersome for GENI wireless users to do so for a variety of scenarios. It may be worthwhile to distinguish between “long running services” and “short-term experiments” when it comes to specifying the requirements.

Specifying requirements of “short-term wireless experiments” can be tricky. In order to increase the adoption of GENI among a broader group of wireless researchers and users, it is mandatory that some rich higher level abstractions are provided to enable specification of fairly sophisticated experiments. For example, a vehicular wireless experimenter may need to specify a “downtown” driving scenario with 20 cars (mobile nodes) within 1 sq. mile: 2 clusters each with 10 cars; each cluster in the proximity of an Access Point connected to the wired backbone. While it may be possible to specify requirements for individual mobile nodes using rspec, it may not be easy to specify the topology and/or the

environment for carrying out the experiment unless there is a higher-level abstraction to do so.

4.2 Admission Control

Overbooking of network may lead to non-availability of resources in GENI for certain experiments. One way of dealing with the situation is to blindly accept all requests and try to provide best-effort service to all. The other approach is for the management and control software to maintain state of the network over which it is responsible for allocating resources, and use that information to do "admission control". Thus request for a specific slice of the network may or may not be granted based on availability of resources. For example, a GENI user experimenting with a "cache and forward" peer-to-peer network architecture may require a minimum number of nodes with rich connectivity and a minimum amount of storage per node so that she can explore sophisticated caching and routing algorithms. However, such a requirement may not be met because of lack of resources in GENI. For example, there may not be enough number of nodes to satisfy the minimum number of nodes required by the experimenter. Note that in this context, nodes are nothing but slivers in the virtualized GENI network. Even if the requirement of minimum number of nodes can be satisfied, the available nodes may not have the desired amount of resources in terms of storage/node because of prior allocations.

It should be noted that admission control is not specific to wireless networks. However, it can be argued that it is "more" critical for wireless experiments where the environment can change rather quickly unless the Management and Control plane is extra careful.

4.3 Service Level Agreement (SLA)

While many experiments are based on best effort networks, there are several experiments that are best carried out in networks that provide a guarantee on availability of certain minimum amount of resources. For example, a GENI Mobile Virtual Network Operator (MVNO) providing a Voice-over-IP (VoIP) service over hybrid WiFi-Cellular network islands in major US cities interconnected over a wired wide area network would expect minimal QoS support in terms of bandwidth, delay, and jitter from the GENI slice allocated to her. While asking for "guaranteed" amount of QoS from GENI in int-serv style would become very expensive and may lead to significant waste of resources, asking for a "softer" guarantee on the resources would be highly desirable. In the context of this document, SLA refers to a "soft" or "probabilistic" guarantee on availability of resources as opposed to a "hard" guarantee.

Another viewpoint would be to make GENI users pay for QoS guarantees. Stricter the requirements on QoS, higher are the prices. This approach would prevent frivolous QoS requests and would be able to generate enough revenue to pay for additional resources needed to support stricter QoS guarantees.

For wireless experiments that require certain topology among its network elements, SLA might take a slightly different meaning. SLA, in such case, would not necessarily mean bandwidth/delay/jitter rather it would mean some level of assurance from the network that

topology would not change (due to say change in signal and/or noise power levels of some nodes in a different slice that interferes with the operation of the slice under consideration).

SLA seems to be very closely related with the next item in the list, namely, “monitoring”.

Also, as in the case of admission control, it should be noted that SLA is not specific to wireless networks. However, it can be argued that it is "more" critical for wireless experiments where the environment can change rather quickly unless the Management and Control plane is extra careful.

4.4 Scheduling

In a virtualized network where the resources are shared across multiple slices, one of the key requirements is “scheduling”. There would be a variety of experiments and/or services with heterogeneous requirements in terms of resources, and SLAs. Given a snapshot of the available resources and the experiments at hand, the scheduler would have to decide which experiments/services would be scheduled to run at a given instant of time.

While this is a requirement across wired and wireless networks, given that many wireless experiments would be short-lived and would have more stringent requirements in terms of radio and network environments, scheduling would be very challenging. Bringing cognitive radio experiments in the mix of existing types of wireless networks makes scheduling even more challenging because of the adaptive nature of the radio.

Also, as in the case of admission control and SLA, it should be noted that scheduling is not specific to wireless networks. However, it can be argued that it is "more" critical for wireless experiments where the environment can change rather quickly unless the Manangement and Control plane is extra careful.

4.5 Discovery

Topology in a virtualized wireless networks depends on a variety of factors, including transmit power, noise level, frequency, time-slots, codes, mobility and others. As a result, it is imperative that GENI provides a “discovery” service such that the researchers, before requesting resources for their experiments, would know what is available in a given wireless network.

Ideally a visual tool depicting the available resources in a wireless network should be made available to the GENI wireless researchers.

4.6 Monitoring

Monitoring is critical to the success of GENI. It should happen in a distributed manner starting at the level of individual components going up to the level of aggregate

components that can maintain snapshots of the availability of resources across individual components of the aggregate. Component level monitoring would guarantee that users of a given sliver do not violate resources granted to them. In the context of wireless, this might mean that the transmitter in a given sliver does not exceed its allocated transmit power or say an application does not use more channels and/or time-slots than those allocated to its corresponding sliver. In addition to monitoring resource consumption of slivers in a component, it is also important to monitor extraneous users who might disturb the perceived topology of the slices by injecting noise in the network or even prevent communication within a slice by doing channel jamming. Monitoring and security are intimately related more so in the context of wireless networks.

4.7 Measurement/Instrumentation

A Measurement framework for collecting relevant performance statistics is critical for wireless GENI users. Users should be able to specify:

- What performance metrics (throughput, latency, packet error rate, SNR) to collect
- How frequently (every minute, every 5 minutes, every hour) to collect
- What granularity of results (every sample, mean, distribution)

4.7.1 Detailed Requirements

There are two types of metrics: (1) Generic or Experiment-independent, and (2) Specific or Experiment-dependent. The Generic or Experiment-independent metrics can be collected by GENI measurement/ instrumentation framework and provided as a service to a researcher. However, for Specific or Experiment-dependent metrics, researchers would have to write their own instrumentation code.

The following is a list of Generic metrics that are of interest to capture on a long-term basis. However, a specific GENI user may request a subset of the metrics for his/her own experiments:

Flow- and Packet-Level Parameters:

- At each node:
  - Header Snap Length: 250 bytes
    - includes MAC, IP, Transport, Application layer headers
  - Packet Size
  - Flow length (number of packets)
  - Physical/MAC layer information: signal strength of received packet, rate of transmission, backoff interval, retransmission attempts, reason for packet loss, channel busy time as sensed by radio
  - Packet transmit power
  - BER and PER (per transmission)
  - Number of retransmissions per data packet
  - Average back-off per data packet

- Data rate per packet

- At destination:
  - Packet interarrival time (jitter)
  - Packet arrival time (assuming synchronized clocks)

- At source:
  - Packet send time (assuming synchronized clocks)
  - Flow interarrival frequency
  - Number of DHCP requests (per time period)

Mobility Parameters:

- At mobile nodes:
  - Physical coordinates per time unit – obtained through either GPS or localization techniques

Wireless Parameters:

- At all wireless nodes:
  - Packet probe to obtain delay to each neighbor (periodically measured)
  - Number of association messages (per time period)
  - Time between associations requests
  - Number of other control packets per time period (RTS/CTS/Probes)
  - Number of mesh routers (MRs)/access points (APs) from which beacons are received (per time period)
  - Number of unique ESSIDs observed
  - Number of handoffs (per time period)
  - SNR between mobile node and each MR/AP
  - Surrounding interference levels and source of interference (ex: microwave, wifi network, etc.)

RF Parameters:

- At all wireless nodes:

  –

4.7.2 Storage Requirements

Given that there is a long list of Generic metrics as specified in section 4.5.1, the storage requirement for collecting and storing these measurements would be enormous, especially if some of these metrics are collected at a fine timing granularity.

Anticipating the huge storage requirements of measurement data, there would be a need for a distributed architecture for storing, filtering, aggregating and transporting data to the interested entities.

While this is a broad architecture question for the GENI-wide storage infra-structure, there is also an issue of storage requirement per researcher. In fact, a critical requirement for a researcher might be to specify the "storage" needs for collecting measurement data for her experiments. If a researcher does (can) not specify the storage requirements, there will be a default allocation of storage and the collected data will be deleted on a first in first out basis.

5 Wireless Components

5.1 Types of Wireless Components

Each GENI node would have two types of interfaces: (1) Management-plane interface and (2) Data/Bearer-plane interface. Based on the number and type of Data-plane interfaces, primary function of a node (packet forwarding vs. computation), and mobility, we can define 5 kinds of wireless components:

1) **Access Point** (which has at least one wired interface + at least one wireless interface)

2) **Forwarding Node** (which has at least two wireless interfaces)

    a. An Access Point that uses "ad hoc" mode of operation instead of "infra-structure" mode of operation at MAC layer is a forwarding node.

3) **Mobile Node** is a Forwarding node that is mobile

    a. Mobility can be specified over an area by the number of mobiles within a geographic bounding box;

    b. For specifying mobility of individual nodes, it would be desirable to specify velocity (speed and direction) of the node in addition to its starting position. However, the velocity and/or the starting position of a mobile may not be controllable!

4) **Sensor Node** (which will have a single wireless interface)

5) **Sensor Gateway Node** (which is a Forwarding Node representing an aggregate of sensor nodes)

While a sensor node, as described above, is the simplest wireless component, a mobile node, as described above, is the most complex one. Thus, if we can specify a mobile node, all other kinds of wireless nodes can be specified as a degenerate version of the mobile node. Specifically, if we specify a mobile node with two interfaces that are wireless and with some computation capability, we can specify:

1. A static Forwarding Node by removing the attribute of "mobility" and "computation" from the Mobile Node spec.

2. A Sensor Node by removing one interface from the Mobile Node spec.

3. A static Sensor Gateway Node by removing the attribute of "mobility" from the Mobile Node spec.

4. An Access Point by removing the attribute of "mobility" from the Mobile Node spec., and by changing the mode of operation from "ad-hoc" to "infra-structure".

5.2 Rspec of Wireless Components

At the highest level, there is no distinction between a wired node and a wireless node. Every node has a set of interfaces. A node with at least one wireless interface is referred to as a wireless node. A wireless node would be hierarchically specified:

First, the wireless interface will be specified using a "type" field [802.11a/b/g/n/p, CDMA, OFDM, Ev-DO, HSDPA]. Then, each type of wireless interface will be specified using [name, value] pairs for attributes unique to the specific "type" of wireless interface. Cognitive radios will be another "type" of wireless interface with their own set of [name, value] pairs of attributes.

A **critical** requirement for a researcher might be to specify the "storage" needs for collecting measurement data for her experiments. If a researcher does (can) not specify the storage requirements, there will be a default allocation of storage and the collected data will be deleted on a first in first out basis.

This will achieve two goals:

1) Extensibility: we could plug in a new type of radio in future

2) Refinement: we do not have to specify all the detailed [name, value] pairs for all wireless interfaces upfront but can add them as and when needed.

5.3 Example Rspec of a Wireless Component

Keeping in mind that a variety of wireless components can be specified as a special case of a Mobile Node, we provide an example Rspec for a Mobile Node with two WiFi interfaces and a Cellular interface.

Example sliver asks for a set of fair-share and best-effort resources on a single wireless node.

- Sliver description:
  - On node orbit2.winlab.rutgers.edu
  - In a Linux vserver-based 'virtual machine'
  - With fair-share CPU scheduling
  - With a best-effort amount of RAM
  - With a best-effort amount of disk space

- With a WiFi interface,
    - type 802.11 a,
        - SSID winmain
        - rate auto
        - mode ad-hoc
        - number of channels 2 [label A]
        - transmit power 10mw
        - carrier sense threshold 2db
- With a WiFi interface,
    - type 802.11 a,
        - SSID orbit
        - rate 54mbps
        - mode ad-hoc
        - number of channels 2 [label B]
            - where B!=A
        - transmit power 10mw
        - carrier sense threshold 2db
- With a Cellular interface,
    - type CDMA 1x Ev-DO Rev.A
        - scheduling proportional fair
        - downlink data rate 500 kbps
        - uplink datarate 100 kbps
        - transmit power 50DBm
- With GPS,
- With Mobility

6 Mapping Wireless Management and Control Requirements to GMC

Having specified the requirements for wireless GENI experiments, we explore which of these requirements are addressed and where are the gaps in terms of supporting these requirements in the current version of GMC.

6.1 Specification of Requirements

As described above, it is possible to specify individual slivers on wireless components using Rspec. However, most of the challenges in wireless experiments lie with specifying the experimental "environment" and/or topologies of slices. Topology, in wireless networks, is trickier than in wired networks because of the following reasons:

1) Topology in wireless networks is identified by multiple dimensions, such as, channel frequency, signal strength, mode of operation etc. For example, topology of a wireless slice can change if the signal strength of one or more nodes change or an external node injects noise into the network. A change in allocation of channel frequency can also change the topology of a slice even though the nodes are static. An 802.11 node can be configured either in the "ad hoc" mode or in the "infra-structure" mode. If the mode of operation is changed for a node from "infra-structure" to "ad hoc", the topology of a slice containing that node will change.

2) Topology and nodes may not both be chosen. A wireless user can either specify a topology, such as, 2 WiFi end-nodes connected to an Access Point or can pick 3 nodes and let the system decide the "available" topologies corresponding to those 3 nodes. Note that if a user chooses 3 wireless nodes and asks for a topology where 2 nodes are connected to an Access Point (configured in infra-structure mode), it may not be possible to get that topology because none of the chosen 3 nodes might be configured in the "infra-structure" mode. In other words, none of the chosen 3 nodes could be used as an "Access Point" without changing topologies of already existing slices.

3) Only 2 out of 3 variables, namely power, rate and frequency, can be independent. Once 2 out of the above 3 variables are chosen, the third is automatically fixed. These variables together determine a topology in a wireless network.

6.2 Admission Control

Coordination Aggregate (CA) component for a wireless subnet is expected to maintain state of the components belonging to the aggregate and perform admission control. Note that each component (such as, Access Point, Sensor Node etc.) will be advertising their

own availability of resources and the CA for the corresponding wireless subnet will aggregate the information from individual components to have a "global" view of resource availability within its aggregate subnet. This information can be used to allow or disallow request for a given slice.

6.3 Service Level Agreement (SLA)

Providing some sort of an SLA to a slice in wireless networks is clearly dependent on a combination of "resource availability advertisements", "monitoring" and "admission control". Resource availability advertisements by individual components within a wireless subnet enable the CA component to keep track of the "global" resource availability within the wireless subnet. This, in turn, helps the CA to determine the kind of performance guarantees that can be provided to a new slice. Monitoring plays a key role in ensuring that the state of the aggregate as perceived by the CA component does not change either due to misbehavior of the existing slices, or due to unexpected behavior of extraneous elements. Admission control being a controlled way of adding new load to the system would ensure, in the absence of misbehavior by existing slices or by extraneous wireless nodes, that SLAs would not be violated. Furthermore, monitoring would enable the CA to determine if and when the state of the network changes, and for how long, and estimate the violation of SLA.

6.4 Scheduling

Coordination Aggregate (CA) component for a wireless subnet would be responsible for scheduling experiments in the aggregate wireless subnet. After all, it has the complete visibility of the availability of resources and the needs of the experiments that are waiting to be scheduled.

6.5 Discovery

Coordination Aggregate (CA) component for a wireless subnet would be responsible for discovery of topology in the aggregate wireless subnet, mainly because it has the complete visibility of the aggregate wireless subnet in terms of transmit power, noise level, frequency, time-slots, codes, mobility etc. of each of the individual components in its aggregate wireless subnet.

6.6 Monitoring

Monitoring has to happen both at individual component level where a monitoring software on each component would oversee the adherence of the individual slivers (on the component) to their allocated "quota" of resources, and also at the aggregate level where the monitoring software of the aggregate component would oversee the adherence of the individual slices (in the wireless subnet) to their allocated "quota" of resources. In addition, the aggregate monitoring component would also oversee that no external source

of noise/interference disturbs the state of resource allocation in the corresponding wireless subnet aggregate.

It may not be feasible to prevent an external source from interfering with the state of a wireless network, but when such an event happens it would be useful to flag it to all relevant parties.

6.7 Measurement/Instrumentation

Measurement/Instrumentation can either be a "service" provided by GENI to the wireless users or it could be left up to the individual users to write their own instrumentation code. A hybrid model would be more appropriate where most of the common measurements could be collected (to avoid duplication of effort by each experimenter trying to gather the same measurements), and delivered to GENI users as per their requests by GENI measurement/instrumentation framework, and the experiment-specific measurements could be collected by the individual experimenters by writing their own instrumentation code.

7 Additional Thoughts