Synthetic Traffic Generation: a Tool for Dynamic Interconnect Evaluation

W. Heirman, J. Dambre, J. Van Campenhout
ELIS Department, Ghent University, Belgium

Sponsored by IAP-V PHOTON & IAP-VI photonics@be,
Belgian Science Policy Office
Outline

• Introduction
• Synthetic traffic generation
• Results
• Conclusions
Distributed shared-memory architecture

Network is part of the memory hierarchy

- Instruction: 0.5 ns
- Cache: 5 ns
- DDR: 50 ns
- Network: 500 ns

Supercomputer

Server

UltraSPARC-Core
Interconnect requirements

Non-uniform network traffic in space and time

=⇒ Reconfigurable network?
Reconfiguration implementation: base network + extra reconfigurable links

other ‘dynamic networks’: e.g. per-link voltage scaling
Evaluate networks with synthetic traffic

- Mimics the behavior of real traffic
- But without the computational cost of modeling application, OS, CPUs, caches, ...

![Diagram showing network traffic simulation with synthetic traffic generator and reconfigurable network simulators.](diagram.png)
We need better synthetic traffic

Reconfiguration exploits low-frequency dynamics in the network traffic

• Trace-driven simulation using static traffic patterns (uniform, hotspot, shuffle, …) won’t do!
• Full execution-driven simulation (traffic is driven by application: FFT, weather forecast, database) is too slow!
Outline

• Introduction
• Synthetic traffic generation
• Results
• Conclusions
Realistic synthetic traffic generation

- One execution-driven simulation
- Resulting traffic profile re-used many times

Application
OS
CPU
Caches

Parameter extraction

Statistical traffic profile

Synthetic traffic generator

Reconfigurable network

network traffic

Application
OS
CPU
Caches

×1

×n
Preserve packet-interdependencies by using packet groups

Packets are processed/generated in groups, corresponding to one memory operation each.
Distribution of # involved nodes

![Graph showing the distribution of involved nodes with relative occurrence on the y-axis and number of nodes on the x-axis. The graph includes a bar chart and a small inset diagram illustrating connections between nodes labeled 2, 3, and n.]
Reuse distance of home nodes: introduce locality
Computation or ‘think’ time
Models time delay between subsequent requests
Outline

- Introduction
- Synthetic traffic generation
- Results
- Conclusions
Simulations

- Simulation platform: Simics, providing *functional* multiprocessor simulation
  - 16 UltraSPARC III processors
  - SPLASH-2 parallel benchmarks

- Timing model:
  - Computes the latency for each memory access
  - Models caches, interconnection network
  - Base network: 4x4 torus
  - Extra links: configurable number, fan-out, reconfiguration interval
Simulations

Once per benchmark:

Simulate execution of the benchmark, base network only, measuring traffic profile \(^{(1)}\)

For each set of extra link parameters:

- Execution-driven simulation with reconfigurable network \(^{(2)}\)
  \(\Rightarrow\) “correct” result

- Trace-driven simulation using (simplified) traffic from \(^{(2)}\)
  \(\Rightarrow\) *tracing error*

- Trace-driven simulation using (simplified) traffic from \(^{(1)}\)
  \(\Rightarrow\) *traffic-dependence on network*

- Trace-driven simulation using synthetic traffic
  \(\Rightarrow\) *total error*
Several parameters can be measured

- Execution-driven
- Trace (this network)
- Trace (base network)
- Synthetic

$n = 2$
$f = 2$
$\Delta t = 1\text{ ms}$

$n = 8$
$f = 2$
$\Delta t = 10\text{ ms}$

$E[\text{packet latency}]$
$E[\text{memop latency}]$
$E[\text{distance(packets*size)}]$
$P[\text{link-congestion>0}]$
$P[\text{distance(packets*size)>2}]$
$P[\text{packet-congestion>0}]$
$EA. cost$
Detailed view of “average packet latency”

![Bar chart showing average packet latency for different scenarios and time steps.](chart.png)
Variability for shorter traces

- Synthetic traffic
- Execution-driven

Packet latency:
- 270 to 265
- 260
- 255

CPU time:
- 10000 to 100
- 100
- 1

160k, 640k, 3M, 10M, 40M, 89M

- Trace-driven
- +profiling*
- Exec-driven

* assuming traffic profile is re-used 100 times
Outline

• Introduction
• Synthetic traffic generation
• Results
• Conclusions
Conclusions

- Synthetic traffic generation was extended to
  - shared-memory cache-coherence protocols,
  - reconfigurable networks
- Good relative accuracy for different network topologies
- Much less computationally expensive (x10), even more so for shorter traces (x100)
- Reproducibility equal to or better than execution-driven simulations