Developing Common Abstractions for Improving Interoperability of Federated Cyberinfrastructure

Zongming Fei and James Griffioen

Laboratory for Advanced Networking
Department of Computer Science, University of Kentucky
{fei,griff}@netlab.uky.edu

1 Introduction

Federation of resources from different and diverse components into an integrated networking cyberinfrastructure is attractive, because it can make more efficient use of available resources and, more importantly, it enables larger scale experiments that cannot be supported before. One prominent problem facing a federated system is interoperability, especially for one with heterogeneous components. Focusing on resource sharing, we can define three levels of interoperability for a federated cyberinfrastructure from a user’s perspective.

The basic level of interoperability requires that a user of any component can use resources from other components of a federated system, subject to policies of these components. This needs mechanisms for authentication and authorization of a user from a different component. The credential of a user from one component can be used for access to resources from other components. With this level of interoperability, a user may have to use resources from different components at different time or for different experiments.

The intermediate level of interoperability requires that a user can use resources from different components at the same time, or within the same experiment. We need to make sure that the elements from different components can be connected to form a networked experiment and can communicate with each other using some selected protocols.

The advanced level of interoperability requires that the user has access to a unified interface to interact with all components of a federated system and resources from these components. Because of the heterogeneity of components, it is often the case that when a new component is federated into the system, the user has to learn new concepts and/or new tools in order to use the resources from that component. This can pose a substantial burden on the user and make the federated infrastructure under-utilized. With the advanced level of interoperability, the user can use the same interface and tool to access all components of a federated system.

We are interested in improving the interoperability level of federated cyberinfrastructure. Our approach is to develop common abstractions and define common interfaces for both inter-component communications and for user-system interactions. The objective is to hide the differences from the users while maintaining unique features provided by each component. We describe our experience in building GENI aggregates and developing GENI software, and extend it to address the interoperability issue of federated cyberinfrastructure.

1A component can have policies such as giving priority to those users within the same organization or who have made reservations.
2 Developing Common Abstractions

A federated system can be built either bottom-up or top-down. In the bottom-up approach, components are developed more or less independently, with no or few common designs to be followed. In GENI, there are different control frameworks that are developed relatively independently. When these components are federated, there are significant differences between them. When requesting resources from these components, a user may need to master separate conceptual models.

For example, when requesting resources from PlanetLab control framework, a user gets a group of independent PC/VMs, without any links connecting them explicitly. The communications among these PCs are through usual Internet connections. We have a different model when requesting resources from ProtoGENI. A user can specify a set of PC/VMs and the topology about how they are connected. These links will constitute the data plane for experiment traffic to go through. The ORBIT control framework allocates a set of wireless nodes that can be connected to a WiMAX access point. The FOAM aggregate manager can allocate a flowspace from an OpenFlow-enabled switch. The user can write a controller to manage the flowspace for the experiment. The observation is that a common abstraction can be defined for different control frameworks so that the user does not have to deal with them in different ways. The unified approach can help users access different resources more easily.

In the top-down approach, we can have a unified conceptual structure. Components of a federated system all follow the same framework. These homogeneous components are much easier to be federated. For example, different ProtoGENI aggregates in GENI all run the same code base, but are responsible for different sets of resources. One problem we found in these systems is that there are different tools developed over time by different aggregates, such as allocation of resources, control of elements in an experiment, instrumentation and measurement of performance, and data analysis and archive. Building a unified seamless user interface needs to have some common abstractions so that it can integrate various tools and give users a unified view of the system. The users can also have a single point of entry to explore the system.

3 Previous Experience with GENI

We have been actively participating in GENI development efforts. The INSTOOLS project was an early GENI project focusing on instrumentation and measurement of user experiments. The ShadowNet project explored virtualizing physical Juniper routers so that user experiments can include a virtual Juniper router. The GEMINI project was one of two major measurement projects in GENI and it helps users collect measurement data.

More recently, we have been working on the GENI Desktop project, which has been used widely by GENI researchers for their experiments and educators for their classes. The goal is to develop a unified framework and interface for accessing, controlling, and interacting with GENI experiments throughout the life cycle of an experiment. GENI Desktop identified the key abstractions that all tools share in common, and then developed a single interface around those abstractions. In GENI, the unifying abstractions shared by all GENI tools are slices and the slivers that make up a slice. Users select a slice to work and then interact with the slice’s resources via a single common interface. The operations that can be applied to resources are defined as modules in the GENI Desktop. Example services that can be invoked on slice resources (i.e., slivers/nodes) include node login services (e.g., ssh), file management, code execution, instrumentation, measurement and monitoring, and archiving of experiment data. GENI Desktop is extensible in the sense that developers can write their own modules using the common abstraction defined and thus add new services and functionalities to GENI Desktop.