Facilitating Interoperation in Research Cyberinfrastructure
Workshop on the Development of a Next Generation Cyberinfrastructure
Mark Berman, GENI Project Office

Not too long ago, the web community discovered the potential for innovation via mashup, the unanticipated and intentionally unmanaged combination of seemingly disparate capabilities by creative users with diverse interests and skills. Arming a large user base with open environments that encourage innovation has proven to be a simple recipe for a large number of new innovations, ranging from the stupid to the spectacular.

Of course, exchanging data and computation in unexpected ways is not a unique web phenomenon. Ask anyone who has ever performed a database join or executed a UNIX command pipeline.

The research cyberinfrastructure community is well positioned to benefit from a similar approach. Testbeds initially designed for specific purposes in one community are likely candidates for deconstruction and recombination. We should begin by enabling the obvious pairings; they offer sufficient benefit to merit our effort. If additional, unanticipated combinations arise, that result will be all the better.

Past experience suggests a small number of strategies that are likely to lead to rapid growth in user base and encourage creative application.

- **Embrace liberal access policies.** For the purposes of this workshop, the first step to interoperability among cyberinfrastructure testbeds is probably for testbed managers to grant routine access to each other’s user communities. Innovation comes from routine use, particularly by students. It’s often too hard to set up coherent configurations that involve heterogeneous resources. Special permission is required, and resource reservation and configuration are accomplished via telephone, rather than programmatically. This situation may be OK for a demo, if you know the right people to get on the phone, but it’s not going to happen on a whim for a student with a clever idea.

- **Establish simple, effective paths for data movement.** Security and performance considerations often drive the design of our research cyberinfrastructure to testbed-specific networking configurations, making internetworked configurations more challenging. By shifting to greater reliance on dynamic provisioning services available from research and education networks, we can lower barriers to quick provisioning of multi-testbed research configurations.

- **Cultivate a thriving ecosystem of researcher support tools.** Decoupling and providing open API access to various testbed functions (e.g., resource
discovery, reservation, configuration, experiment orchestration, data gathering, and analysis) is perhaps the quickest way to expand the population of contributors to testbed design and development.

- **Encourage transparent and maximally permissive policy.** Policies that restrict full access to cyberinfrastructure are an unfortunate reality of our world. As we work to enable interoperation among research cyberinfrastructure testbeds, we should keep the resulting system as open as possible, while respecting the limitations of the various participants. There are a handful of elements that will facilitate this process.

  - **Maintain local autonomy over policy definition and enforcement.** Each participant in an interoperating federation should retain the responsibility for defining and enforcing its own local policy. There are several reasons for this approach. First, it is the resource owner who is responsible for the correct administration of policy at each testbed. Second, because a testbed may well be participating in multiple federation agreements, and serving its own local users outside of any federation, no other authority can be expected to have full knowledge of the complete set of policies to be enforced. Finally, a testbed will generally already have an existing admission and allocation procedure encoded in software, saving additional development effort.

  - **Use an explicit, externally represented, and transparent policy mechanism.** Policies that are deeply embedded into testbed code are harder to understand, modify, and customize than those that are explicitly represented in an externalized format. When restrictions are exposed to users and to the software tools that support users, it is easier for them to design configurations that meet their needs, while working within the rules.