Flexlab: A Realistic, Controlled, and Friendly Environment for Evaluating Networked Systems

Jonathon Duerig, Robert Ricci, Junxing Zhang, Daniel Gebhardt, Sneha Kasera, Jay Lepreau

University of Utah

HotNets-V
November 30, 2006
Emulators (Emulab Sucks)

- Examples: Modelnet & Emulab
- The Good: Control, repeatability, wide variety of network conditions
- The Bad: Artificial network conditions
Overlay Testbeds (PlanetLab Sucks)

- Examples: RON & PlanetLab
- The Good: Real network conditions
- The Bad: Overloaded, No privileged operations, Poor repeatability, Hard to develop/debug
Goal: Best of Both Worlds (Don’t Suck)
Model-driven Emulation
(How not to suck)
Key Points

- Flexlab is an emulation framework into which different network models may be plugged
- Exploit an overlay testbed to generate measurements for some example models
  - Models make different fidelity, overhead, and repeatability trade-offs
- Application-Centric Internet Modeling
Flexlab: Application
Flexlab: Application Monitor
Flexlab: Network Model

Offered Load Model -> Network Model

Emulab Host

Application

App Monitor

Application Traffic

Emulab Host

Application

App Monitor
Flexlab: Measurement Repository
Flexlab: Path Emulator
ACIM: Application-Centric Internet Modeling
Imagine Ideal Fidelity
ACIM Challenges

- Hardening implementation to deal with PlanetLab unreliability
- CPU starvation on PlanetLab
  - Host artifacts in throughput
  - Packet loss from libpcap
- Reverse path congestion
- Measuring bottleneck queue size in time
- Discovering when bottleneck link is saturated
ACIM Available Bandwidth

- Throughput == available bandwidth
  iff agent is saturating
  && bottleneck link is saturated
- Agent saturating ≈ socket buffer full
- Bottleneck queue saturated
  ≈ queue filling up
  ≈ RTT increasing recently
Sample Results

[Graph showing Emulab (Throughput) with throughput in bytes/sec on the y-axis and time in seconds on the x-axis.]
Sample Results
Sample Results
Sample Results

PlanetLab (Throughput)

Throughput (bytes/sec)

Emulab (Throughput)

Throughput (bytes/sec)

Time (seconds)
Network Model Trade-offs

- Emulab
- Static
- Dynamic
- General Internet Model
- ACIM
- PlanetLab

More Repeatable vs. Higher Network Fidelity
Sample Real Application: BitTorrent with Static Model
BitTorrent w/ ACIM Model
BitTorrent w/ PlanetLab

What is “correct”? Challenging to determine; work-in-progress.
Conclusions

• Contribution: Modeling Framework for Emulation
  – Models can allow the experimenter to trade-off fidelity, 
    repeatability, and overhead

• Contribution: Application-Centric Internet Modeling

• Contribution: Running on Emulab and PlanetLab in 
  alpha stage
Backup Slides
Why not just add more nodes to every PlanetLab site? (cf. public review)

- Remaining problems:
  - Poor repeatability
  - Hard to develop/debug
  - No privileged operations
- Malicious traffic cannot be tested
- Some Flexlab network models reduce network load
- Emulab node pool stat muxed and shared more efficiently than per-site pools
- Overload can (will?) still happen with PL’s pure shared-host model
- Major practical barriers: admin, cost
PlanetLab Overload (What)
PlanetLab Overload (Why)

- Only a few nodes per site
  - Sites supply their own nodes
  - No incentive to increase number of nodes
- No admission control
- No resource guarantees
- No incentive to minimize usage
- Typically tedious to set up experiments (exceptions: Emulab portal, Plush, other?)
Network Model 1: Static

All-sites PlanetLab Measurements

Static Network Model

Datapositionary

Network Characteristics to Path Emulator
Static Trade-offs

- Low fidelity
- Fixed continuous overhead
- Complete repeatability
Network Model 2: Dynamic

All-sites PlanetLab Measurements

Dynamic Network Model

Datapository

Application Network Model from Monitor

Network Characteristics to Path Emulator
Dynamic Trade-offs

- Moderate fidelity
- Overhead proportional to number of paths used
- High repeatability
## Low-Frequency Measurements Miss Changes (Changepoint Analysis)

<table>
<thead>
<tr>
<th>Path</th>
<th>20 Sec. Period</th>
<th>2 Sec. Period</th>
<th>Avg magnitude of 2 sec changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Src</td>
<td>Dest</td>
<td>Count</td>
<td>Count</td>
</tr>
<tr>
<td>Commodity</td>
<td>Commodity</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>Commodity</td>
<td>Internet2</td>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td>Internet2</td>
<td>Internet2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Flexlab and VINI

 Entirely different kinds of realism and control

- Flexlab: passes “experiment” traffic over shared path
  - Real Internet conditions from other traffic on same path, but app. traffic is not from real users
  - Control: of all software
  - Environment: friendly local dev. environ, dedicated hosts

- VINI: can pass “real traffic” over dedicated link
  - Real routing, real neighbor ISPs, potentially traffic from real users, but network resources are not realistic/representative
  - Dedicated pipes with dedicated bandwidth, that insulate experiment from normal Internet conditions
  - Control: restricted to VINI’s APIs (Click, XORP, etc)
  - Environment: distributed environ; shared host resources.
Dealing with PlanetLab Unreliability

- Our initial design was optimistic
- Nodes fail
  - There is no set of ‘good nodes’
  - Agents must react robustly to node failure
- Most errors are transient
  - Log everything
  - Replay packet analysis
CPU Starvation on PlanetLab

- **Host Artifacts**
  - Long period when agent can’t read or write
  - Empty socket buffer or full receive window
  - Solution: Detect and ignore

- **Packet loss from libpcap**
  - Long period without reading libpcap buffer
  - Many packets are dropped at once
  - Solution: Detect and ignore
Handling Reverse Path Congestion

• Can cause ack compression
• Throughput Measurement
  – Throughput numbers become much noisier
  – We abuse the TCP timestamp option
  – PlanetLab: homogenous OS environment
  – Extending it would require hacking client
• RTT Measurement
  – Future work
Measuring Bottleneck Queue Size

• Important to emulate loss episodes due to congestion
• No one knows how in terms of bytes/packets
• Easier to measure in terms of time:
  – full = RTT when queue is full
  – empty = RTT when queue is empty
  – queue_time = full - empty
Initial Conditions

• Needed to bootstrap ACIM
  – ACIM uses traffic to generate conditions
  – But conditions must exist for first traffic
• We created a measurement framework
  – All pairs of sites are measured
  – Put data into measurement repository
• Set initial conditions to latest measurements
Path Emulator (detail)