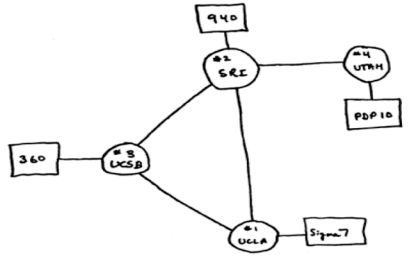
Remote Control: Distributed Application Configuration, Management, and Visualization with Plush

Jeannie Albrecht, Ryan Braud, Darren Dao,
Nikolay Topilski, Christopher Tuttle,
Alex C. Snoeren, and Amin Vahdat

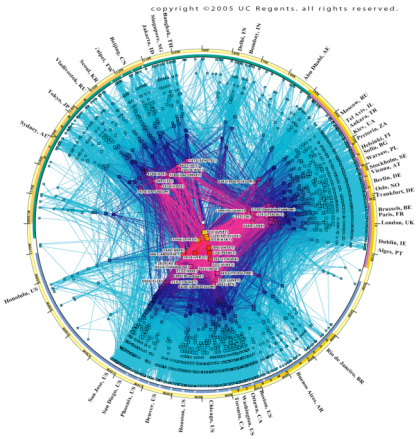
Williams College & UC San Diego

Meeting Current Demands

- 1 billion people worldwide use the Internet
- 500 million people surf the Web each week
- Services must support increasing user demand
  - Online banking, media downloads, news websites, search engines
- Demand is only satisfied using distributed applications running on tens of thousands of resources worldwide
  - Google uses 450,000+!



1969 Internet Map



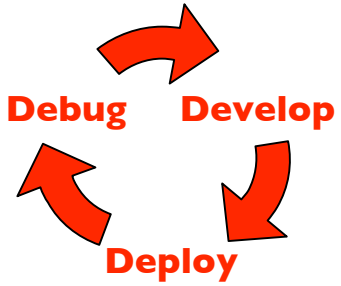
2005 Internet Map

Google Lime wire CNN.com BitTorrent Bank of America Higher Standards

Distributed Applications

- Have many advantages, but also introduce new challenges
  - + Increased computing power can improve scalability and fault tolerance
  - − Building and managing distributed applications is difficult

- Building applications: Develop-Deploy-Debug cycle
  - Develop software
  - Deploy on distributed machines
  - Debug code when problems arise



- Key management challenges
  - Locating and configuring distributed resources
  - Detecting and recovering from failures
  - Achieving availability, scalability, fault tolerance

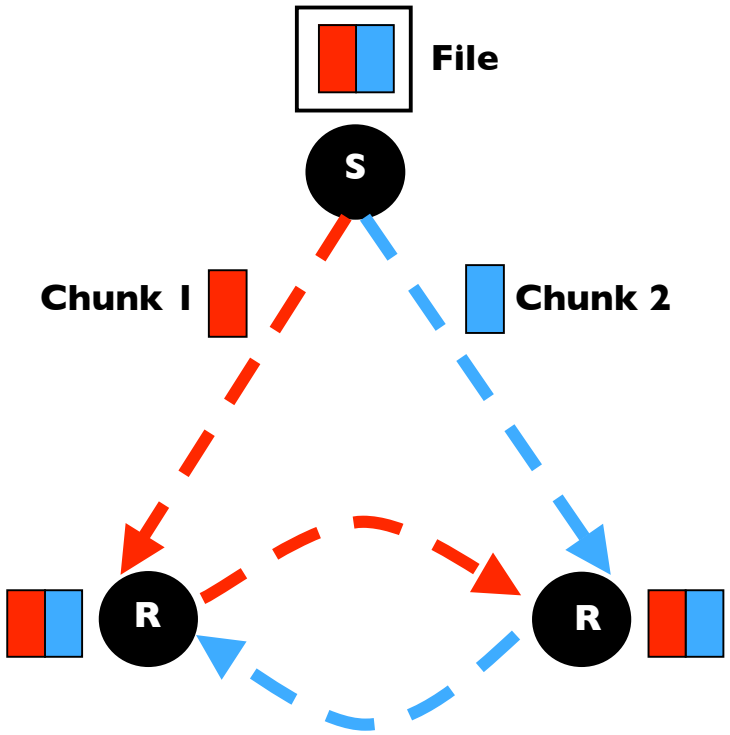
Overview

- Goal: Develop abstractions for addressing the challenges of managing distributed applications
  - We want to provide support for a *broad range of applications* run in a *variety of execution environments*

- Talk overview
  - Discuss a specific distributed application: ByteTorrent
  - Examine a specific execution environment: PlanetLab
  - Configure & manage ByteTorrent on PlanetLab: Plush
  - Closing remarks

Example Application: ByteTorrent

- Suppose we build ByteTorrent, a "new" file distribution service
  - Sender (S) sends file to Receivers (R)
  - Sender splits large file into "chunks"
  - Two phases of execution
    - Phase 1 – Join ByteTorrent network
    - Phase 2 – Transfer file
- We want to evaluate performance achieved on resources spread across the wide-area



PLANETLAB

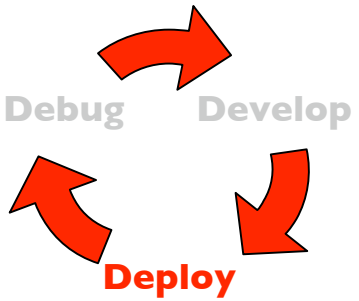
An open platform for developing, deploying, and accessing planetary-scale services

- Network of 800+ Linux computers at 400+ sites in 40+ countries
- Allows deployment of distributed applications around the world
- Can be a volatile working environment
  - High contention for machines (especially near paper deadlines)
  - Common problems: low disk space, clock skew, connection errors

http://www.planet-lab.org

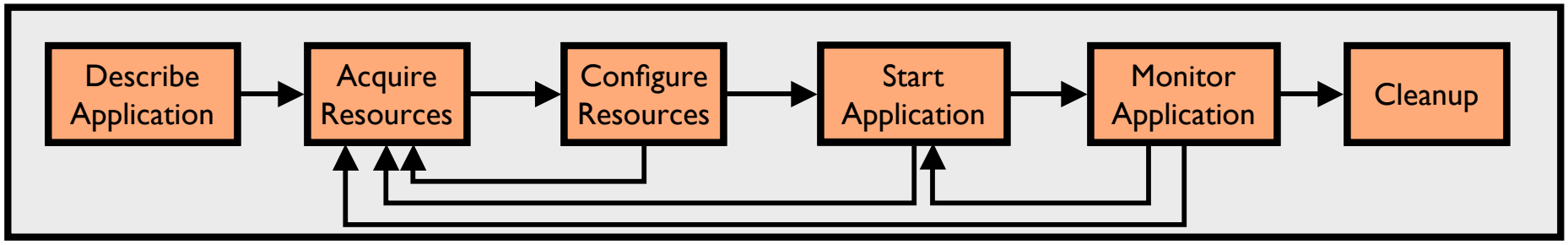
Deploying ByteTorrent

- 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 Plush



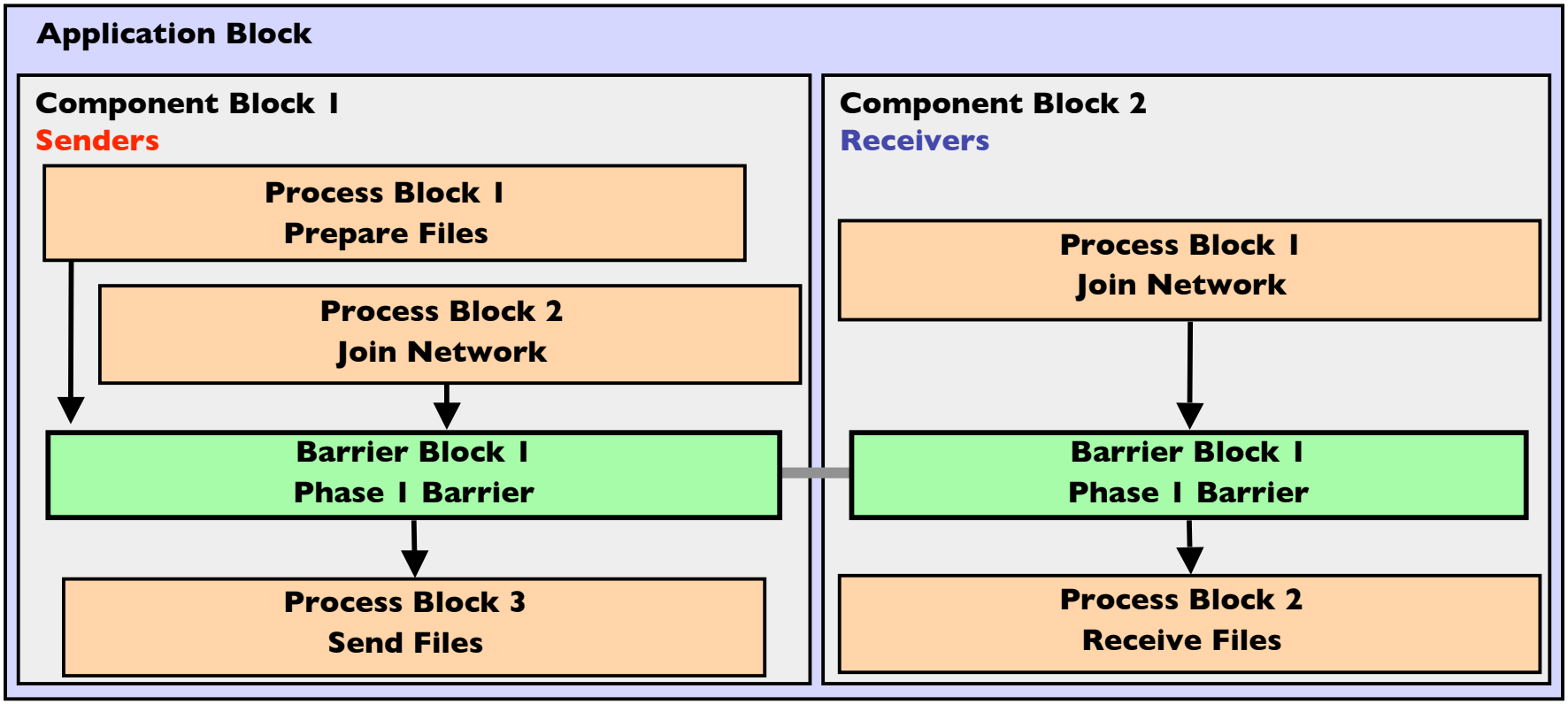
Plush

- A distributed application management infrastructure
  - Designed to simplify deployment of distributed applications
  - Provides abstractions for configuration and management
  - Allows users to "remotely control" computers running distributed applications worldwide

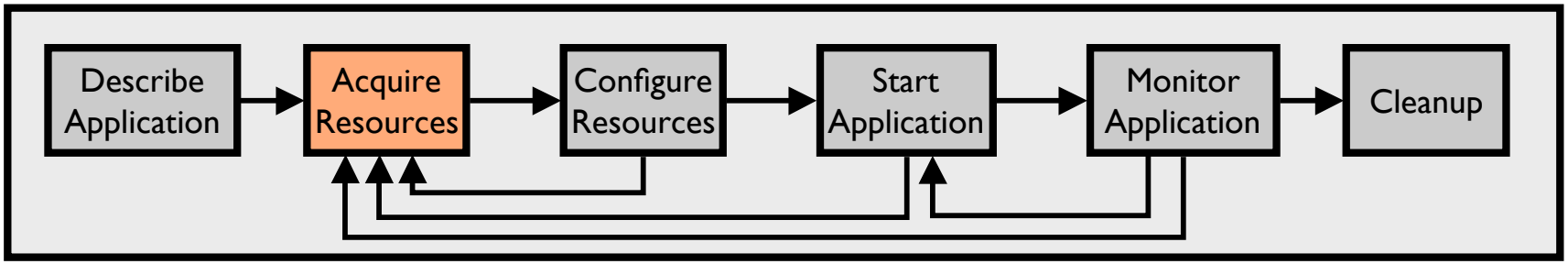


Step 1: Describe Application

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

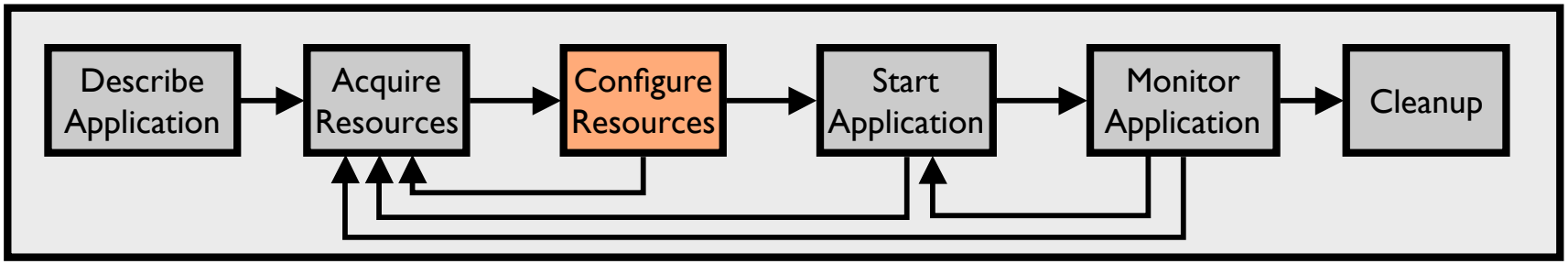


Step 2: Acquire Resources

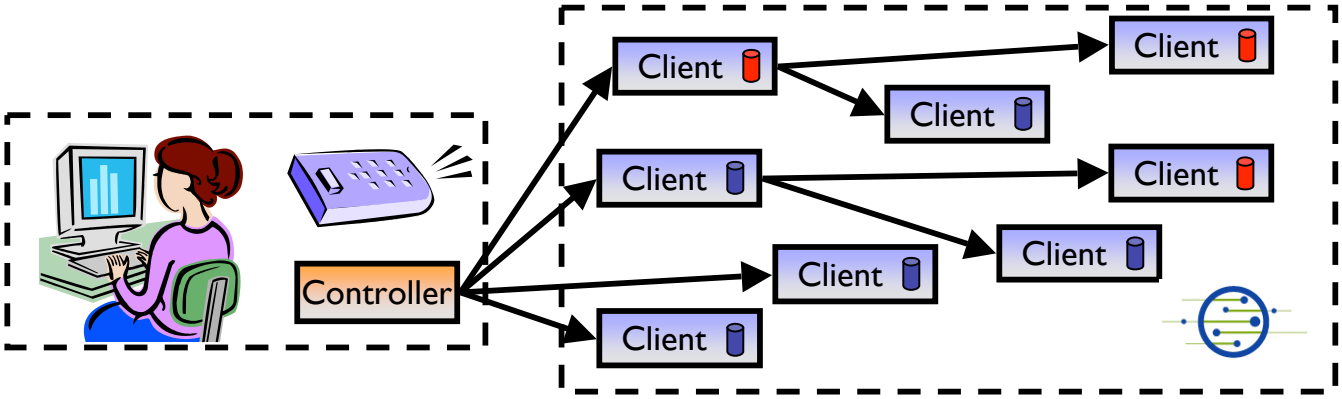


- How can we find "good" machines?
  - We want machines with specific characteristics
    - High bandwidth, fast processors, ample disk space
  - PlanetLab services perform *resource discovery*
  - Services find machines that satisfy our requirements
- Plush interfaces directly with these services

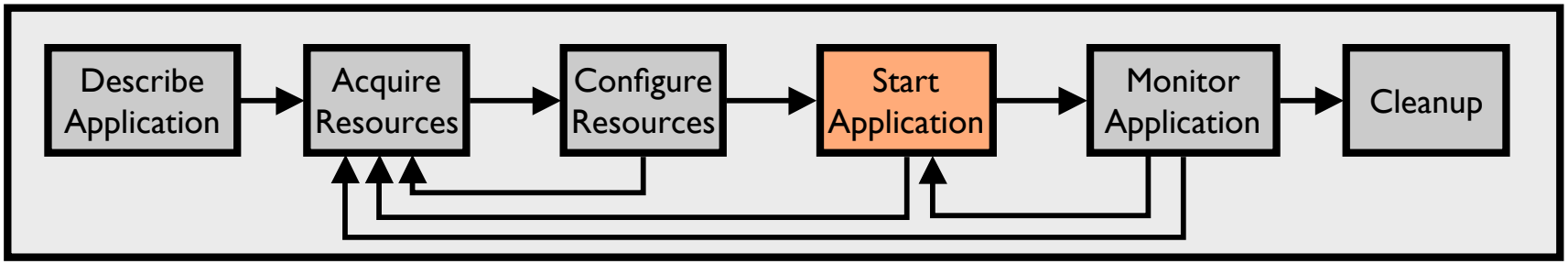
Step 3: Configure Resources



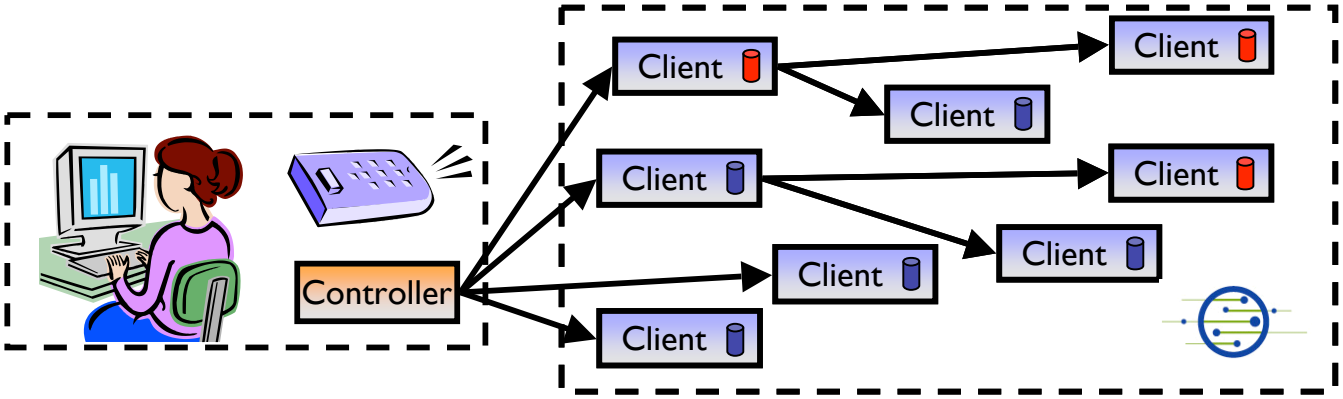
- Connect to and configure selected resources
  - Create a tree for achieving scalability in communication
  - **Controller** "remotely controls" the **clients** on our behalf
  - Install software on clients (some are senders, some are receivers)



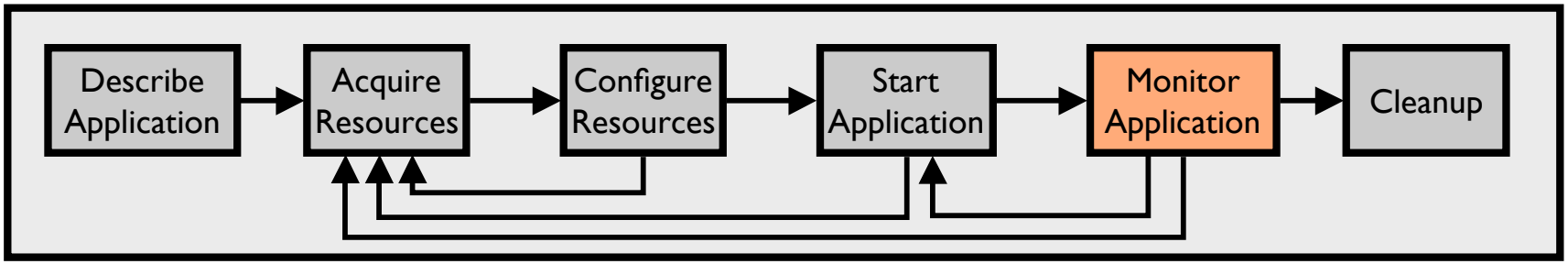
Step 4: Start Application



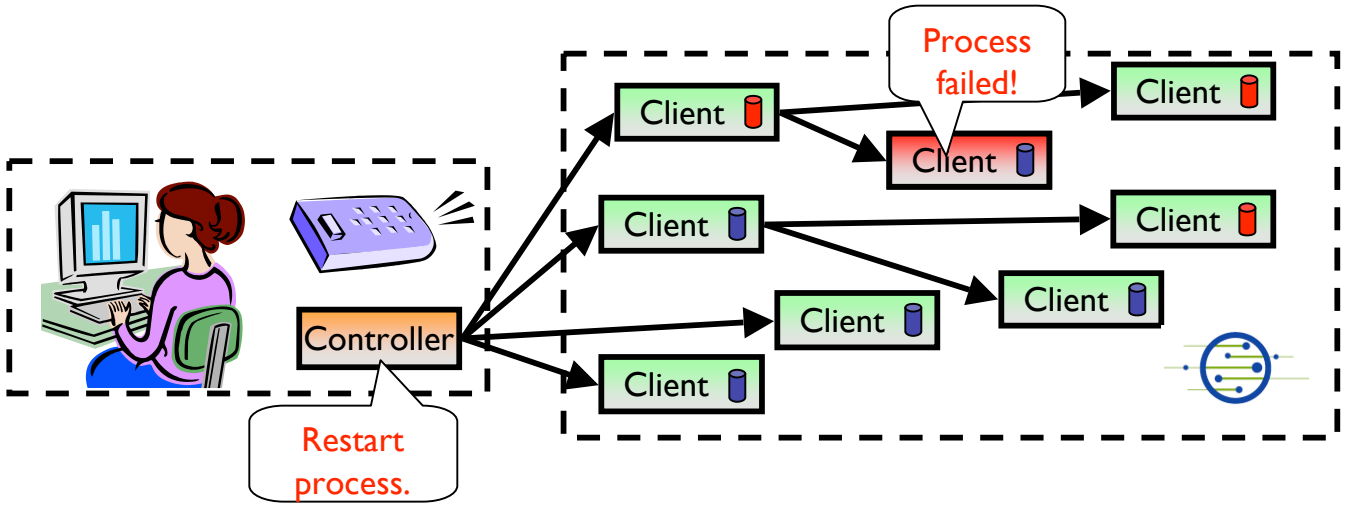
- Controller issues commands to clients telling them to start running our application
  - ByteTorrent senders begin running sender processes
  - ByteTorrent receivers begin running receiver processes



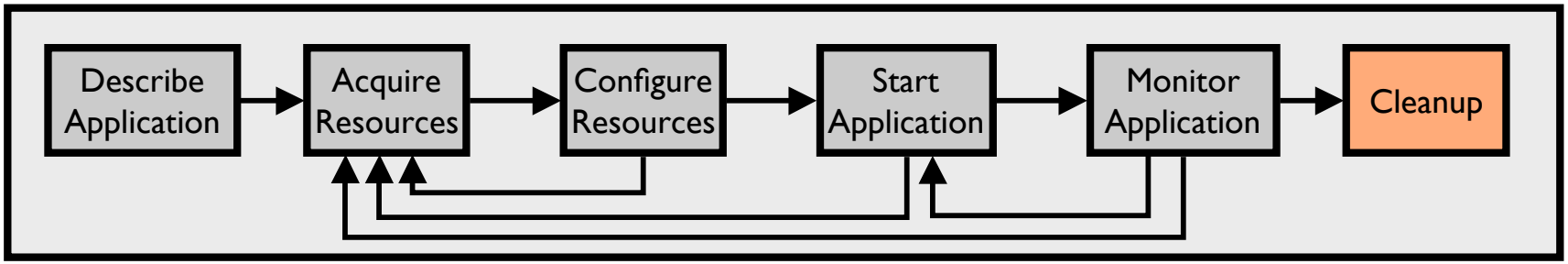
Step 5: Monitor Application



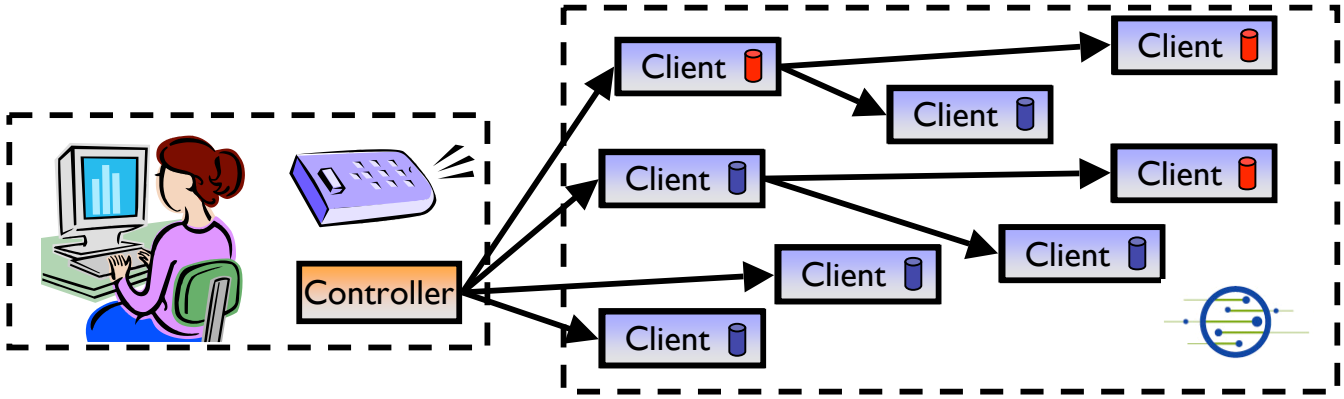
- We want to make sure the processes keep running
- Plush clients monitor ByteTorrent processes for failures
  - If a failure is detected, client notifies controller
  - Controller decides to tell client to restart failed program or process



Step 6: Cleanup

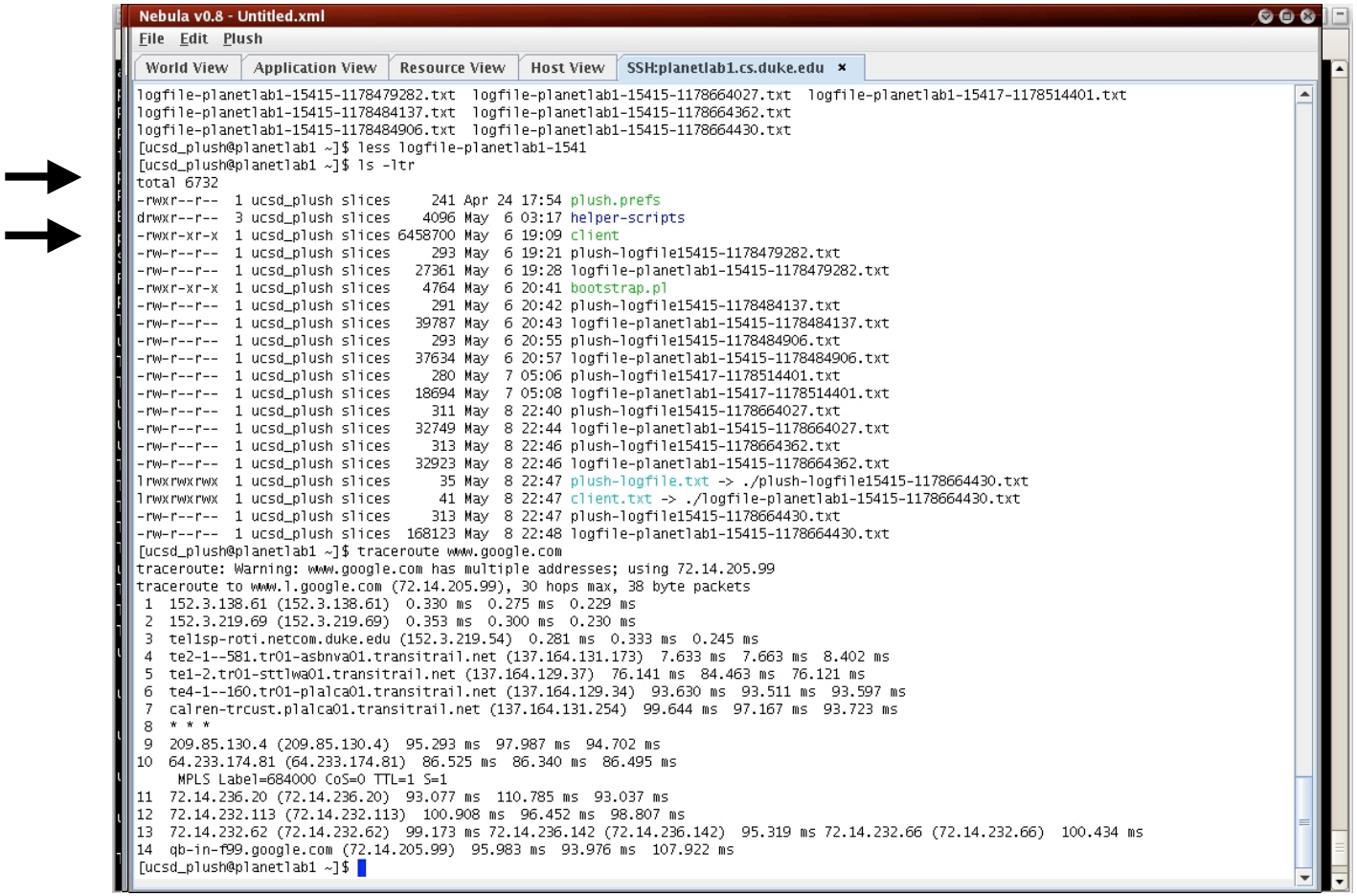


- Plush clients make sure all programs exited cleanly
- Remove logs and software from remote machines
- Disconnect clients from controller



Plush 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 programatically



Summary

- Plush provides abstractions for managing distributed applications
  - Supports a range of applications using "building blocks" that define customized control flow
  - Supports several execution environments
- Reduces the burden of deploying and debugging distributed applications so software developers can focus more on developing
- Next steps: Attract more users and obtain user feedback to enhance usability
  - Plush in the classroom?

Thanks!

For more info, visit

http://plush.cs.williams.edu

Email:

jeannie@cs.williams.edu