# Beyond Simulation: Large-Scale Distributed Emulation of P2P Protocols

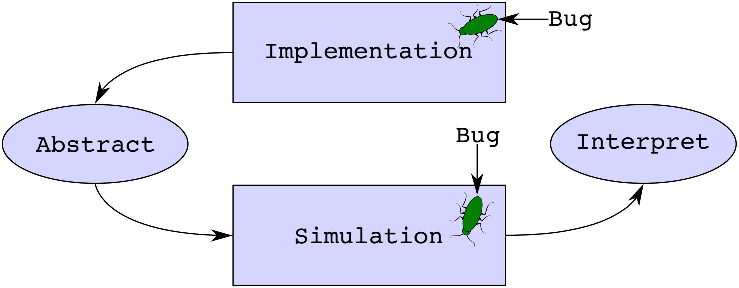
Nathan S. Evans    Christian Grothoff

Technische Universität München

August 8, 2011

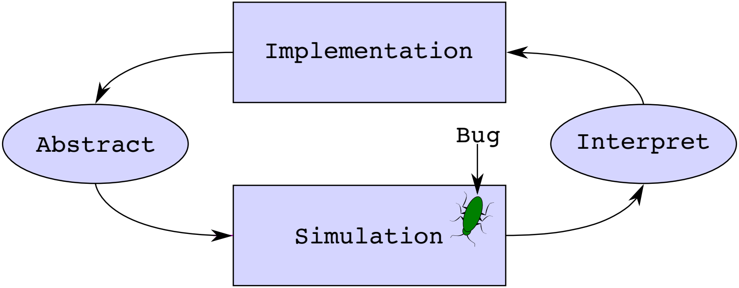
**Presented by: Bartlomiej Polot and Matthias Wachs**

# Systems Research — Simulation



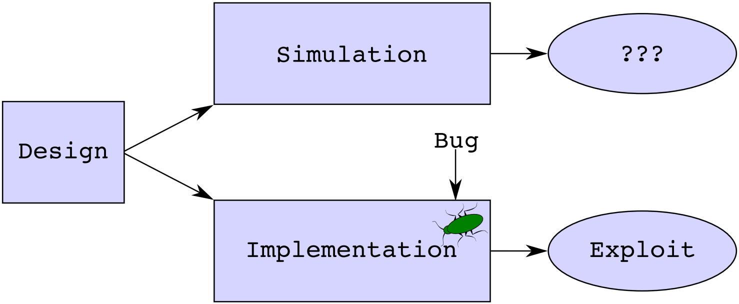

Ideal use of simulation for security testing.

# Systems Research — Simulation




Ideal use of simulation for security testing.

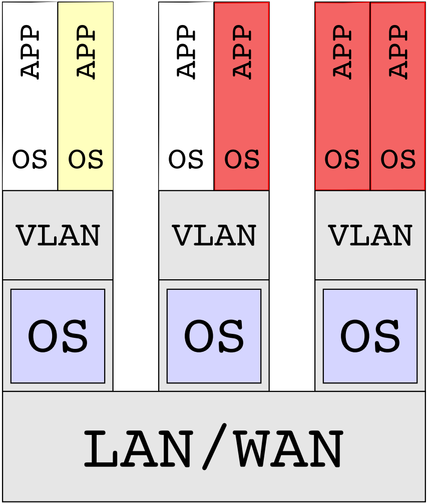
# Systems Research — Simulation



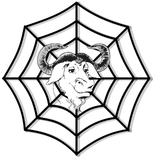

Reality for most simulation usage.

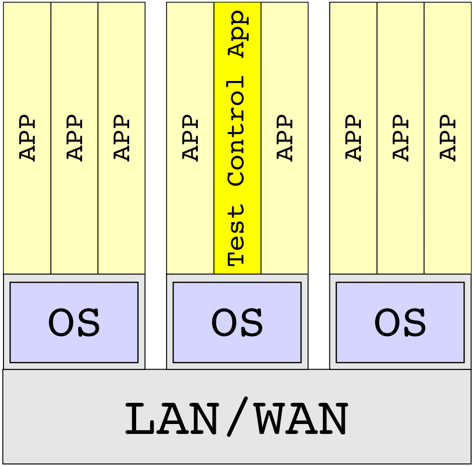
# Systems Research — Emulation

DETER



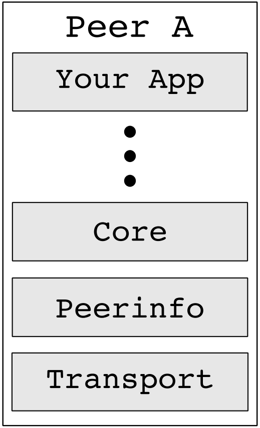
# Our Emulation Approach





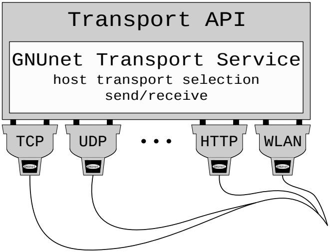
# GNUnet Architecture

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



# The Transport Service

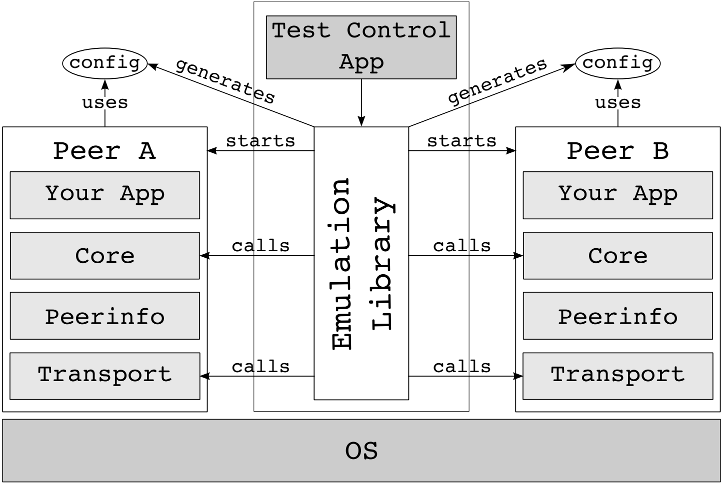
- Low-level P2P connectivity
- Transport plugins:
  provide many connection options
- Unix domain sockets
- Blacklisting & whitelisting



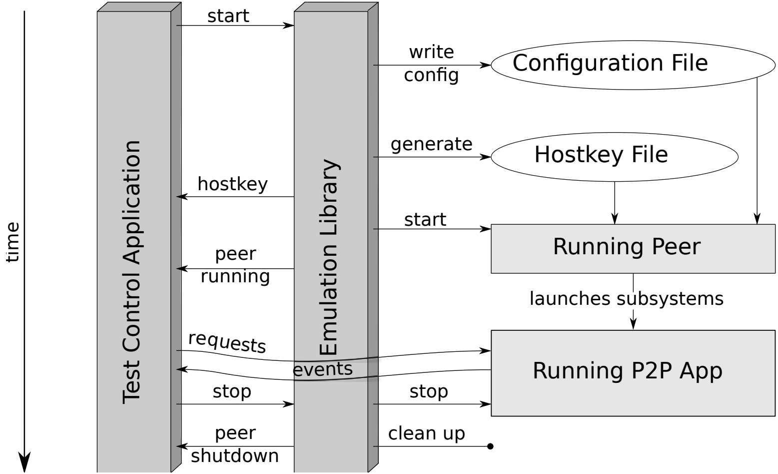
# P2P Emulation Steps

- 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



# 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

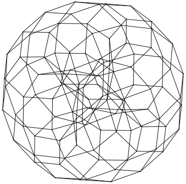
# Peer Group Startup, Code Example

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

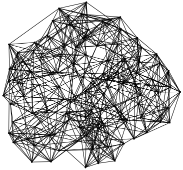
```c
GNUNET_CONFIGURATION_load (testing_cfg, "~/test.conf");
struct MyClosure *data; /* your data here */
struct GNUNET_TESTING_Host *hosts = GNUNET_TESTING_HOSTS_load("~/hosts.conf");
pg = GNUNET_TESTING_peergroup_start (testing_cfg, 20000, TIMEOUT, &connect_cb,
                                    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

# Network Topologies

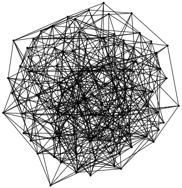
*Simple* topology creation/import/export



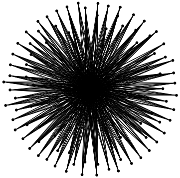
(a) 2d-grid



(b) Small-World

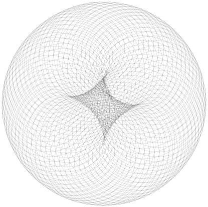


(c) Erdos-Renyi

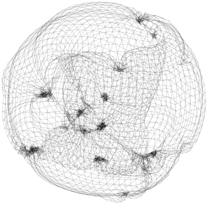


(d) InterNAT

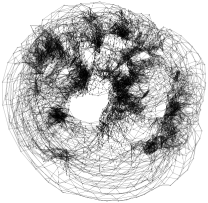
# Topology Generation and Evolution



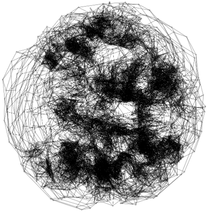
(a) Initial



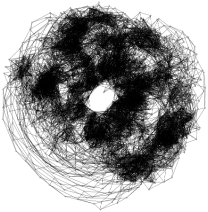
(b) 2 Minutes



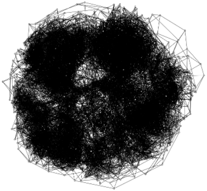
(c) 5 Minutes



(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

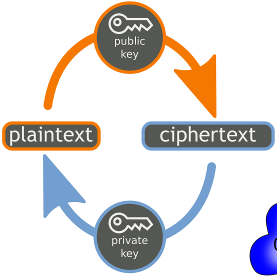
- Speed
- Shared IP/hostnames
- Peer diversity
- GNUnet

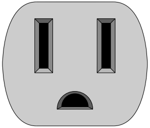
# Overcoming Limitations

- Single OS per peer
  $\Rightarrow$ 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

| Service | Non-shared | Heap | Shared |
|---|---|---|---|
| supervisor | 228 KB | 32 KB | 2,364 KB |
| transport | 359 KB | 99 KB | 2,888 KB |
| core | 300 KB | 84 KB | 2,428 KB |
| dht | 536 KB | 240 KB | 3,684 KB |
| total | 1,424 KB | 456 KB | 11,364 KB |

# Peer and Emulation Performance

| Architecture | Hosts | Cores (Total) | Memory (Total) | Peers | Connections per second | Time to start peer |
|---|---|---|---|---|---|---|
| Cortex-A8 | 1 | 1 | 512 MB | 100 | $\sim$ 1 | $\sim$ 206 ms |
| Xeon W3505 | 1 | 2 | 12 GB | 2,025 | $\sim$ 60 | $\sim$ 12 ms |
| Xeon W3520 | 1 | 8 | 12 GB | 2,025 | $\sim$ 188 | $\sim$ 5 ms |
| Opteron 8222 | 1 | 16 | 64 GB | 10,000 | $\sim$ 327 | $\sim$ 27 ms |
| Opteron 850 | 31 | 124 | 217 GB | 80,000 | $\sim$ 559 | $\sim$ 1 ms |

# 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?

?