Seattle: The Internet as an Educational Testbed

Justin Cappos
NYU Poly
Computer Science and Engineering
The “Dark Ages”
The “Golden Age”
The Path to Enlightenment
Common use

- **Students**
  - Request resources via website
  - Use shell to start programs on computers all around the world
    - Not just the machines at your university!
  - View program output / tracebacks

- **Students can also test on a local install**
  - Easy way to test quickly or when disconnected

- **Instructor or students optionally install Seattle**
  - Each install allows the use of 10 more VMs
How Seattle Works

- **An installer is downloaded**
  - Can be installed in a restricted account
    - Admin access not needed
  - Many platforms supported (Windows, Linux, Mac, BSD, some smartphones, tablets, etc.)
- Runs a few processes
- No IT management overhead
Programs are run in a virtualized environment

- Safety / Security

- Performance isolation (similar to Xen, VMWare)
  - A buggy program can’t slow down machines

- Node Manager allows remote users to control programs

- Portability

- Programming language VM based upon Python
  - Students (and instructors!) find it easy to learn
    - Chord implementation (~300 LOC) in 3 weeks!
Demonstration

- Seattle Clearinghouse
  - Register an account
  - Install Seattle
  - Acquire resources
  - Download demokit / shell

- Use shell to control resources
  - Deploy all pairs ping
Educational use

• Classroom experience
  • Released in Spring 2009
  • Used in more than two dozen classes (so far)
  • 3 tutorials, 3 library references, etc.
  • 10 battle tested assignments
    • Overlay routing, flow control, NAT / Non-transitive connectivity, Chord (DHT), web / chat servers, reference monitors, NAT tunneling, etc.
  • OS classes are coming
  • Advanced projects
    • MapReduce, Distributed Web Servers, etc.

• Community support
  • Supported by educational groups
  • SIGCSE paper, 3 CCSC workshops, etc.
  • Top ranked SIGCOMM Educational Resource
  • Coming in Computer Networking by Kurose & Ross
    • Most popular networking book!
Summary

Seattle testbed

• Real system deployed around the world
  • Geographic diversity, network diversity, device diversity...
  • Real networks!

• Battle tested educational platform!
  • Free, simple and safe to use
  • Open participation / open source
  • Broad community
  • Easy to drop into a class

https://seattle.cs.washington.edu/
## Current Node Composition

<table>
<thead>
<tr>
<th>Node Type</th>
<th>Quantity*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Testbed</td>
<td>791</td>
</tr>
<tr>
<td>University nodes</td>
<td>1720</td>
</tr>
<tr>
<td>Home machines</td>
<td>2849</td>
</tr>
<tr>
<td>Phone in name</td>
<td>67</td>
</tr>
<tr>
<td>Unknown nodes</td>
<td>3370</td>
</tr>
<tr>
<td>Total</td>
<td>8797</td>
</tr>
</tbody>
</table>

About 1% phones, 9% testbed, 20% university, 71% (likely) home nodes

* Nodes by IP address that accessed the Seattle software updater from Nov 2010 to Nov 2011. Location information by pygeoip.
Easy To Code

UDP ping server (4 LOC)

```python
def got_message(srcip, srcport, mess, ch):
    sendmess(srcip, srcport, mess)
    if callfunc == 'initialize':
        recvmess(getmyip(), 54321, got_message)
```

UDP ping client (6 LOC)

```python
def got_reply(srcip, srcport, mess, ch):
    print 'received:', mess, 'from', srcip, srcport
    if callfunc == 'initialize':
        recvmess(getmyip(), 43210, got_reply)
    # send the second arg to the first arg's IP
    sendmess(callargs[0], 54321, callargs[1], getmyip(), 43210)
    # exit in one second
    settimer(1, exitall, ())
```
All Pairs Ping

Send periodic UDP pings
15 LOC

Handle incoming UDP pings
7 LOC

Format latency data into HTML
9 LOC

Return a webpage
11 LOC

Initialization
15 LOC

Handle incoming UDP pings
7 LOC

Format latency data into HTML
9 LOC

Return a webpage
11 LOC

Initialization
15 LOC

Send periodic UDP pings
15 LOC

Handle incoming UDP pings
7 LOC

Format latency data into HTML
9 LOC

Return a webpage
11 LOC

Initialization
15 LOC

Send periodic UDP pings
15 LOC

Handle incoming UDP pings
7 LOC

Format latency data into HTML
9 LOC

Return a webpage
11 LOC

Initialization
15 LOC

Send periodic UDP pings
15 LOC

Handle incoming UDP pings
7 LOC

Format latency data into HTML
9 LOC

Return a webpage
11 LOC

Initialization
15 LOC

Send periodic UDP pings
15 LOC

Handle incoming UDP pings
7 LOC

Format latency data into HTML
9 LOC

Return a webpage
11 LOC

Initialization
15 LOC

Send periodic UDP pings
15 LOC

Handle incoming UDP pings
7 LOC

Format latency data into HTML
9 LOC

Return a webpage
11 LOC

Initialization
15 LOC

Send periodic UDP pings
15 LOC

Handle incoming UDP pings
7 LOC

Format latency data into HTML
9 LOC

Return a webpage
11 LOC

Initialization
15 LOC

Send periodic UDP pings
15 LOC

Handle incoming UDP pings
7 LOC

Format latency data into HTML
9 LOC

Return a webpage
11 LOC

Initialization
15 LOC

Send periodic UDP pings
15 LOC

Handle incoming UDP pings
7 LOC

Format latency data into HTML
9 LOC

Return a webpage
11 LOC

Initialization
15 LOC

Send periodic UDP pings
15 LOC

Handle incoming UDP pings
7 LOC

Format latency data into HTML
9 LOC

Return a webpage
11 LOC

Initialization
15 LOC
Research use

- **Projects**
  - YouTube CDN mapping
  - Wireless mobility patterns
  - Network heterogeneity
  - Overlay routing across P2P networks
  - P2P resource allocation fairness
  - Etc.

- **Community support**
  - Port to N900 by Nokia researchers
  - Runs on PlanetLab, Emulab, GpENI, DOME, etc.
  - GENI workshops, PyCon, etc.
  - NaCl integration by U Victoria / HP Labs
  - iPad 2 port, tun / tap support, Android, etc. by academics in Europe