

Scalable Multi-Purpose Network Representation for Large Scale Distributed System Simulation

Laurent Bobelin¹, **Arnaud Legrand**¹,
David A. González Márquez² Pierre Navarro¹,
Martin Quinson³, Frédéric Suter⁴, Christophe Thiéry³

¹ LIG, Grenoble University, France

² Departamento de Computacion, Universidad de Buneos Aires, Argentina

³ LORIA, Nancy University, France

⁴ IN2P3 Computing Center, CNRS/IN2P3 Lyon-Villeurbanne, France

AGENCY FOR NATIONAL RESEARCH
ANR

ANR 08 SEGI 022
ANR 11 INFRA 13



CCGrid 2012

- LSDS** (clusters, P2P, grid, volunteer computing, clouds, ...) are a pain
- ▶ analytic methods quickly become intractable and often fail to capture key characteristics of real systems
 - ▶ experiments on the field are tedious, time-consuming, non-reproducible, sometimes even impossible

- LSDS** (clusters, P2P, grid, volunteer computing, clouds, ...) are a pain
- ▶ analytic methods quickly become intractable and often fail to capture key characteristics of real systems
 - ▶ experiments on the field are tedious, time-consuming, non-reproducible, sometimes even impossible

Hence, lots of research in our area rely on **simulation**

- LSDS** (clusters, P2P, grid, volunteer computing, clouds, ...) are a pain
- ▶ analytic methods quickly become intractable and often fail to capture key characteristics of real systems
 - ▶ experiments on the field are tedious, time-consuming, non-reproducible, sometimes even impossible

Hence, lots of research in our area rely on **simulation**

LSDS simulation challenges

- ▶ **scalability** (both in terms of speed and memory)
- ▶ **accuracy**/validity/realism (a very context-dependent notion)
- ▶ **genericity**

- LSDS** (clusters, P2P, grid, volunteer computing, clouds, ...) are a pain
- ▶ analytic methods quickly become intractable and often fail to capture key characteristics of real systems
 - ▶ experiments on the field are tedious, time-consuming, non-reproducible, sometimes even impossible

Hence, lots of research in our area rely on **simulation**

LSDS simulation challenges

- ▶ **scalability** (both in terms of speed and memory)
- ▶ **accuracy**/validity/realism (a very context-dependent notion)
- ▶ **genericity**

Most works trade everything for scalability although. . .

Premature optimization is the root of all evil
– D.E.Knuth

Networking Protocol design requires accurate packet-level simulations

Networking Protocol design requires accurate packet-level simulations

Not everyone has such needs

Networking Protocol design requires accurate packet-level simulations

Not everyone has such needs

P2P DHT geographic diversity, jitter, churn

↪ no need for contention, only delay

Networking Protocol design requires accurate packet-level simulations

Not everyone has such needs

P2P DHT geographic diversity, jitter, churn

↪ no need for contention, only delay

P2P streaming network proximity, asymmetry, interference on the edge

↪ ignore the core

Networking Protocol design requires accurate packet-level simulations

Not everyone has such needs

P2P DHT geographic diversity, jitter, churn

↪ no need for contention, only delay

P2P streaming network proximity, asymmetry, interference on the edge

↪ ignore the core

Grid heterogeneity, complex topology, contention w. large transfers

↪ no need to focus on packets

Validity: Community Requirements

Networking Protocol design requires accurate packet-level simulations

Not everyone has such needs

P2P DHT geographic diversity, jitter, churn

↪ no need for contention, only delay

P2P streaming network proximity, asymmetry, interference on the edge

↪ ignore the core

Grid heterogeneity, complex topology, contention w. large transfers

↪ no need to focus on packets

Volunteer Computing dynamic availability, heterogeneity

↪ little need for networking

HPC complex communication workload, protocol peculiarities

↪ build on regularity and homogeneity

Cloud mixture of previous requirements

Consequence: most simulators are **ad hoc and domain-specific**

read “dead within a year or so”



Packet-level simulation Networking community has standards, many popular open-source projects (NS, GTneTS, OmNet++,...)

- ▶ full simulation of the whole protocol stack
- ▶ complex models \rightsquigarrow hard to instantiate
- ▶ inherently **slow**

Packet-level simulation Networking community has standards, many popular open-source projects (NS, GTneTS, OmNet++,...)

- ▶ full simulation of the whole protocol stack
- ▶ complex models \rightsquigarrow hard to instantiate
- ▶ inherently **slow**

Delay-based models The simplest ones...

- ▶ communication time = constant delay, statistical distribution, LogP
 $\rightsquigarrow(\Theta(1)$ footprint and $O(1)$ computation)
- ▶ coordinate based systems to account for geographic proximity
 $\rightsquigarrow(\Theta(N)$ footprint and $O(1)$ computation)

Although very scalable, these models ignore network congestion and typically assume large bisection bandwidth

Flow-level models A communication (flow) is simulated as a single entity:

$$T_{i,j}(S) = L_{i,j} + S/B_{i,j}, \text{ where } \begin{cases} S & \text{message size} \\ L_{i,j} & \text{latency between } i \text{ and } j \\ B_{i,j} & \text{bandwidth between } i \text{ and } j \end{cases}$$

Estimating $B_{i,j}$ requires to account for interactions with other flows

Flow-level models A communication (flow) is simulated as a single entity:

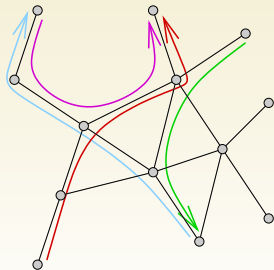
$$T_{i,j}(S) = L_{i,j} + S/B_{i,j}, \text{ where } \begin{cases} S & \text{message size} \\ L_{i,j} & \text{latency between } i \text{ and } j \\ B_{i,j} & \text{bandwidth between } i \text{ and } j \end{cases}$$

Estimating $B_{i,j}$ requires to account for interactions with other flows

Assume steady-state and **share bandwidth** every time a new flow appears or disappears

Setting a set of flows \mathcal{F} and a set of links \mathcal{L}

Constraints For all link j : $\sum_{\text{if flow } i \text{ uses link } j} \rho_i \leq C_j$



Network Communication Models (cont'd)

Flow-level models A communication (flow) is simulated as a single entity:

$$T_{i,j}(S) = L_{i,j} + S/B_{i,j}, \text{ where } \begin{cases} S & \text{message size} \\ L_{i,j} & \text{latency between } i \text{ and } j \\ B_{i,j} & \text{bandwidth between } i \text{ and } j \end{cases}$$

Estimating $B_{i,j}$ requires to account for interactions with other flows

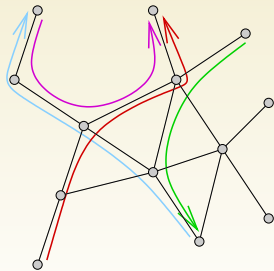
Assume steady-state and **share bandwidth** every time a new flow appears or disappears

Setting a set of flows \mathcal{F} and a set of links \mathcal{L}

Constraints For all link j : $\sum_{\text{if flow } i \text{ uses link } j} \rho_i \leq C_j$

Objective function

- ▶ Max-Min $\max(\min(\rho_i))$
- ▶ or other fancy objectives
e.g., Reno $\sim \max(\sum \log(\rho_i))$



Such **fluid models can account** for TCP key characteristics

- ▶ slow-start
- ▶ flow-control limitation
- ▶ RTT-unfairness
- ▶ cross traffic interference

They are a very reasonable approximation for most LSDC systems

Yet, many people think they are too complex to scale.

Let's prove them wrong! 😊

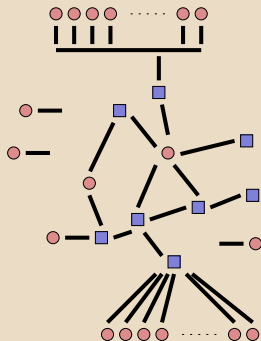
How to achieve scalability

Platform description

Main issues with topology

- ▶ description size, expressiveness
- ▶ memory footprint
- ▶ computation time

N nodes and E links



Representation

Input

Footprint

Parsing

Lookup

How to achieve scalability

Platform description

Main issues with topology

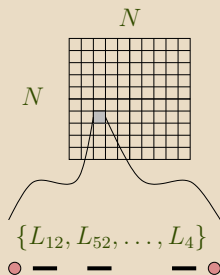
- ▶ description size, expressiveness
- ▶ memory footprint
- ▶ computation time

Classical network representation

1 Flat representation

5000 hosts doesn't fit in 4Gb!

N nodes and E links



Representation	Input	Footprint	Parsing	Lookup
Flat	N^2	N^2	N^2	1

How to achieve scalability

Platform description

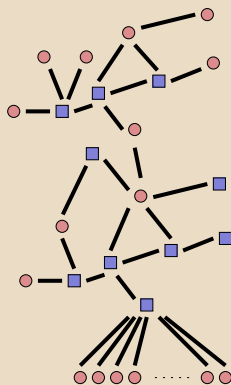
Main issues with topology

- ▶ description size, expressiveness
- ▶ memory footprint
- ▶ computation time

Classical network representation

- 1 Flat representation
5000 hosts doesn't fit in 4Gb!
- 2 Graph representation assuming shortest path routing

N nodes and E links



Representation	Input	Footprint	Parsing	Lookup
Dijkstra	$N + E$	$E + N \log N$	$N + E$	$E + N \log N$
Floyd	$N + E$	N^2	N^3	1

How to achieve scalability

Platform description

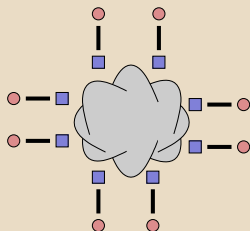
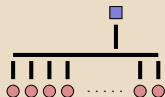
Main issues with topology

- ▶ description size, expressiveness
- ▶ memory footprint
- ▶ computation time

Classical network representation

- 1 Flat representation
5000 hosts doesn't fit in 4Gb!
- 2 Graph representation assuming shortest path routing
- 3 Special class of structures (star, cloud, ...)

N nodes and E links



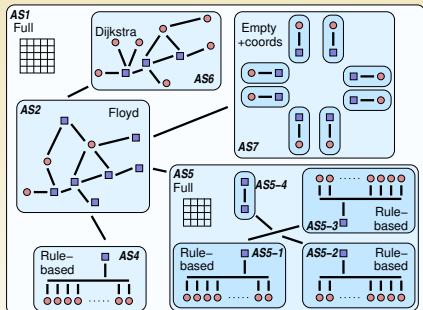
Representation	Input	Footprint	Parsing	Lookup
Star	1	N	N	1
Cloud	N	N	N	1

Our proposal

Every such representation has drawbacks and advantages

Let's build on the fact that *most* networks are *mostly* hierarchical

- 1 Hierarchical organization in AS
 - ~> cuts down complexity
 - ~> recursive routing
- 2 Efficient representation of classical structures
- 3 Allow bypass at any level



This approach has been integrated into the open-source SIMGRID simulation toolkit

Size of platform description file

Community	Scenario	Size
P2P	2,500 peers with Vivaldi coordinates	294KB
VC	5120 volunteers	435KB + 90MB
Grid	Grid5000: 10 sites, 40 clusters, 1500 nodes	22KB
HPC	1 cluster of 262144 nodes	5KB
HPC	Hierarchy of 4096 clusters of 64 nodes	27MB
Cloud	3 small data centers + Vivaldi	10KB

Size of platform description file

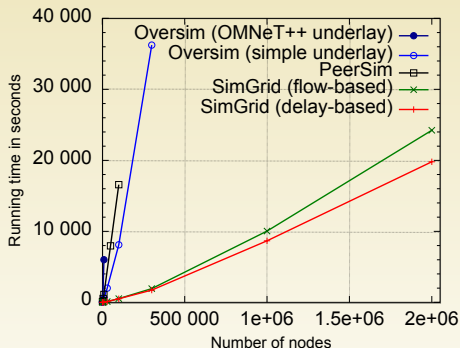
Community	Scenario	Size
P2P	2,500 peers with Vivaldi coordinates	294KB
VC	5120 volunteers	435KB + 90MB
Grid	Grid5000: 10 sites, 40 clusters, 1500 nodes	22KB
HPC	1 cluster of 262144 nodes	5KB
HPC	Hierarchy of 4096 clusters of 64 nodes	27MB
Cloud	3 small data centers + Vivaldi	10KB

Grid Scenario a master distributes 500,000 fixed size jobs to 2,000 workers in a round-robin way

	GRIDSIM	SIMGRID
Network model	delay-based model	flow model
Topology	none	Grid5000
Time	1h	14s
Memory	4.4GB	165MB*

★ 5.2Mb are used to represent the Grid 5000. Stack size not optimized (80KB/worker)

- ▶ Scenario: Initialize Chord, and simulate 1000 seconds of protocol
- ▶ Arbitrary Time Limit: 12 hours (kill simulation afterward)

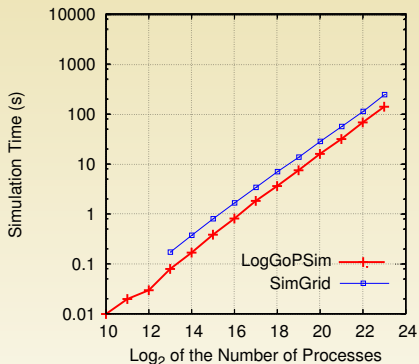


Largest simulated scenario

Simulator	size	time
OverSim (OMNeT++)	10k	1h40
OverSim (simple)	300k	10h
PeerSim	100k	4h36
SG (flow-based)	10k	130s
	300k	32mn
	2M*	6h23
SG (delay-based)	2M	5h30

* 36GB = 18kB/ process (16kB for the stack)

- ▶ SIMGRID is orders of magnitude more scalable than state-of-the-art P2P simulators
- ▶ Using the precise flow-based model incurs a limited ($\approx 20\%$) slow-down, while simulation accuracy is improved



Simulating a binomial broadcast:

- ▶ SIMGRID is roughly 75% slower than LOGGOPSIM
- ▶ SIMGRID is at least 20% more fat than LOGGOPSIM (15GB required for 2^{23} processors)

The genericity of SIMGRID data structures comes at the cost of a slight overhead

This demonstrates that scalability does not necessarily come at the price of realism (e.g., ignoring contention on the interconnect)

Take away message

- ▶ The widespread belief that “scalable simulations require to oversimplify the network models and avoid the use of threads” is erroneous
- ▶ SIMGRID is open-source, mature, and does not trade accuracy and meaning for scalability \leadsto use it instead of rewriting ad hoc simulators

<http://simgrid.gforge.inria.fr>



Future plan

- ➊ Further reduce platform description size (hence parsing time) and memory footprint by exploiting stochastic regularity and improving programmable description approach
- ➋ Consider the specifics of emerging computing systems such as clouds or exascale platforms:

<http://infra-songs.gforge.inria.fr/>