

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

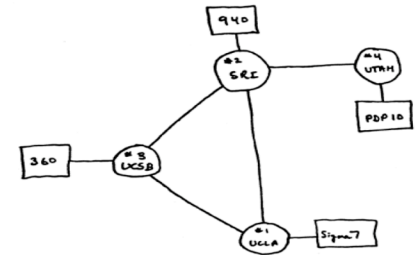
**Jeannie Albrecht**, Ryan Braud, Darren Dao,  
Nikolay Topilski, Christopher Tuttle,  
Alex C. Snoeren, and Amin Vahdat

Williams College & UC San Diego

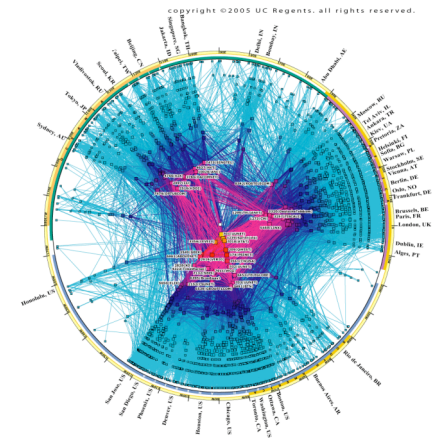


# Meeting Current Demands

- 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

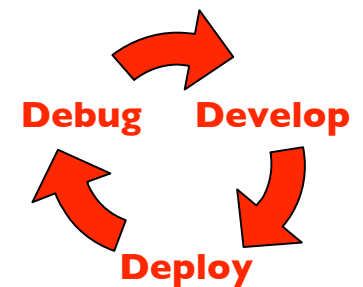


2005 Internet Map



# 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

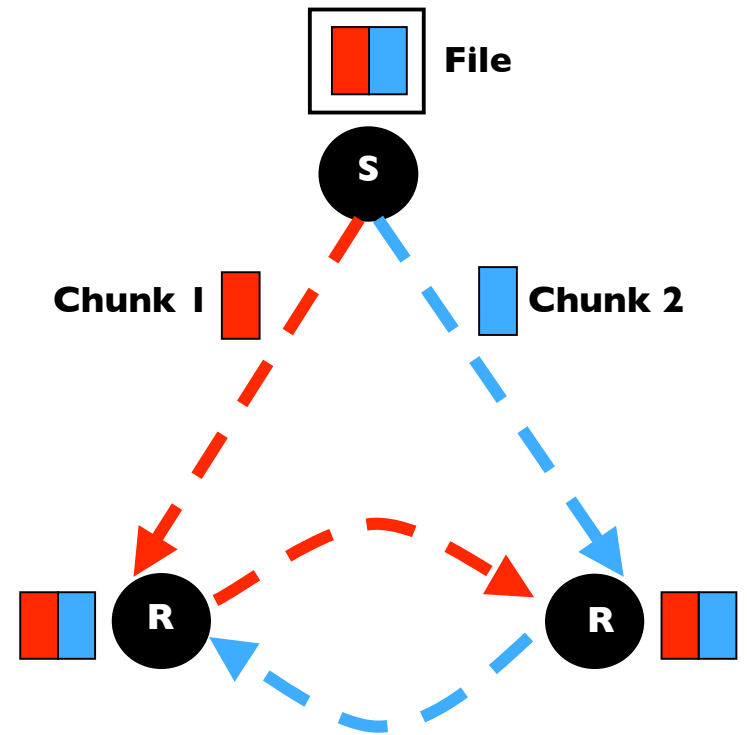


# Overview

- 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

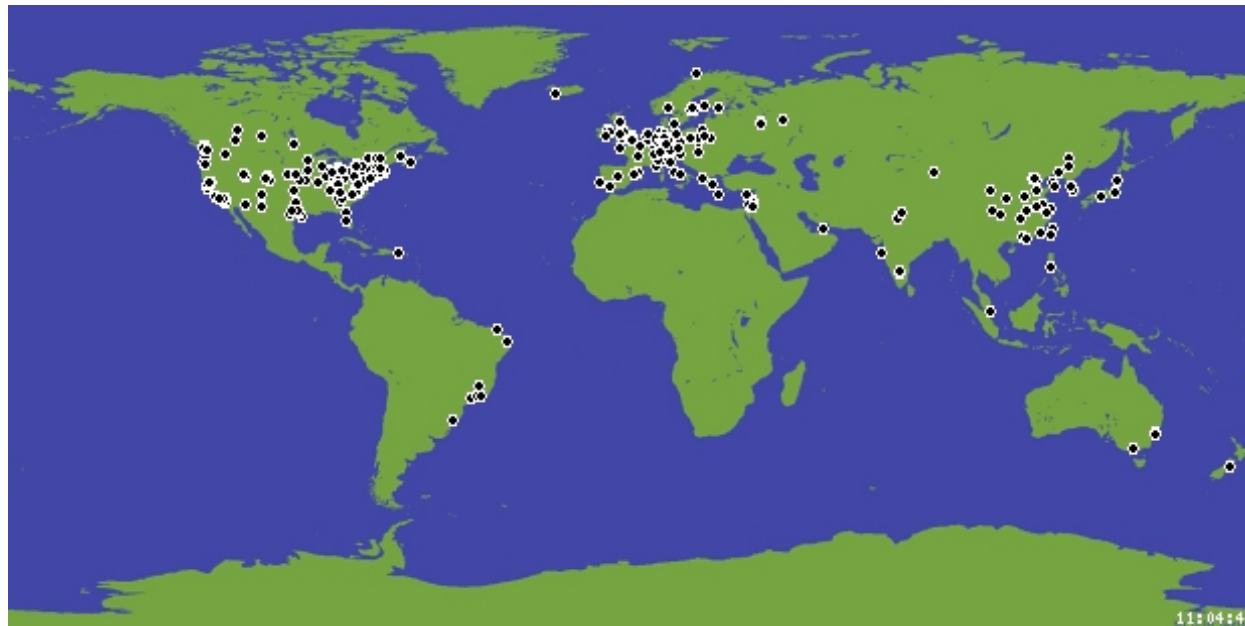
- Suppose we build **ByteTorrent**, a “new” file distribution service
  - Sender (S) sends file to Receivers (R)
  - Sender splits large file into “chunks”
  - Two phases of execution
    - Phase 1 – Join ByteTorrent network
    - Phase 2 – Transfer file
- We want to evaluate performance achieved on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





# PLANETLAB

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

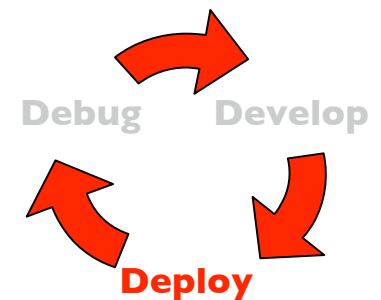


- Network of 800+ Linux computers at 400+ sites in 40+ countries
- Allows deployment of distributed applications around the world
- Can be a volatile working environment
  - High contention for machines (especially near paper deadlines)
  - Common problems: low disk space, clock skew, connection errors

<http://www.planet-lab.org>

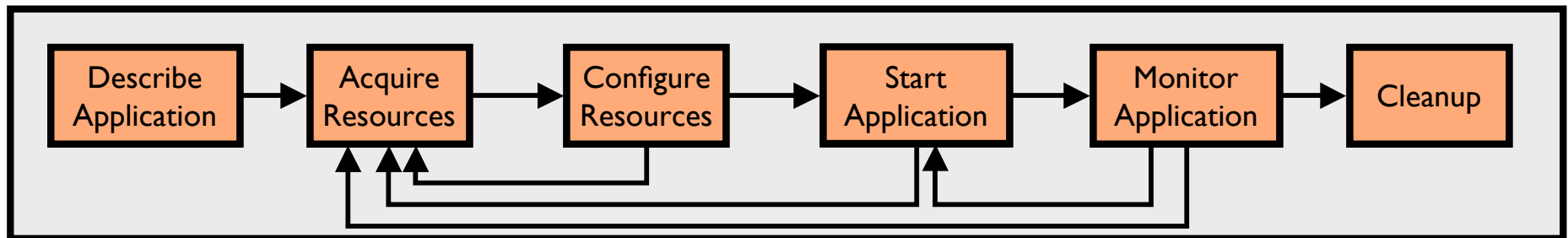
# Deploying ByteTorrent

- 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

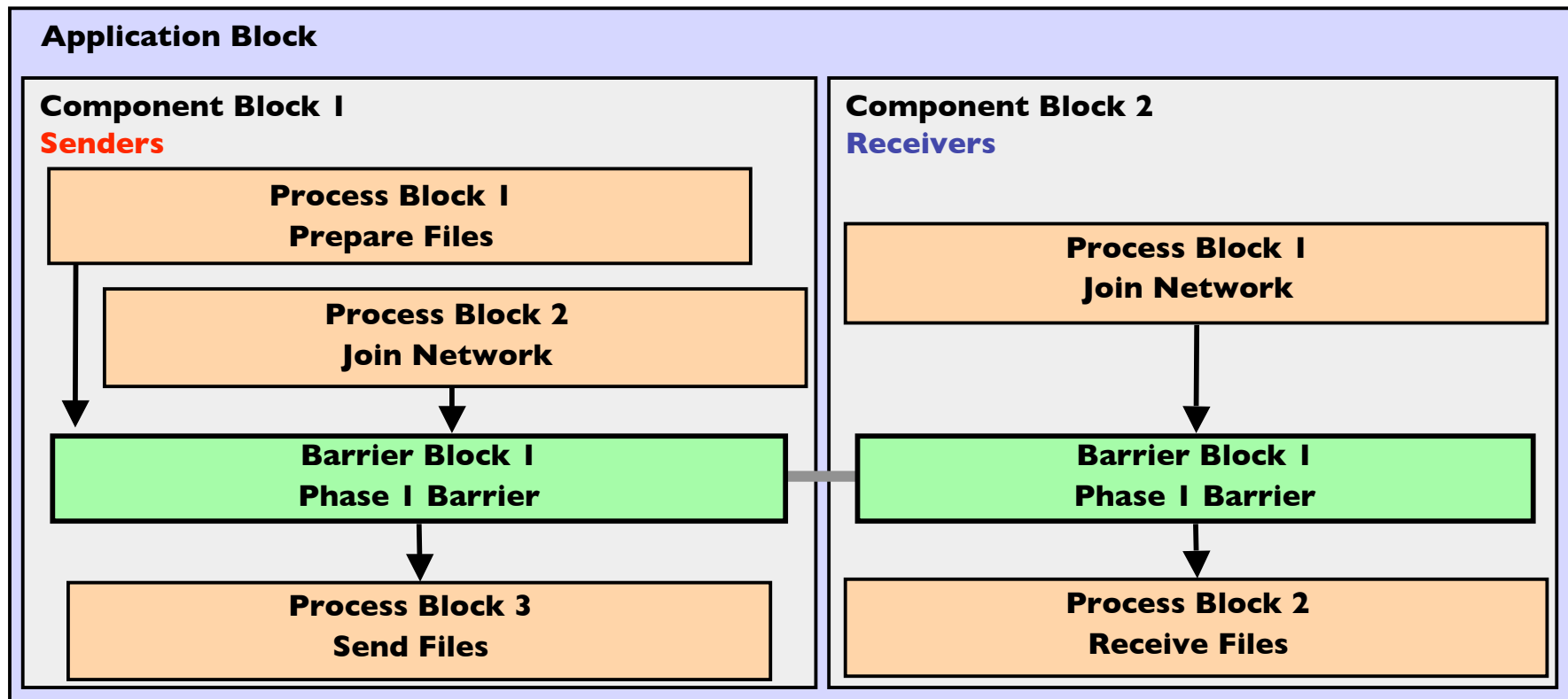
- A distributed application management infrastructure
  - Designed to simplify deployment of distributed applications
  - Provides abstractions for configuration and management
  - Allows users to “remotely control” computers running distributed applications worldwide



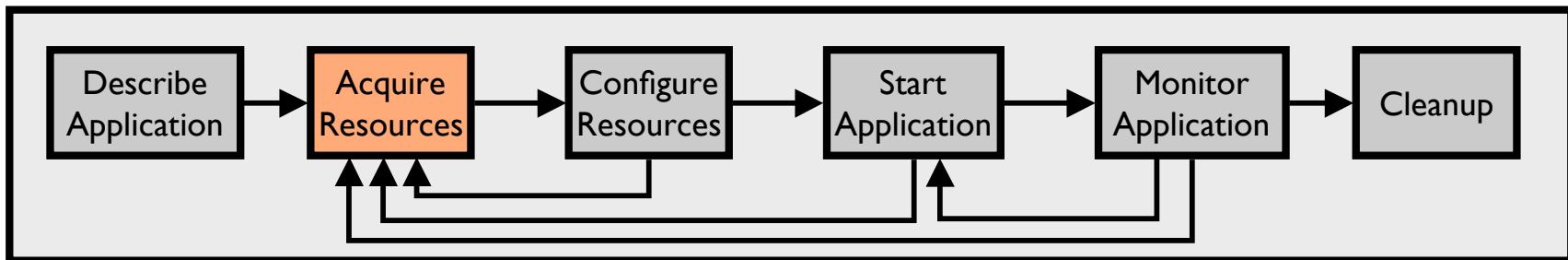


# Step I: Describe Application

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

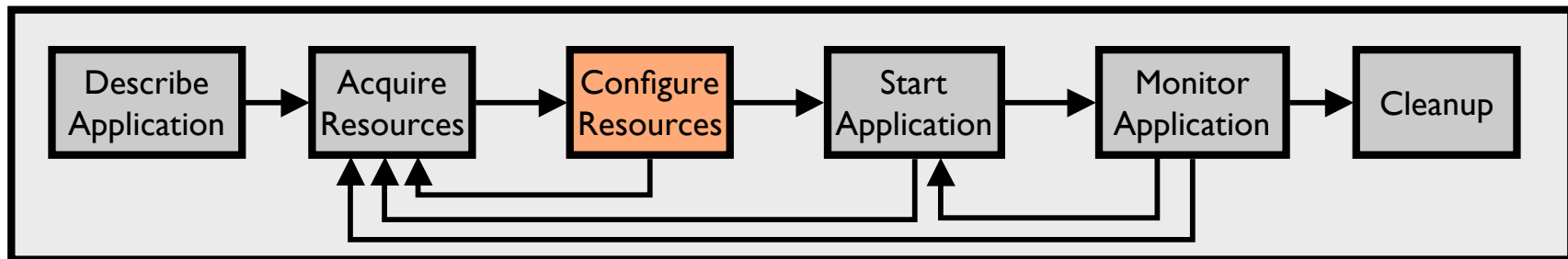


## Step 2: Acquire Resources

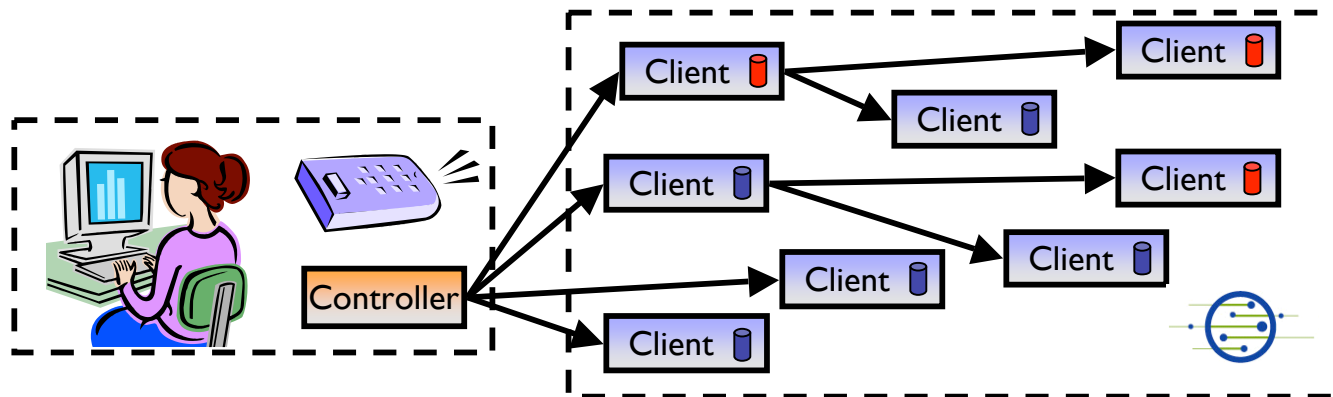


- How can we find “good” machines?
  - We want machines with specific characteristics
    - High bandwidth, fast processors, ample disk space
  - PlanetLab services perform *resource discovery*
  - Services find machines that satisfy our requirements
- Plush interfaces directly with these services

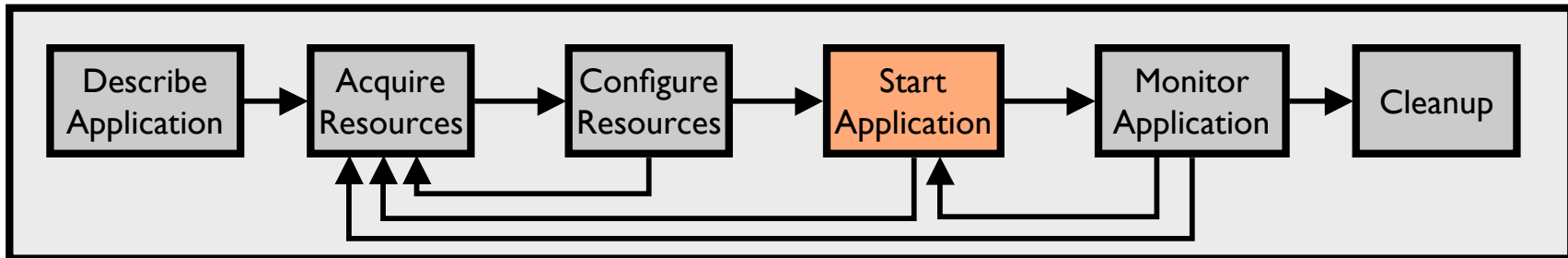
# Step 3: Configure Resources



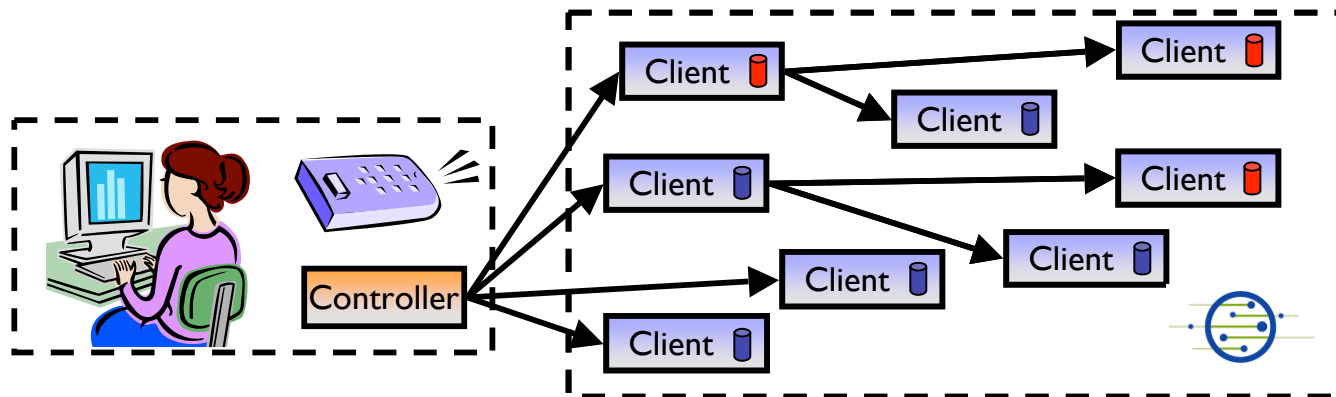
- Connect to and configure selected resources
  - Create a tree for achieving scalability in communication
  - **Controller** “remotely controls” the **clients** on our behalf
  - Install software on clients (some are **senders**, some are **receivers**)



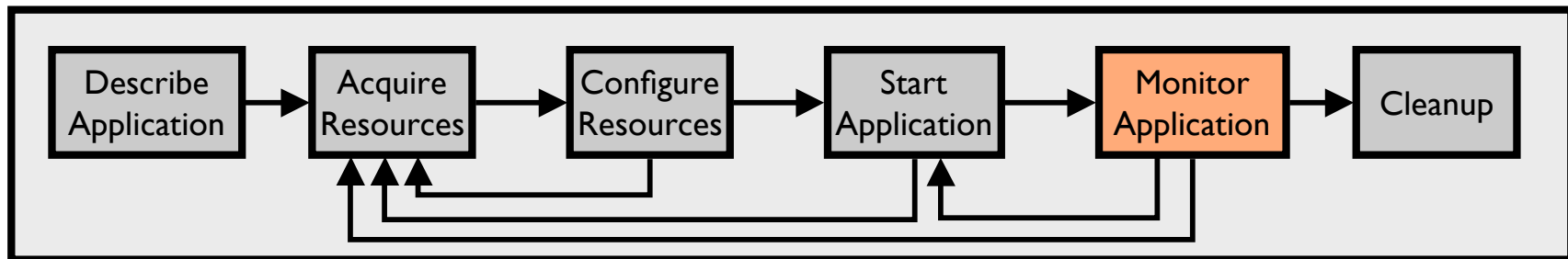
# Step 4: Start Application



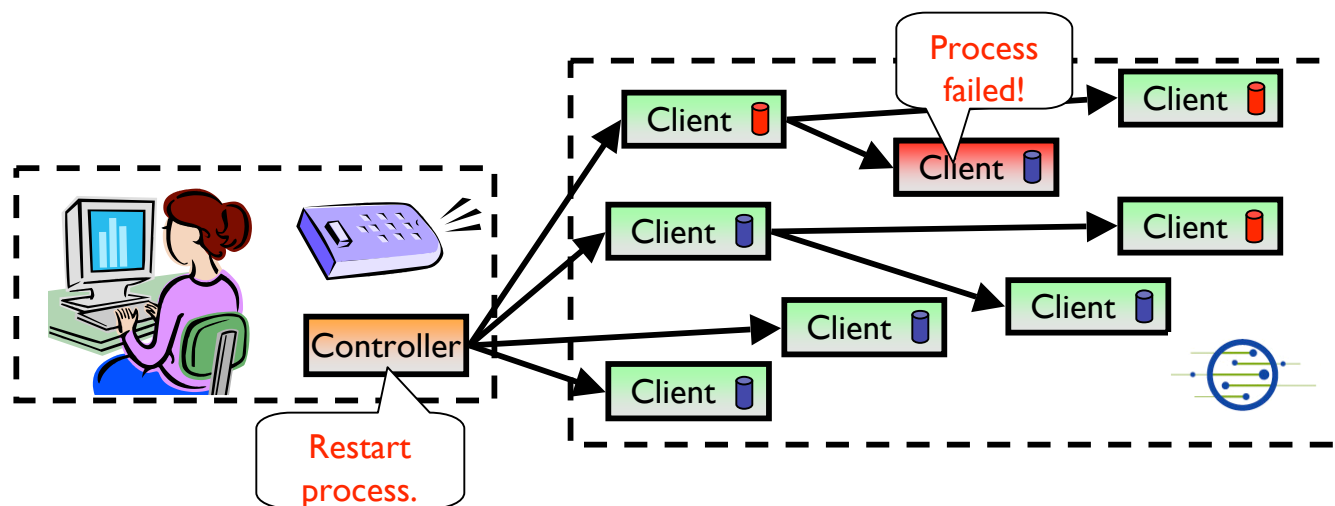
- Controller issues commands to clients telling them to start running our application
  - ByteTorrent **senders** begin running sender processes
  - ByteTorrent **receivers** begin running receiver processes



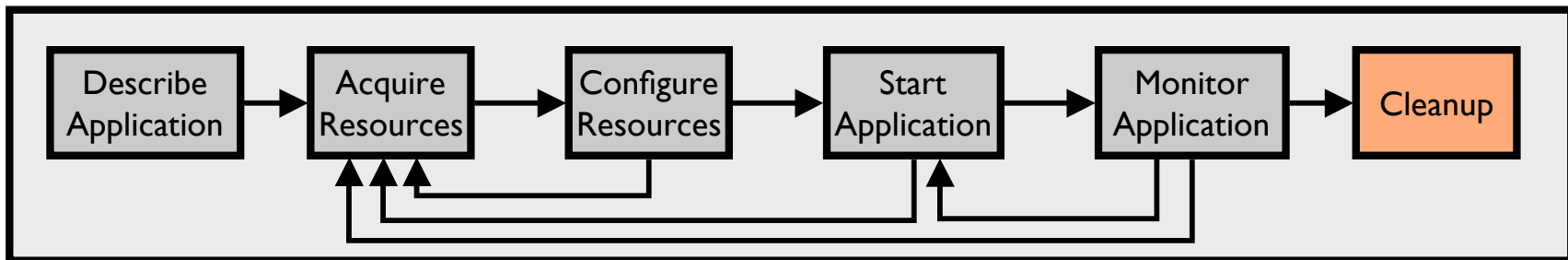
# Step 5: Monitor Application



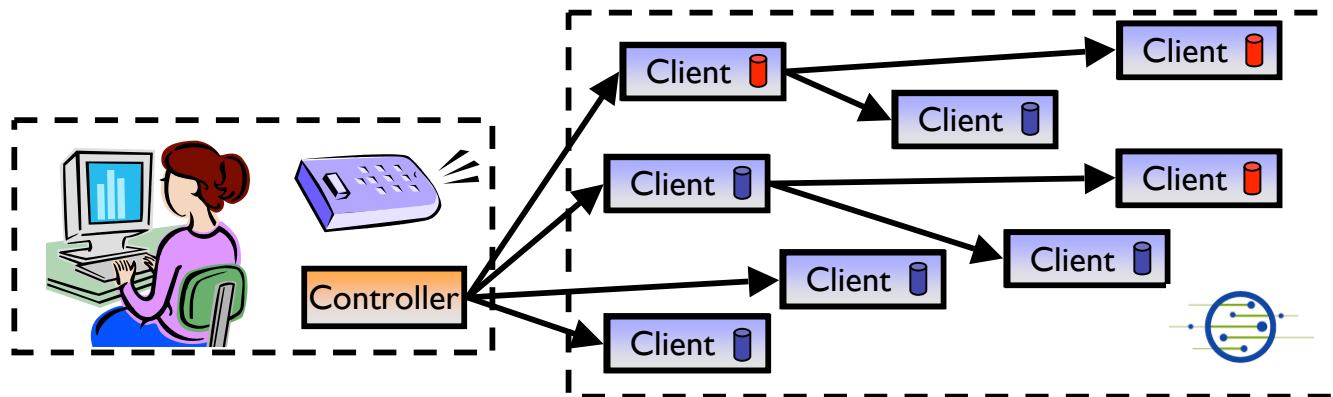
- We want to make sure the processes keep running
- Push clients monitor ByteTorrent processes for failures
  - If a failure is detected, client notifies controller
  - Controller decides to tell client to restart failed program or process



# Step 6: Cleanup

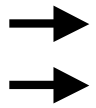


- Push clients make sure all programs exited cleanly
- Remove logs and software from remote machines
- Disconnect clients from controller



# 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 programmatically



```
Nebula v0.8 - Untitled.xml
File Edit Plush
World View Application View Resource View Host View SSH:planetlab1.cs.duke.edu x
logfile-planetlab1-15415-1178479282.txt logfile-planetlab1-15415-1178664027.txt logfile-planetlab1-15417-1178514401.txt
logfile-planetlab1-15415-1178484137.txt logfile-planetlab1-15415-1178664362.txt
logfile-planetlab1-15415-1178484906.txt logfile-planetlab1-15415-1178664430.txt
[ucsd_plush@planetlab1 ~]$ less logfile-planetlab1-1541
[ucsd_plush@planetlab1 ~]$ ls -ltr
total 6732
-rwxr--r-- 1 ucsd_plush slices 241 Apr 24 17:54 plush.prefs
drwxr--r-- 3 ucsd_plush slices 4096 May 6 03:17 helper-scripts
-rwxr-xr-x 1 ucsd_plush slices 6458700 May 6 19:09 client
-rw-r--r-- 1 ucsd_plush slices 293 May 6 19:21 plush-logfile15415-1178479282.txt
-rw-r--r-- 1 ucsd_plush slices 27361 May 6 19:28 logfile-planetlab1-15415-1178479282.txt
-rwxr-xr-x 1 ucsd_plush slices 4764 May 6 20:41 bootstrap.pl
-rw-r--r-- 1 ucsd_plush slices 291 May 6 20:42 plush-logfile15415-1178484137.txt
-rw-r--r-- 1 ucsd_plush slices 39787 May 6 20:43 logfile-planetlab1-15415-1178484137.txt
-rw-r--r-- 1 ucsd_plush slices 293 May 6 20:55 plush-logfile15415-1178484906.txt
-rw-r--r-- 1 ucsd_plush slices 37634 May 6 20:57 logfile-planetlab1-15415-1178484906.txt
-rw-r--r-- 1 ucsd_plush slices 280 May 7 05:06 plush-logfile15417-1178514401.txt
-rw-r--r-- 1 ucsd_plush slices 18694 May 7 05:08 logfile-planetlab1-15417-1178514401.txt
-rw-r--r-- 1 ucsd_plush slices 311 May 8 22:40 plush-logfile15415-1178664027.txt
-rw-r--r-- 1 ucsd_plush slices 32749 May 8 22:44 logfile-planetlab1-15415-1178664027.txt
-rw-r--r-- 1 ucsd_plush slices 313 May 8 22:46 plush-logfile15415-1178664362.txt
-rw-r--r-- 1 ucsd_plush slices 32923 May 8 22:46 logfile-planetlab1-15415-1178664362.txt
lrwxrwxrwx 1 ucsd_plush slices 35 May 8 22:47 plush-logfile.txt -> ./plush-logfile15415-1178664430.txt
lrwxrwxrwx 1 ucsd_plush slices 41 May 8 22:47 client.txt -> ./logfile-planetlab1-15415-1178664430.txt
-rw-r--r-- 1 ucsd_plush slices 313 May 8 22:47 plush-logfile15415-1178664430.txt
-rw-r--r-- 1 ucsd_plush slices 168123 May 8 22:48 logfile-planetlab1-15415-1178664430.txt
[ucsd_plush@planetlab1 ~]$ traceroute www.google.com
traceroute: Warning: www.google.com has multiple addresses; using 72.14.205.99
traceroute to www.1.google.com (72.14.205.99), 30 hops max, 38 byte packets
 1 152.3.138.61 (152.3.138.61) 0.330 ms 0.275 ms 0.229 ms
 2 152.3.219.69 (152.3.219.69) 0.353 ms 0.300 ms 0.230 ms
 3 telisp-roti.netcom.duke.edu (152.3.219.54) 0.281 ms 0.333 ms 0.245 ms
 4 te2-1--581.tr01-asbnva01.transitrail.net (137.164.131.173) 7.633 ms 7.663 ms 8.402 ms
 5 te1-2.tr01-sttlwa01.transitrail.net (137.164.129.37) 76.141 ms 84.463 ms 76.121 ms
 6 te4-1--160.tr01-plalca01.transitrail.net (137.164.129.34) 93.630 ms 93.511 ms 93.597 ms
 7 calren-trcust.plalca01.transitrail.net (137.164.131.254) 99.644 ms 97.167 ms 93.723 ms
 8 * * *
 9 209.85.130.4 (209.85.130.4) 95.293 ms 97.987 ms 94.702 ms
10 64.233.174.81 (64.233.174.81) 86.525 ms 86.340 ms 86.495 ms
   MPLS Label=684000 CoS=0 TTL=1 S=1
11 72.14.236.20 (72.14.236.20) 93.077 ms 110.785 ms 93.037 ms
12 72.14.232.113 (72.14.232.113) 100.908 ms 96.452 ms 98.807 ms
13 72.14.232.62 (72.14.232.62) 99.173 ms 72.14.236.142 (72.14.236.142) 95.319 ms 72.14.232.66 (72.14.232.66) 100.434 ms
14 qb-in-f99.google.com (72.14.205.99) 95.983 ms 93.976 ms 107.922 ms
[ucsd_plush@planetlab1 ~]$
```

# 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](mailto:jeannie@cs.williams.edu)