Beyond Simulation: Large-Scale Distributed Emulation of P2P Protocols

Nathan S. Evans    Christian Grothoff

Technische Universität München

August 8, 2011

Presented by: Bartlomiej Polot and Matthias Wachs
Ideal use of simulation for security testing.
Systems Research — Simulation

Ideal use of simulation for security testing.

Implementation

Abstract

Simulation

Interpret

Bug
Systems Research — Simulation

Reality for most simulation usage.
### Systems Research — Emulation

<table>
<thead>
<tr>
<th>APP</th>
<th>APP</th>
<th>APP</th>
<th>APP</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>OS</td>
<td>OS</td>
<td>OS</td>
</tr>
<tr>
<td>VLAN</td>
<td>VLAN</td>
<td>VLAN</td>
<td>VLAN</td>
</tr>
<tr>
<td>OS</td>
<td>OS</td>
<td>OS</td>
<td>OS</td>
</tr>
</tbody>
</table>

**LAN/WAN**
Our Emulation Approach

![Diagram of LAN/WAN, APP, OS, Test Control App, LAN/WAN]
GNUnet Architecture

- P2P framework
- Focus on security
- Written in C
- Portable & extendable
- Multi-process architecture & IPC
- Extensive utility library
The Transport Service

- Low-level P2P connectivity
- Transport plugins: provide many connection options
- Unix domain sockets
- Blacklisting & whitelisting
P2P Emulation Steps

- Design P2P application
- Implement as GNUnet service
- Use built-in statistics or design logging facility
- Create test control application
  - Links against emulation library
  - Peer group startup/shutdown
  - Utilizes API to access service
Our Emulation Approach
Single Peer Startup Sequence

- **Running Peer**
  - **Configuration File**
  - **Hostkey File**

- **Test Control Application**
  - **Running P2P App**
  - **Emulation Library**

- **Start**
- **Stop**
- **Shutdown**
- **Generate**
- **Write**
- **Events**
- **Requests**
- **Time**

**GNUnet**

N. Evans, C. Grothoff (TUM)

August 8, 2011
Peer Group

- “Peer group” is the handle to running peers
- Layering — peer group reuses single peer startup code
- Peer group features
  - Configuration mangling
  - Resource allocation, throttling
  - Connects peers in desired topology
  - Capture running topology/statistics
  - Start/stop/reconfigure peers
  - Induce churn
  - Provide handles to specific peers
Peer Group Startup, Code Example

```
struct GNUNET_TESTING_PeerGroup *
GNUNET_TESTING_peergroup_start(const struct GNUNET_CONFIGURATION_Handle *cfg,
    unsigned int total,
    struct GNUNET_TIME_Relative timeout,
    struct GNUNET_TESTING_NotifyConnection connect_cb,
    struct GNUNET_TESTING_NotifyCompletion peergroup_cb,
    void *peergroup_cls,
    const struct GNUNET_TESTING_Host *hosts);
```

```
GNUNET_CONFIGURATION_load (testing_cfg, "~/test.conf");
struct MyClosure *data; /* your data here */
struct GNUNET_TESTING_Host *hosts = GNUNET_TESTING_HOSTS_load("~/hosts.conf");
pg = GNUNET_TESTING_peergroup_start (testing_cfg, 20000, TIMEOUT, &connect_cb,
    &peergroup_cb, data, hosts);
/* peergroup_cb must eventually call: */
GNUNET_TESTING_daemons_stop (pg, TIMEOUT, &shutdown_cb, data);
```
Network Topologies

Simple topology creation/import/export

(a) 2d-grid

(b) Small-World

(c) Erdos-Renyi

(d) InterNAT
Topology Generation and Evolution

(a) Initial  
(b) 2 Minutes  
(c) 5 Minutes  
(d) 10 minutes  
(e) 15 minutes  
(f) 30 minutes
Limitations of Emulation

- Timing accuracy
  - Network latency
  - Throughput
- Underlying OS interference
  - CPU scheduling
  - Disk access
  - Memory usage
- Speed
  - Shared IP/hostnames
  - Peer diversity
  - GNUnet
Overcoming Limitations

- Single OS per peer
  ⇒ Testing framework can be used on lower level emulators focused on timing accurate results
    - PlanetLab, Emulab, DETER, etc.
    - Sacrifice scalability
- Shared IP/hostnames — Virtual addresses, VMs
- Peer diversity — Configure per-peer bandwidth, VMs
- GNUnet — Benefit and limitation
Important Lessons Learned

- Cryptography
- Start-up time
- Periodic tasks
- Sockets
- Memory
# Peer and Emulation Performance

## Memory consumption

<table>
<thead>
<tr>
<th>Service</th>
<th>Non-shared</th>
<th>Heap</th>
<th>Shared</th>
</tr>
</thead>
<tbody>
<tr>
<td>supervisor</td>
<td>228 KB</td>
<td>32 KB</td>
<td>2,364 KB</td>
</tr>
<tr>
<td>transport</td>
<td>359 KB</td>
<td>99 KB</td>
<td>2,888 KB</td>
</tr>
<tr>
<td>core</td>
<td>300 KB</td>
<td>84 KB</td>
<td>2,428 KB</td>
</tr>
<tr>
<td>dht</td>
<td>536 KB</td>
<td>240 KB</td>
<td>3,684 KB</td>
</tr>
<tr>
<td>total</td>
<td>1,424 KB</td>
<td>456 KB</td>
<td>11,364 KB</td>
</tr>
</tbody>
</table>
## Peer and Emulation Performance

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Hosts</th>
<th>Cores (Total)</th>
<th>Memory (Total)</th>
<th>Peers</th>
<th>Connections per second</th>
<th>Time to start peer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cortex-A8</td>
<td>1</td>
<td>1</td>
<td>512 MB</td>
<td>100</td>
<td>~ 1</td>
<td>~ 206 ms</td>
</tr>
<tr>
<td>Xeon W3505</td>
<td>1</td>
<td>2</td>
<td>12 GB</td>
<td>2,025</td>
<td>~ 60</td>
<td>~ 12 ms</td>
</tr>
<tr>
<td>Xeon W3520</td>
<td>1</td>
<td>8</td>
<td>12 GB</td>
<td>2,025</td>
<td>~ 188</td>
<td>~ 5 ms</td>
</tr>
<tr>
<td>Opteron 8222</td>
<td>1</td>
<td>16</td>
<td>64 GB</td>
<td>10,000</td>
<td>~ 327</td>
<td>~ 27 ms</td>
</tr>
<tr>
<td>Opteron 850</td>
<td>31</td>
<td>124</td>
<td>217 GB</td>
<td>80,000</td>
<td>~ 559</td>
<td>~ 1 ms</td>
</tr>
</tbody>
</table>
Example: Comparison of DHT Performance

- Performance comparison of different DHT implementations
- 60,000 peers
- Specific peers were changed into malicious sybil nodes
- Success rate of requests measured
Example: NSE Implementation

- Network Size Estimation algorithm
- 2 days to implement
- 2 weeks from idea to paper
- Single host: 4,000 peers
Conclusion

- Framework available at https://gnunet.org
- We encourage people to use our framework
- 80,000 peers on cluster:
  what happens on supercomputer?
- at least consider:
  emulation vs. simulation even at large scale
Questions?