A Toolchain for Symplifying Network Simulation Setup

Lorenzo Saino, Cosmin Cocora and George Pavlou

Communications and Information Systems Group
Department of Electrical and Electronics Engineering
University College London
{l.saino,c.cocora,g.pavlou}@ee.ucl.ac.uk

http://fnss.github.com

Outline

Fast Network Simulation Setup (FNSS) toolchain

- Background and motivations
- Workflow
- Functional features

Modelling link capacity assignments

- Problem statement
- Proposed solution
- Evaluation

Summary and conclusions

Background and motivation

Setting up a network simulation is a cumbersome and error-prone task. This requires to:

- select a suitable topology,
- configure it with link capacities, weights, delays, buffer sizes, protocol stacks and applications,
- generate a traffic matrix or configure traffic sources,
- deploy all this in the target simulator.

No tools currently exist taking care of all these steps. All currently available tools only provide capabilities to parse or generate unconfigured topologies or are bound to a specific simulator.

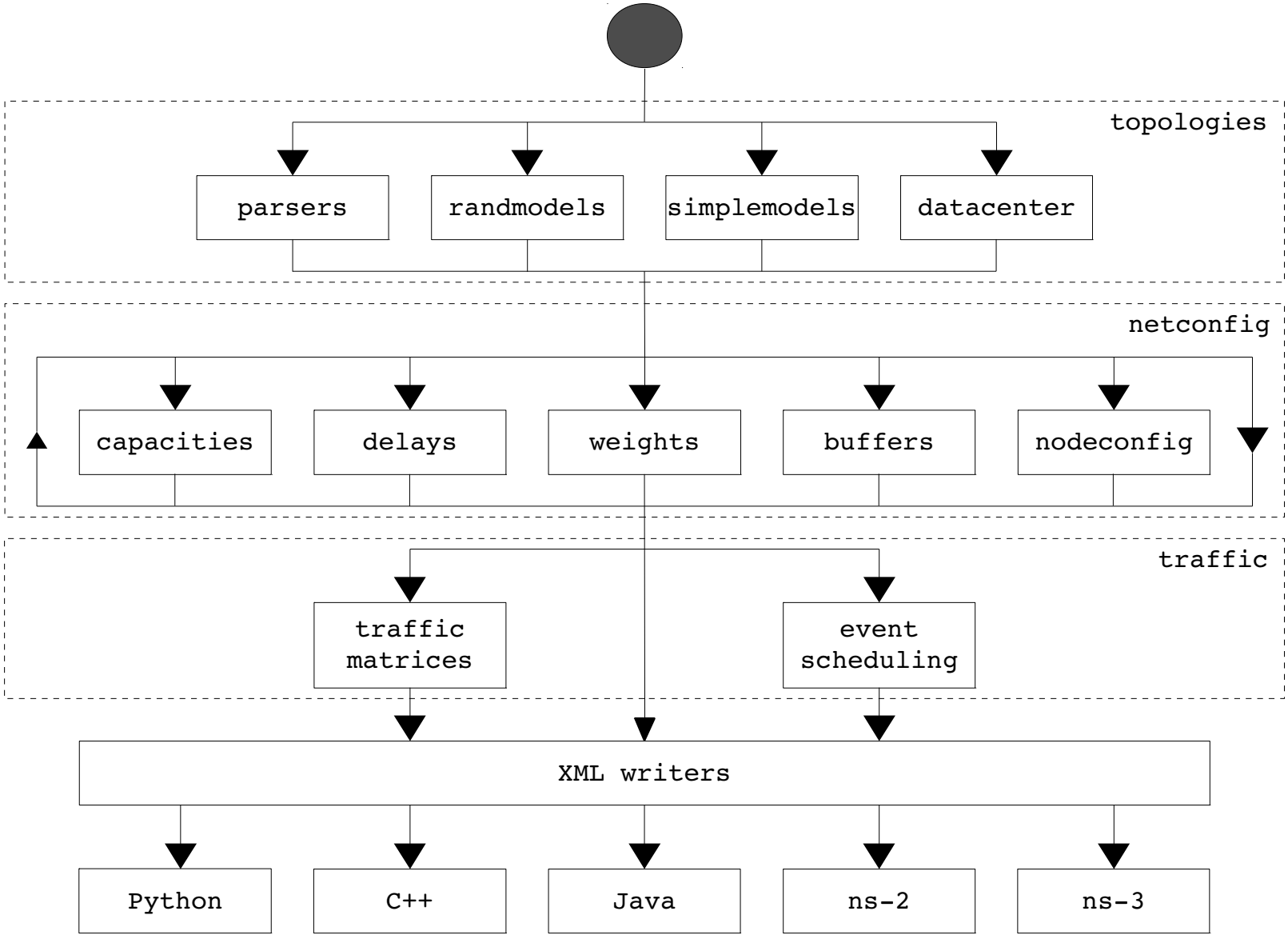
Introducing FNSS

Fast Network Simulation Setup (FNSS) is a toolchain allowing network researchers and engineers to easily set up a network simulation scenario (topology, traffic matrix, event schedule) and deploy it in the preferred target simulator.

FNSS is made of a core library and a number of APIs and adapters:

- The core library, written in Python, generates simulation scenarios and export them to XML files.
- Adpaters and APIs can be used to import a scenario in the target simulator:
  - APIs: Java, C++, Python
  - Adapters: ns-2, ns-3

Workflow


topologies
parsers
randmodels
simplemodels
datacenter
netconfig
capacities
delays
weights
buffers
nodeconfig
traffic
traffic matrices
event scheduling
XML writers
Python
C++
Java
ns-2
ns-3

Topology creation

Import:

- **from other generators:** BRITE, INET, aSHIIP
- **from datasets:** RocketFuel, CAIDA AS relationships, Topology Zoo, Abilene

Generate:

- **random topologies:** Barabási-Albert, extended Barabási-Albert, Erdős-Rényi, Waxman, Generalized Linear Preference (GLP).
- **datacenter topologies:** two-tier, three-tier, fat tree, B-cube
- **simple topologies:** star, ring, line, dumbbell, tree, mesh

Topology configuration

- **Link capacities assignment:** constant, manual, random (uniform, power-law, Zipf-Mandlebrot, user-defined pdf), new centrality-based models (we'll talk about them later)
- **Link delays assignment:** constant, manual, proportional to link length
- **Link weight assignment:** constant, manual, proportional to link delay, proportional to inverse of capacity
- **Buffer sizes assignment:** constant, manual, bandwidth-delay product, proportional to link capacity
- **Protocol stack and applications**: each node can be assigned one stacks and several applications

Event scheduling

FNSS can produce event schedules and export them to an XML file. An event schedule is a list of events labelled with an execution time. An event is modelled as a dictionary of key-value attributes.

For example, an HTTP request event could be:

```
event = {
    'client_ip':    '192.168.1.24'
    'proxy':        '192.168.1.100:8080'
    'method':       'GET'
    'url':          'http://www.ucl.ac.uk/'
    'User-Agent':   'fnss-client'
    'Connection':   'keep-alive'
}
```

The target simulator must be able to interpret the meaning of the event attributes.

Traffic Matrix

FNSS can synthetically generate traffic matrices according to the *Ranking Metrics Heuristic* method.[1]

It can generate the following traffic matrices:

- **Static:** traffic volumes at single point in time
- **Stationary:** series of traffic volumes suitable for modelling network traffic variations over short timescales (up to 1-1.5 hours)
- **Cyclostationary:** series of traffic volumes suitable for modelling diurnal patterns in network traffic

[1] A. Nucci, A. Sridharan, and N. Taft. The problem of synthetically generating ip traffic matrices: initial recommendations. ACM SIGCOMM Computer Communication Review, 35(3):1932, 2005.

Deployment in target simulator

Supported target simulators are integrated using different strategies:

- **Java, C++ or Python API**: Parse topology, traffic matrix and event schedule XML files and convert them to objects of the target language
- **ns-2 adapter**: Python script which parses topology XML file and converts it to a Tcl script
- **ns-3 adapter**: ns-3 C++ module capable of parsing topology and event schedule XML files, deploy the topology, schedules all the events and start the simulation
- **Omnet++ adapter**: Python script which parses topology XML file and converts it to a NED script

Modelling link capacity assignments

Background and problem statement

Background

- Inferred or synthetically generated network topologies do not include link capacities.
- Little research effort has focused on inferring and modelling link capacity assignments in an ISP network.
- Common practice in network simulations is to randomly assign link capacities following various distributions and analyse results sensitivity.

Problem definition: Let $G(E, V)$ be a graph with vertices $V = \{v_1, v_2, ...v_n\}$ and edges $E = \{e_1, e_2, ..., e_m\}$ representing the topology of an ISP backbone network whose edges have capacities belonging to the set $C = \{c_1, c_2, ..., c_p\}$, with $|C| \ll |E|$, assigned according to $f_c : E \rightarrow C$. The objective is to find the most realistic estimate $\hat{f}_c$ of f_c, assuming that no additional information about the network apart from G and C is available.

Modelling link capacity assignments

Proposed solution

Our argument is that a key factor influencing the link capacity assignment is the *importance* of the link itself which also depends on the *importance* of the nodes that the link connects. *Important* links are more likely to have higher capacities.

How do we quantify the *importance* of links?

- Edge betweenness centrality
- Degree centrality gravity
- Communicability centrality gravity

Our solution is to assign values c of link capacity proportionally to the value m of the centrality metric considered:

$$c(u, v) \propto m(u, v) \qquad c(u, v) \in C$$

Modelling link capacity assignments

Evaluation methodology

Datasets: we evaluated the performance of our method on five networks with known topology and link capacity assignments:

- GEANT: European academic network
- GARR: Italian academic network
- WIDE: Japanese academic network
- RedIris: Spanish academic network
- Uninett: Norwegian academic network

Methodology:

- Assign link capacities according to our method
- Measure how different this assignment is from the real network.
- Random assignment (uniform) used as baseline. This is the most commonly used method in literature.

Modelling link capacity assignments

Performance metrics

Matching Capacity Ratio (MCR): the fraction of links whose capacity assignment matches the real capacity

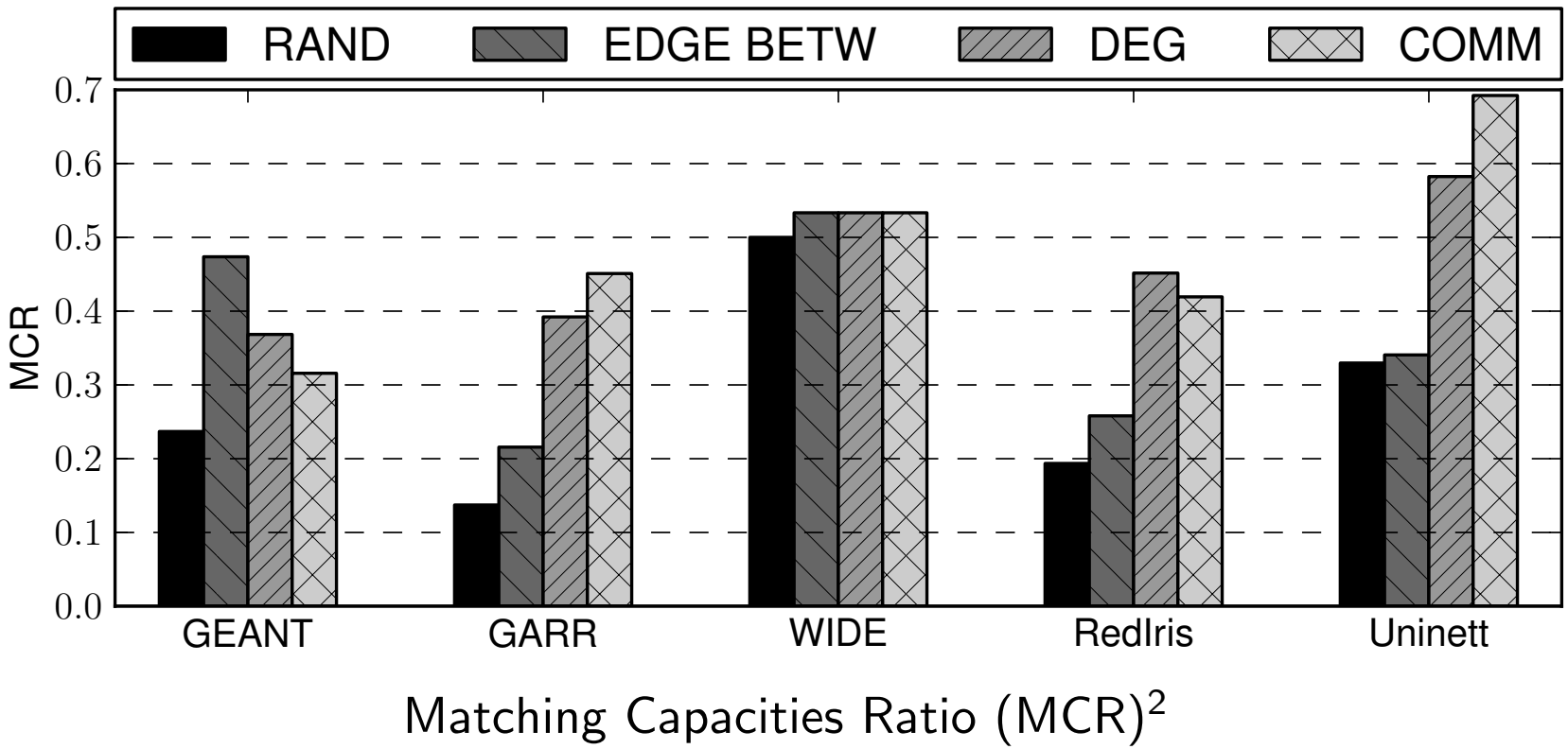
$$MCR = \frac{|\{e \in E | \hat{f}(e) = f(e)\}|}{|E|}$$

Capacity Rank Error (CRE): RMSE between the rank in C of real and inferred capacity normalized by $|C|$

$$CRE = \frac{1}{|C|} \times \sqrt{\frac{1}{|E|} \sum_{e \in E} \left[R(\hat{f}_e) - R(f_e) \right]^2}$$

Modelling link capacity assignments

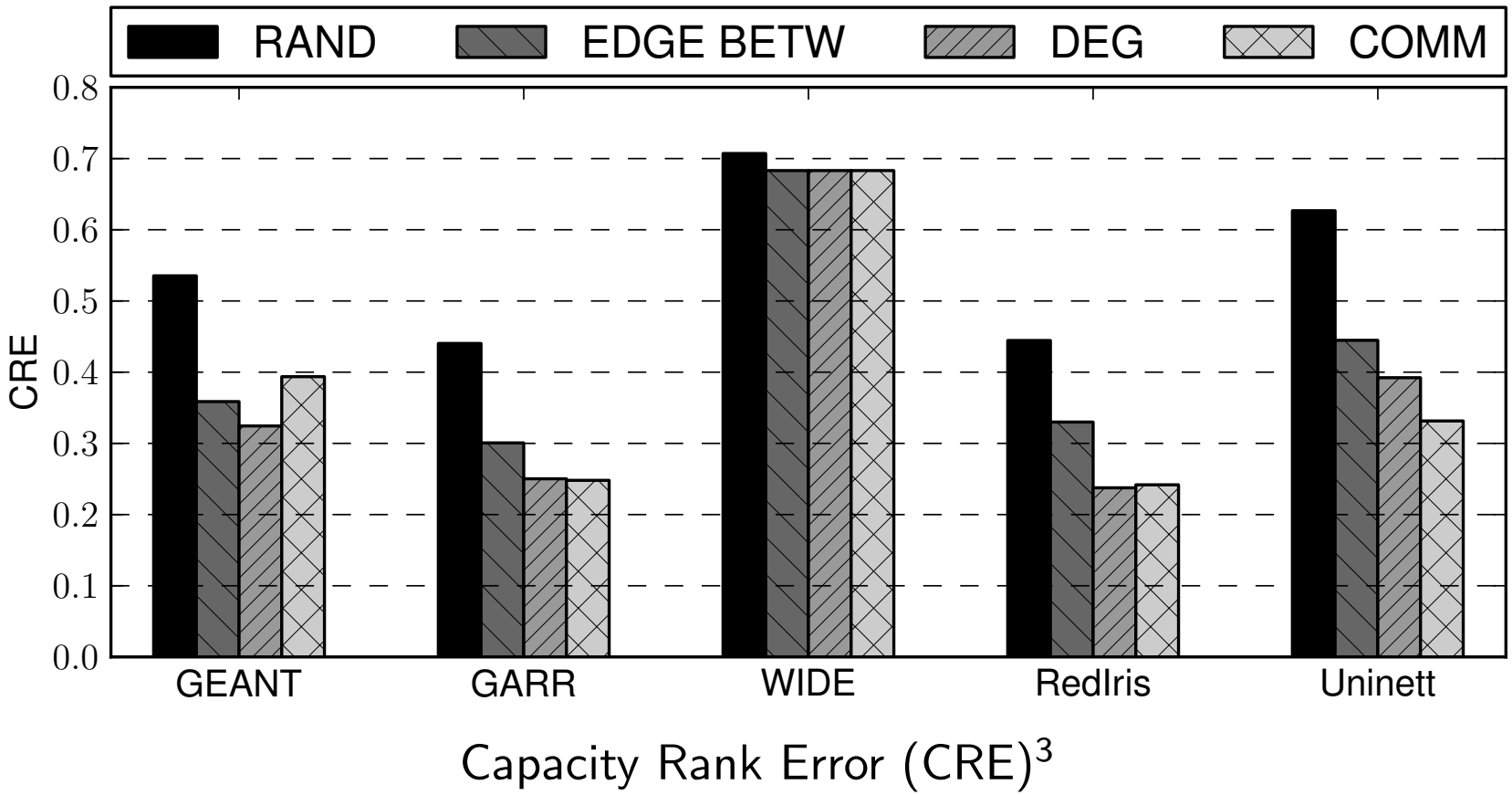
Evaluation Results



[2]RAND refers to median MCR value calculated analytically

Modelling link capacity assignments

Evaluation Results



[3]RAND refers to median CRE value calculated analytically

Summary and Conclusions

With this work, we provided two main contributions: the FNSS toolchain and a novel model of link capacity assignment in an ISP backbone network.

FNSS toolchain:

- Supports Python, C++, Java, ns-2 and ns-3. Omnet++ support planned for future release
- Open-sourced, documented, ready to use

Centrality-based models of link capacity assignment:

- Simple model but performance gain is consistent
- Further evaluation and model extension ongoing

http://fnss.github.com