AN/-Sim/1
AN/-Sim/:ActiveNetworkSimulator
Samrat Bhattacharjee
Ken Calvert
Ellen Zegura

Network and Telecommunications Group

Georgia Institute of Technology
College of Computing
Atlanta, Georgia, USA.

http://www.cc.gatech.edu/projects/canes
AN-Sim is:

An active network simulator

AN-Sim

Designed to work with large topologies

Yet configurable (at runtime)

Written mostly in C, parts in C++

(Yet another general purpose discrete event simulator)
AN-Sim : Design Goals

- Gracefully incorporate arbitrary node designs
  - able to experiment with different active functions
  - and combination of active and non-active nodes
- Be able to
  - support wide-area topologies — $O(10^3)$ nodes
  - simulate total number events — $O(10^7)$ events
  - support several (text/graphical) front ends
- ... and be reasonably quick
Users not averse to writing in non-scripting languages

Not for detailed simulation of small systems

Correlation of property with topology
- e.g. How does the protocol work when the number of active nodes is doubled/halved?
- e.g. How does cache hit ratio change with increase in degree of backbone nodes?
- Correlation of property with topology
- Interested in topology, network design properties

AN-Sim: Assumptions
AN-Sim: Topologies

The AN-Sim project uses the Georgia Tech Internet Topology Toolkit (http://www.cc.gatech.edu/projects/gt-tim) to provide models of network geography, i.e., structure that goes beyond simple topology to include policy and other considerations, including known scaling properties. Compositional techniques for abstracting large internets as aggregates of smaller geographical components.

gt-tim provides:

- Models of network geography, i.e., structure that goes beyond simple topology to include policy and other considerations, including known scaling properties.
- Compositional techniques for abstracting large internets as aggregates of smaller geographical components.
AN-Sim: Design

- Designed with autonomous code modules that use *Synchronous Events* to communicate.

- Graph models*, Module Registry, Event Registry, and Event Invocation are the only “core” parts of the simulator.

- Rest are configured at run time.
In general, modules not aware of other modules installed/active.

Thus, events provide an anonymous publish-subscribe interface.

Exports handlers for synchronous events.

May bind to and raise events.

Implements specific functionality.

Named code blocks.

---

**Modules and Events**
Currently available modules:

- event-gen
- dist-zipf
- multicast
- node-arch-0
- node-arch-1
- text-log
- socket
Arbitrary number of handlers for each event

Event Registry — dynamic event list

Example:

Generic event interface

Event handlers bind to specific events

Mechanism for composition and communication
Function pointers

Instead of invoking event for these functions, use a generic

uniformly at random between 0..n.

E.g. Distribution and generation functions - generate integer

Some events will only have one function bound

minor Detail: Generic Functions
<table>
<thead>
<tr>
<th>Event Module</th>
<th>Event Handle</th>
<th>Event Log</th>
<th>Event Node</th>
<th>Log Event</th>
<th>UNICAST DG ARRIVAL</th>
<th>UNICAST DG FORWARD</th>
<th>AN ACTIVATE BEGIN</th>
<th>AN EVENT COMPLETE</th>
<th>AN NULL EVENT LIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>unicast-reply</td>
<td>n-de-log</td>
<td>unicast-fwd</td>
<td>an-fn-evaluator</td>
<td>log-node-arch</td>
<td>UNICAST DG ARRIVAL</td>
<td>UNICAST DG FORWARD</td>
<td>AN ACTIVATE-BEGIN</td>
<td>AN EVENT-COMPLETE</td>
<td>AN NULL-EVENT-LIST</td>
</tr>
<tr>
<td>log-event</td>
<td>generate-events</td>
<td>gen-object-td</td>
<td>gen-pop-dest</td>
<td>gen-query-source</td>
<td>dist-zipf</td>
<td>an-object-td</td>
<td>an-dest</td>
<td>dist-zipf</td>
<td>an-source</td>
</tr>
</tbody>
</table>

A Complete Example
Evaluation of Active Caching

- Caching as a module in Vo.1
- Several different access policies
- Cache Sizes — $O(10^3)$ items per interface
- Topologies: 700-2000 nodes
- Several different caching policies
- Evaluation of active caching

Case Study: Caching (Vo.0)
Representative Caching Result

- \(10^6\) Queries — 130 seconds (383 seconds routing setup)

- In general simulation: 4–20 minutes (Ultra-1, 167 MHz)

- ROUTING SETUP: approx. 30–40 seconds

- Hits, Misses, Flushing, Occupancy

- Round trip latencies, Average Lifetimes

Output:

- \(2 \times 10^9\) Objects, \(10^6\) Queries-Response pairs

- 1500 nodes, 150 servers, 60–1500 caches, 3.71 avg. degree

---

Simulation:

- Ultra-1, 167 MHz

Routing Setup:

- Approx. 30–40 seconds
Useful version available early next year

- Modules for Version 0.1 being implemented now

- Version 0.1 implemented in Summer–Fall 1997

- Version 0.0 implemented in Summer 1996