Statistical approach to networks-on-chip

Statistical Approach to Networks-on-Chip

Itamar Cohen, Ori Rottenstreich, and Isaac Keslassy, *Member, IEEE.*

Abstract—Chip multiprocessors (CMPs) combine increasingly many general-purpose processor cores on a single chip. These cores run several tasks with unpredictable communication needs, resulting in uncertain and often-changing traffic patterns. This unpredictability leads network-on-chip (NoC) designers to plan for the worst-case traffic patterns, and significantly over-provision link capacities. In this paper, we provide NoC designers with an alternative statistical approach. We first present the traffic-load distribution plots (T-Plots), illustrating how much capacity over-provisioning is needed to service 90%, 99%, or 100% of all traffic patterns. We prove that in the general case, plotting T-Plots is #P-complete, and therefore extremely complex. We then show how to determine the exact mean and variance of the traffic load on any edge, and use these to provide Gaussian-based models for the T-Plots, as well as guaranteed performance bounds. We also explain how to practically approximate T-Plots using random-walk-based methods. Finally, we use T-Plots to reduce the network power consumption by providing an efficient capacity allocation algorithm with predictable performance guarantees.

Index Terms—Networks-on-Chip, Chip Multiprocessors, Capacity Allocation, Traffic Load Distribution Plot.

1 Introduction

The multi-core era is here. Today, chip multiprocessors (CMPs) combine increasingly many general-purpose processor cores on a single chip [1], [2], [3], [4], [5], [6], [7]. As shown in Fig. 1, these processor cores can be placed in regular and identical tiles, interconnected in a network-on-chip (NoC) using links and switches. Such a regular network-based design enables a lower design complexity, scalable and predictable layout properties, a high level of parallelism and modularity, and an efficient statistical capacity sharing [8], [9], [10], [11], [12], [13].

The processor cores in CMPs run many software processes belonging to a wide variety of possible applications, with unpredictable communication needs between the cores. As a result, the traffic pattern in the NoC is uncertain and often-changing. The challenge is to allocate NoC link bandwidth capacities efficiently so as to service the many possible traffic patterns, and at the same time not to use excessive link area and power — especially given that the NoC architecture consumes a significant portion of CMP resources [1], [2], [14], [15]. Note that this capacity allocation problem is different from traditional application-specific systems-on-chip (ASSoCs), in which a limited set of applications is statically mapped onto the cores and generates a well-known traffic pattern [16], [17], [18], [19]. It also differs from traditional chip-to-chip multiprocessor interconnects, in which the link bandwidth capacities cannot be easily adjusted [20].

The link bandwidth capacity allocation algorithm needs to trade off the different bandwidth requirements of the many traffic patterns, and the possible capacity over-provisioning. On the one hand, the usual method of sizing the network for some typical *average* traffic pattern [13], [21], such as a uniform traffic pattern, can completely miss the widely different bandwidth demands of the other traffic patterns, which appear when running different applications. On the other hand, planning for the *worst case* among all possible traffic patterns [3], [22], [23], [24] can potentially necessitate significant link bandwidth capacities that are rarely fully utilized and consume expensive power resources. In fact, such a scheme does not fully exploit the statistical multiplexing properties of the NoC, which are increasingly significant as the number of cores increases.

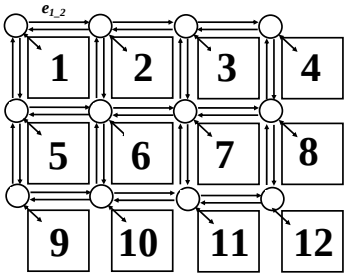


Fig. 1. 3×4 NoC-based CMP architecture.

The main contribution of this paper is the introduction of a *statistical* approach to NoC design and capacity allocation. To do so, we introduce a novel method to represent and analyze the full spectrum that lies between the *average* and the *worst-case* traffic patterns. Then, we argue that the NoC designer should consider the trade-off between link capacity and performance guarantee, where the performance is measured by the fraction of traffic patterns that can be fully served. For instance, the NoC designer should know that instead of some worst-case capacity allocation in which the NoC can guarantee service to 100% of traffic patterns, some other statistical

- *Itamar Cohen is with the Jerusalem College of Engineering, Israel. This work was done as he was with the department of Electrical Engineering, Technion, Haifa 32000, Israel.*
  E-mail: itamarco@post.jce.ac.il.
- *Ori Rottenstreich and Isaac Keslassy are with the department of Electrical Engineering, Technion, Haifa 32000, Israel.*
  E-mail: ori.rot@gmail.com, isaac@ee.technion.ac.il.

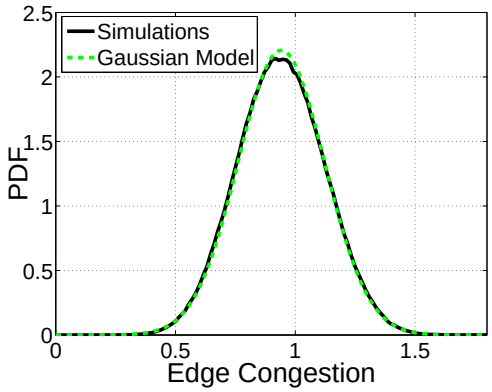


Fig. 2. PDF T-Plot of the edge congestion on e_{6_7}.

capacity allocation can guarantee service to 99.99% of traffic patterns in exchange for a reduction in the total link capacity. It is then up to the NoC designer to determine whether the 0.01% of traffic patterns are worth this additional capacity and the necessary additional power resources; or whether they can potentially be dropped, for instance if they belong to a best-effort quality-of-service traffic class [13], [25].

To support our statistical approach, we introduce the *T-Plots*, or Traffic Load Distribution Plots — a class of plots illustrating the distribution of the load generated by the set of traffic patterns, and providing a synthetic view of the network performance. For instance, Fig. 2 illustrates such a T-Plot, showing the probability density function (PDF) of the normalized load on edge e_{6_7}. The T-Plot is generated using the set of all the possible traffic patterns in the CMP, and measuring the distribution of the load caused by each of these traffic patterns on edge e_{6_7}. The graph is of course normalized so that the area below it sums up to 1 (other simulation details are in Section 9).

As shown in the T-Plot, the average load generated on this link is 0.94. Further, the worst-case load can be found to be exactly 2 (with a negligible density), and the 99.99%-cutoff load is a bit below 1.59. In other words, this T-Plot shows that when the link capacity equals 21% less than the worst-case load, 99.99% of the traffic patterns can already be serviced. Thus, using this T-Plot, a NoC designer can directly evaluate the performance of a capacity allocation scheme, and clearly see the tradeoff between performance guarantee and capacity over-provisioning. The NoC designer might decide, for instance, that the marginal benefit of allocating more capacity beyond 1.59 is not worth the cost. Incidentally, note that this T-Plot can be closely modeled as Gaussian. In fact, it passes both the Pearson's χ^2 and the Cramér-von Mises normality tests with a standard significance level of 5%. The paper will later expand on this point.

In this paper, we demonstrate that the exact computation of T-Plots is #P-complete, and therefore without known polynomial-time algorithms. We later show how to practically approximate T-Plots using random-walk-based methods, and how to analytically calculate the mean and the variance of the edge loads in a combinatorial way. We further show that knowing the mean and the variance is often sufficient to approximate the whole distribution, as the edge T-Plots may be frequently closely modeled as Gaussian. We also suggest some simple bounds and models for the global T-Plot, which models the performance of the whole network. Finally, we suggest a very simple, yet efficient, capacity allocation algorithm with predictable performance guarantees.

We would like to emphasize that this paper provides *fundamental* results on the maximum available capacity on any link [23]. The available capacity might become lower following protocol-specific constraints, like control overhead and packet retransmissions, and topology-specific constraints, like limited buffer sizes and lossy links [26]. These are beyond the scope of this paper.

In addition, the paper relies on several simplifying assumptions. Routing is assumed to be fixed (oblivious) and given, thus excluding any adaptive routing with traffic-dependent load-balancing. The set of traffic patterns is assumed to be given as well. For instance, it could be the set of all theoretically-possible traffic patterns, or a set of traffic patterns sampled in a large number of traces, or a set of possible traffic bursts. The analysis in this paper allows for any such set, but neglects the temporary effects of traffic pattern transitions, which are beyond the scope of this paper.

In our view, a key aspect of the statistical approach is its potential for a wide range of applications, including in non-CMP NoC-based architectures. For instance, while not applicable to simple ASSoCs with a single traffic pattern, the statistical approach can be highly useful for more complex ASSoCs with dozens of basic use-cases and potentially thousands of compound use-cases [27], [28]. Likewise, the statistical approach can be used in NoC-based FPGAs to allocate the available bandwidth capacity of the higher-performance hard-wired non-programmable links, thus providing the designer with performance guarantees for a significantly large number of traffic patterns [29]. Finally, the statistical approach can be combined with other approaches to provide quality-of-service (QoS) guarantees, for instance by using worst-case analysis for high-priority control and delay-sensitive traffic, and statistical analysis for the remaining best-effort traffic [13], [25]. This is because the statistical approach presented in this paper deals with the fundamental link capacity needs, and not protocol-specific needs.

This work is structured as follows. After formulating the T-Plot model in Section 2, we prove its #P-completeness in Section 3. Then, in Sections 4 and 5, we provide a Gaussian view of the edge T-Plots as well as strict performance guarantees, and generalize these results to global T-Plots in Section 6. Finally, in Sections 7 and 8, we introduce a simple capacity allocation scheme, which we evaluate, together with the other results, in Section 9.

2 T-PLOT MODEL

Network — The NoC architecture is modeled as a directed graph $G(V, E)$ with n=$|V|$ nodes (processor cores) and $|E|$ edges (links). For instance, Fig. 1 illustrates such a graph with 12 nodes (each corresponding to a processor core and its associated switch), and 34 edges between them.

Let $D = [D_{ij}]_{1 \leq i,j \leq n}$ denote the traffic matrix in the NoC. For each i, j, D_{ij} represents the traffic rate from source i to destination j. In this paper, we consider a normalized homogeneous CMP in which each processor core works at the same frequency and can send (receive) at most one data word every clock cycle, but we do not make any assumption on the destination (source) of its traffic (see [3], [22], [23], [24] for more details on this standard model). Therefore, the possible set $\mathcal{A}$ of traffic matrices in the NoC, called the *T-Set* $\mathcal{A}$, is defined as

$$\mathcal{A} = \left\{ D \in \mathbb{R}_+^{n \times n} | \forall i : \sum_j D_{ij} \leq 1, \sum_j D_{ji} \leq 1 \right\} \quad (1)$$

For example, assuming a data width of 32 bits and a frequency of 200 MHz, we get a maximum input/output rate of $32 * 200/8 = 800$ MByte/s for each processor core of the network, and the T-Set $\mathcal{A}$ is the set of all the possible traffic matrices that respect this maximum.

We can generalize the results to different T-Sets. For instance, in a general heterogeneous NoC architecture, node *i* may send (receive) traffic at any rate up to q_i (r_i), and we will consider the T-Set $\mathcal{H}$ defined as

$$\mathcal{H} = \left\{ D \in \mathbb{R}_+^{n \times n} | \forall i : \sum_j D_{ij} \leq q_i, \sum_j D_{ji} \leq r_i \right\} \quad (2)$$

Likewise, we will consider the set $\mathcal{P}$ of permutation traffic matrices, in which each processor core transmits (receives) at maximum rate to (from) a unique processor core.

Given a T-Set, we will usually assume below that any traffic matrix in a T-Set is always equally likely. However, we will also briefly mention *weighted T-Sets* in which specific subsets are more likely, for instance by giving more weight to traffic matrices that carry more traffic.

We will also assume that each edge *e* is allocated a positive capacity $c(e) > 0$. An edge e is a *strictly minimal edge* if $c(e') > c(e)$ for each edge *e'* different from e, and a *bridge* if removing e would increase the number of components in the graph.

Routing — A *routing* is classically defined as a set of $(n^2|E|)$ variables $\{f_{ij}(e)\}$, where $f_{ij}(e)$ denotes the fraction of the traffic from node i to node j that is routed through edge e. In other words, the total *flow* crossing e when routing the traffic matrix D is $\sum_{i,j} D_{ij} f_{ij}(e)$, where D_{ij} is the $(i,j)^{\text{th}}$ element of matrix D.

Such a routing is oblivious in the sense that the routing variables are independent of the current traffic matrix. The routing is assumed to satisfy the classical linear flow conservation constraints [30]. An example of routing scheme is dimension-ordered routing (DOR) [31], also called XY routing, a simple NoC mesh routing algorithm in which packets are routed along one dimension first and then along the next dimension (we assumed an "X then Y" routing). Further, when the T-Set is $\mathcal{A}$, the *most loaded edge* is the edge e that maximizes $\sum_{ij} f_{ij}(e)$.

Congestion — The *edge congestion* (or load) on edge e is equal to the total flow crossing it divided by the edge capacity, i.e.

$$EC(e, f, D) = \frac{\sum_{i,j} D_{ij} f_{ij}(e)}{c(e)} \quad (3)$$

When the edge congestion on e is at least 1, i.e. when the routing algorithm would like to send more traffic on edge e than the edge capacity would strictly allow, we will say that e is *saturated*. Further, a network is saturated if at least one edge in it is saturated. The *global congestion* of routing D using f will be obtained by maximizing the edge congestion over all the edges, that is:

$$GC(f, D) = \max_{e \in E} \{EC(e, f, D)\} \quad (4)$$

For a saturated network, the *throughput* is defined as the inverse of the global congestion, and is otherwise made not to exceed 100%:

$$TP(f, D) = \min\{GC(f, D)^{-1}, 1\} \quad (5)$$

T-Plot — *Edge* (*global*) *T-Plots* show the distribution of the edge (global) congestion generated by traffic matrices in the T-Set. T-Plots can be represented as plots of the cumulative distribution function (CDF) or the probability density function (PDF). For example, the value of the *edge T-plot CDF* at point *L* is the probability that the edge congestion imposed on that edge by a traffic matrix selected from the T-Set $\mathcal{T}$ would be at most *L*:

$$EC_{CDF}^{\mathcal{T}}(e, f, L) = \Pr\{EC(e, f, D) \leq L | D \in \mathcal{T}\} \quad (6)$$

3 T-PLOTS ARE #P-COMPLETE

We will now prove that computing the T-Plots is *#P-complete* [32], which implies that it cannot be done using any known polynomial-time algorithm.

Intuitively, #P-complete problems are hard *counting* problems without known polynomial-time solution, in the same way as NP-complete problems are hard *decision* problems without known polynomial-time solution. In fact, NP is a subset of #P, and therefore #P-complete problems are at least as hard as NP-complete problems: while a typical NP-complete problem is to decide whether there exists *at least one* solution, the related #P-complete problem is to *count* the number of solutions, which can make it quite harder.

We will first show the #P-completeness of computing *edge T-Plots*, and then the #P-completeness of computing *global T-Plots*. For ease of reading, the proofs are moved to the appendix, together with some standard definitions.

Theorem 1: When the T-Set is the set of permutations $\mathcal{P}$, finding the edge T-Plot of a non-bridge edge e is *#P-complete*.

Proof: See Appendix A. □

Corollary 1: In the general case, finding the edge T-Plot is *#P-complete*.

Proof: Since at least one sub-class of edge T-Plot problems is *#P-complete*, the general class of edge T-Plot problems is clearly *#P-complete* as well. □

We showed above that computing *edge* T-Plots is *#P-complete*. We will now show that computing *global* T-Plots is *#P-complete* as well.

Theorem 2: When the T-Set is the set of permutations $\mathcal{P}$, finding the global T-Plot of a graph that includes a strictly minimal edge is *#P-complete*.

Proof: See Appendix B. □

As above, we can directly derive the following corollary.

Corollary 2: In the general case, finding the global T-Plot is *#P-complete*.

Since exact T-Plot computation proves elusive, we can only try to approximate or bound it. This will be a recurring theme in this paper.

4 Exact mean and variance of edge T-Plots

We just proved that in the general case, computing edge T-Plots is *#P-complete*, and therefore extremely complex. Thus, we will strive to look for good approximations and bounds. We will now present a straightforward method to calculate the mean and variance of the edge congestions. This will enable us to obtain an overview of the network bottlenecks by running a simulation at most once during topology synthesis, instead of running repeatedly extended simulations each time we change the routing algorithm or the capacity allocation.

In the remainder, we will see that these values will be enough to provide both a Chebyshev-based deterministic bound (Section 5) and a Gaussian-based model (Section 7).

4.1 Finding the mean and variance of the load for the set of permutations

Let's illustrate the computation of the mean and variance of the edge congestion when the T-Set is the set of permutations $\mathcal{P}$. In this case, the average-case edge congestion on edge e using routing f is:

$$\begin{aligned} EC^{\mathcal{P}}_{ac}(e,f) &= \frac{1}{n!}\sum_{D\in\mathcal{P}} EC(e,f,D) \\ &= \frac{1}{n!}\sum_{D\in\mathcal{P}} \frac{\sum_{ij} D_{ij} f_{ij}(e)}{c(e)} \\ &= \frac{1}{n!c(e)}\sum_{ij} f_{ij}(e) \sum_{D\in\mathcal{P}} D_{ij} \\ &= \frac{1}{nc(e)}\sum_{ij} f_{ij}(e), \end{aligned} \tag{7}$$

The last equality relies on the fact that a given flow (i,j) is only used in $\frac{1}{n}^{\text{th}}$ of the permutations, where n is the number of nodes.

Likewise, the variance is calculated using the variance formula

$$Var_{D\in\mathcal{P}}[EC(e,f,D)] = E[EC^2] - E^2[EC], \tag{8}$$

with

$$E[EC^2] = \frac{\sum_{ijkl} f_{ij}(e) f_{kl}(e) \left(\sum_{D\in\mathcal{P}} D_{ij} D_{kl}\right)}{n! c(e)^2}, \tag{9}$$

where the expectations are with respect to the random variable $D \in \mathcal{P}$ and the parameters of EC are implicit. Using basic combinatorial considerations,

$$\sum_{D\in\mathcal{P}} D_{ij} D_{kl} = \begin{cases} (n-1)! & i=k \wedge j=l \\ (n-2)! & i\neq k \wedge j\neq l \\ 0 & i=k \wedge j\neq l \\ 0 & i\neq k \wedge j=l \end{cases} \tag{10}$$

Note that for a given topology and routing, these values are fixed, and are independent of the capacity allocation. They need to be computed only once per topology during topology synthesis.

4.2 Mean and variance of the load for continuous T-Sets

We will now use the same techniques to compute the mean and variance of the load for continuous T-Sets, such as $\mathcal{A}$.

First, given a general T-Set $\mathcal{T}$, the average-case edge congestion is the average congestion caused by a matrix D randomly selected in $\mathcal{T}$:

$$EC^{\mathcal{T}}_{ac}(e,f) = \frac{\int_{\mathcal{T}} EC(e,f,D)ds}{\int_{\mathcal{T}} 1 ds}, \tag{11}$$

with the simplified formula:

$$\begin{aligned} \int_{\mathcal{T}} EC(e,f,D)ds &= \frac{1}{c(e)} \int_{\mathcal{T}} \sum_{ij} \left[D_{ij} f_{ij}(e)\right] ds \\ &= \frac{1}{c(e)} \sum_{ij} f_{ij}(e) \int_{\mathcal{T}} D_{ij} ds \end{aligned} \tag{12}$$

It is possible to predetermine the value of $\int_T D_{ij} ds$ *at most once* during topology synthesis, thus eliminating the need to run extensive repeated simulations each time we change the routing algorithm or the capacity allocation. For instance, when the T-Set is $\mathcal{A}$, it can be done using Monte Carlo simulations with an arbitrarily small error, as explained in Chapter 9.

The variance of the edge congestion is calculated as in Equations (8) and (9), using:

$$E_{\mathcal{T}}[EC(e,f,D)^2] = \frac{\sum_{ijkl} f_{ij}(e) f_{kl}(e) \int_{\mathcal{T}} D_{ij} D_{kl} ds}{c(e)^2 \cdot \int_{\mathcal{T}} 1 ds} \tag{13}$$

By separating the mutual relations of *i,j,k,l* to the same 3 cases as in (10), it is possible to predetermine the value of $\int_T D_{ij} D_{kl} ds$ — again, at most once for each qualified network.

4.3 Mean and variance of the load for heterogeneous and weighted T-Sets

Our CMP model assumes that all processor cores behave in an identical way. Let's briefly describe how this model can be generalized to heterogeneous and weighted NoC architectures. Let D be a traffic matrix, either in an heterogeneous T-Set $\mathcal{H}$ as defined in Section 2, or in a weighted T-Set. The average and the variance of the edge congestion in D may be calculated by methods similar to those detailed above. Note that as the values of the entries in D are not identically distributed anymore, using Equation (11) requires calculating the integrals $\int_T D_{ij} ds$ separately for each pair (i,j) in the worst case, i.e. n^2 times. Using Equation (13) requires calculating up to n^4 integrals - one for each chosen tuple $(i,j,k,l) \in [1,n]^4$. Again, it is possible to calculate each of these integrals *only once* during topology synthesis, even when trying many different routing algorithms or capacity allocations, thus saving on the respective repeated simulations.

In the next section, we will show how the mean and variance of the edge congestions can provide us with deterministic congestion guarantees.

5 Congestion guarantees for edge T-Plots

We are interested in providing performance bounds that are guaranteed independently of the shape of the edge T-Plot. We will now show that it is indeed possible to bound the probability that the congestion on some edge exceeds a given value.

Let *X* denote the congestion imposed on a given edge *e* by a traffic matrix *D* generated from the T-Set. Further, let μ and σ be the average and standard-deviation of the edge congestion on e . Then, by Chebyshev's one-tailed inequality with $k \geq 0$,

$$Pr(X \geq \mu + k\sigma) \leq \frac{1}{1+k^2} \tag{14}$$

By definition, *e* is saturated iff $X \geq c(e)$. Therefore, the probability for e to be saturated is upper-bounded as follows:

$$Pr(X \geq c(e)) \leq \frac{1}{1+\left[\frac{c(e)-\mu}{\sigma}\right]^2} \tag{15}$$

Alternatively, given a desired congestion guarantee level $0 < G < 1$, it is possible to calculate a capacity *c'(e)* that guarantees that at least a fraction *G* of the allowable traffic matrices would be served without saturating *e*. Transforming Equation (15), we get:

$$c'(e) = \mu + \sigma\sqrt{\frac{G}{1-G}} \tag{16}$$

For instance, for $G = 99\%$, we need $c'(e) = \mu + 9.95\sigma$; i.e., with this edge capacity, we are guaranteed that at least 99% of the matrices can be served without saturating e.

These performance bounds are particularly useful when the T-Plot is not exactly Gaussian and cannot be easily characterized. This is true in particular for heterogeneous and weighted T-Plots, for which the Gaussian approximation might be harder to obtain.

Example — Fig. 3(a) provides an example of CDF T-Plot of an edge congestion. It is compared with the Chebyshev-based deterministic guarantee presented above, as well as a Gaussian-based model (as developed in Section 7). This plot was obtained on edge e_{6_7} in the 3×4 mesh of Fig. 1, using DOR routing. It is a CDF plot, corresponding to the PDF plot of Fig. 2.

As seen in Fig. 3(a), the Chebyshev-based deterministic congestion guarantees are rather far below the simulated CDF. This is because the Chebyshev inequality is known to be a very loose bound. On the contrary, in this case, the Gaussian model does very well for this edge, to the point that the plots of the congestion and its Gaussian model can barely be distinguished (as mentioned earlier, the plot passes both the Pearson's χ^2 and the Cramér-von Mises normality tests with a standard significance level of 5%).

As an example, consider an edge congestion of 1.25 (on the x-axis): as shown by simulations, 96% of the matrices cause an edge congestion under this value. By contrast, the Chebyshev-based deterministic approach only guarantees that at least 76% of the matrices will be under this value.

In the same way, the same T-Plot can also be represented as the CCDF (Complementary CDF) of the throughput, as seen in Fig. 3(b). Again, a throughput of at least $\frac{1}{1.25} = 80\%$ is provided to 96% of the matrices on this edge, while the Chebyshev-based performance bound can only guarantee this for 76% of the matrices.

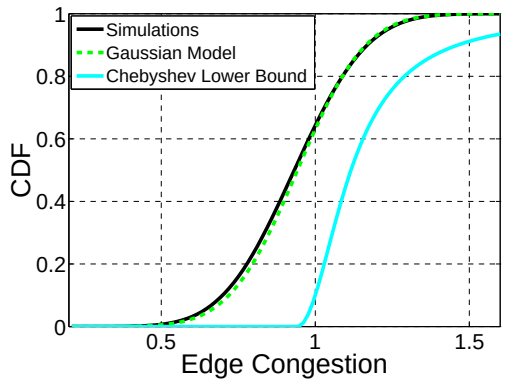
6 Model and bounds of global T-Plots

So far, we have mainly dealt with *edge* congestions. We will now deal with *global* congestions. Of course, succeeding to well approximate the *global* T-Plots would mean obtaining a performance model for the whole network. We will first provide a simple model assuming independence, and then an upper-bound.

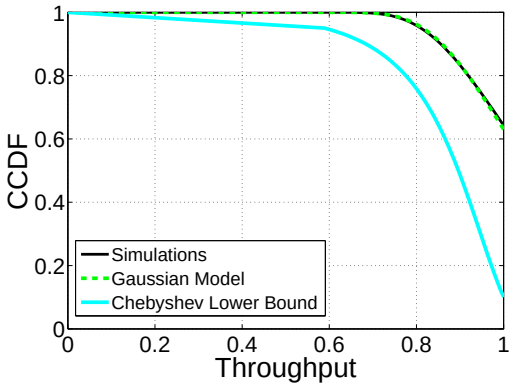
6.1 Edge-independent and independent-Gaussian models

Assuming that all edge congestions are independent, i.e. traffic matrices cause congestion at different links in an independent manner, provides the following *edge-independent model*:

$$\begin{aligned} GC_{CDF}(f,L) &= \Pr\left(\max_{e\in E}[EC(e,f,D)] \leq L\right) \\ &\approx \prod_{e\in E} \Pr\left(EC(e,f,D) \leq L\right) \\ &= \prod_{e\in E} EC_{CDF}(e,f,L) \end{aligned} \tag{17}$$



(a) Edge congestion CDF



(b) Edge throughput CCDF

Fig. 3. Two views of the same T-Plot (edge e_{6_7} in the 3×4 mesh)

This edge-independent model is not *always* a good approximation, because matrices might cause loads in a positively correlated way. However, it plays the role of an intuitive lower bound (though it can be shown that it is not always a lower bound).

Further, this model can be extended to an even simpler *independent-Gaussian model*, in which the distributions of all edge congestions are assumed to be Gaussian. In Section 4, we determined their exact average and standard-deviation at each edge e, denoted $\mu(e)$ and $\sigma(e)$. Therefore, this model is fully and exactly determined:

$$GC_{CDF}(f,L) \approx \prod_{e\in E} \Phi\left(\frac{L-\mu(e)}{\sigma(e)}\right), \qquad (18)$$

where Φ denotes the normalized Gaussian CDF.

In the simulations section, we will show that the independent-Gaussian model performed surprisingly well.

6.2 Global upper bound

Let's now look for an upper bound on the CDF T-Plot of the global congestion. The global congestion is the maximum edge congestion across all edges, and therefore it is at least as large as the congestion of the most loaded edge in the network. Thus, we get the following upper bound on the probability of *not* being congested:

$$GC_{CDF}(f,L) \leq \min_{e\in E}\{EC_{CDF}(e,f,L)\}. \qquad (19)$$

Further, if e_1 and e_2 are two edges (e.g. the two most loaded edges in the network, which could be the two different directions of the same link), then:

$$\begin{aligned} Pr(GC>x) &\geq Pr(EC(e_1)>x \ \vee \ EC(e_2)>x) \\ &= Pr(EC(e_1)>x)+Pr(EC(e_2)>x) \\ &- Pr(EC(e_1)> \ x \ \wedge EC(e_2)>x) \\ &\geq Pr(EC(e_1)>x)+Pr(EC(e_2)>x) \\ &- Pr(EC(e_1)+EC(e_2)>2x), \end{aligned} \qquad (20)$$

where $Pr(EC(e_1)+EC(e_2) > 2x)$ is equal to $1 - EC_{CDF}(\hat{e},2x)$, using a dummy edge $\hat{e}$ for which $f(\hat{e}) = f(e_1)+f(e_2)$. Using $\hat{e}$, a similar upper bound can be obtained as follows:

$$\begin{aligned} Pr(GC\leq x) &\leq Pr(EC(e_1)\leq x \ \wedge \ EC(e_2)\leq x) \\ &\leq EC_{CDF}(\hat{e},2x) \end{aligned} \qquad (21)$$

A stricter global upper bound may finally be defined as the minimum of the three bounds (19), (20) and (21).

7 Capacity Allocation for Edge T-Plots

Our goal is now to propose a simple, yet efficient, statistical capacity allocation algorithm, which would enable significant savings in the total capacity, yet achieve full service for the vast majority of the traffic patterns in the CMP. We first explain in this section how our statistical approach enables to dramatically decrease the capacity of a given edge with only a negligible effect on the throughput on that edge. In the next section, we show that our global capacity allocation scheme is optimal in CMP architectures that obey some simplifying assumptions. Finally, the simulations in Section 9 suggest that the capacity allocation scheme is close to optimal in reference CMP architectures as well.

7.1 Gaussian model

We will now prove that when scaling a CMP network that obeys simplifying assumptions, a *statistical* design allows cutting the edge capacity by almost 50%, while still guaranteeing full service with probability arbitrarily close to 1. To do so, we will first show that the normalized edge T-Plot is asymptotically Gaussian.

Consider a CMP with n nodes. Assume that the T-Set $\mathcal{T}$ is defined such that the processes of core i send traffic to each of the other $n-1$ cores j according to some uniform i.i.d. distribution. The uniform distribution is taken so that any core does not exceed its normalized

maximum input/output rate of 1 word per clock-cycle: for $D \in \mathcal{T}$,

$$\forall i \neq j, D_{ij} \sim Uniform\left(\left[0, \frac{1}{n-1}\right]\right). \quad (22)$$

Consider some edge e in the network. We want to characterize the way the network is scaled. Let's denote by $s_e(n)$ the number of (source, destination) flows crossing e. Then we will assume that *as n goes to infinity, $s_e(n)$ goes to infinity as well*. In other words, as the network is scaled, the routing is intuitively scaled accordingly. This assumption holds for most network architectures of interest, as in the following mesh and tree examples.

Example 7.1: Consider a $\sqrt{n} \times \sqrt{n}$ mesh of n modules with DOR (XY) routing. Then we always have $s_e(n) \geq \sqrt{n}(\sqrt{n}-1)$, with the equality being reached for instance in e_{1_2} (the edge connecting the first two nodes, as illustrated in Fig. 1). In particular, *as n goes to infinity, $s_e(n)$ goes to infinity as well*.

Example 7.2: Consider a tree of n modules with hierarchical routing. Then we always have $s_e(n) \geq n-1$, with the equality being reached for instance in any of the leaf directed edges. In particular, *as n goes to infinity, $s_e(n)$ goes to infinity as well*.

We will denote the average, standard-deviation, and maximum of the flow on edge e by μ, σ and w. By Equation (22), it is clear that the maximum flow generated by each (source, destination) pair is $\frac{1}{n-1}$, and therefore $w = \frac{s_e(n)}{n-1}$. Likewise, using independence and summation rules of the expectation and variance, $\mu = \frac{w}{2}$ and $\sigma = \frac{\sqrt{s_e(n)}}{\sqrt{12}\cdot(n-1)}$.

A worst-case deterministic approach would allocate a capacity equal to the maximum possible edge flow: $c(e) = w$. We will now show that as we scale the CMP, the edge flow distribution becomes extremely concentrated around its average $\mu = \frac{w}{2}$. Therefore, by following a statistical design and allocating a capacity just above this average, we can gain nearly 50% capacity with a loss probability going to zero. To prove this, we will first demonstrate that modeling the edge T-Plot as Gaussian is asymptotically correct in this network. We will later use this model to analyze capacity allocation schemes.

Theorem 3: For the T-Set $\mathcal{T}$, as n grows and we scale the CMP architecture, the normalized edge T-Plot of any edge e converges to the normalized Gaussian distribution $\mathcal{N}(0,1)$.

Proof: See Appendix C. □

Example 7.3: Fig. 4 illustrates Theorem 3 using the $\sqrt{n} \times \sqrt{n}$ mesh of Example 7.1. It plots the PDF of the edge congestion on edge e_{1_2}, as the mesh is scaled and the number of nodes n increases, while keeping a DOR routing. It shows that for this very common network, the convergence defined in Theorem 3 is rather fast.

7.2 Statistical capacity allocation

Denote by $\Phi(x)$ the normalized Gaussian CDF, and let $k(n) = \frac{c(e)-\mu}{\sigma}$. Then a consequence of Theorem 3 is that

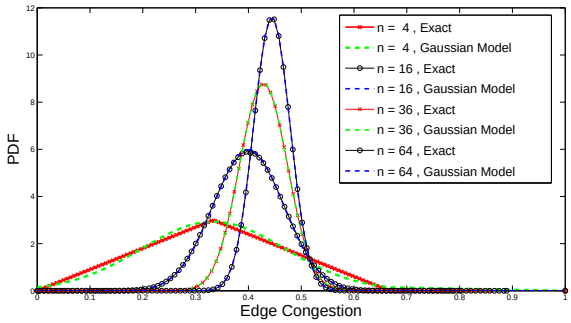


Fig. 4. PDF of the edge congestion on edge e_{1_2} in an n-node mesh

the percentage of traffic matrices that do not saturate edge e (i.e. such that the flow on e is at most $c(e)$) converges to $\Phi(k(n)) = \Phi((c(e)-\mu)/\sigma)$ as n increases. For instance, suppose that we would like to guarantee that at least 99% of the matrices do not saturate e. Since $\Phi^{-1}(0.99) = 2.33$, it suffices to allocate capacity $c(e) = \mu + 2.33\sigma$, rather than allocating the worst-case capacity $c(e) = w = 2\mu$. Asymptotically, we can gain up to 50% capacity if σ/μ goes to zero when n goes to infinity. In fact, the theorem below shows that having a capacity allocation barely above 50% of the worst-case capacity is enough to guarantee any level of performance guarantee on edge e as we scale n.

Theorem 4: For the T-Set $\mathcal{T}$, given any small $\epsilon > 0$, any edge e, and any guaranteed probability $G < 1$, having an edge capacity of $(\frac{1}{2}+\epsilon)$ of the worst-case link capacity and n large enough is sufficient to guarantee full service on edge e for a fraction G of all traffic matrices.

Proof: See Appendix C. □

These results are of course only correct under the assumptions made above; in the general case, we do not claim that the T-Plot necessarily converges to a Gaussian distribution, and leave it to future study.

8 Capacity Allocation for Global T-Plots

Let's denote by c_i the capacity of edge i, and by μ_i, σ_i the mean and standard deviation of the load on edge i, respectively. We now suggest to allocate to edge i a capacity of $c_i = \mu_i + k\sigma_i$, where we use the same value of k for all edges. Therefore, the total capacity C required as a function of k is:

$$C = \sum_{i=1}^{|E|} c_i = \sum_{i=1}^{|E|} \mu_i + k \sum_{i=1}^{|E|} \sigma_i, \quad (23)$$

or, equivalently, for a given total capacity C, we need to use

$$k = \frac{C - \sum_{i=1}^{|E|} \mu_i}{\sum_{i=1}^{|E|} \sigma_i}. \quad (24)$$

Note that when the total capacity is constrained to be smaller than the sum of the average-case edge congestions, k is negative.

The following theorem demonstrates that this capacity allocation minimizes the probability that the network is saturated, in any NoC with any topology and any routing, as long as two approximation assumptions hold: first, the loads on different edges are independent; and second, the edge T-Plots obey a Gaussian model with the same standard-deviation.

Theorem 5: Assume that the T-Plots of all edges i are independent and Gaussian of mean μ_i and same standard-deviation σ. Then allocating to each edge i a capacity $c_i = \mu_i + k\sigma$, where k is a real constant, minimizes the probability that the network is saturated.

Proof: See Appendix D. □

In the simulations, we will evaluate the performance of this capacity allocation algorithm in NoC architectures.

9 Simulations

9.1 T-Set representation

To perform simulations, we need the ability to represent the T-Set. We proved above that this is intrinsically hard (Theorem 1). Therefore, we want to pick traffic matrices uniformly at random from the T-Set in order to approximate their full representation – and to do so, we use *random-walk sampling* [33], [34], [35]. In the simulations below, sampling is always done using one million samples, unless mentioned otherwise. It is also assumed that nodes don't send traffic to themselves. After starting from some matrix in the T-Set (e.g. the null matrix), we compute a new matrix each time. To do so, at each step, we pick some small random matrix change Δ, and add it to the current matrix D, so that the new matrix is $D' = D + \Delta$. If D' is in the T-Set, we move to it, and reject it otherwise. (In weighted T-Sets, we simply move to D' with a probability that is proportional to its weight.)

In continuous T-Sets such as $\mathcal{A}$, there are several possible ways to choose Δ. Below, we use a *Gaussian random walk*, where Δ is normally distributed with distribution $\Delta_{i,j} \sim \mathcal{N}(0, \sigma^2)$ for each pair (i, j), given some standard-deviation σ (more formally, this is a random-walk Metropolis-Hastings algorithm in which the proposal density follows a multivariate normal distribution [34], [35]). We also compare it with a *uniform random walk*, in which Δ is distributed uniformly so that $\Delta_{i,j} \sim Uniform([-\Delta_{max}, \Delta_{max}])$ for each pair (i, j), where the standard-deviation of $\Delta_{i,j}$ is σ, i.e. $\Delta_{max} = \sqrt{3}\sigma$. For both walks, we use $\sigma = \frac{1}{2n}$. We show below that they both yield similar performances.

9.2 Correctness of the T-Set Representation

Although we cannot guarantee that the sample distribution from the random walk will always converge to the uniform distribution on the T-Set, we conduct several tests to verify that it does in smaller networks, and that it satisfies some expected properties in larger networks.

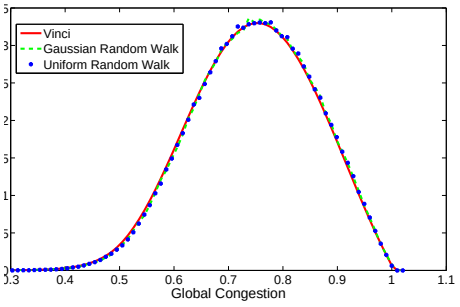


Fig. 5. Global congestion PDF of the 2×2 mesh

First, we use the fact that computing the T-Plots when the T-set is $\mathcal{A}$ may also be expressed as a volume computation problem [36]. For each load value L, the region

$$\mathcal{R} = \{D \in \mathcal{A} | GC(f, D) < L\}$$

forms a convex polytope. $GC^T_{CDF}(f, L)$, the fraction of matrices with global congestion over L, is equal to the ratio between the volume of $\mathcal{R}$ and the volume of the polytope $\mathcal{A}$.

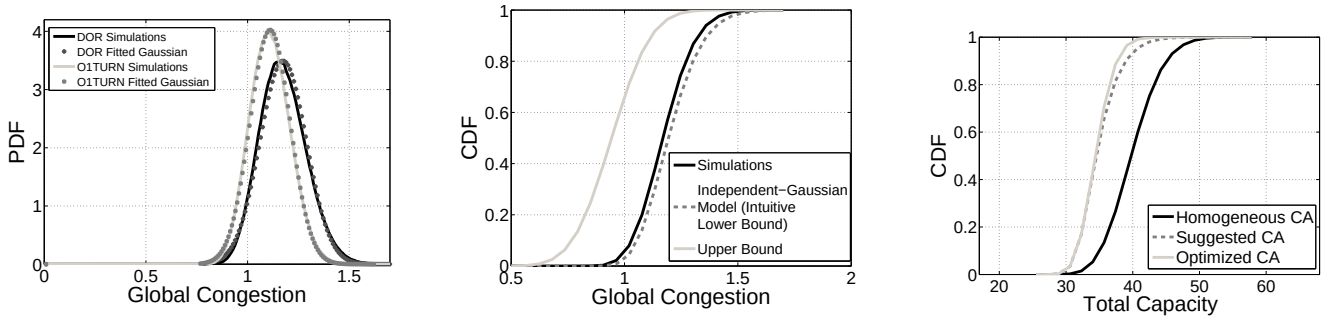
We compare the results of our Monte Carlo simulations using 1 million matrices generated uniformly at random to the exact results obtained by Vinci [37], a standard convex-polytope volume computation tool. The general computation of the volume of polytopes is *#P-complete* [38], and therefore the network cannot be scaled in Vinci to more typical sizes. We use a 2×2 mesh and DOR routing.

Fig. 5 is a T-Plot of the global congestion at this network. The graph shows that the simulation results are extremely close to Vinci's exact results, both when the random walk is Gaussian and uniform. In fact, we have also found that both random walks have about the same convergence speed. In the remainder, we only show the results of Gaussian random walks.

The comparison with Vinci strongly validates the approach with small n. However, as n becomes larger, because of the #P-completeness, there is no way to check the correctness of the results. For these cases we check two other, much weaker properties, which indicate that the results are not biased. First, we check that the resulting T-Plot is independent of the initial matrix. Second, after k experiments of points generated uniformly at random, we verify that the variance does indeed decrease as $O(1/k)$ and obey the formula $var[X] = \frac{p(1-p)}{k}$, where X is the indicator random variable representing the fact that the sample is in some bin and $p = EX$.

9.3 Global T-Plot

We already saw simulation results of *edge congestion T-Plots* in Figures 2 and 3. Let's now look at *global congestion T-Plots*, which show the distribution of the maximum load across all edges. In Figures 6(a), 6(b),



(a) Global congestion PDF: DOR vs O1TURN (b) Global congestion CDF: model and bound (c) Global congestion CDF: CA schemes

Fig. 6. Global congestion T-Plots for the 3x4 mesh

and 6(c), we analyze several parameters of the global congestion T-Plots for the 3x4 mesh.

First, Fig. 6(a) shows the PDF of the global congestion, with the routing algorithm being either DOR or O1TURN [39]. The graph shows that O1TURN does a much better job than DOR in load-balancing the load across different links, and thus has less chances of reaching high link loads (for instance, the area under the PDF to the right of 1.4 is much smaller in O1TURN). Incidentally, both graphs seem well fitted to Gaussian distributions, and both pass the Pearson's χ^2 and the Cramér-von Mises normality tests with a standard significance level of 5%. This is a surprising result, as the maximum of many i.i.d. Gaussian random variables does *not* behave as a Gaussian random variable (it follows a Gumbel distribution [40]). Nevertheless, the Gaussian distribution proves a better fit than the Gumbel distribution. More precisely, the mean-squared error between the CDF of each plot and the CDF of the corresponding Gaussian (resp. Gumbel) plot is $2.6 \cdot 10^{-5}$ (resp. $5.6 \cdot 10^{-4}$) for O1TURN, and $6.2 \cdot 10^{-5}$ (resp. $4.5 \cdot 10^{-4}$) for DOR. The question of when a *global* T-Plot can be approximated by a Gaussian distribution is left for future research.

Fig. 6(b) shows the CDF of the global congestion in the same network, using DOR routing. The independent-Gaussian model and the upper bound are presented in Section 6. We can see that the independent-Gaussian model assuming independent edge congestions with Gaussian distributions is rather close to the exact results. The upper bound, however, is rather loose, which is explained by the fact that it is based on the two most loaded edges in the network, while our network contains many other highly-loaded edges, which may raise the global congestion. Other simulations (not shown here) show that this upper bound is stricter in networks in which there exist only very few highly-loaded edges.

Fig. 6(b) can be used to determine the required capacity overprovisioning: for instance, the CDF for a global congestion of 1 is 0.053. Therefore, without overprovisioning, only 5.3% of the traffic matrices in the T-Set would be fully served. Since the CDF for a global congestion of 1.2 is 0.604, an overprovisioning of 20% would guarantee that 60.4% of the traffic matrices would be fully served.

9.4 Capacity allocation algorithms

Until now, all of our T-Plots were realized without doing any optimization, by simply measuring the distribution of the link load. We will now show that our statistical approach using T-Plots can do more than just *measure*: it can also help *optimize*.

Fig. 6(c) illustrates the performance of different capacity allocation (CA) algorithms on the 3×4 mesh network with 34 edges (presented in Fig. 1). For each total capacity, it shows the fraction of matrices that would be served under a given CA algorithm. It compares three CA algorithms: the homogeneous CA assumed above, the simple CA based on means and variances suggested in Section 8, and an optimized CA explained below.

For instance, assume that the average capacity per edge is 1.2, i.e. the total capacity is $1.2 \cdot 34 = 40.8$. The homogeneous CA algorithm would allocate a capacity of exactly 1.2 to each edge. It would only be able to service 60.4% of the matrices (as seen above as well with Fig. 6(b)).

On the contrary, our simple CA scheme suggested in Section 8 would distribute the total capacity differently among the edges, according to their congestion average and variance. With this total capacity of 40.8, it would be able to service 96.4% of the matrices, hence improving noticeably on the homogeneous scheme.

Finally, to examine the quality of our simple capacity allocation scheme, we compare it to an optimized CA, which was obtained after extensive brute-force simulations. Using this optimized CA, we can service 99.2% of the matrices, hence slightly improving on our simple suggested CA. In fact, the plot suggests that our simple suggested heuristic CA is not too far from optimum.

Note that to obtain this optimized CA, we ran 10,000 iterations for each total capacity value. At each iteration, a new CA is taken at the neighborhood of the old one using a Gaussian random-walk algorithm, and is only accepted if it fares better (as explained above). The optimization was done using 200,000 sample matrices from

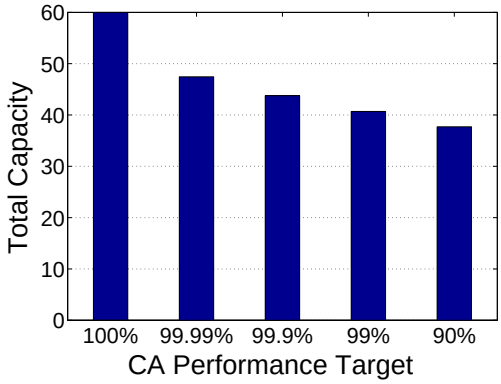


Fig. 7. Total capacity required for various CA targets

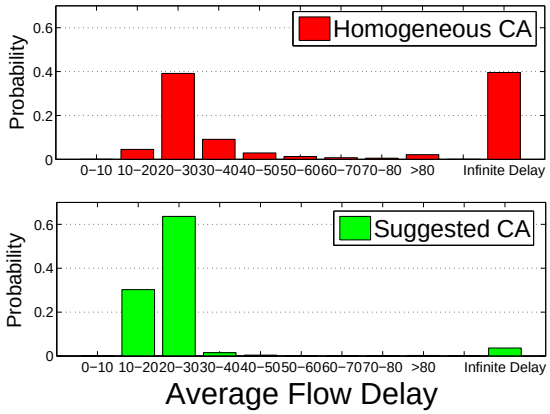


Fig. 8. Average flow delay distribution over all traffic matrices, for two different CA schemes

$\mathcal{A}$, and the results were computed on 200,000 different matrices. We also checked that starting from different points yields the same end result.

9.5 Capacity allocation and throughput guarantee

We will now exemplify how our statistical approach enables a drastic capacity saving with only negligible deterioration in performance.

Fig. 7 compares the total capacities needed by the optimized CA algorithm with 5 different performance targets, in the 3×4 mesh. The first bar represents a *worst-case approach*, in which each edge is allocated a capacity according to the worst-case flow *on this edge*, thus guaranteeing that 100% of traffic matrices will be fully served. It is loosely based on the worst-case approach adopted in [3], [22], [23]. On the contrary, the other bars represent the statistical approach, with increasingly loose levels of statistical-based capacity allocation schemes. Their values can be retrieved from Fig. 6(c). For instance, for $G = 99.9\%$, the amount of provisioning needed is $CDF^{-1}(0.999) = 43.8$.

Fig. 7 shows that switching from a worst-case to a statistical CA approach may save up to 37% of the total required capacity in this network, for a capacity guarantee at a 90% level. Likewise, planning for a very stringent 99.99% cutoff decreases the amount of total capacity used by 21%. As an aside, note that we didn't even compare with the naive *homogeneous* worst-case approach — such a comparison would have yielded even greater savings!

9.6 Delay distribution

Our objective is to obtain some intuition on the different distributions of the expected flow delays using different CA algorithms. In order to do so, we model the delay at each edge with the simple M/M/1 model, using an arrival rate equal to the edge flow and a service rate equal to the edge capacity. (Of course, this is just a toy model: the deterministic nature of the services would probably decrease the average delays, and the wormhole scheduling [20] would increase them.) The average delay of a flow is the sum of its average edge delays. Finally, for each given traffic matrix, we compute the average flow delay across all flows. Note that a saturated edge results in an infinite edge delay, and therefore an infinite average flow delay.

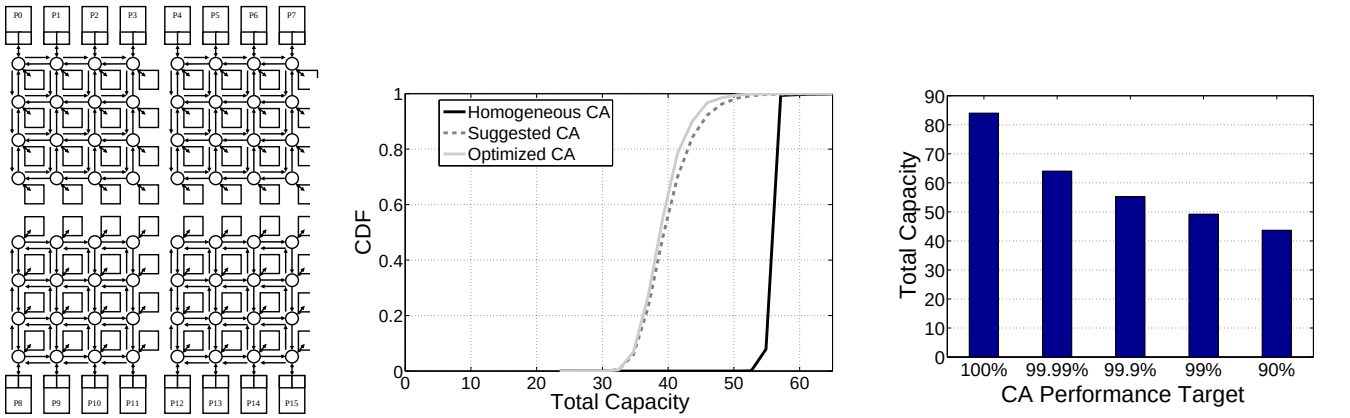
Fig. 8 compares the distributions of the average flow delays for both the homogeneous CA and our simple suggested CA, for the 3×4 mesh with average edge capacity 1.2. As expected, our CA scheme has significantly less traffic matrices with infinite average flow delay; in addition, on the remaining matrices, the average flow delay also tends to be lower. Thus, this plot confirms that our simple CA tends to significantly outperform the homogeneous CA.

9.7 NUCA network

Finally, we considered a different CMP architecture model based on a NUCA (non-uniform cache architecture) network. As shown in Fig. 9(a) (based on [41] with sharing degree 4), the network contains 4 sub-networks, each with 4 processor cores and 16 caches, hence with a total number of 80 nodes and 224 edges. Each core may only send (receive) traffic to (from) caches in its sub-network, and each cache may only send (receive) traffic to cores in its sub-network, with a maximum node transmission (reception) rate of 1.

Fig. 9(b) compares the different CA schemes on this NUCA network (simulated using 100,000 samples). For example, with a total capacity of 50, using our suggested CA dramatically increases the probability that the NUCA network is not saturated from less than 1% to 98%. Again, it is very close to the optimized envelope.

Likewise, Fig. 9(c) shows that the total capacity required to fully serve 99.99% of the matrices is lower than the total capacity in the worst-case approach by 24%, and in the 90% cutoff case by 48%. Thus, this confirms the intuition that as networks grow in size, the gains in the statistical approach tend to grow as well - intuitively confirming Theorem 4 as well.



(a) NUCA architecture (based on [41] with sharing degree 4)

(b) Global congestion CDF: CA schemes

(c) Total capacity required for various CA targets

Fig. 9. NUCA network: topology and performance

10 Conclusion

In this paper, we introduced the T-Plots, which can provide a common foundation to quantify, design, optimize and compare NoCs architectures and routing algorithms. We showed that an accurate computation of T-Plots is #P-complete, but that they can sometimes be modeled as Gaussian, providing a full link load distribution model using only two variables. Further, we provided bounds that can be the basis of strict throughput performance guarantees. We finally showed how T-Plots can be used to develop a simple, yet efficient, capacity allocation scheme. We believe and hope that this work will contribute to lay the ground to a common basis in future NoC design research.

Appendix A
Finding the edge T-Plot is #P-Complete

We will now provide some standard formal definitions [32].

Definition 1: Let f be a function. We say that $f \in$ #*P* if there exists a binary relation R such that:

- If $(x, y) \in R$ then the length of y is polynomial in the length of x.
- It can be verified in polynomial time that a pair (x, y) is in R.
- For every $x \in \Sigma^*$ (the set of all 0-1 strings), $f(x) = |\{y : (x, y) \in R\}|$

Definition 2: Given two functions f, g:

- There is a *polynomial Turing-reduction* from g to f (and denote $g \propto f$) if the function g can be computed in polynomial time using an oracle to f.
- A function $f : \Sigma^* \rightarrow N$ is *#P-Hard* if for every $g \in$ #*P* there is a polynomial reduction $g \propto f$.
- A function f is *#P-complete* iff it is both *#P-Hard* and in #*P*.

Definition 3: Given an $n \times n$ matrix A, the permanent of A is defined as

$$Perm(A) = \sum_{D \in \mathcal{P}} \prod_{i=1}^{n} A_{i,D(i)} \tag{25}$$

We denote the problem of computing the permanent of a binary 01-matrix as *01-Perm*. It was shown in [32] that 01-Perm is #P-complete. Our goal will be to find a polynomial-time reduction from the problem of *01-Perm* to the T-Plot computation problem. This reduction relies on the following lemma.

Lemma 1: Let $G(V, E)$ be a connected graph. Let e be a non-bridge edge. Then, for each pair of nodes (s, d), there exists at least one walk, which uses e exactly once, and at least one additional walk, which doesn't use e.

Proof: Let's denote $e = (a, b)$. Since e is not a bridge, for each (source, destination) pair of nodes (s, d), there exists a walk that doesn't use e. Let's also construct a walk from s to d that uses e exactly once. First, from s to a, we use a walk that doesn't use e (such a walk exists, because e is not a bridge). Then, from a to b, we cross through e. Finally, from b to d, we use another walk that doesn't use e (again, such a walk exists, because e is not a bridge). □

We will now prove that when the T-Set is the set of permutations $\mathcal{P}$, finding the edge T-Plot of a non-bridge edge e is *#P-complete*. To do so, we will first demonstrate that the problem is #*P* (Lemma 2), and then show a polynomial-time reduction from the problem of *01-Perm* (Lemma 3). Intuitively, given an $n \times n$ matrix A and a non-bridge edge e, this reduction computes a routing algorithm f that routes packets from source s to destination d via e iff $A_{sd} = 1$ (this is possible by Lemma 1). We will show that as a consequence, the value of the permanent of A is equal to the number of permutations, for which all the flow is routed via e. Therefore, using this equality, an oracle to a specific point of the PDF of the congestion on e suffices to calculate the permanent of A.

Lemma 2: Let $G(V, E)$ be a directed graph, in which

the traffic is routed according to an oblivious routing algorithm f. Let e be a non-bridge edge. Then, $EC^{\mathcal{P}}_{PDF}(e,f,L)$ is #*P*.

Proof: Let us denote $L=\frac{n}{c(e)}$. Let's define the binary relation R as follows:

$$((e,f,L),D)\in R \Leftrightarrow EC(e,f,D)=L \tag{26}$$

- The size of the representation of a permutation is polynomial in the representation of (e,f).
- It can be verified in polynomial time whether $((e,f,L),D)\in R$ by calculating $EC(e,f,D)$.
- As $EC^{\mathcal{P}}_{PDF}(e,f,L)$ represents the probability of a randomly chosen permutation to impose congestion L on edge *e*, and $|\mathcal{P}|=n!$,

$$n!EC^{\mathcal{P}}_{PDF}(e,f,L)=|\{D\in\mathcal{P}:EC(e,f,D)=L\}| \tag{27}$$

Note that the multiplication by $n!$ doesn't affect $EC^{\mathcal{P}}_{PDF}(e,f,L)$ being #*P*. □

Lemma 3: Let *A* be an $n\times n$ 01-matrix. Let D be a permutation of $\{1,2,...,n\}$. Then, it is possible to construct in polynomial time a routing algorithm f s.t.

$$\prod_{i=1}^{n}A_{i,D(i)}=1\Leftrightarrow EC(e,f,D)=L \tag{28}$$

Proof: We will construct an oblivious routing algorithm f s.t.

$$\forall i,j\in V: f_{ij}(e)=A_{ij} \tag{29}$$

In other words, if $f_{sd}(e)=1$, f will use *e* exactly once when routing packets from s to d. Otherwise, f will not use *e* when routing packets from s to d. By the construction:

$$\begin{aligned}\prod_{i=1}^{n}A_{i,D(i)}=1 &\Leftrightarrow \sum_{ij}D_{ij}A_{ij}(e)=n\\ &\Leftrightarrow \sum_{ij}D_{ij}f_{ij}(e)=n\\ &\Leftrightarrow EC(e,f,D)=\frac{n}{c(e)}=L\end{aligned} \tag{30}$$

If $A_{sd}=0$, f routes the packets from s to d via a walk, which does not use e. Else, f routes the packets from s to d via a walk which uses *e* exactly once, as described in the proof of Lemma 1. f can be calculated by running the Dijkstra Algorithm n times on the graph $G'(V,E\backslash\{e\})$, each time taking a different node in the graph as the source node. Each edge in $E\backslash\{e\}$ has a unit weight. The complexity of the construction is polynomial, as it requires $O(|V|)$ runs of the Dijkstra Algorithm, where each run takes polynomial time. □

We are finally ready to demonstrate the #P-completeness of the edge T-Plot computation.

Proof of Theorem 1: Let A be an $n\times n$ 01-matrix. By Lemma 2, $EC^{\mathcal{P}}_{PDF}(e,f,L)$ is #*P*. Successively using Lemma 3 and Equation (27):

$$\begin{aligned}Perm(A) &= |\{D\in\mathcal{P}:\prod_{i=1}^{n}A_{i,D(i)}=1\}|\\ &= |\{D\in\mathcal{P}:EC(e,f,D)=L\}|\\ &= n!EC^{\mathcal{P}}_{PDF}(e,f,L)\end{aligned} \tag{31}$$

This is a polynomial reduction from *01Perm* to $EC^{\mathcal{P}}_{PDF}$. As *01Perm* is #P-Hard, $EC^{\mathcal{P}}_{PDF}$ is #P-complete, where even the task of computing the value of the distribution at an arbitrary point is #P-complete. Consequently, $EC^{\mathcal{P}}_{CDF}$ is #P-complete as well. □

APPENDIX B
FINDING THE GLOBAL T-PLOT IS #P-COMPLETE

We will now prove that plotting a global T-Plot is #P-complete as well. In our proof, we will use a network with a strictly minimal edge e_m, so that the worst-case congestion on e_m is higher than the worst-case congestion on any other edge in the network. Thus, e_m's worst-case edge congestion is equal to the network's *global* congestion. Thus, a similar construction to that used when proving Theorem 1 would suffice to prove that $GC^{\mathcal{P}}_{CDF}$ is *#P-complete* as well.

Proof of Theorem 2: Let us denote: $L=\frac{n}{c(e_m)}$. Let *A* be an $n\times n$ 01-matrix. $GC(f,L)\in$ #*P* by the same proof as that of Lemma 2. By Lemma 3, there exists a polynomial-time construction of a routing algorithm f s.t.

$$\prod_{i=1}^{n}A_{i,D(i)}=1\Leftrightarrow EC(e_m,f,D)=L \tag{32}$$

As e_m is strictly minimal, its worst-case congestion is higher than the worst-case congestion on every other edge, and therefore its worst-case edge congestion is equal to the network's *global* congestion. We finally get:

$$Perm(A)=n!EC^{\mathcal{P}}_{PDF}(e_m,f,L)=n!GC^{\mathcal{P}}_{PDF}(f,L)$$

As in the proof of Theorem 1, even the task of computing the value of $GC^{\mathcal{P}}_{PDF}$ at an arbitrary point is #P-complete. Consequently, $GC^{\mathcal{P}}_{CDF}$ is #P-complete as well. □

APPENDIX C
GAUSSIAN PROPERTIES OF EDGE T-PLOTS

Proof of Theorem 3: Let $S_e(n)$ denote the flow on edge e. Since it is the sum of $s_e(n)$ i.i.d. uniform random variables, which we will rewrite as $Z_1,\cdots,Z_{s_e(n)}$, we have:

$$S_e(n)=\sum_{i=1}^{s_e(n)}Z_i. \tag{33}$$

By definition, $Z_i\sim Uniform\left(\left[0,\frac{1}{n-1}\right]\right)$, therefore we also have

$$S_e(n)=\frac{1}{n-1}\cdot\sum_{i=1}^{s_e(n)}Z'_i, \tag{34}$$

where $Z'_i \sim Uniform([0,1])$. Define the normalized flow

$$S'_e(n) = \sum_{i=1}^{s_e(n)} Z'_i. \quad (35)$$

Then $S'_e(n)$ is the sum of $s_e(n)$ i.i.d. normalized uniform random variables of mean $\mu' = \frac{1}{2}$ and standard deviation $\sigma' = \frac{1}{\sqrt{12}}$. Further, by assumption, when n goes to infinity, $s_e(n)$ also goes to infinity. Thus, by the Central Limit Theorem, the distribution of $\frac{S'_e(n) - s_e(n)\mu'}{\sqrt{s_e(n)}\sigma'}$ converges (in distribution) towards the standard normal distribution $\mathcal{N}(0,1)$.

Finally, by definition, $S_e(n) = \frac{1}{n-1} S'_e(n)$, $\mu = \frac{1}{n-1} s_e(n)\mu'$, and $\sigma = \frac{1}{n-1}\sqrt{s_e(n)}\sigma'$. Therefore, we get that $\frac{S_e(n)-\mu}{\sigma}$ converges to $\mathcal{N}(0,1)$ as well. (Of course, note that μ and σ are also implicitly dependent on n and e.) □

Proof of Theorem 4: The proof is in two steps. Let $0 < \epsilon' < 1 - G$. We first show that for n large enough, the total flow on the edge is extremely close to half the worst-case capacity, and therefore, with an edge capacity of $(\frac{1}{2} + \epsilon)$ of the worst-case capacity, a T-Plot following a Gaussian distribution would obtain a guarantee level of at least $G + \epsilon'$. But, of course, our T-Plot is not exactly Gaussian. Therefore, in a second step, we use the result of the Central Limit Theorem (Theorem 3) and the fact that our distribution is indeed within ϵ' of a Gaussian distribution, and we finally obtain a guarantee level of at least $G + \epsilon' - \epsilon' = G$.

First, let $k(n) = \frac{c(e)-\mu}{\sigma}$. Then

$$\begin{aligned} k(n) &= \frac{(\frac{1}{2}+\epsilon)w - \mu}{\sigma} \\ &= \frac{\epsilon w}{\sigma} \\ &= \frac{\epsilon \frac{s_e(n)}{n-1}}{\frac{\sqrt{s_e(n)}}{\sqrt{12}(n-1)}} \\ &= \epsilon\sqrt{12 s_e(n)} \end{aligned} \quad (36)$$

By assumption, when n goes to infinity, $\epsilon\sqrt{12 s_e(n)}$ also goes to infinity as well, and therefore will become larger than any fixed value. In particular, there exists some large n' such that $\epsilon\sqrt{12 s_e(n')} \geq \Phi^{-1}(G + \epsilon')$, so for any $n \geq n'$,

$$\begin{aligned} k(n) &= \epsilon\sqrt{12 s_e(n)} \\ &\geq \epsilon\sqrt{12 s_e(n')} \\ &\geq \Phi^{-1}(G + \epsilon'). \end{aligned} \quad (37)$$

Thus, when n becomes large, we have

$$\Phi\left(\frac{c(e)-\mu}{\sigma}\right) = \Phi(k(n)) \geq G + \epsilon'. \quad (38)$$

Second, by the Central Limit Theorem (Theorem 3), the normalized edge T-Plot converges to the normalized Gaussian distribution $\mathcal{N}(0,1)$. Hence, for any fixed parameter, the difference between its CDF and Φ will converge to zero, and therefore will become lower than ϵ' for some large value. Put mathematically, for any $\alpha > 0$, there exists some $n'' \geq n'$, so that for any $n \geq n''$,

$$Pr\left(\frac{S_e(n)-\mu}{\sigma} < \alpha\right) \geq \Phi(\alpha) - \epsilon'. \quad (39)$$

Using $\alpha = \Phi^{-1}(G + \epsilon')$, we get

$$\begin{aligned} Pr\left(\frac{S_e(n)-\mu}{\sigma} < \Phi^{-1}(G+\epsilon')\right) &\geq (G+\epsilon') - \epsilon' \\ &= G. \end{aligned} \quad (40)$$

Finally,

$$\begin{aligned} Pr(S_e(n) < c(e)) &= Pr\left(\frac{S_e(n)-\mu}{\sigma} < \frac{c(e)-\mu}{\sigma}\right) \\ &= Pr\left(\frac{S_e(n)-\mu}{\sigma} < k(n)\right) \\ &\geq Pr\left(\frac{S_e(n)-\mu}{\sigma} < \Phi^{-1}(G+\epsilon')\right) \\ &\geq G, \end{aligned} \quad (41)$$

where the last two inequalities follow from Equations (37) and (40). In other words, a fraction of at least G of the traffic matrices generate a flow on the edge that is under the edge capacity. □

APPENDIX D
GAUSSIAN PROPERTIES OF GLOBAL T-PLOTS

Proof of Theorem 5: We denote the total available capacity by C , i.e., a legal capacity allocation satisfies $\sum_i c_i \leq C$. We would like to maximize the probability that the global congestion is below 1. Formally, we want to find

$$\underset{\{c_1,\cdots,c_{|E|}\}}{\text{argmax}} \; Pr(GC(f, c_1, \cdots, , c_{|E|}) < 1)$$

$$\text{s.t.} \sum_{i=1}^{|E|} c_i = C$$

Using the independence assumption, we want to maximize the product of the probabilities of edge congestions on all edges:

$$Pr(GC(f, c_1, \cdots, c_{|E|}) < 1) = \prod_{i=1}^{|E|} EC_{CDF}(e_i(c_i), f, 1) \quad (42)$$

Let $\Phi(x)$ be the standard Gaussian CDF, and $\phi(x)$ the PDF (its derivative). Then, by the definitions in Section 7,

$$EC_{CDF}(e_i(c_i), f, 1)) = \Phi((c_i - \mu_i)/\sigma). \quad (43)$$

Using Lagrange multipliers and differentiating Equation (42), we find that the objective function is maximized when for each $i \neq j$,

$$\frac{EC_{PDF}(e_i(c_i), f, 1))}{EC_{CDF}(e_i(c_i), f, 1))} = \frac{EC_{PDF}(e_j(c_j), f, 1))}{EC_{CDF}(e_j(c_j), f, 1))}, \quad (44)$$

i.e.

$$\frac{\phi((c_i-\mu_i)/\sigma)}{\Phi((c_i-\mu_i)/\sigma)} = \frac{\phi((c_j-\mu_j)/\sigma)}{\Phi((c_j-\mu_j)/\sigma)}. \quad (45)$$

Equation (45) is solved when $c_i = \mu_i + k\sigma$ for each i, where k is *the same* constant for each i, because all the PDF probabilities are then equalized (the standard-deviations are equal) as well as the CDF probabilities. Therefore, this solution maximizes the objective function. Note that for different standard deviations, this Lagrangian-based method can also directly bring the optimal, yet less elegant, capacity allocation solution, by solving Equation (45) using methods of numerical analysis. □

ACKNOWLEDGMENT

This work was partly supported by European Research Council Starting Grant No. 210389.

REFERENCES

[1] M.B. Taylor et al., "The raw microprocessor: a computational fabric for software circuits and general purpose programs," *IEEE Micro*, vol. 22, no. 2, pp. 25-35, April 2002.

[2] L. Shang, L.-S. Peh, A. Kumar, and N. K. Jha, "Thermal modeling, characterization and management of on-chip networks," *MICRO*, Portland, Oregon, December 2004.

[3] S. Murali, D. Atienza, P. Meloni, S. Carta, L. Benini, G. De Micheli, and L. Raffo, "Synthesis of predictable network-on-chip-based interconnect architectures for chip multiprocessors," *IEEE Transactions on VLSI Systems*, vol. 15, no. 8, pp. 869–880, August 2007.

[4] Rick Merritt, "AMD, Intel square off in quad-core processors," *EE Times*, September 2007.

[5] AMD, "Quad-core processors," multicore.amd.com/us-en/quadcore/

[6] Intel, "Teraflops research chip," techresearch.intel.com/articles/Tera-Scale/1449.htm

[7] R. Kalla et al., "IBM Power5 chip: a dual-core multithreaded processor," *IEEE Micro*, vol. 24, no. 2, pp. 40-47, March/April 2004.

[8] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet-switched interconnections," *DATE '00*, pp. 250-256, Paris, France, March 2000.

[9] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," *DAC 01*, pp. 684-689, Las Vegas, USA, June 2001.

[10] L. Benini and G. De Micheli, "Networks on chip: a new SoC paradigm," *IEEE Computer*, vol. 35, no. 1, Jan. 2002.

[11] A. Radulescu, and K. Goossens, "Communication services for networks on chip," *Domain-Specific Processors: Systems, Architectures, Modeling, and Simulation*, Marcel Dekker, pp. 193–213, 2004.

[12] R. Mullins, A. West, and S. Moore, "The design and implementation of a low-latency on-chip network", *ASP–DAC*, pp. 164–169, 2006.

[13] Z. Guz, I. Walter, E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "Network delays and link capacities in application-specific wormhole NoCs", *VLSI Design*, May 2007.

[14] J. Kim, M. Taylor, J. Miller and D. Wentzlaff, "Energy characterization of a tiled architecture processor with on-chip networks," *International Symposium on Low-Power Electronics and Design*, 2003.

[15] H. Wang, L.S. Peh and S. Malik, "Power-driven design of router microarchitectures in on-chip networks", *International Symposium on Microarchitecture*, pp. 105–116, San Diego, CA, December 2003.

[16] J. Hu and R. Marculescu, "Exploiting the routing flexibility for energy/performance aware mapping of regular NoC architectures," *DATE 00*, 2003.

[17] S. Murali et al., "Mapping and physical planning of networks-on-chip with quality-of-service guarantees," *ASP–DAC*, pp. 27-32, 2005.

[18] A. Hansson et al., "A unified approach to constrained mapping and routing on network-on-chip architectures," *ISSS*, pp. 75-80, 2005.

[19] K. Srinivasan et al., "An automated technique for topology and route generation of application specific on-chip interconnection networks," *ICCAD*, pp. 231-237, 2005.

[20] W. J. Dally and B. Towles, "Principles and practices of interconnection networks," Morgan Kaufmann, 2004.

[21] A. Baron, R. Ginosar, and I. Keslassy, "The capacity allocation paradox," *IEEE Infocom '09*, Rio de Janeiro, Brazil, April 2009.

[22] B. Towles and W. J. Dally, "Worst-case traffic for oblivious routing functions," *ACM SPAA*, pp. 1-8, 2002.

[23] B. Towles, W. J. Dally, and S. Boyd, "Throughput-centric routing algorithm design," *ACM SPAA*, pp. 200-209, 2003.

[24] N. Duffield, P. Goyal, and A. Greenberg, "A flexible model for resource management in virtual private networks," *ACM SIGCOMM*, 1999.

[25] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "QNoC: QoS architecture and design process for Network on Chip," *The Journal of Systems Architecture*, December 2003.

[26] P. Bogdan and R. Marculescu, "Quantum-like effects in network-on-chip buffers behavior," *DAC – Session: Wild and crazy ideas*, Pittsburgh, PA, pp. 266 – 267, July 2007.

[27] S. Murali, M. Coenen, A. Radulescu, K. Goossens, and G. D. Micheli, "Mapping and configuration methods for multi-use-case networks on chips," *ASP–DAC*, pp. 146-151, 2006.

[28] S. Murali, M. Coenen, A. Radulescu, K. Goossens, and G. De Micheli, "A methodology for mapping multiple use-cases onto networks on chips," *DATE '06*, pp. 118–123, 2006.

[29] R. Gindin, I. Cidon, and I. Keidar, "NoC-based FPGA: architecture and routing," *NOCS '07*, May 2007.

[30] Y. Azar, E. Cohen, A. Fiat, H. Kaplan, and H. Racke, "Optimal oblivious routing in polynomial time," *ACM Symposium on the Theory of Computing*, pp. 383–388, 2003.

[31] H. Sullivan and T. R. Bashkow, "A large scale, homogeneous, fully distributed parallel machine," *ISCA*, pp. 105–117, ACM Press, 1977.

[32] A. Ben-Dor and S. Halevi, "Zero-one permanent is #P-complete, a simpler proof," *Israel Symposium on Theory of Computing and Systems*, IEEE Press, 1993.

[33] S. Vempala, "Geometric random walks: a survey," *MSRI volume on Combinatorial and Computational Geometry*, 2005.

[34] S. Chib and E. Greenberg, "Understanding the Metropolis-Hastings algorithm," *American Statistician*, vol. 49, no. 4, pp. 327-335, 1995.

[35] S. P. Brooks, "Markov chain Monte Carlo method and its application," *The Statistician*, vol. 47, no. 1, pp. 69–100, 1998.

[36] N. Dukkipati, Y. Ganjali, and R. Zhang-Shen, "Typical versus Worst Case Design in Networking," *HotNets-IV*, College Park, November 2005.

[37] VINCI, www.lix.polytechnique.fr/Labo/Andreas.Enge/Vinci.html

[38] M. E. Dyer. and A. M. Frieze, "On the complexity of computing the volume of a polyhedron," *SIAM Journal on Computing*, vol. 17, no. 5, pp. 967–974, 1988.

[39] D. Seo, A. Ali, W. T. Lim, N. Rafique, and M. Thottethodi, "Near-optimal worst-case throughput routing for two-dimensional mesh networks," *ISCA*, pp. 432–443, June 2005.

[40] E. J. Gumbel, "Multivariate extremal distributions," *Bulletin de l'Institut International de Statistique*, vol. 37, pp. 471–475, 1960.

[41] J. Huh, C. Kim, H. Shafi, L. Zhang, D. Burger, and S.W. Keckler, "A NUCA substrate for flexible CMP cache sharing," *ICS '05*, June 2005.

Itamar Cohen received the M.S. degree in electrical engineering from the Technion, Haifa, Israel, in 2007. He is a lecturer in the Jerusalem College of Engineering, Israel. His research interests include the design and analysis of routing algorithms.

Ori Rottenstreich received the B.S. degree in computer engineering from the electrical engineering department of the Technion, Haifa, Israel in 2008. He is now pursuing a Ph.D. degree in the same department. He is mainly interested in computer networks, algorithm design and analysis, and TCAM architectures.

Isaac Keslassy (M'02) received the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, in 2000 and 2004, respectively. He is currently a faculty member in the electrical engineering department of the Technion, Haifa, Israel. His recent research interests include the design and analysis of high-performance routers and on-chip networks. The recipient of the Yigal Alon Fellowship, the ATS-WD Career Development Chair and the ERC Starting Grant, he is a member of the IEEE.