GENI Experiment Control Using Gush

Jeannie Albrecht and Amin Vahdat

Williams College and UC San Diego

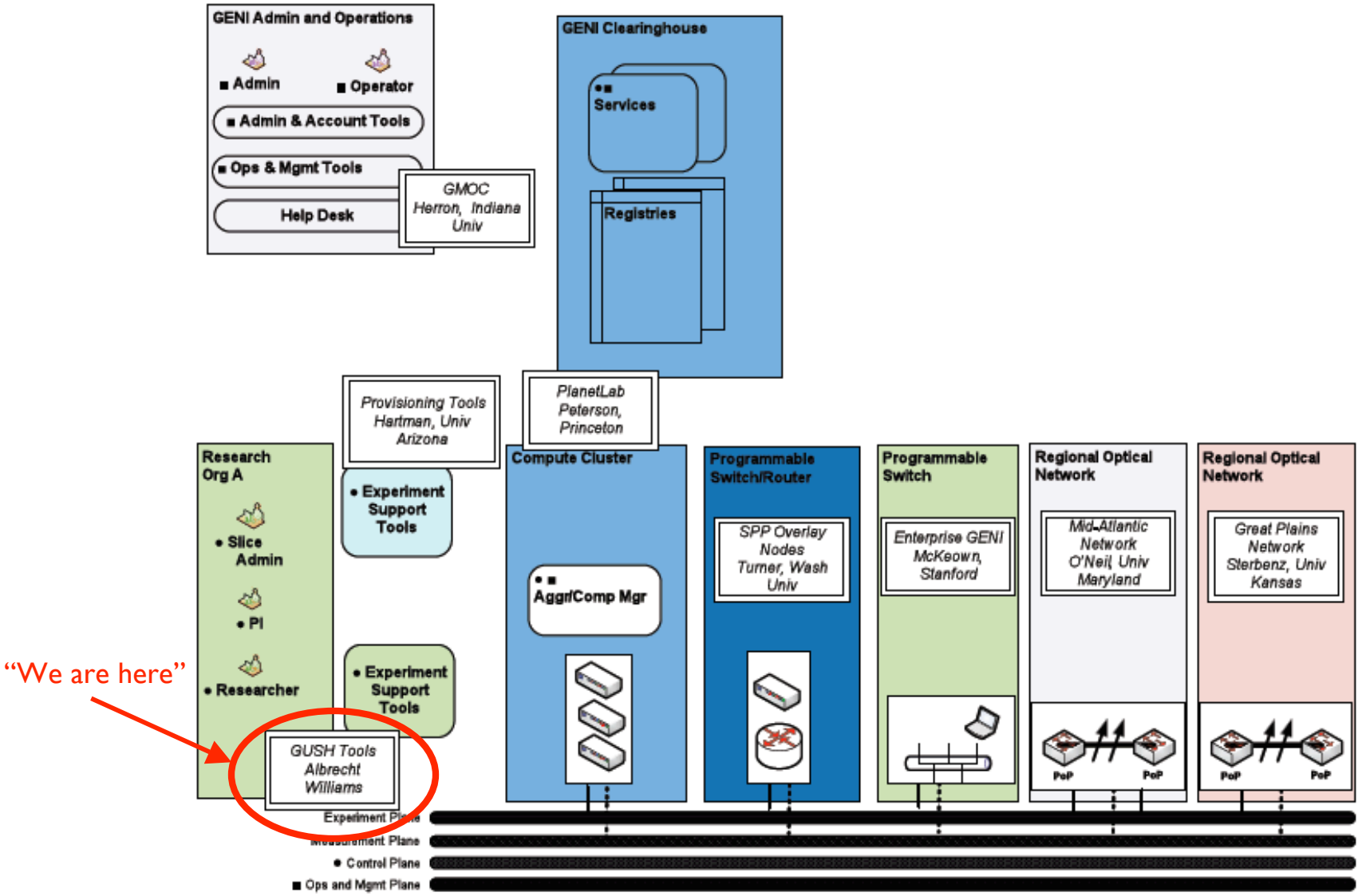
UCSDCSE

Computer Science and Engineering

Overview

- How do we actually use GENI?
- Goal: Develop abstractions and tools for addressing the challenges of managing distributed applications
  - Make it easy for a range of users to run a variety of experiments on GENI
- Talk outline
  - Summarize existing work and design of Gush
  - Describe implementation/integration plan and timeline
  - Stimulate a broader discussion about the general requirements of an "experiment controller" in GENI

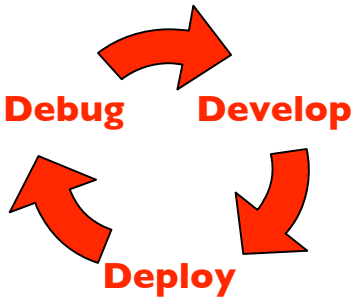
Cluster B: PlanetLab CF



"We are here"

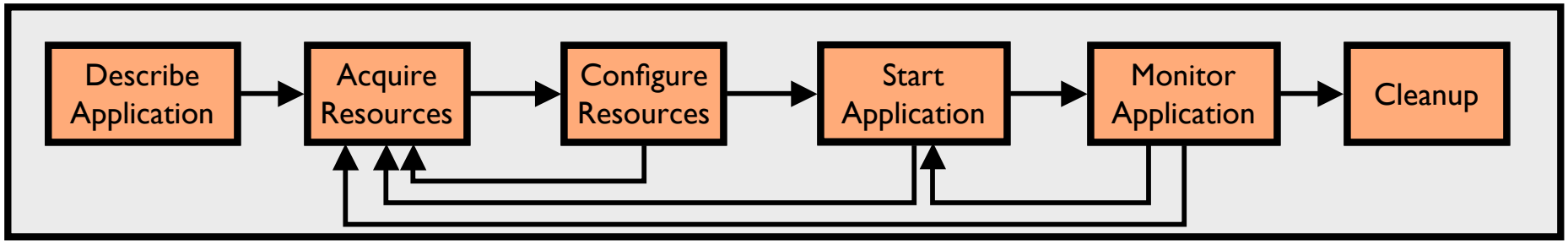
Running an Experiment

- Suppose we have written our software and are ready to deploy on PlanetLab for the first time
- We could…
  1. Connect to each of the 800+ PlanetLab machines
  2. Download software (no common file system)
  3. Install software
  4. Run application and analyze performance
  5. Check for errors on each machine
  6. When we find an error, we start all over…
- Or we could use Gush



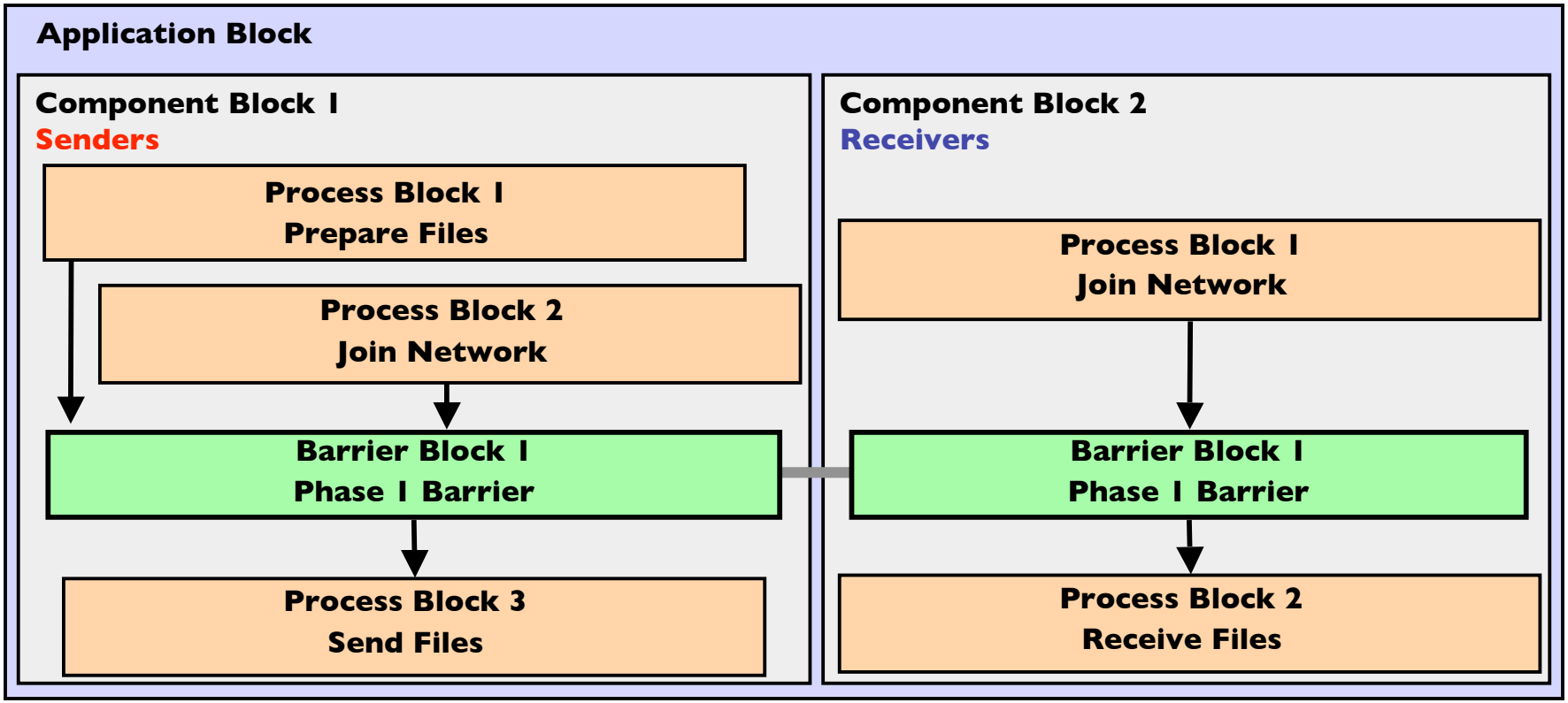
Gush

- A distributed application management infrastructure
  - Extends functionality of Plush to support experiment control on GENI
  - Designed to simplify deployment of distributed applications
  - Provides abstractions for configuration and management
  - Allows users to "remotely control" computers running distributed applications worldwide

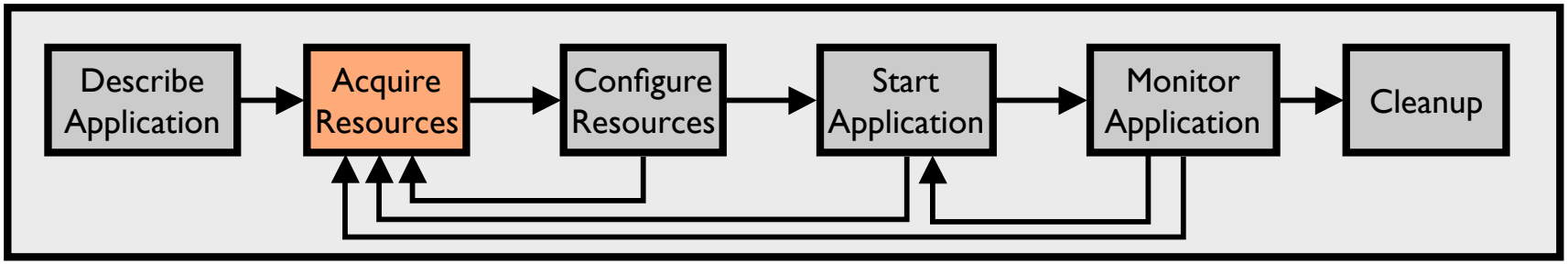


Step 1: Describe Application

- Describe experiment using application "building blocks"
- Create customized control flow for distributed applications
- Application specification blocks are described using XML

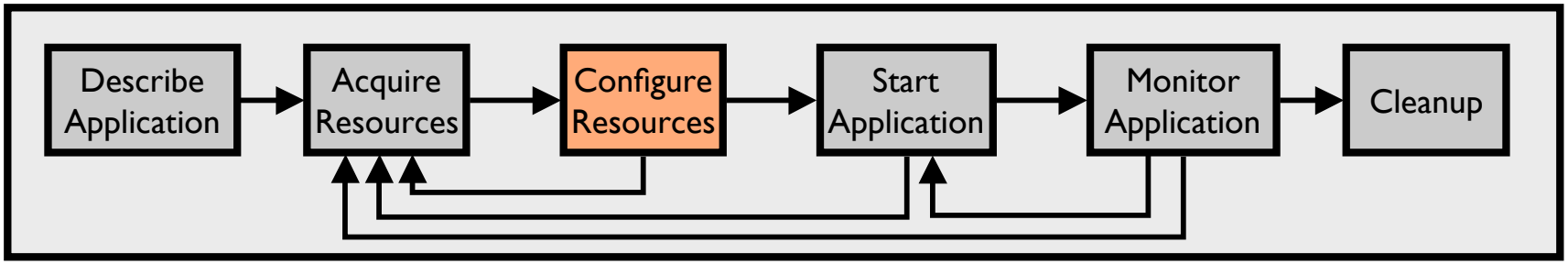


Step 2: Acquire Resources

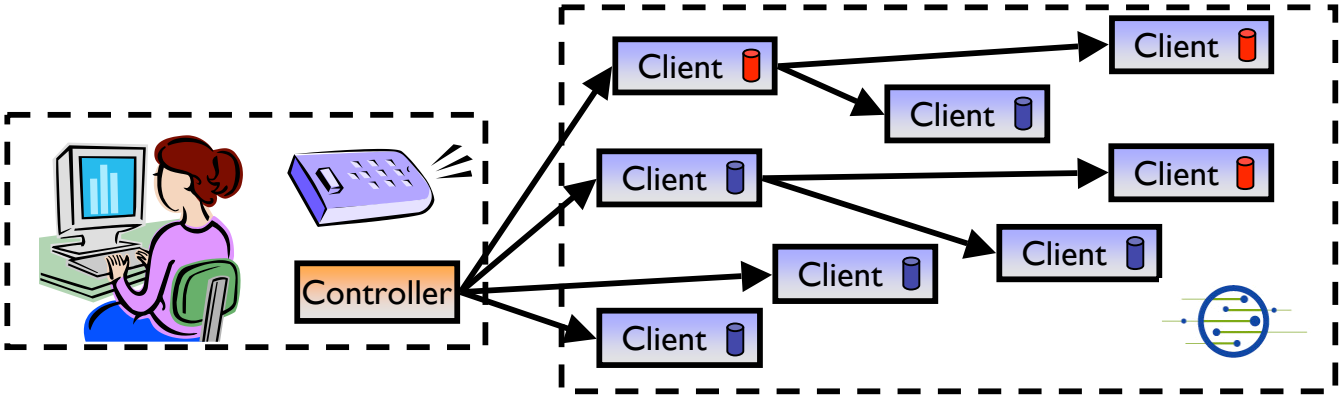


- How can we find "good" machines?
  - We want machines with specific characteristics
  - PlanetLab services perform *resource discovery* to find machines that satisfy our requirements
- Gush interfaces directly with these services
  - Eventually Gush will talk directly to GENI Clearinghouses to find resources

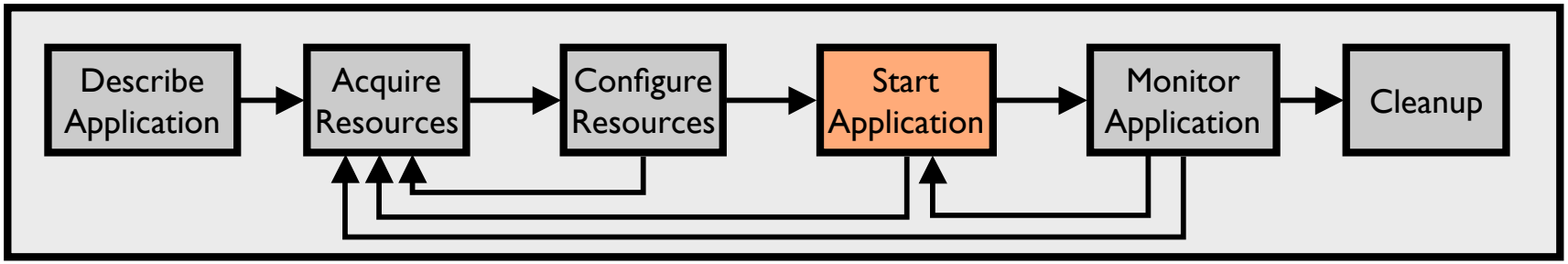
Step 3: Configure Resources



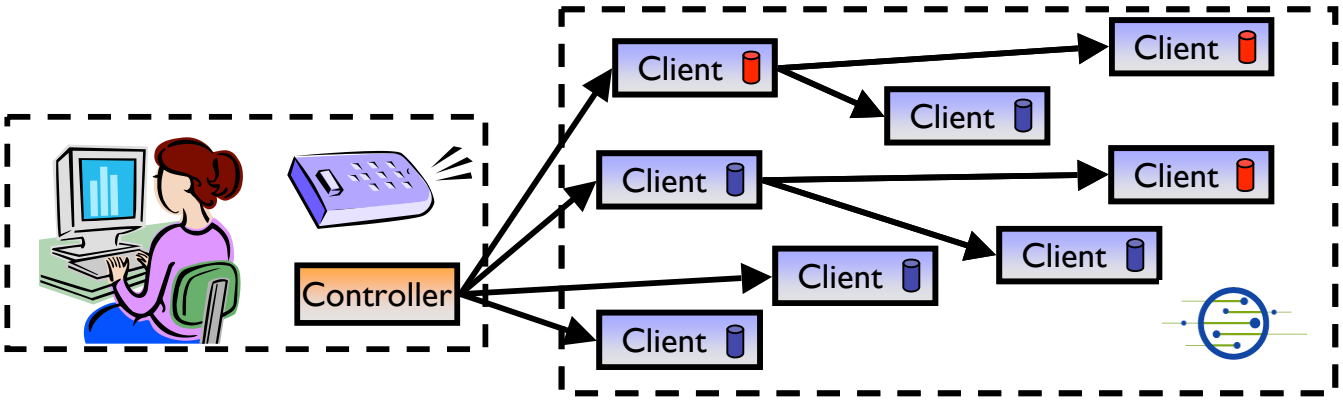
- Connect to and configure selected resources
  - Optionally create a tree for achieving scalability in communication
  - **Controller** "remotely controls" the **clients** on our behalf
  - Install software on clients (some are senders, some are receivers)



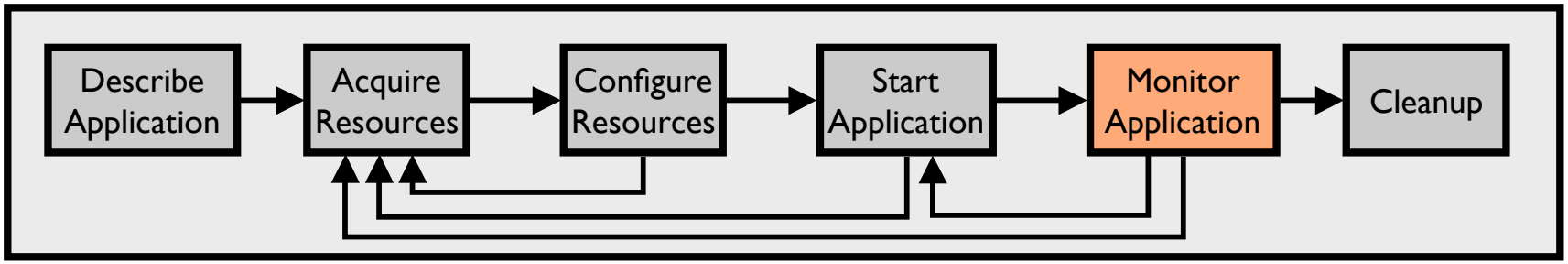
Step 4: Start Application



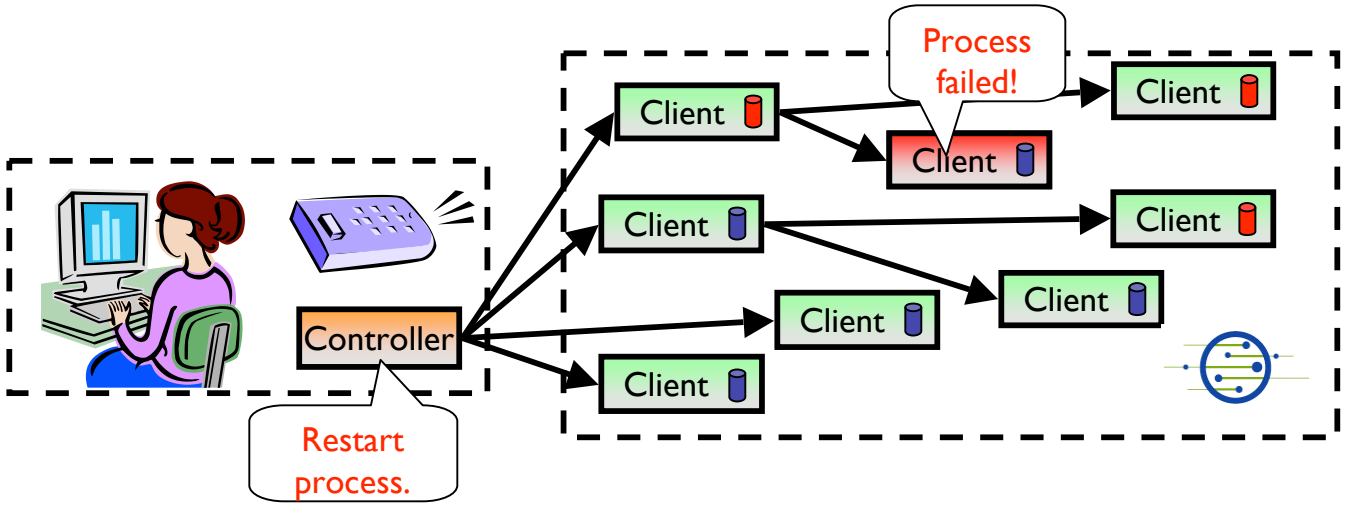
- Controller issues commands to clients telling them to start running our application
  - Senders begin running sender processes
  - Receivers begin running receiver processes



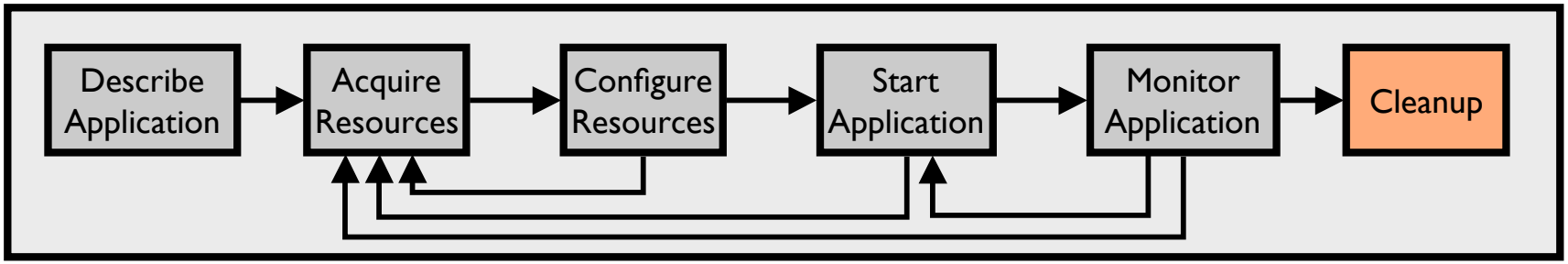
Step 5: Monitor Application



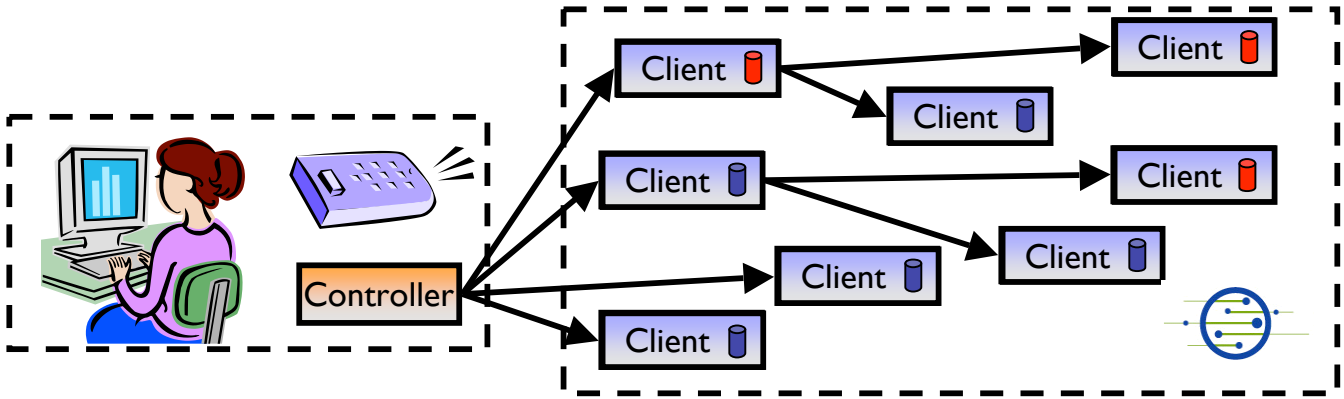
- We want to make sure the processes keep running
- Gush clients monitor experiment processes for failures
  - If a failure is detected, client notifies controller
  - Controller decides to tell client to restart failed program or process



Step 6: Cleanup

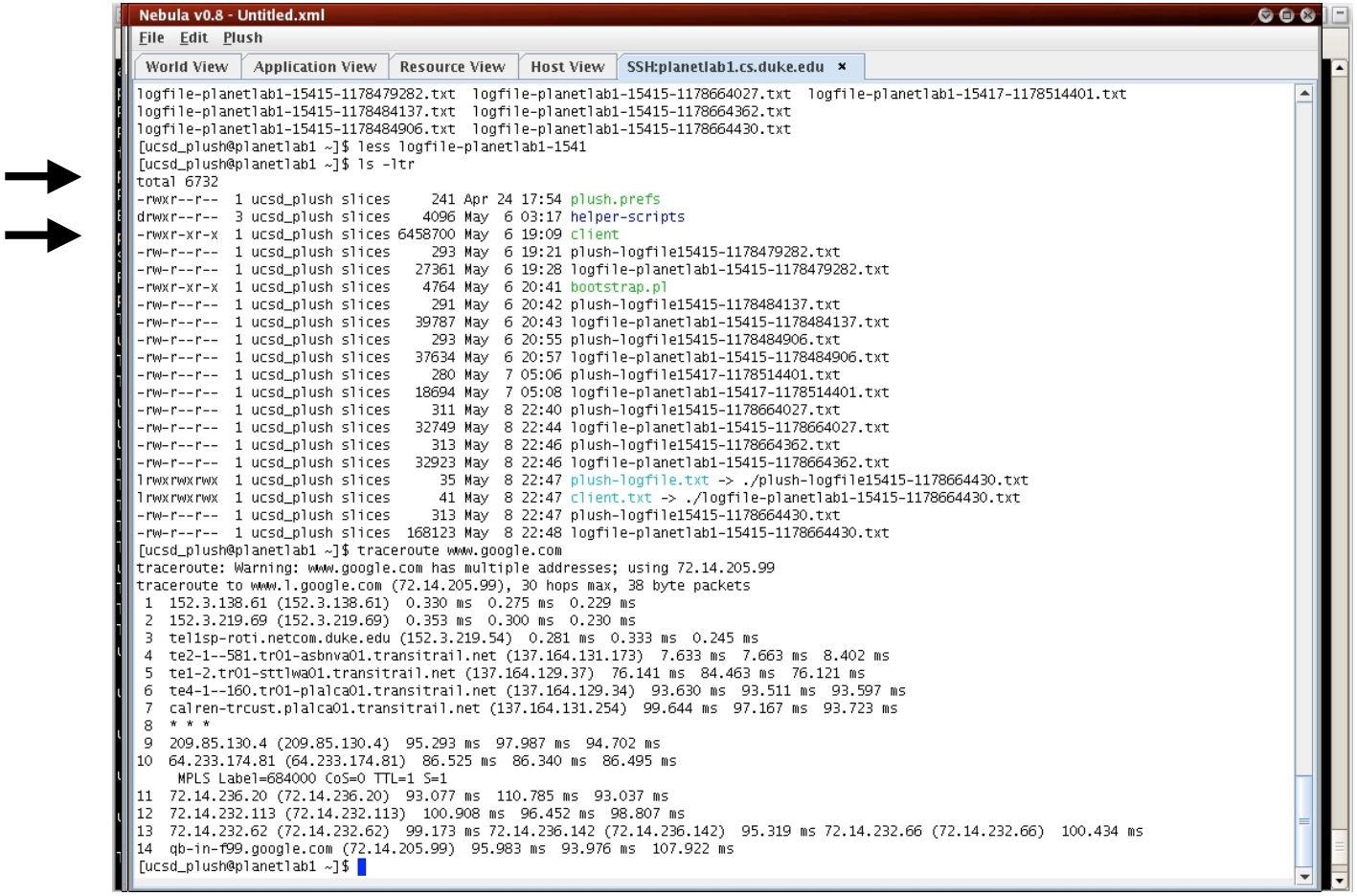


- Gush clients make sure all programs exited cleanly
- Remove logs and software from remote machines
- Disconnect clients from controller



Gush User Interfaces

- Command-line interface used to interact with applications
  - Provides single point of control for remotely controlling resources
- Nebula (GUI) allows users to describe, run, monitor, & visualize applications
- XML-RPC interface for managing applications programmatically



Current Status

- Gush works with current PlanetLab API
  - Will continue to modify Gush as GENI APIs are implemented and released
- Preliminary user study completed
- Student involvement
  - Undergraduate research projects at Williams College
    - Gush web interface (GoogleMaps)
    - Batch scheduler that uses Gush XML-RPC interface for "automated" experiment control on PlanetLab
  - Gush in the classroom
    - "Bringing Big Systems to Small Schools: Distributed Systems for Undergraduates," To appear in SIGCSE 2009.

Summary

- Gush provides abstractions for managing a range of distributed applications on GENI
  - Provides three different user interfaces to meet the needs of a variety of researchers with varying levels of expertise
- Next steps (6-12 months)
  - Continue obtaining user feedback to enhance usability and provide additional functionality
  - Develop user documentation
  - Integrate with other Cluster B GENI services (e.g., Stork/Raven)
  - Demo different user interfaces at future GECs
- Long term goals
  - Build on prior Plush/Orca integration to support experiments on sensor/mobile networks (e.g., DieselNet)

Thanks!

For more info, visit

http://gush.cs.williams.edu

Email:

jeannie@cs.williams.edu

Discussion Questions

- What experiment control functionality do researchers want/need on GENI?
  - Support for both novice and advanced users
- How much control should be exposed?
- How do I specify an experiment?
- What is the API between experiment controllers and the underlying control frameworks?
- What is the API between the experiment controller and GENI Clearinghouses?