Staged Simulation

Improving Scale and Performance of Wireless Network Simulations

Kevin Walsh

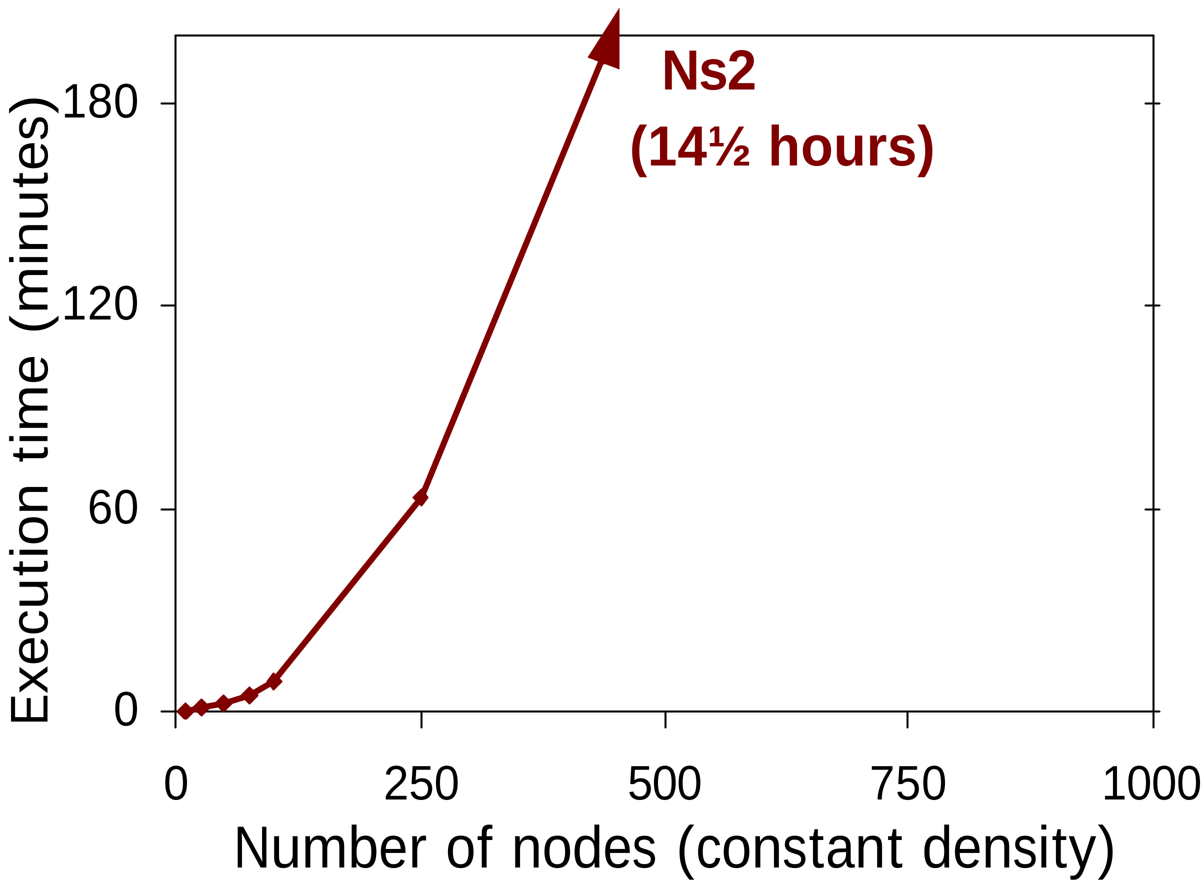
Emin Gün Sirer

Cornell University

Wireless Simulation

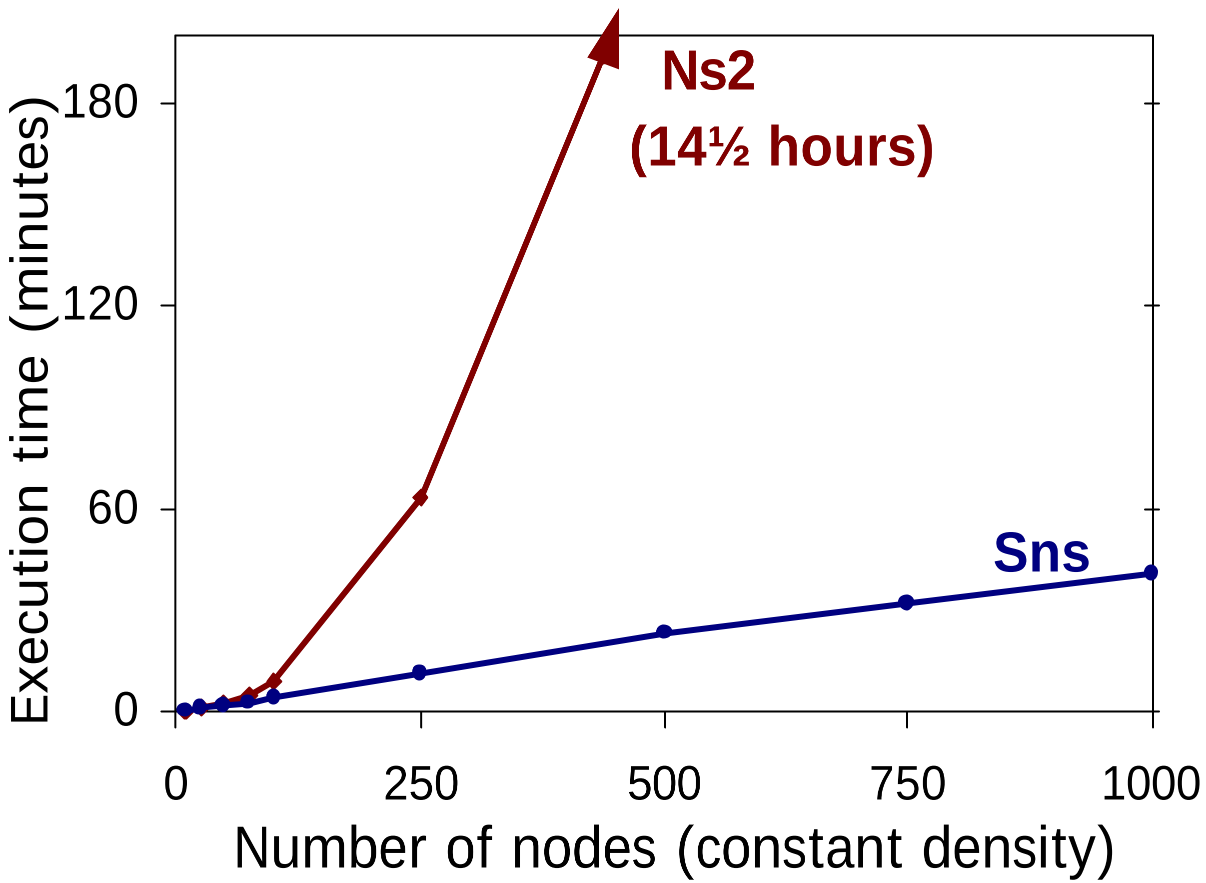
- Simulation for networking research
  - Ad hoc routing protocols
  - Mobile applications
  - Sensor networks
- Wireless simulation is often slow and does not scale
  - Takes too long
  - Works only for small networks

How bad is it?



Ns2
(14½ hours)
Execution time (minutes)
0
60
120
180
0
250
500
750
1000
Number of nodes (constant density)

How bad is it?


Ns2
(14½ hours)
Sns
Execution time (minutes)
0
60
120
180
0
250
500
750
1000
Number of nodes (constant density)

Insight

- Redundant computations, due to conservative structure of simulator
  - Can be highly dynamic
  - Recompute state whenever inputs *may* have changed
- State recomputed more often than strictly necessary

Redundancy Across Simulator Runs

- Simulator often used for large batch runs
  - Nearly identical scenarios/parameters
- Leads to redundancy across runs
  - Users can identify or predict

Contributions

- Identify problem
  - Redundant computations
- Propose approach: Staging
  - Reduce time spent in redundant computations
  - Intra- & inter-simulation staging
- Demonstrate speedup with practical simulator
  - No loss of accuracy or change in interface

Staging: Approach

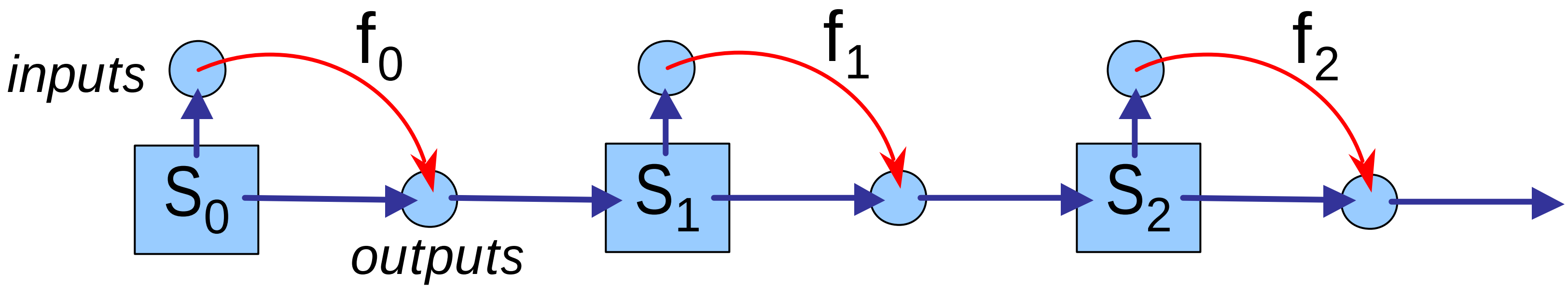
- Basic approach: Result caching and reuse
  - Limited use, due to real valued and continuously varying inputs: e.g. current simulation time
- Restructure events within a discrete event simulator to make them amenable to caching
  - Expose redundancy for caching and reuse
  - Leads to small, time-independent computations
- Use time-shifting where possible
  - Reorder events for maximum efficiency

Simulator Model

- Progression of *states* S_i
- *An event* e_i acts on S_i and produces S_{i+1}
  - Ex. Packet transmission, topology computation

Simulator Model

- Progression of *states* S_i
- *An event* e_i acts on S_i and produces S_{i+1}
  - Ex. Packet transmission, topology computation
- Can view events as functions
  - Inputs taken from current state
  - Output describes modifications to get next state



Restructuring Events

- Use currying
  - Decompose event into multiple events
  - Group part of computation that depends on slowly varying or discrete inputs
  - Example: node position nearly static

time

$$r = f(t, p_1, p_2, p_3, \ldots)$$

$$r' = f(t', p_1, p_2, p_3, \ldots)$$

Restructuring Events

- Use currying
  - Decompose event into multiple events
  - Group part of computation that depends on slowly varying or discrete inputs
  - Example: node position nearly static

time

$$r = f(t, p_1, p_2, p_3, \ldots) \qquad\qquad r = f'(t, g(p_1, p_2, p_3, \ldots))$$

$$r' = f(t', p_1, p_2, p_3, \ldots) \qquad\qquad r' = f'(t', g(p_1, p_2, p_3, \ldots))$$

Restructuring Events

- Use currying
  - Decompose event into multiple events
  - Group part of computation that depends on slowly varying or discrete inputs
  - Example: node position nearly static

time

$r = f(t, p_1, p_2, p_3, \ldots)$ $\Rightarrow$ $r = f'(t, \boxed{g(p_1, p_2, p_3, \ldots)})$

$r' = f(t', p_1, p_2, p_3, \ldots)$ $\Rightarrow$ $r' = f'(t', \boxed{g(p_1, p_2, p_3, \ldots)})$

Restructuring Events

- Use incremental computation
  - Reuse results of similar computations
  - Often relies on continuity w.r.t. an input
  - Example: topology similar at nearby times

time

$$r = f(t, p_1, p_2, p_3, \ldots)$$

$$r' = f(t', p'_1, p'_2, p'_3, \ldots)$$

Restructuring Events

- Use incremental computation
  - Reuse results of similar computations
  - Often relies on continuity w.r.t. an input
  - Example: topology similar at nearby times

time

$r = f(t, p_1, p_2, p_3, \ldots)$

$r' = f(t', p'_1, p'_2, p'_3, \ldots)$

$r = f(t, p_1, p_2, p_3, \ldots)$

$r' = f(t', p'_1, p'_2, p'_3, \ldots, r, t)$

Restructuring Events

- Use auxiliary results
  - Compute and save additional information
  - Example: bounded node speed

time

$r = f(t, p_1, \ldots)$

$r' = f(t', p'_1, \ldots)$

Restructuring Events

- Use auxiliary results
  - Compute and save additional information
  - Example: bounded node speed

time

$$r = f(t, p_1, \ldots) \qquad \{r, a, b\} = f'(t, p_1, \ldots)$$

$$r' = f(t', p'_1, \ldots) \qquad \{r', a', b'\} = f'(t', p'_1, \ldots, a, b)$$

Time-shifting

- Restructuring and caching provide opportunities for changing the time at which computations are performed
  - Smaller, time-independent events
- Architectural benefits
  - Better working set and cache performance using precomputation and event reordering
- Algorithmic benefits
  - More efficient algorithms using batch processing

Staging in Practice: Sns

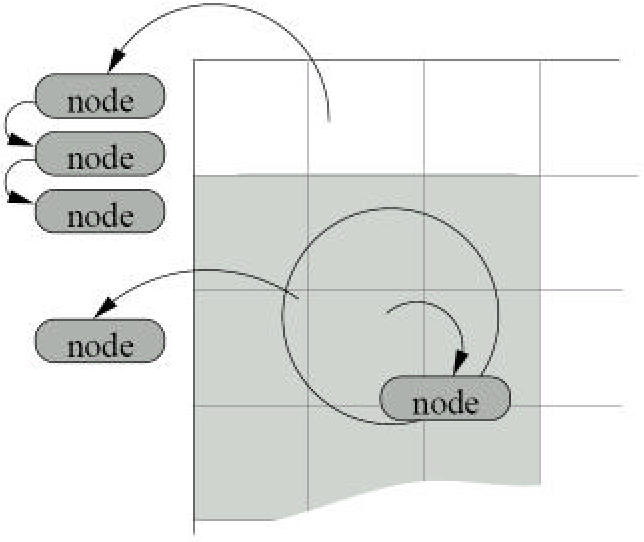
- Based on ns-2 wireless network simulator
  - Ns-2 wireless is slow; scales poorly
  - Implementation is typical
- Inter- and intra-simulation staging
- No change in accuracy or interface

Neighborhood Computation

- During each packet send, compute nodes in neighborhood of sender
- Expensive in ns-2
  - Full scan of network on every packet
  - Leads to redundant computations for typical networks
- Staging applied to neighborhood computation

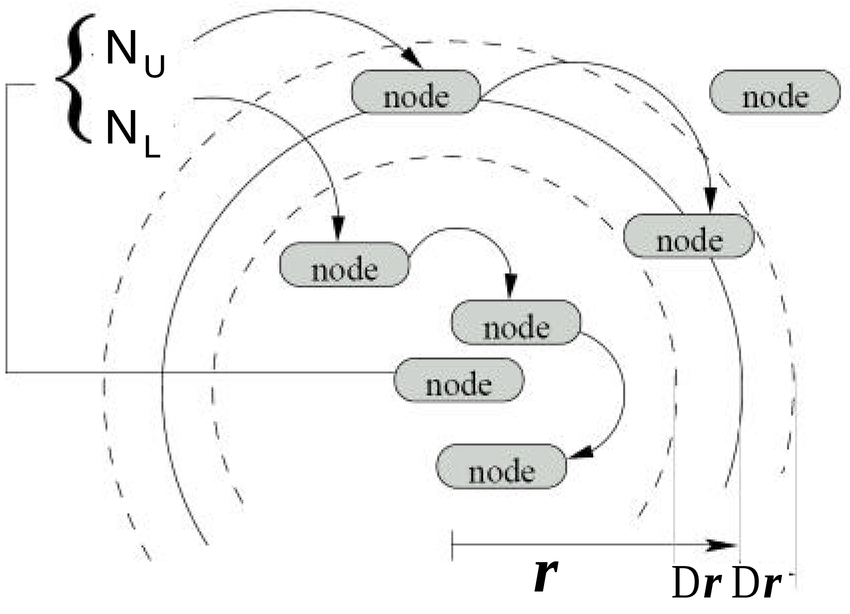
1: Grid-based Neighborhood Computation

- Use a grid to compute nodes within range
  - Reuse results for nearby nodes
  - Share grid maintenance across calls
- Reduces number of nodes examined
- Elementary form of staging
  - Currying and incremental computation



2: Neighborhood Caching

- Compute upper/lower bounds on neighborhood set, with expiration time ?*t*
- Refine bounds into exact result
- Many computations share same bounds
- Reduces number of nodes examined
- Staging by auxiliary results



3: Time-shifting

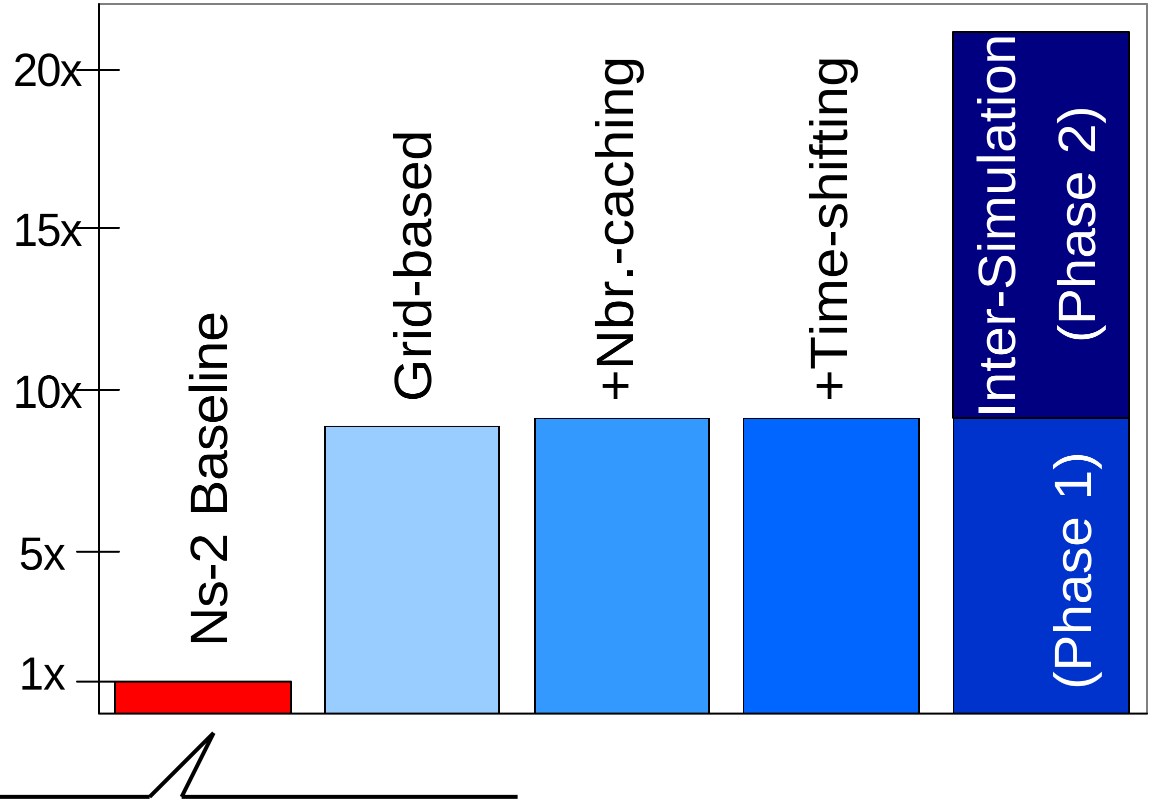
- Precompute all neighborhood cache entries together
- Compute on schedule, every ? *t* epoch
- Reduces total work (batching)
- Improves memory locality
- May introduce new work
  - Perform only under heavy load

4: Inter-simulation Staging

- Reuse neighborhood sets across simulation runs
  - All runs have same mobility scenario
  - Other simulation parameters may differ
- *Generate phase:* write neighborhood sets to disk
  - First run in batch
- *Use phase:* read sets from disk
  - No grid or topology needed if cache is complete
  - Need only the previously computed result cache

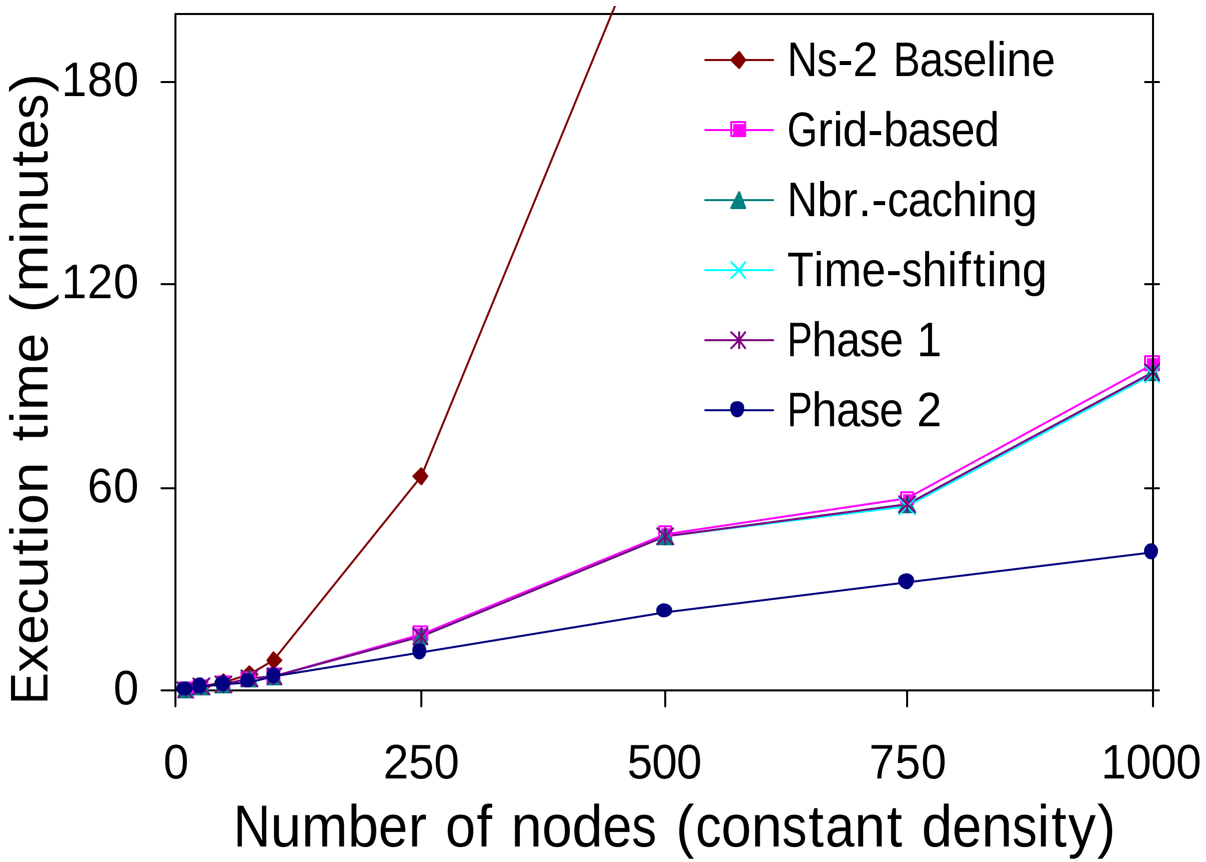
Execution Time Speedup

- 1000 nodes
- AODV routing
- Setup as in
  [Broch et al., 1998]



Baseline is approx. 2x faster than stock ns-2
(using standard optimization techniques)

Effect of Network Size



Related & Prior Work

- Existing instances of intra-simulation staging
  - NixVectors for wired networks [Riley et al. 2000]
  - Selective packet transmission [Wu & Bonnet 2002]
- Instances of inter-simulation staging
  - Splitting [Glasserman et al. 1996]
  - Cloning [Hybinette & Fujimoto 1997]
  - Updateable Simulations [Ferenci et al. 2002]
- Staging in other domains
  - Compilation [Chambers 2002], iterative programming, memoization

Conclusions

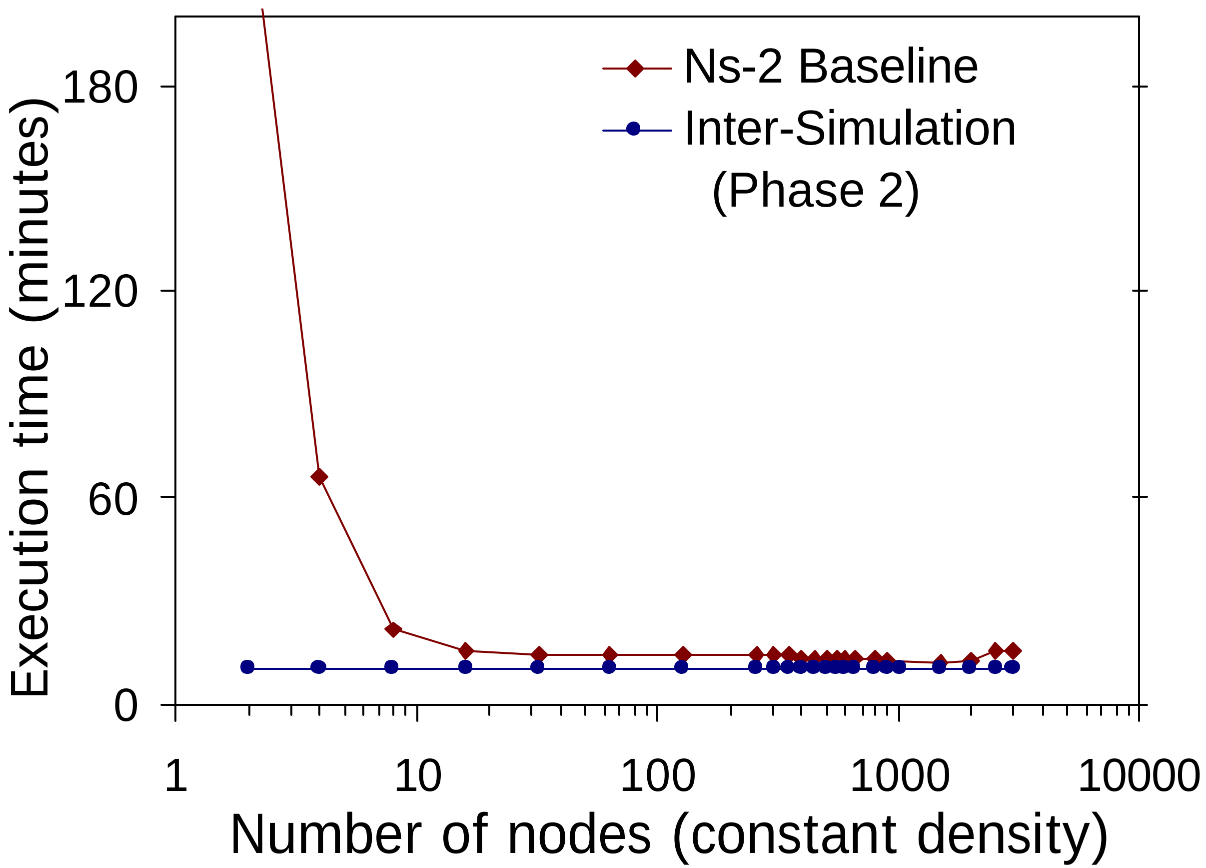
- Insight: Redundant computations are main bottleneck for wireless simulation
- Staging improves speed & scale by eliminating redundant computation
- No loss in accuracy
- Applicable to a variety of simulation engines
- $O(n^2)$ to $O(n)$ speedup for ns-2

http://www.cs.cornell.edu/People/egs/sns/

Staged Simulation for Improving the Scale and Performance of Wireless Network Simulations. Kevin Walsh and Emin Gün Sirer. In *Procedings of the Winter Simulation Conference*. December 2003.

Staged Simulation: A General Technique for Improving Simulation Scale and Performance. Kevin Walsh and Emin Gün Sirer. In *ACM Transactions on Modeling and Computer Simulation (TOMACS)*. April 2004.

Grid Performance



Sensitivity to Parameters

- Initial application of staging (grid):
  - Very sensitive to granularity (both in memory and CPU)
  - Poorly tuned grid much worse than baseline
- Successive applications of staging:
  - Reduce sensitivity to granularity
  - Less sensitive to other parameters
  - 10% variation in execution time over wide range of parameters

Memory Use and Performance

- Ns2 severely memory constrained
  - Artifact of simulator implementation
- Grid-based staging adds:
  - Typical: 1-10 KB, many new events
  - Poorly-tuned: 1-100+ MB, many new events
- Other intra-simulation staging adds:
  - Typical: 20-200 KB, few to no new events
  - Uses grid, but avoids worst-case scenarios
- Inter-simulation staging:
  - Eliminates grid entirely