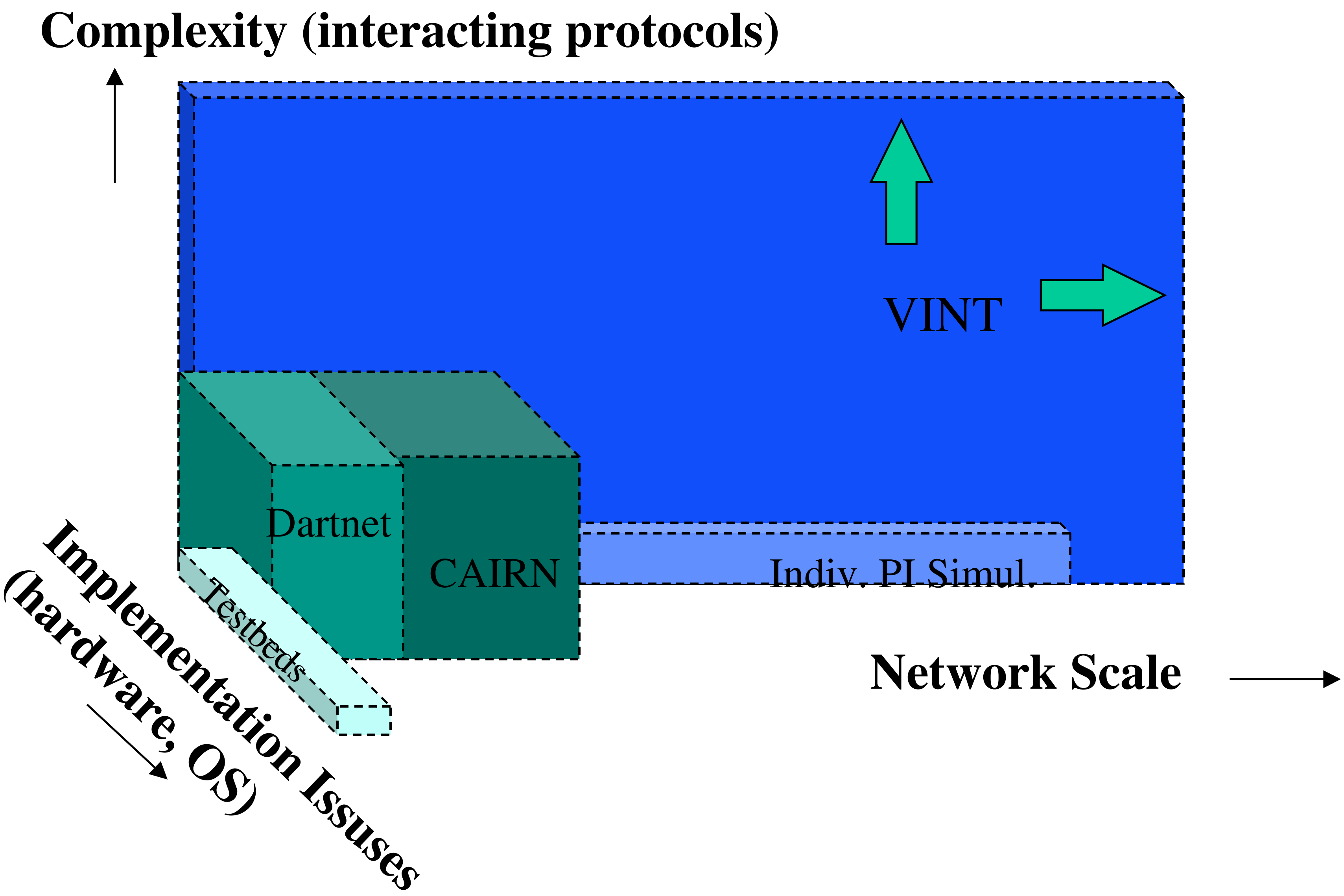
Virtual InterNet Testbed (VINT)

- Contributors (* Co-PIs):
  - USC/ISI: Deborah Estrin (*), Mark Handley, John Heideman, Ahmed Helmy, Polly Huang, Satish Kumar, Kannan Varadhan, Haobo Yu, Daniel Zappala
  - LBNL: Kevin Fall, Sally Floyd (*), Van Jacobson
  - UCBerkeley: Elan Amir, Steve McCanne (*), Teck-Lee Tung, Daedelus group
  - Xerox PARC: Lee Breslau (*), Scott Shenker (*)
  - other ns users

http://netweb.usc.edu/vint

http://www-mash.cs.berkeley.edu/ns/ns.html

- Funded under BAA 96-06.



Complexity (interacting protocols)

VINT

Dartnet

CAIRN

Testbeds

Indiv. PI Simul.

Network Scale

Implementation Issues (hardware, OS)

Outline

- Background: VINT goals and ns-v2 architecture
- In-use and in-progress protocol modules
- Research issues/projects

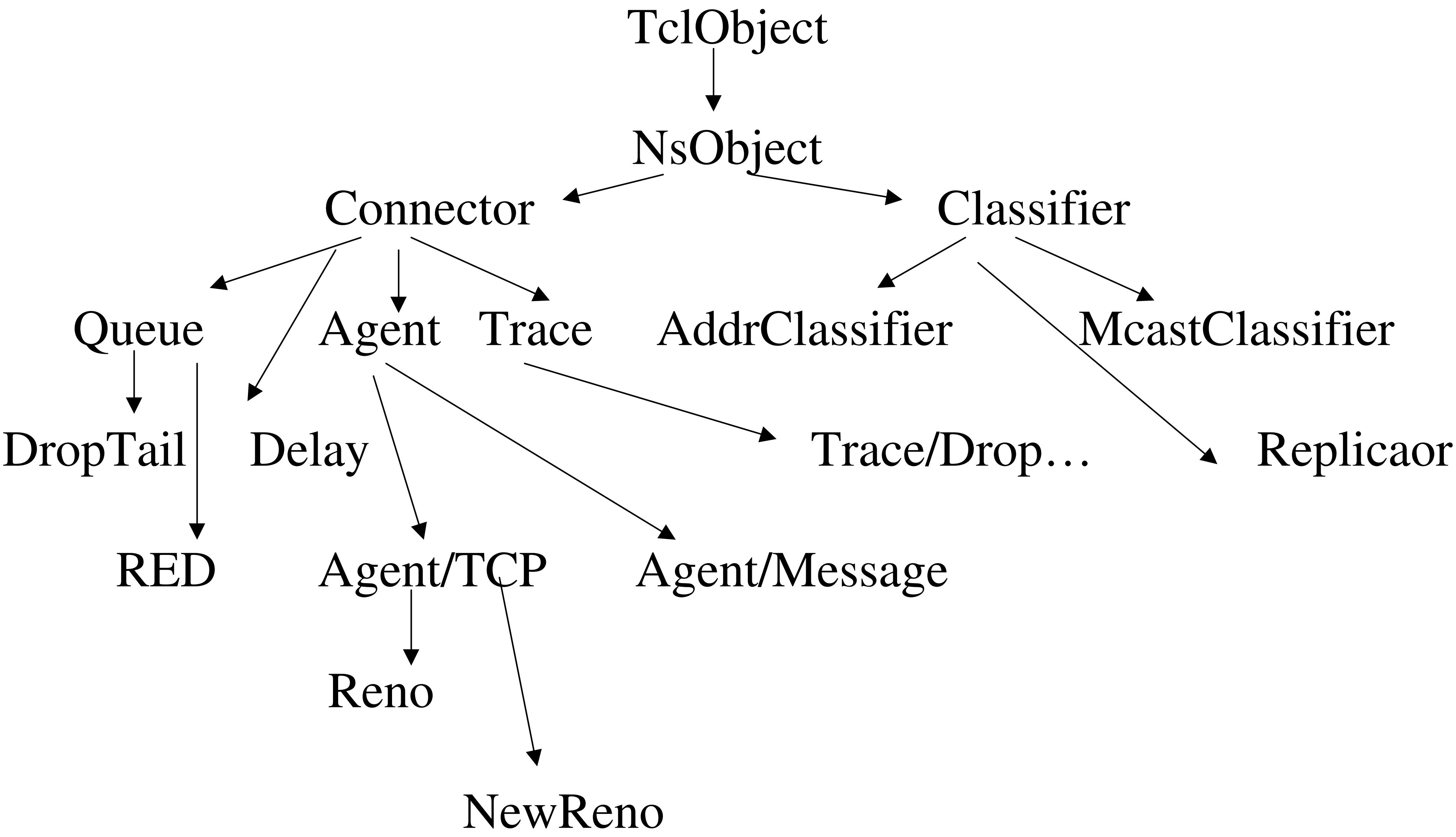
VINT goals

- Simulation infrastructure to study protocol design and behavior
- Protocol interaction
- Scaling
- Evaluation methods
- Visualization
- Emulation

VINT is built on ns
(originally developed at LBNL)

- C++ simulation engine, and for performance-intensive activities
- protocol definition and simulation configuration using a rich scripting language (OTcl)
- flexible object composition
- modules from multiple contributors mirrors modularity of protocols/development
- separate visualization, network animator (nam)

Class Hierarchy



TclObject

NsObject

Connector Classifier

Queue Agent Trace AddrClassifier McastClassifier

DropTail Delay Trace/Drop… Replicaor

RED Agent/TCP Agent/Message

Reno

NewReno

Protocol Modules

- Source agents
- TCP
- Queueing
- Unicast routing
- Multicast routing
- Reliable multicast
- MAC/LANs

Source Agents

- Telnet, Bulk file transfer
- CBR: both constant rate and periodic/quasi-periodic bursts
- UDP: variable pkt sizes
- Traffic generation based on distributions
- Others: CBR with RTP header, TAP emulation interface agent

TCP

- Many flavors: Tahoe, Reno, Sack, Vegas, Two-way
- Commonalities: configurable pkt size, fast retransmit, slow-start and congestion avoidance, dynamic RTT and RTX timeout
- Well developed validation test suites
- Examples of ongoing work: TCP pacing (Heideman), Emulation (Fall)

Packet Queueing/Scheduling

- Buffer management (Drop tail FIFO, RED)
- Packet scheduling (FIFO, CBQ, RoundRobin, FQ/WFQ)
- Trace (event triggered entry) and Monitoring (collect statistics) objects
- Examples of ongoing work:
  - nonconforming flow identification in RED routers (Fall and Floyd)

Unicast Routing

- Unicast routing precomputed
- Dynamic unicast recomputes on topology change
- Multi-path and asymmetric path routing
- Topology dynamics
  - Link failures may be exponential, deterministic, trace-driven, manual
- Hierarchical routing in progress for scalability

Multicast routing

- Two types supported:
  - Basic “mbone like” broadcast and prune (with static or dynamic topologies)
  - Centralized tree computation
  - Flexibly tradeoff detail and scaling
- Examples of ongoing work:
  - BGMP simulation
  - Detailed analysis of PIM protocols

Multicast transport

- implemented for programmability of: timers, session message handling, protocol event handling
- Reliable multicast (SRM-fixed and adaptive timers)
- RTP/RTCP
- Examples of ongoing work:
  - SRM adapted to non-transactional databases, Scalable session messages, SRM dynamics
  - Other RM schemes (RM research grou)
  - Multicast congestion control

MAC

- Multi-access LAN
- Wireless
- Developed at UCBerkeley under Daedelus project.

TAP

- Emulation interface
- ns controls packet drops, delays
- Examples:
  - McCanne: study srm/media board subject to wide range of simulated events
  - LBNL: tcpanaly on real host running across ns-simulated network
  - ISI: study some distributed application protocols by running across ns-simulated network and in simulated time.
- Not currently tackling hard timing issues.

Topology Generation

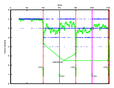
	Type of Graph	**Edge Models**
Ntg	n-level hierarchy	user configurable distributions
GT-ITM	flat, random n-level hierarchies, transit-stub nets	many
TIERS	3-level hierarchy	minimum spanning tree and random placement
rtg	flat random	Waxman

Validation/Test Suites

- In preparation for “all” ns functions
- Essential for managing multiple, distributed code contributors.
- Ongoing work: automated running of suites and comparison of results.

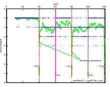
Protocol interaction studies

- End-to-end (transport) protocols and dynamic routing
  - TCP and SRM
- Multicast routing and Multiaccess LAN with selective packet loss
  - PIM-SM bug discovery



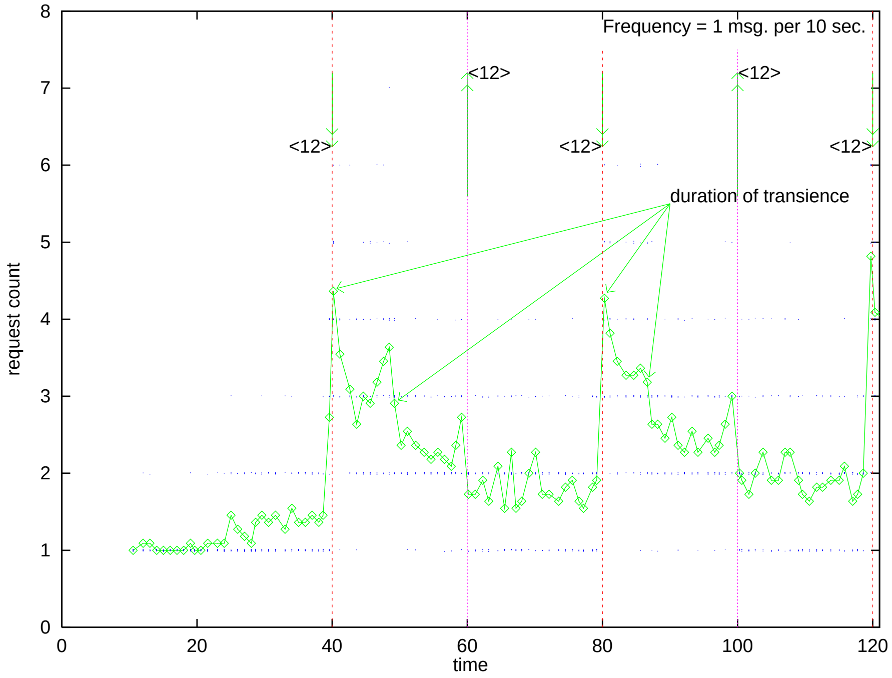
Example of variations in protocol activity following topology change

- Timers in SRM are a function of topological distance
- Requests to recover a lost packet increase with some changes



Example of design tradeoff associated with network dynamics

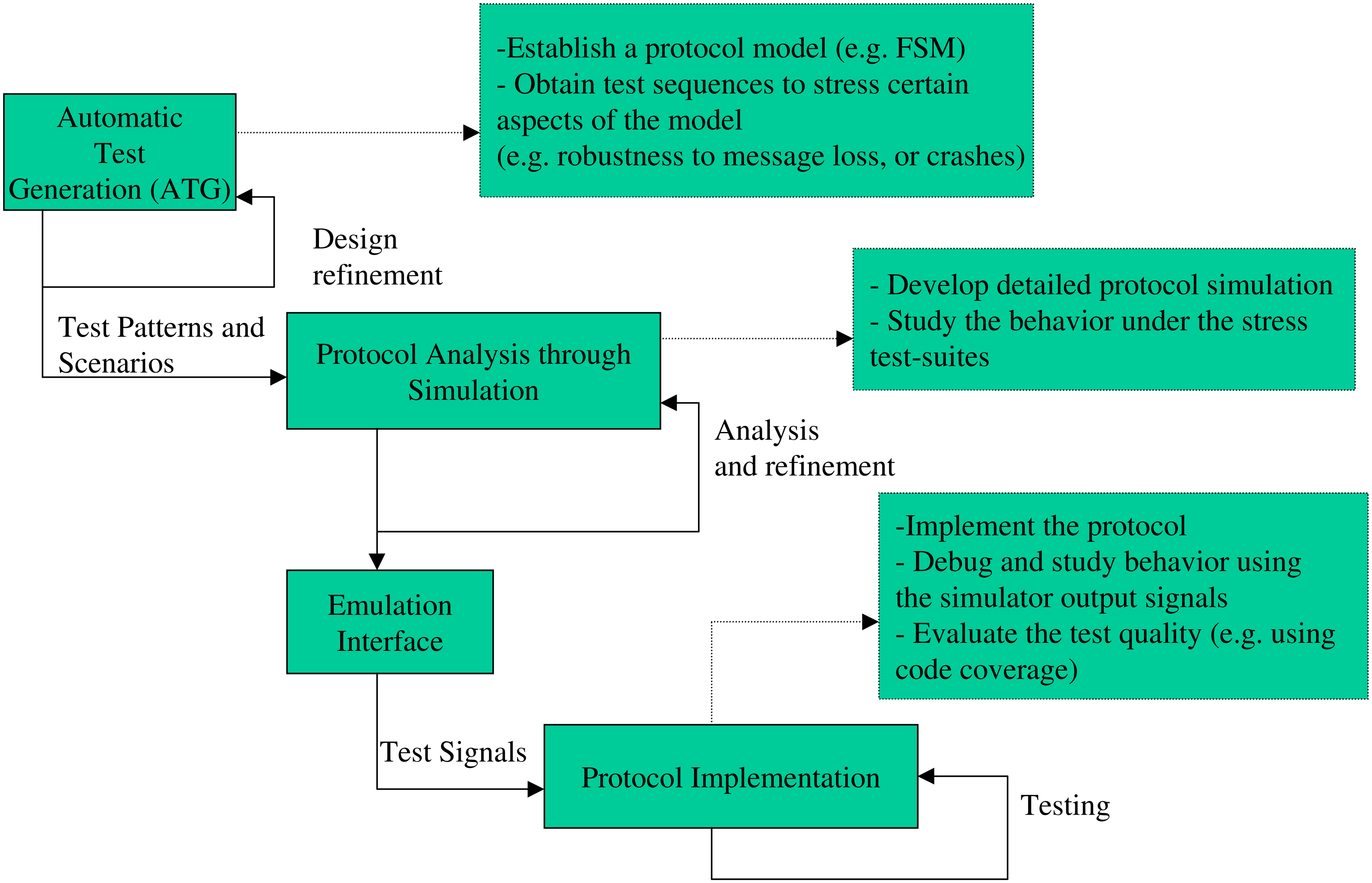
- Distance Estimation in SRM uses periodic session messages
- Tradeoff between session message overhead and prompt error recovery


Frequency = 1 msg. per 10 sec.
<12>
<12>
<12>
<12>
<12>
duration of transience
request count
time
0
1
2
3
4
5
6
7
8
0
20
40
60
80
100
120

Systematic protocol testing

- STRESS: drive simulator through systematically/automatically derived states.
- Applied to multicast routing protocol analysis
  - disclosed several corner-case bugs in PIM-SM design

A Framework for Design and Testing of Multicast Routing Protocols:



Automatic Test Generation (ATG)

-Establish a protocol model (e.g. FSM)
- Obtain test sequences to stress certain aspects of the model
(e.g. robustness to message loss, or crashes)

Design refinement

Test Patterns and Scenarios

Protocol Analysis through Simulation

- Develop detailed protocol simulation
- Study the behavior under the stress test-suites

Analysis and refinement

Emulation Interface

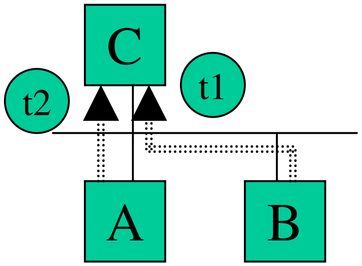
-Implement the protocol
- Debug and study behavior using the simulator output signals
- Evaluate the test quality (e.g. using code coverage)

Test Signals

Protocol Implementation

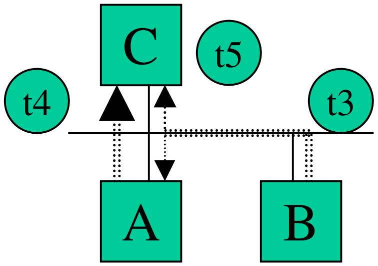
Testing

Example (selective) loss scenarios discovered using STRESS



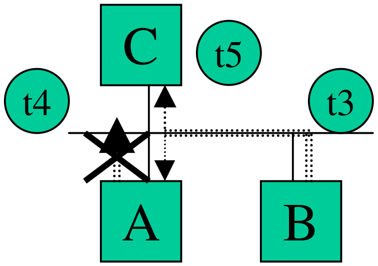
t1) *B* joins the group and send a join message
t2) *A* joins the group and sends a join message

- Case 0 (no loss):



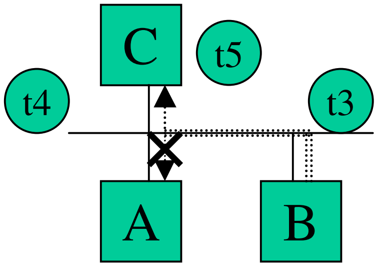
t3) *B* leaves group & mcasts prune onto LAN
t4) *A* receives prune & sends a join to override
t5) *C* receives join & does not delete mcast entry

- Case 1:



t3) *B* leaves group & mcasts prune onto the LAN
t4) *A* receives prune & sends join to override. Join is lost.
t5) *C* receives prune & deletes mcast entry.

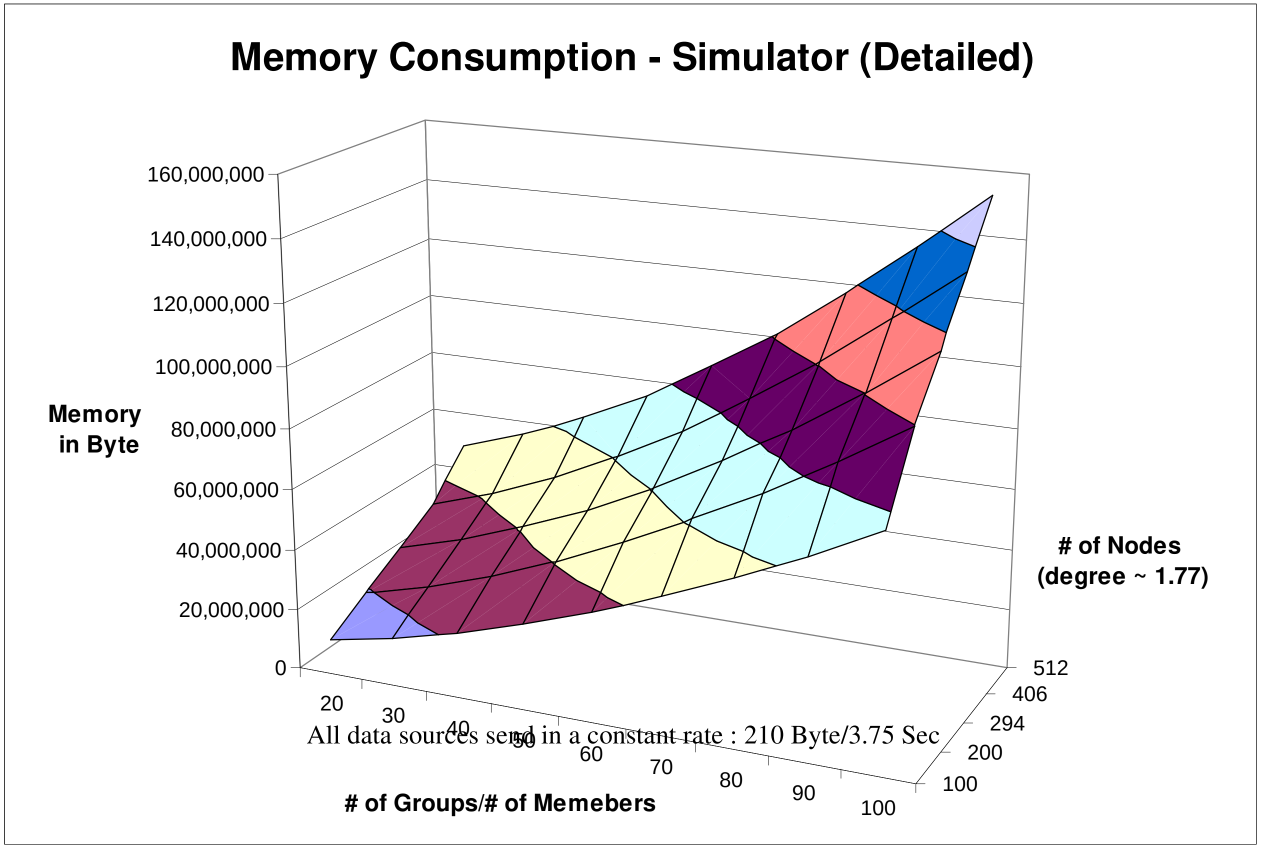
- Case 2:



t3) *B* leaves group & mcasts prune onto the LAN
t4) *A* does not receive the prune & does not override.
t5) *C* receives prune and deletes the mcast entry.

Scaling

- Multiple dimensions must be considered: nodes, links, group members, sources, groups, packets…



All data sources send in a constant rate : 210 Byte/3.75 Sec

Target Numbers

- Single user workstation (e.g. 64 to 128 MB)
- detailed simulations currently up to ~150 nodes, ~2000 links
- “session level” simulations currently achieving ~2000 nodes, ~8000 links...and still increasing
- ultimate goal is ~50,000 node network simulations

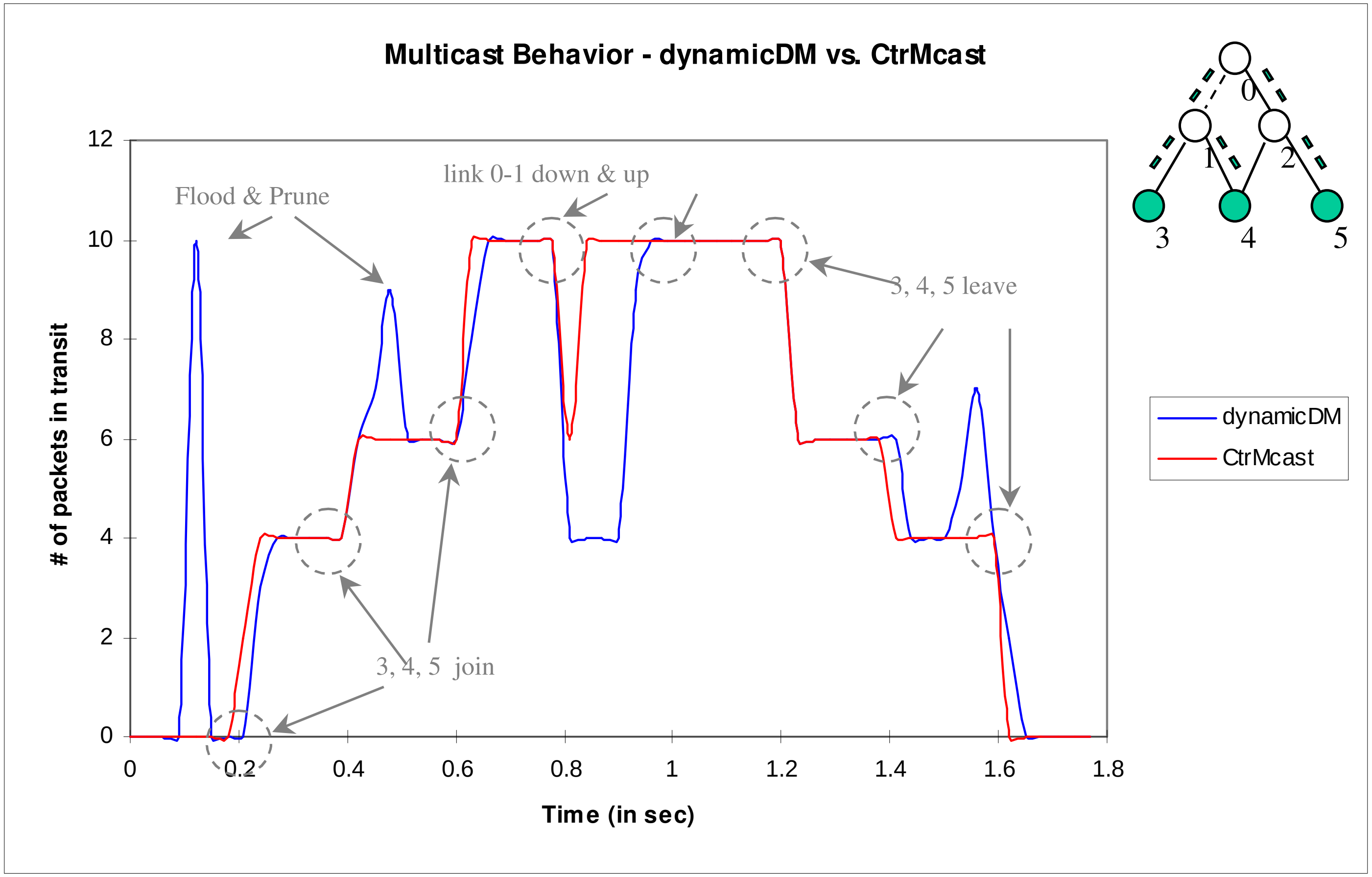
Scaling through Abstraction

- Common interfaces with different levels of detail allow detailed and session level simulations in the same environment
- Example: multicast transport or application with different degrees of underlying protocol abstraction
  - packets sent across centrally computed multicast trees, forwarded hop-by-hop through router queues
  - packets delivered subject to computed delay and loss models

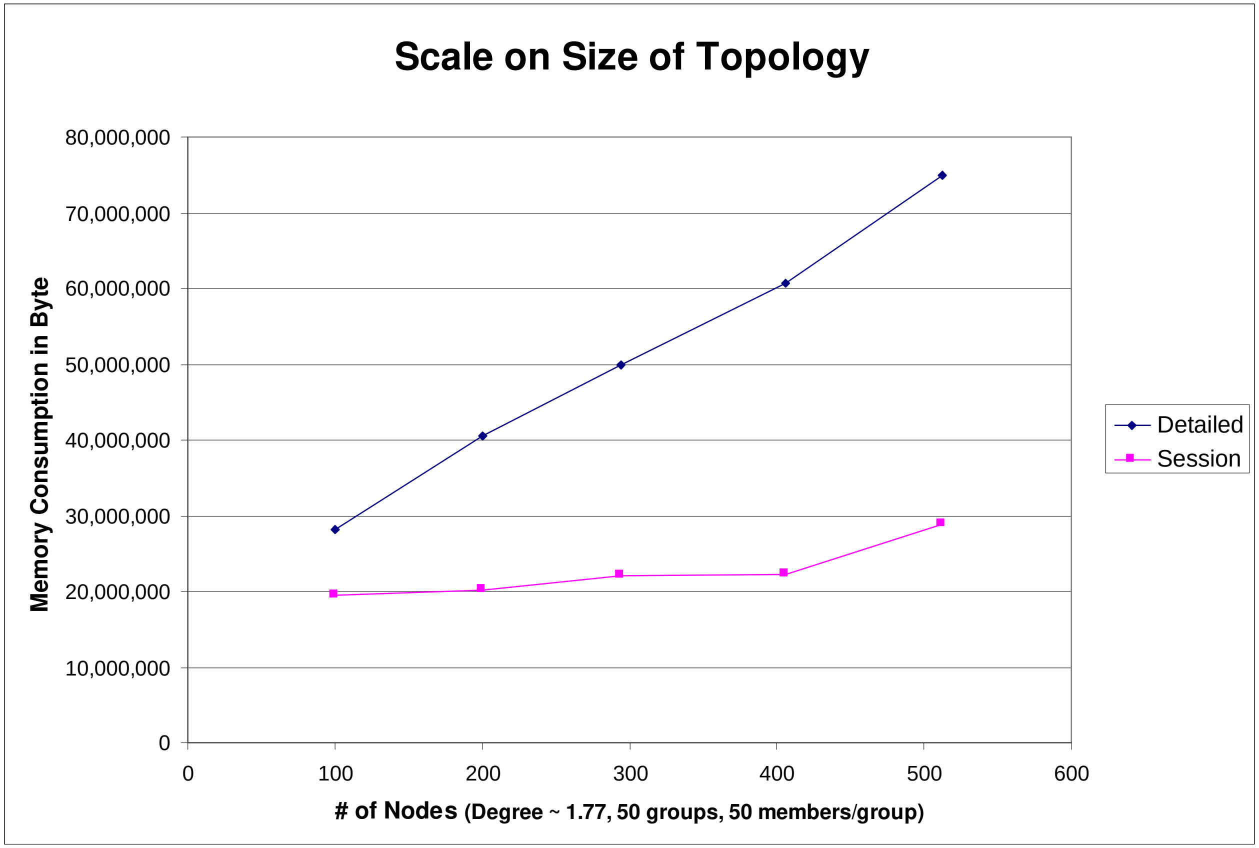
Validation of abstraction techniques

- Critical to compare results from detailed and session level simulations
- Example:
  - multicast routing: compare number of packets in transit for detailed and session level
- Ongoing work to characterize “approximation error” for wider range of protocols

Difference in Behavior - Multicast

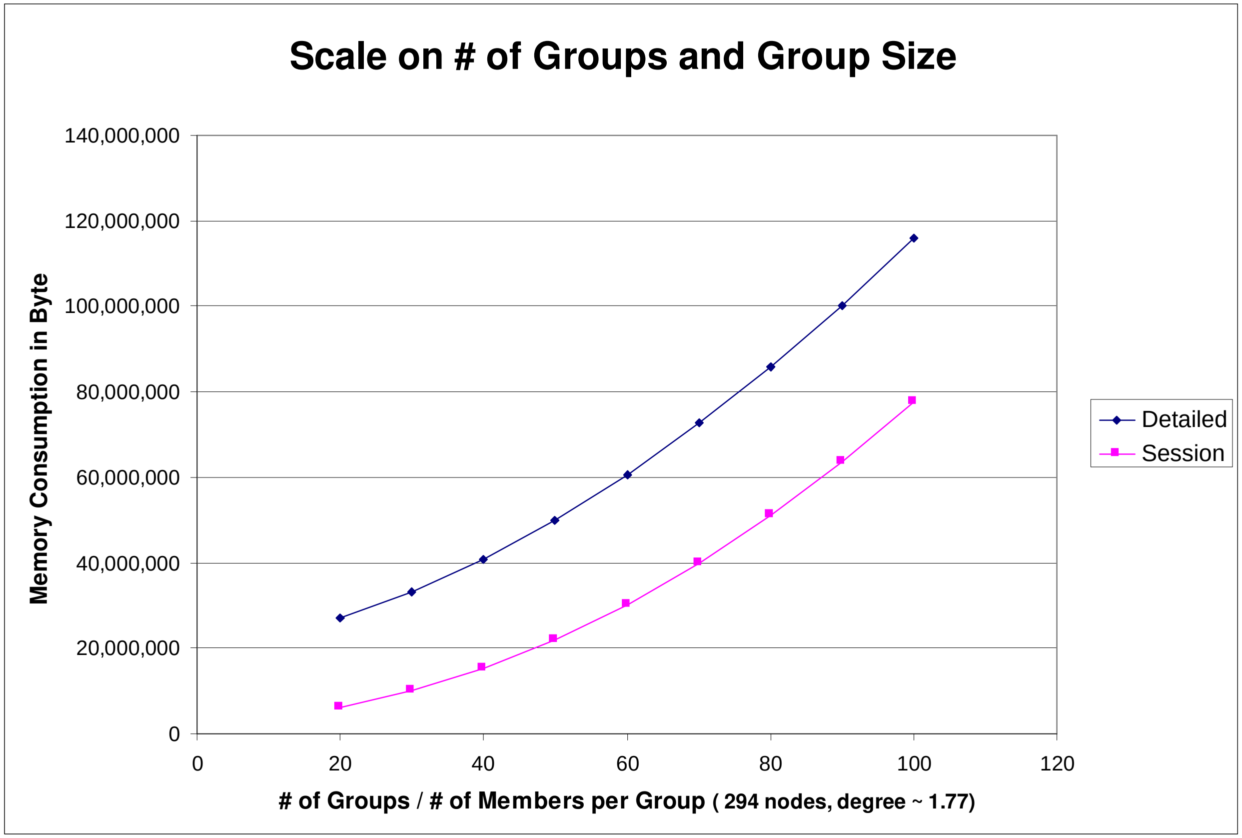


Ns Scaling - # Nodes



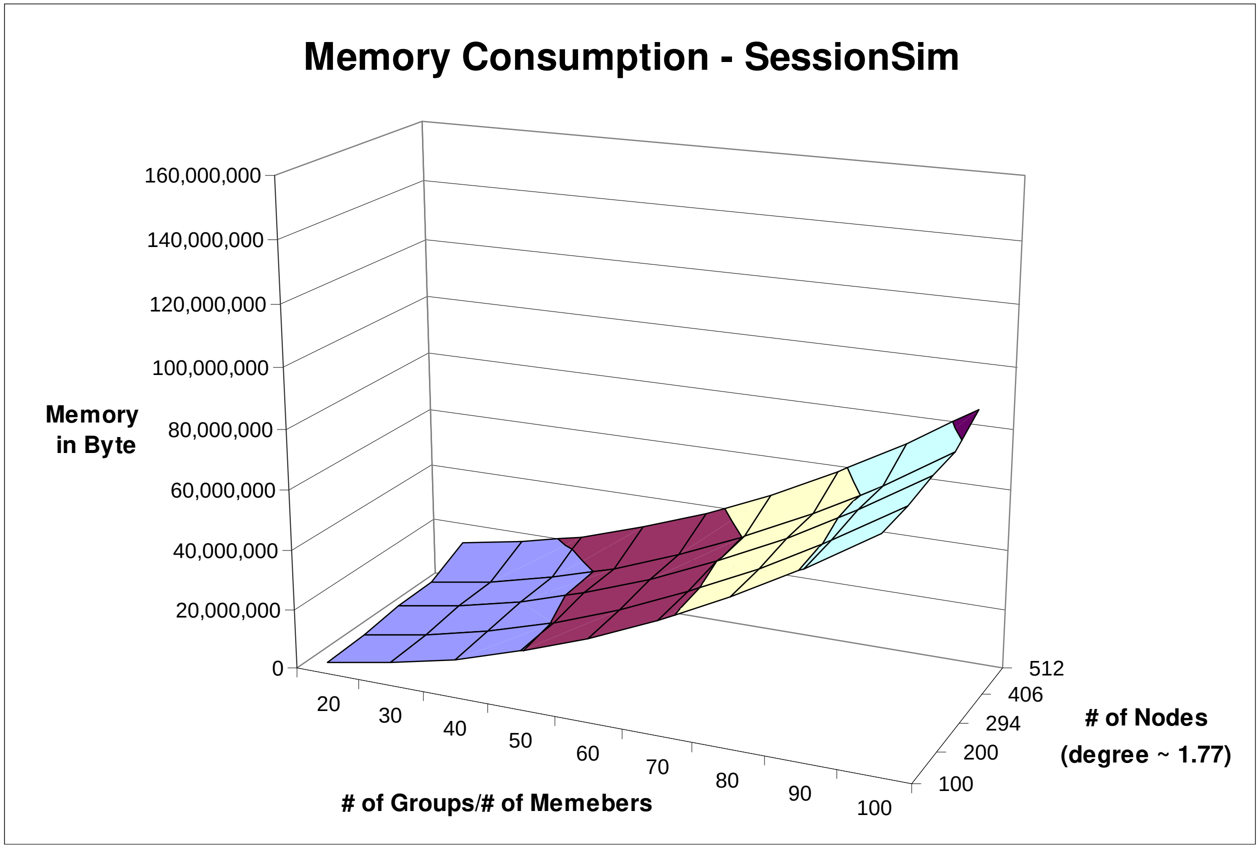
All data sources send in a constant rate : 210 Byte/3.75 Sec

Ns Scaling - # of Groups/Members



All data sources send in a constant rate : 210 Byte/3.75 Sec

Ns Scaling - Session



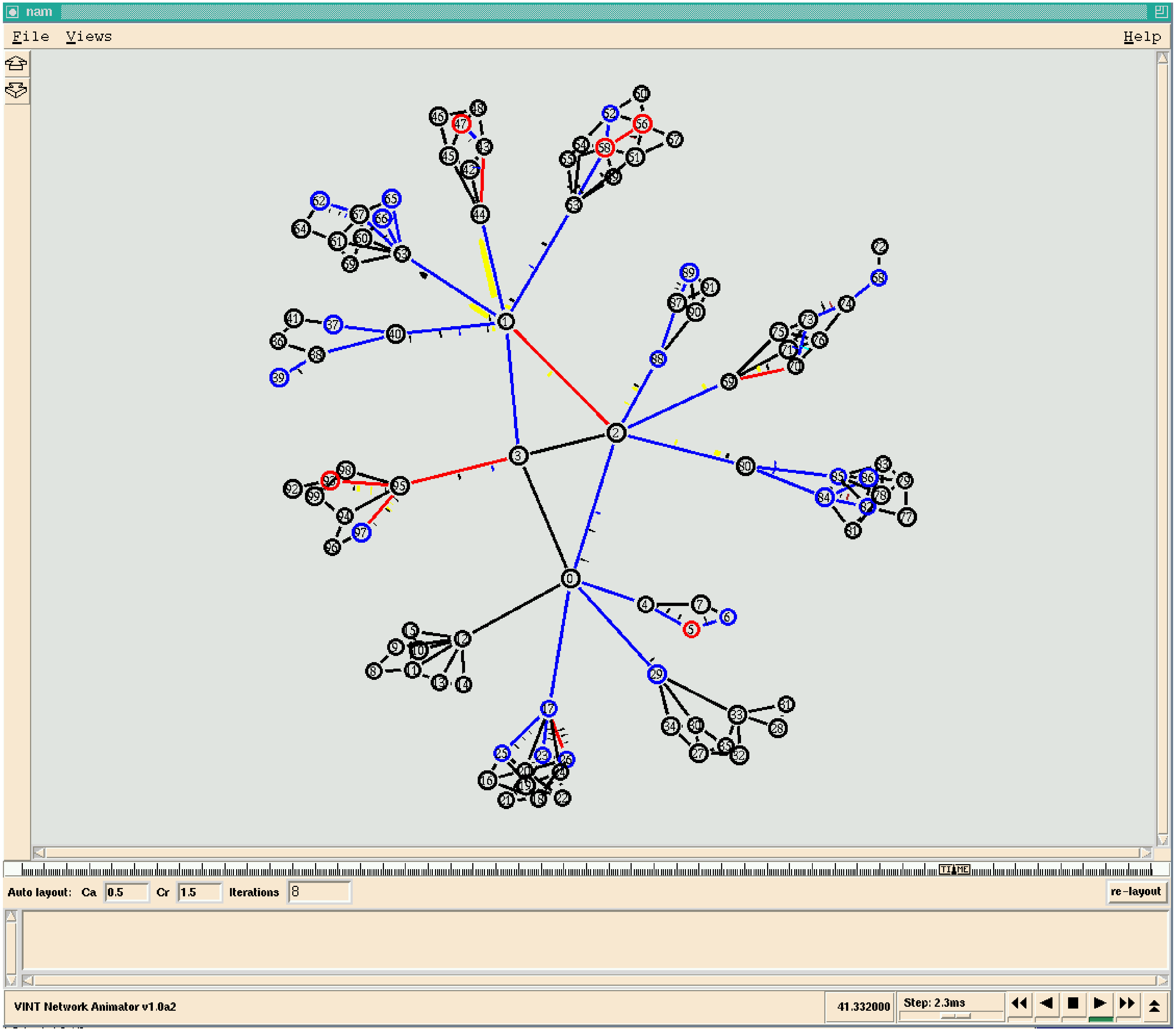
All data sources send in a constant rate : 210 Byte/3.75 Sec

Scaling Objective 50,000 node network

- Represent flexibly-defined “clouds” to achieve additional order of magnitude in scale.
  - few real networks/protocols have 50K interacting elements without clustering/hierarchy
- Other forms of selective filtering/modeling of irrelevant detail from parts of the topology
- Continue to work on “thinning” ns base objects in general.

Protocol visualization

- network animation for visualizing detailed packet transmission, queueing
  - network animator (nam) of packet flows and protocol states
  - interactive coloring/labeling, flexible annotations to find events, moving backwards
- support under development for scripted visualization of network/protocol objects
  - e.g., reliable multicast sender, loss, timer-events, repairs.
- graphical scenario construction



Auto layout: Ca 0.5 Cr 1.5 Iterations 8

VINT Network Animator v1.0a2

41.332000

Step: 2.3ms

Longer-term projects/goals

- Very large scale, higher-level simulations (e.g. multicast address allocation, distributed measurement system, web caching)
- Generalized methods for moving between and combining detailed and abstract simulations.
- Validation of modules
- Enhanced visualization/graphical front end
- Emergent-behavior studies
- Emulation interface

VINT Innovative Claims

- (100%) Composable component framework
- (50%) Libraries of topologies, workloads, scenarios, protocols
- (50%) Abstraction capabilities such as hybrid session and packet level simulation components with common UI and interfaces
- (25%) Visualization techniques and tools
- (50%) Emulation interface to real implementations
- (50%) Scaling studies of key next generation internet design issues

Conclusions

- ***Still excited by growing momentum behind use of ns and demand for VINT tools/infrastructure !!!***
- URLs:
  - http://netweb.usc.edu/vint
  - http://www-mash.cs.berkeley.edu/ns/ns.html
- Lists:
  - ns-users@mash.cs.berkeley.edu
  - ns-announce@mash.cs.berkeley.edu