

Managing Distributed Applications using Gush

Jeannie Albrecht and Danny Y. Huang
Williams College

<http://gush.cs.williams.edu>

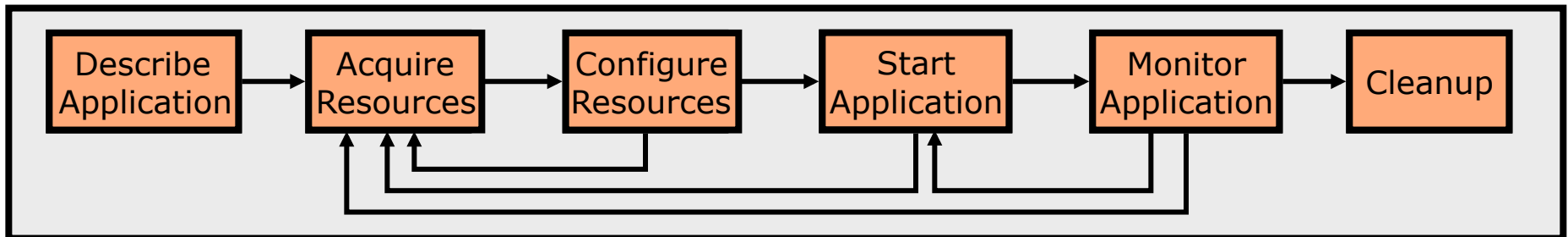


Overview

- How do experimenters actually use GENI?
- Goal: Develop abstractions and tools for addressing the challenges of managing distributed applications
 - Make it easy for a range of users (including students!) to run a variety of experiments on GENI
- Strategy
 - Make minimal assumptions about GENI “resources” and how they are allocated
 - Leverage existence of lower level services to locate resources and obtain credentials
 - Interface with other user tools
 - Hide complexity and use one common user interface to interact with different underlying systems (i.e., PlanetLab, ProtoGENI/Emulab, ORCA)

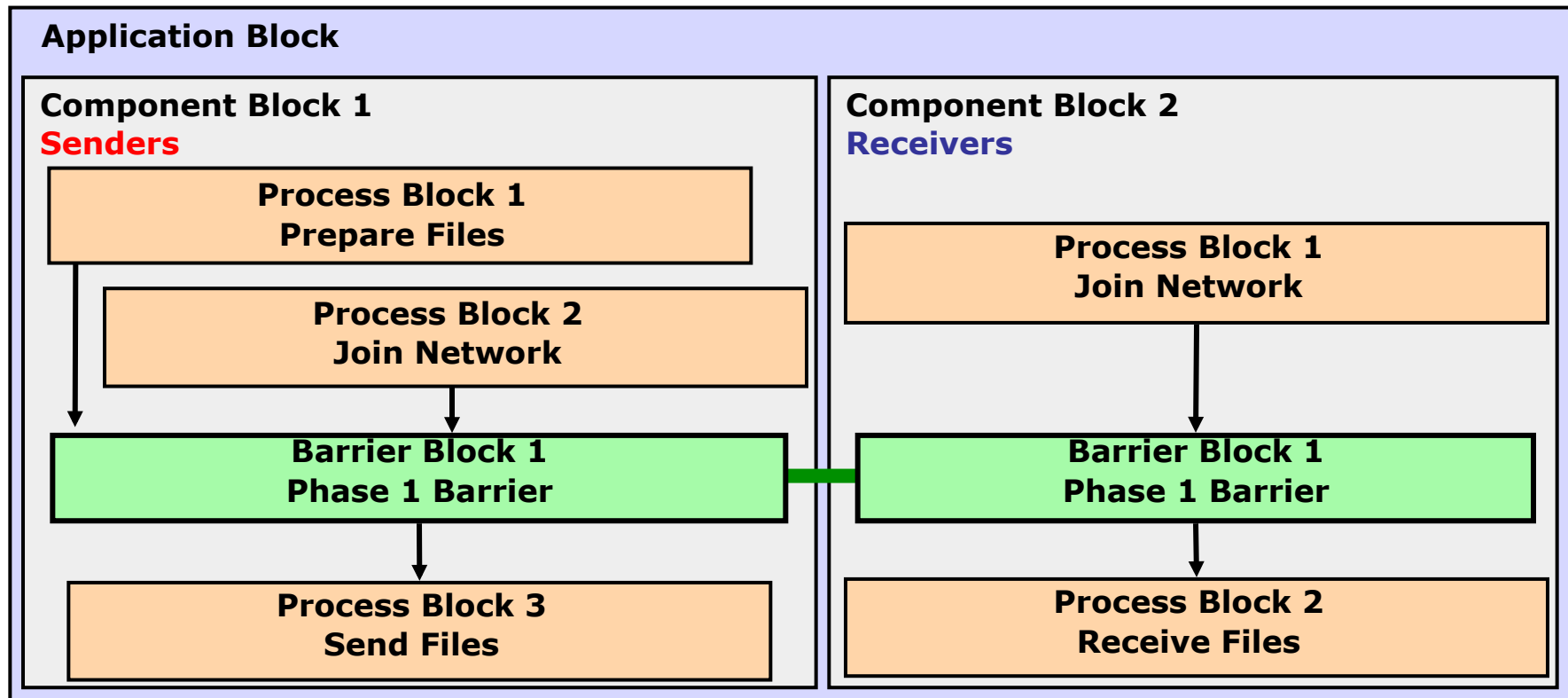
Gush

- 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



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 1: App Spec

```
<gush>
  <project name="simple">
    <software name="SimpleSoftwareName" type="none">
      <package name="Package" type="web">
        <path>http://sysnet.cs.williams.edu/~jeannie/software.tar</path>
        <dest_path>software.tar</dest_path>
      </package>
    </software>
    <component name="Cluster1">
      <rspec>
        <num_hosts>20</num_hosts>
      </rspec>
      <software name="SimpleSoftwareName" />
      <resources>
        <resource type="planetlab" group="williams_gush" />
        <resource type="gpeni" group="gpeni_gush" />
        <resource type="max" group="maxpl_gush" />
      </resources>
    </component>
    <experiment name="simple">
      <execution>
        <component_block name="cb1">
          <component name="Cluster1" />
          <process_block name="p2">
            <process name="cat">
              <path>cat</path>
              <cmdline>
                <arg>software.txt</arg>
              </cmdline>
            </process>
          </process_block>
        </component_block>
      </execution>
    </experiment>
  </project>
</gush>
```

SOFTWARE

DEFINE RESOURCE POOL

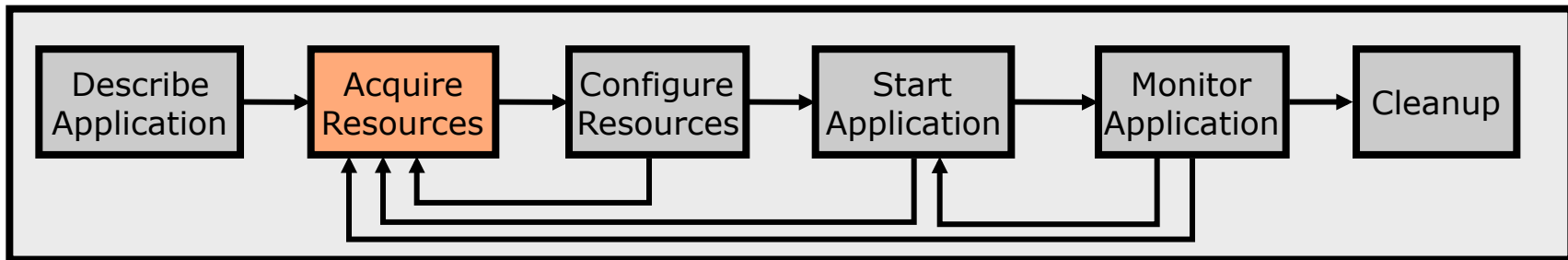
DEFINE PROCESSES (EXECUTION)

Step 1: App Spec

```
<gush>
  <project name="simple">
    ...
    <component name="Cluster1">
      <rspec>
        <num_hosts>20</num_hosts>
      </rspec>
      <software name="SimpleSoftwareName" />
      <resources>
        <resource type="planetlab" group="williams_gush" />
        <resource type="gpeni" group="gpeni_gush" />
        <resource type="max" group="maxpl_gush" />
      </resources>
    </component>
    <component name="Cluster2">
      <rspec>
        <num_hosts>20</num_hosts>
        <orca>
          <num hosts>20</num hosts>
          <type>1</type>
          <memory>784</memory>
          <bandwidth>300</bandwidth>
          <cpu>75</cpu>
          <lease length>12000</lease length>
          <server>http://geni.renci.org/orca:8080</server>
        </orca>
      </rspec>
      <software name="SimpleSoftwareName" />
      <resources>
        <resource type="ssh" group="orca" />
      </resources>
    </component>
    ...
  </project>
</gush>
```

- Application level control framework interoperability in GENI!

Step 2: Acquire Resources



- How can we find “good” resources?
 - We may want machines with specific characteristics
- Gush interfaces directly with lower level services
 - Gush fully supports PlanetLab resources
 - Beta support for ORCA and ProtoGENI resources

PlanetLab Resource Directory

<gush>

<resource_manager type="geni">

<user>plc.williams.jeannie</user>

<config_file>planetlab_sfi_config</config_file> → PlanetLab

<port_map slice="plc.williams.gush" port="15413"/>

</resource_manager>

<resource_manager type="geni">

<user>plc.ksu.jeannie</user>

<config_file>gpeni_sfi_config</config_file>

<port_map slice="plc.ksu.gush" port="15414"/> → GpENI

</resource_manager>

<resource_manager type="geni">

<user>plc.max.jeannie</user>

<config_file>max_sfi_config</config_file>

<port_map slice="plc.max.gush" port="15415"/> → MAX

</resource_manager>

</gush>

ORCA Resource Specification

```
<component name="VMGroup1">
  <rspec>
    <num hosts>20</num hosts>
    <orca>
      <num hosts>20</num hosts>
      <type>1</type>
      <memory>784</memory>
      <bandwidth>300</bandwidth>
      <cpu>75</cpu>
      <lease length>12000</lease length>
      <server>http://geni.renci.org/orca:8080</server>
    </orca>
  </rspec>
  <resources>
    <resource type="ssh" group="orca"/>
  </resources>
</component>
```

- Unlike PlanetLab, ORCA resources do not exist in advance
- ORCA creates VMs on demand and emphasizes resource isolation
- ORCA resources are defined within application specification

ORCA Resource Specification

```
<component name="VMGroup1">
  <rspec>
    <num hosts>20</num hosts>
    <orca>
      <num hosts>20</num hosts>
      <type>1</type>
      <memory>784</memory>
      <bandwidth>300</bandwidth>
      <cpu>75</cpu>
      <lease length>12000</lease length>
      <server>http://geni.renci.org/orca:8080</server>
    </orca>
  </rspec>
  <resources>
    <resource type="ssh" group="orca"/>
  </resources>
</component>
```

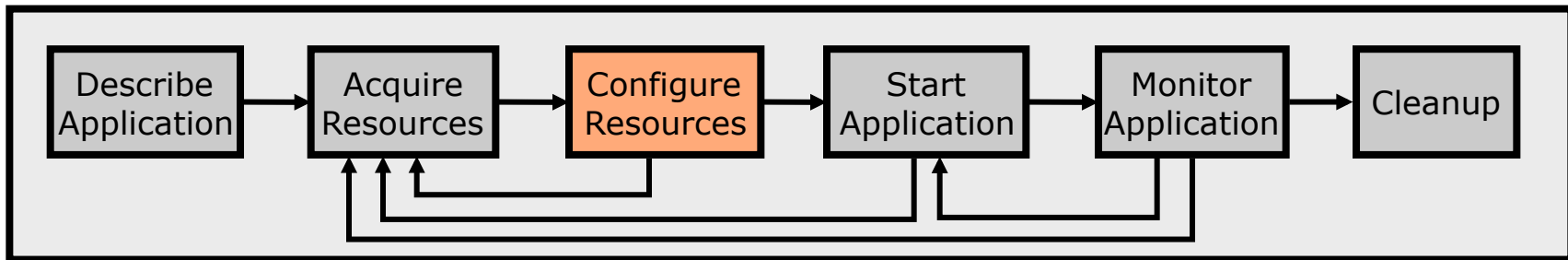
- Gush contacts ORCA slice manager when experiment is started
- ORCA calls back to Gush when resources are ready for use

ProtoGENI Resource Directory

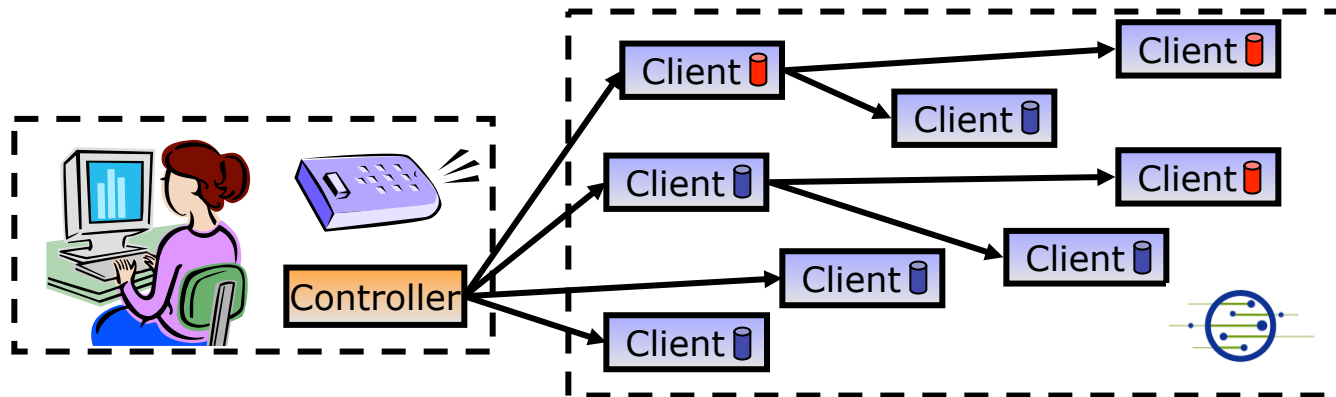
```
<gush>
  <resource manager type="emulab">
    <user>jeannie</user>
    <port>15420</port>
    <EmulabProjectID>Gush</EmulabProjectID>
    <EmulabExperimentID>gush</EmulabExperimentID>
    <EmulabNSFile>nsfile.ns</EmulabNSFile>
  </resource manager>
</gush>
```

- ProtoGENI resources are defined like PlanetLab resources
- Experiments must be swapped in and out before execution
- Like ORCA, ProtoGENI resources are created on demand
- Unlike ORCA, ProtoGENI currently does not provide callbacks to Gush about resource availability

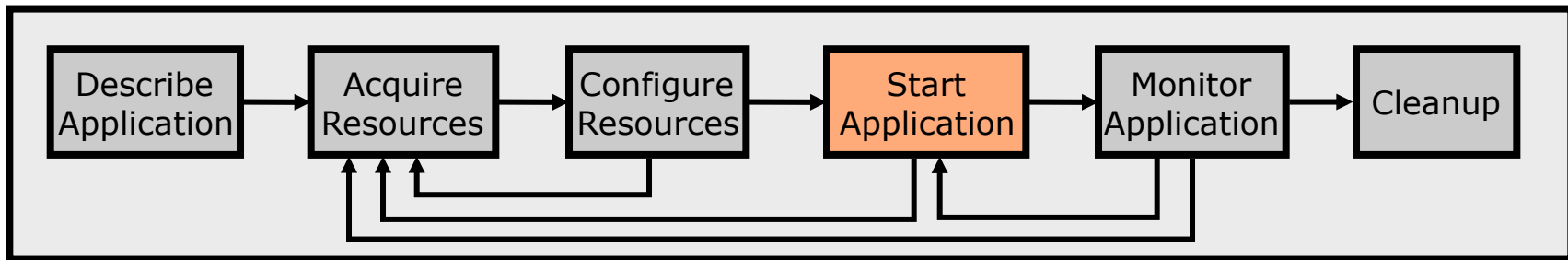
Step 3: Configure Resources



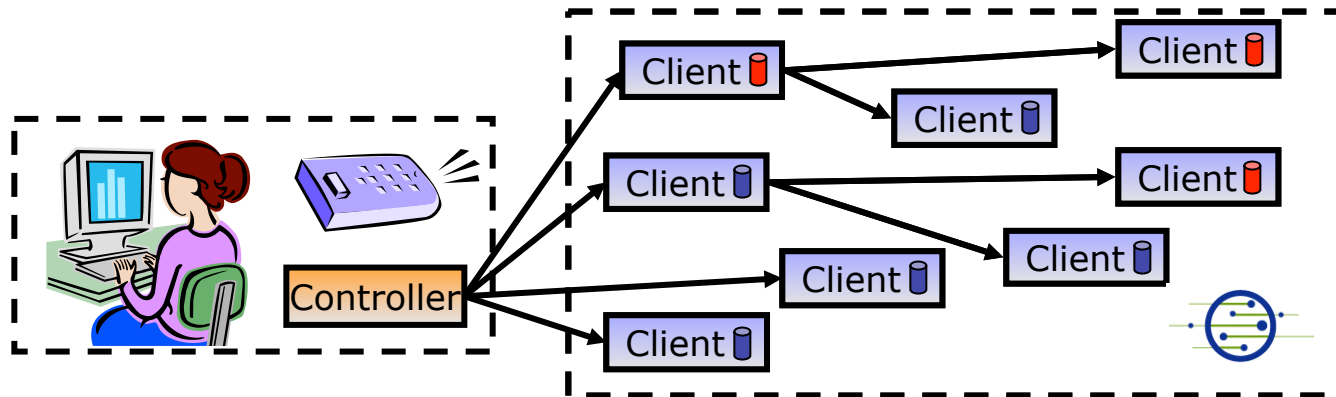
- Connect to and configure selected resources
 - **Controller** “remotely controls” the **clients** on the experimenter’s behalf
 - Install software on clients



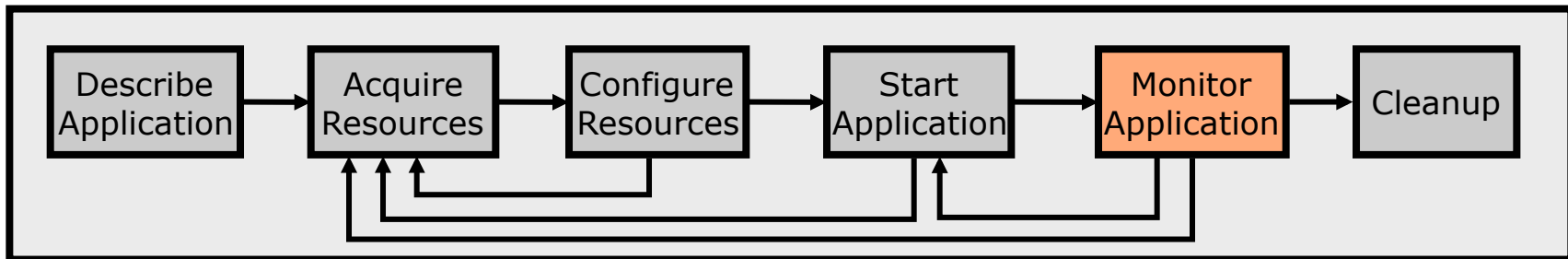
Step 4: Start Application



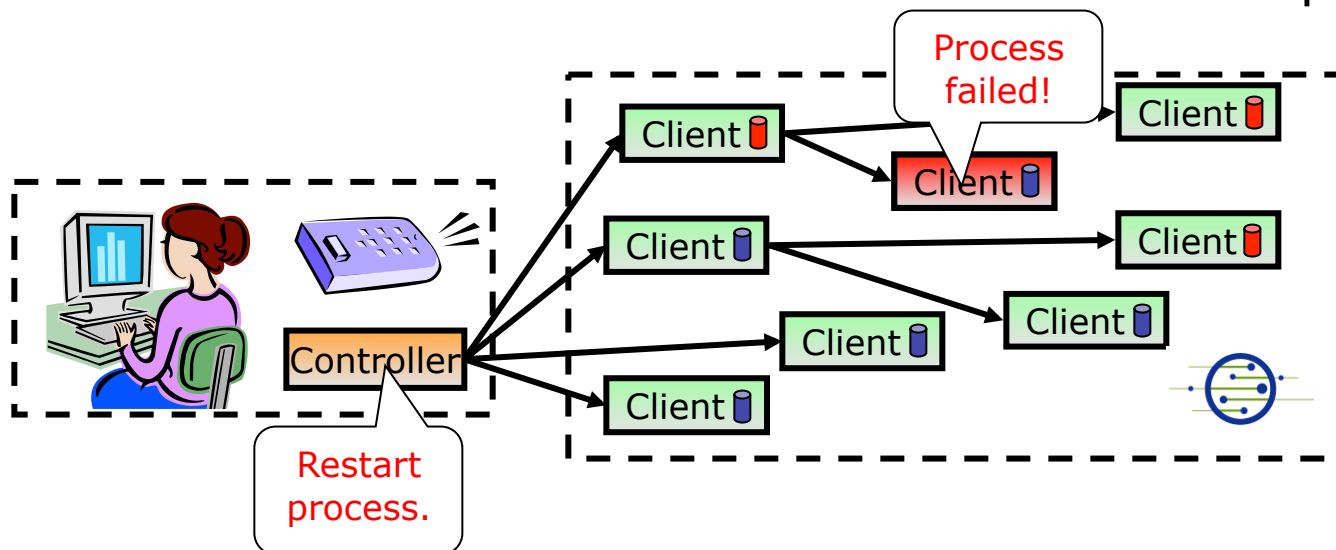
- Controller issues commands to clients telling them to start running applications/experiments
 - **Senders** begin running sender processes
 - **Receivers** begin running receiver processes



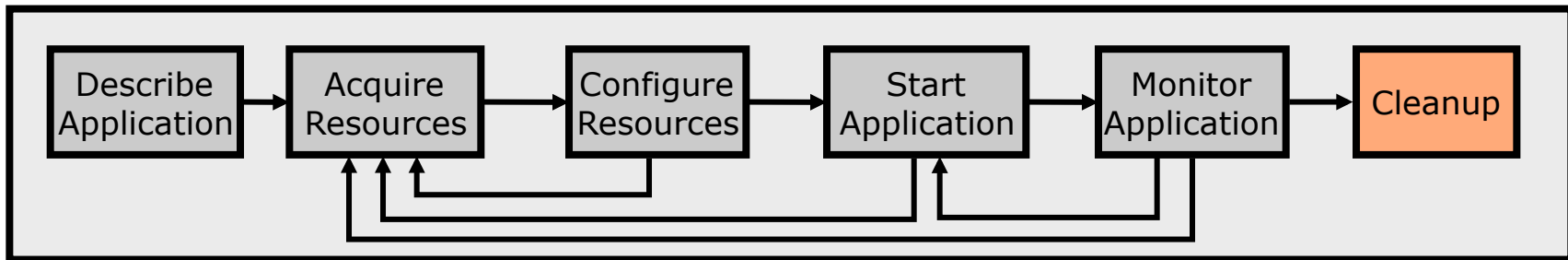
Step 5: Monitor Application



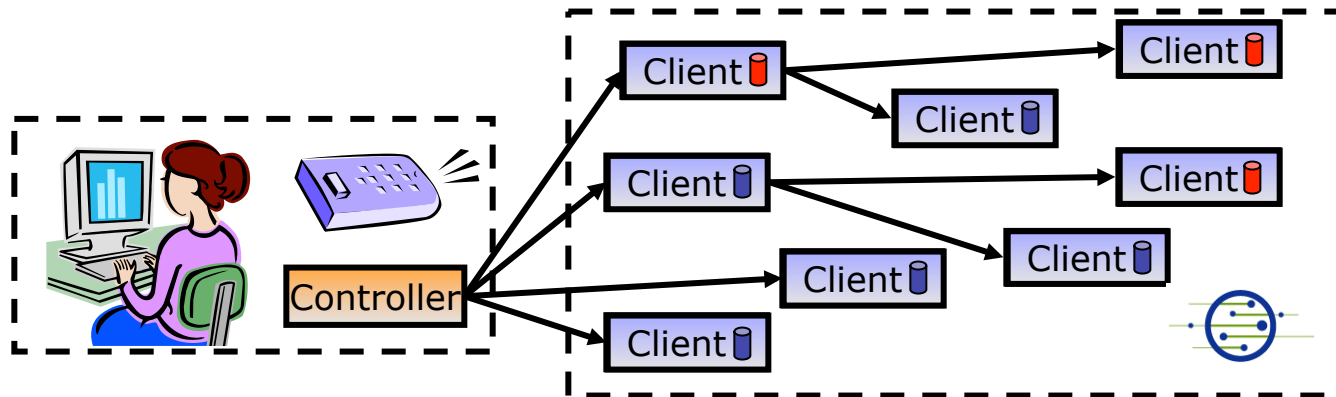
- We want to make sure the processes keep running
- Clients monitor experiment processes for failures
 - If a failure is detected, client notifies controller
 - Controller decides to tell client to restart failed process



Step 6: Cleanup



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



Gush in Action

```
gush> load ./tests/simple.xml
```

Project "simple" is selected.

Experiment "simple" is selected.

```
gush> run
```

Starting experiment run.

Running experiment simple...

```
gush> The configuration matcher has finished matching.
```

The resource allocator has finished successfully.

```
gpeni_gush@geni-planetlab-1.ksu.gpeni.net:15414 has joined the mesh.
```

The file transfer of Package to geni-planetlab-1.ksu.gpeni.net has been completed.

The software installation of Package on geni-planetlab-1.ksu.gpeni.net was successful.

```
williams_gush@planetlab1.williams.edu:15413 has joined the mesh.
```

```
maxpl_gush@planetlab2.dragon.maxgigapop.net:15415 has joined the mesh.
```

The file transfer of Package to planetlab1.williams.edu has been completed.

The software installation of Package on planetlab1.williams.edu was successful.

The file transfer of Package to planetlab2.dragon.maxgigapop.net has been completed.

The software installation of Package on planetlab2.dragon.maxgigapop.net was successful.

```
gpeni_gush@geni-planetlab-1.ksu.gpeni.net:15414,31821: Hello World
```

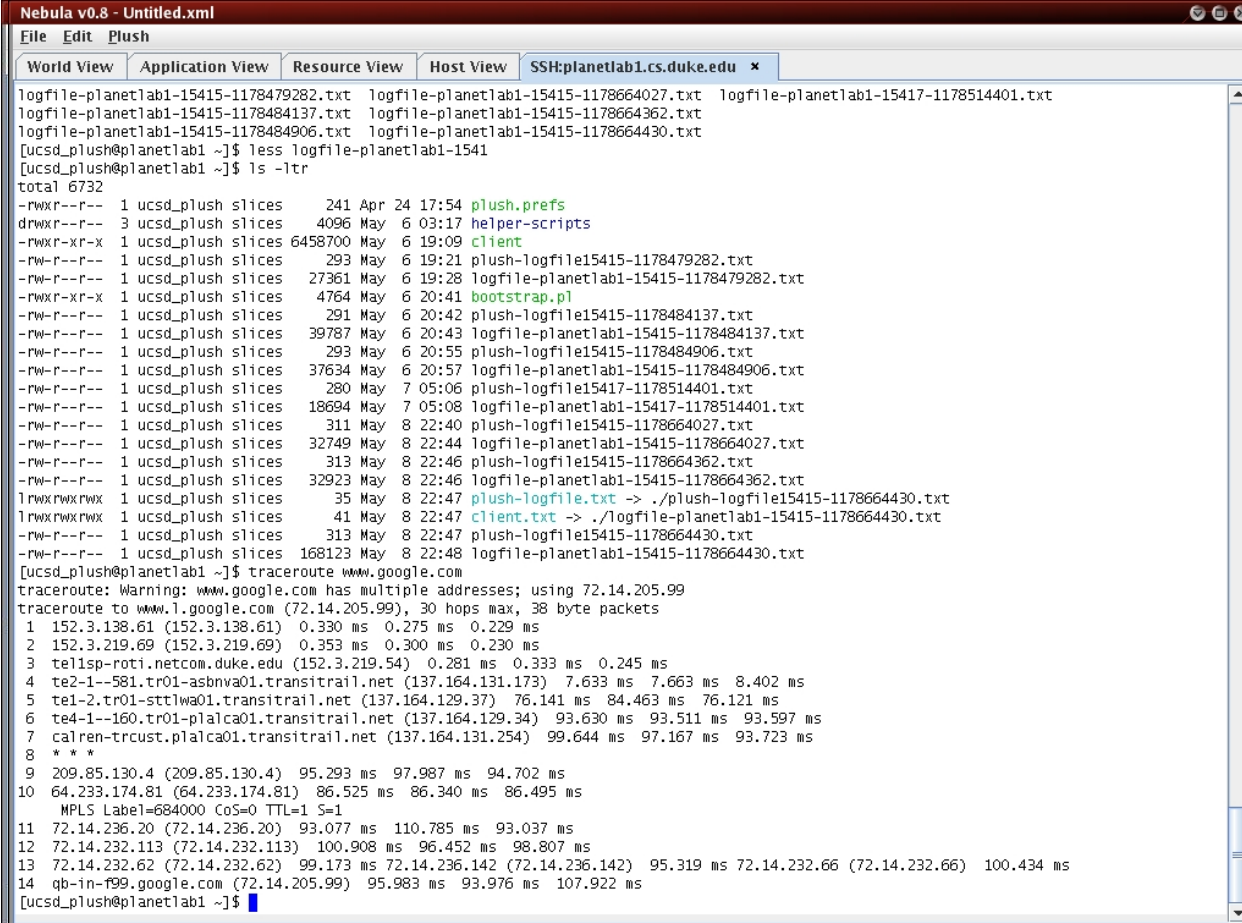
```
williams_gush@planetlab1.williams.edu:15413,19548: Hello World
```

```
maxpl_gush@planetlab2.dragon.maxgigapop.net:15415,26459: Hello World
```

The experiment has ended.

Nebula

- 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 ~]$ 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.l.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 ~]$
```

Future Work and Conclusions

- 18 undergrads at Williams College used Gush and Nebula to run experiments on PlanetLab last fall
 - Gush was stable, Nebula needs work
 - iPod/iPhone interface?
 - 2 undergrads have worked on Gush development
 - 2 more will work on Nebula this summer
- Need better support for wireless/mobile devices
- Gush is probably not the solution for all testbeds
 - But it's a step in the right direction (I hope)!
- Gush has helped identify what users actually want and need
 - Determine the right set of abstractions for experiment management and application control

Thanks!

For more info:

<http://gush.cs.williams.edu>

