Usage Scenarios for Experiment Control

Peter Steenkiste et al.
Draft of Dec 28, 2007

Goal

At the first GENI engineering people, a number of people indicated that "GENI should be easy to use". This sets the high level goal for the services that GENI users (experimenters) will use to specify and control experiments. This document describes a number of usage scenarios that will help identify requirements for those services. The scenarios may have other uses as well.

The goal is not to identify requirements for the specification and control of all possible experiments. The focus of GENI is on large scale network architecture experiments, so experiments that focus on specific technologies, without broader architectural implications, are outside the scope. The distinction between these two types of experiments may not always be clear though.

Background

The Distributed Services Working Group gave a presentation at the GENI Transition Meeting held on July 10, 2007, summarizing that status of its work [1]. The slides present a high level view of the software architecture and a preliminary specification for a number of services. The high level architecture (slide 5) consists of an hourglass with substrate components at the bottom and distributed services at the top. These two layers are separated by a minimal GENI Management Core (GMC) that allows the distributed services to realize the GENI goals on top of the components.

Experiment control corresponds to the Experiment Support services discussed in slides 69-77. A more detailed discussion of the services can be found in Section 6 of [2]. Given the above architecture, the experiment support services will use APIs and functions provided by the GMC. These APIs are critical since they are likely to be longer lived than specific services. The text below focuses on what an end user will have to do to run an experiment, i.e. the focus is on the services that are needed. This will also help define the APIs in a top-down fashion. Controlling experiments potentially involves many services, which means that the usage scenarios will need to be very elaborate.

Several other GENI design documents [3] also describe usage scenarios for GENI or discuss experimental control. [4], for example, uses an example experiment to identify requirements for the control and management of wireless experiments.

Range of Experiments

GENI experiments can differ in many different ways:

- Running time, e.g. short runs for correctness testing versus long running, large scale experiments for performance evaluation with real users. There is a fair bit of experience with the former but not much with the latter. We look at this dimension in more detail below.
- Level of the technology being evaluation, e.g. services versus specific features of optical switches. The issue here is that what you need to specify (i.e. the nature of the controls) is very different and requires different expertise.

- Heterogeneity of the infrastructure used, e.g. backbone only versus end-to-end experiments that include wireless subnets, optical links, etc. This is a hard problem because of the many different diverse aspects of the experiment that need to be controlled. The community has relatively little experience with heterogeneous experiments.
- Novice users versus experts. Ideally, simple experiments would be easy to specify while complex experiments will necessarily be complex.
- Others?

These dimensions create a experimental space that the experiment support services need to cover. We are obviously not looking for different point solutions for each type of experiment since, over time, researchers will need to do experiments that corresponding to different regions in the experimental space.

The level of experience we have with experiment control differs quite a bit in different parts of the space. For example, we can consider the first dimension, and look at three different stages an implementation goes through during the experimentation process (based on text from Steve Schwab):

1 -- **Debugging for initial functionality and correctness** - often this is about making the system work, and work robustly, on a small to moderate number of nodes. This is similar to typical software development in a distributed environment. A number of groups have found that a local or remote controlled testbeds (such a Emulab, DETER, CMU wireless emulator, etc.) are attractive platforms for this style of development. Projecting forward, this suggests that researchers may do the first-cut implementation on a local testbed (e.g. an Emulab cluster), and then migrate to the distributed/wide-area environment (GENI). So tools for debugging, replicating, and replaying problems observed on GENI but not on a local cluster are going to be key to supporting users. At this stage they are likely to use few components in GENI.

2 -- **Performance analysis and tuning "at scale", using synthetic or emulated workloads** - Ideally the experimental results can be compared against a model or simulation to assess whether expectations are met. This model could also be improved over time as a consequence of measuring, investigating root causes, and tuning the implementation. The unexpected interactions of multiple low-level implementation or environment features need to be ferreted out, hopefully resulting in enough understanding to make significant improvements. This step is mainly a challenge because setting up foreground and background workloads is a headache, as well as realistic configurations for every other 'ordinary' or typical aspect of a network, e.g. DNS, routing, addressing, etc. Foreground means a workload that uses the new features of the implementation. Background refers to all the other traffic that we would expect to be present in the real deployed environment, but which doesn't get handled in any special or novel way by the new implementation. A good deal of data collection, repeated runs of various configurations with different parameters, and basic scientific database/notebook support is what is needed here.

3 -- **Operating the implementation with a workload generated by "real" users** - This means to actually run things in a manner so that a significant user base actually makes use of the implementation in the real world, and the experimenter gets to collect a large body of significant real data. This step is very similar to a pilot or beta deployment of the technology (correct?). This is one of the promises of GENI, but it depends on signing up real users. This will only work if there are enough early adopters, and things can be made robust enough to not scare them away. The main difference between step #2 and step #3 is probably the addition of 'management tools' to let someone turn a prototype into a real world (beta-quality) service.

When defining usage scenarios, it may be tempting to pick scenarios that we are familiar with. While it is certainly important that GENI will support experiments we that we already perform

today, this is not an effective way of identifying requirements for experiment support services. We should instead focus on experiments we have no or limited experience with.

Scenario 1: Mobile user scenario (heterogeneity)

Scenario: A group of researchers has developed architectural mechanisms to support mobile users accessing the Internet. They include a new addressing scheme, mechanisms for route adaptation based on user location, and fast proxy deployment to help the user access Internet services. The researchers want to test their prototypes using realistic mobile users, i.e. the dynamics of the wireless connections (e.g. variable bandwidth and delay, temporary disconnects, etc.) must be realistic, using either emulation or modeling. They also want realistic workloads at the level of both individual users and groups of "nearby" users.

Specification: The researchers will need to specify the following aspects of the experiment:

- Resources needed;
- Nodes and their properties;
- "Network location", i.e. location of nodes in a topology;
- Mobility of the wireless nodes. What we mostly care about is the impact of the mobility on (1) the wireless transmission (e.g. what nodes can communicate and the quality of the channel) and (2) the network topology (e.g. AP association). This could be specified in a number of ways. Users could specify how nodes move in a physical space with certain properties; the wireless transmission and network topology properties can then be derived automatically. Alternatively, the users can directly specify the transmission and technology properties, e.g. based on traces.
- Configuration of the nodes: what app should run on the nodes (OS, services, etc.) and how should the node and the software be configured. This includes installing the experimental technologies being tested, e.g. new addressing scheme, routing, etc.
- Traffic loads, e.g. traffic generator or real applications;
- Event timing (runtime control), e.g. the user may want to control the timing of certain events, e.g. node movement, traffic load changes, fault injection, events related to the new technologies, etc.
- Experimental data to collect, including both generic measurements (and thus could potentially be collected by GENI services) and measurements that are specific to the new technologies (collected by probes built and deployed by the user).
- Exception handling, i.e. what should happen if something fails unexpectedly (i.e. other than injected faults);
- Anything else??

Run experiment: To run an experiment, the user goes through the following steps (I believe Jay had a slide on this – expand later):

- Specify the experiment
- Allocate resources
- Configure the resources for the experiment
- Run experiment

- Possibly monitor experiment in real time (especially for long running experiments)
- Collect measurement results
- Clean up, e.g. free resources, etc.

Post mortem: After the experiment is completed a number of other steps are needed:

- The configuration, collection, validation, and publication of measured data. There should be enough information associated with the data so the experiment is fully described (and can be repeated? – code in next bullet).
- The documentation, publication, and maintenance of experimental software for reuse by researchers. The software can be reused for a number of reason: repeat the experiments (e.g. for comparison purposes or to recreate the data collected), as a building block for richer experiments, and a starting point for new experiments.

Variants:

- Same experiment, but replace the emulated wireless subnets by a real user population. This may change the experiment and its specification in a number of ways: exception handling may become more important, and privacy and security becomes more of a concern. Also, since some of the wireless devices may not be reconfigurable, we may want to allow them to keep their old software and emulate the modifications (e.g. new addressing) on the GENI device they directly communicate with.
- The user wants to repeat the same experiment a thousand times with different parameters and traffic loads. What changes?

Scenario 2: Security (focus??)

Scenario: A group or researchers defined a new security architecture that combines a novel key distribution infrastructure with mechanisms on the end systems, in access routers, and in the optical backbone. They want to run an experiment that evaluates how well the design deals with different types of known DOS attacks. (This is naïve – needs to get fixed)

Specification: Similar to Scenario 1, except:

- No mobility issues
- Need ability to replay old attacks
- Containing the attacks to the allocated slices seems more critical here, but that is probably not part of the experiment specification or control.

Run experiment: the same although some of the details change in obvious ways?

Post mortem: similar as scenario 1.

Variants:

- Create new attacks: changes the traffic load that is placed on the system
- Long running attacks with artificial traffic loads and "red teams" trying to bring down the network. Changes who can generate traffic.
- Long running experiments involving real user traffic. Changes the security, privacy and exception handling.

Scenario 3: Getting real (scaling up, combining components)

Scenario: The research groups in Scenarios 1 and 2 get together and figure out how their techniques may or may not work together. They want to run experiments that combine components of their separate experiments.

Specification: Some kind of combination of the specifications of experiments in Scenarios 1 and 2. Can reuse stored experimental software.

Run experiment: Similar as first two scenarios.

Post mortem: Similar to first two scenarios.

Variants: Same variants.

Scenario 4: Novice users (focus??)

Scenario: For any of the above scenarios, researchers want to create "canned" instances that can be run as course projects by undergraduates in a networking course. The students will need to be able to change certain aspects of the experiments (e.g. some parameters, maybe some of the applications on the endpoints), but should not have to deal with the full complexity of the experiments.

References

[1] GENI Distributed Services Preliminary Requirements and Design, available from http://geni.net/office/office_events_71007.html

[2] GENI Distributed Services, Thomas Anderson, Amin Vahdat (Eds), GDD-06-24 available at http://www.geni.net/GDD/GDD-06-24.pdf, November 2006.

[3] GENI Design Documents, http://www.geni.net/documents.html

[4] Requirements for Wireless GENI Experiment Control and Management, Sanjoy Paul, Sachin Ganu, Pandurang Kamat, Elizabeth Belding Royer, GDD-07-43 available at http://www.geni.net/GDD/GDD-07-43.pdf , March 2007.