Statistical approach to NoC design

Statistical Approach to NoC Design

Itamar Cohen, Ori Rottenstreich and Isaac Keslassy
Department of Electrical Engineering
Technion - Israel Institute of Technology
Haifa 32000, Israel
{*ofanan, ori.rot*}*@gmail.com, isaac@ee.technion.ac.il*

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. Finally, we use T-Plots to reduce the network power consumption by providing an efficient capacity allocation algorithm with predictable performance guarantees.

1 Introduction

The multi-core era is here. Today, chip multiprocessors (CMPs) combine increasingly many general-purpose processor cores on a single chip [1–7]. As shown in Figure 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 [8–13].

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 [16–19]. It also differs from traditional chip-to-chip multiprocessor interconnects, in which the link bandwidth capacities cannot be easily adjusted [20].

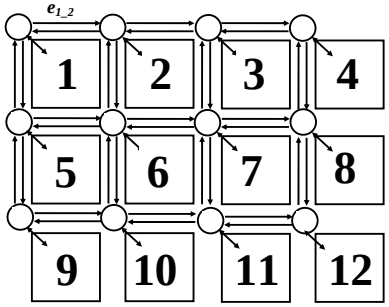


Figure 1. 3×4 NoC-based CMP architecture.

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], 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, 21–23] 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 mul-

tiplexing properties of the NoC, which are increasingly significant as the number of cores increases.

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 tradeoff 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 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.

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, Figure 2 illustrates such a T-Plot, showing the distribution of the normalized load on edge e_{6_7}. The T-Plot is generated using the set of all traffic patterns in the CMP, and 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 overprovisioning. 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 – 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.

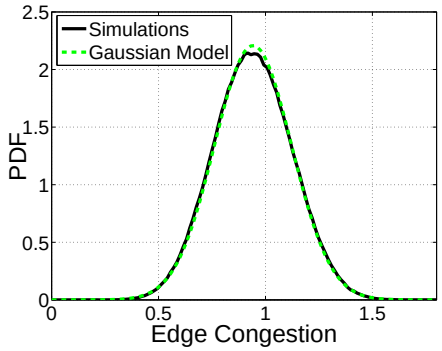


Figure 2. Edge congestion PDF T-Plot on e_{6_7}

We would like to stress that in our view, a key aspect of this 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 [24, 25]. 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 [26]. 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, 27].

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, Figure 1 illustrates such a graph with 12 nodes (each corresponding to a processor core and its associated switch), and 34 edges between them.

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, 21–23] 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

$$\mathcal{A} = \left\{ D | \forall i : \sum_j D_{ij} \leq 1, \sum_j D_{ji} \leq 1 \right\} \quad (1)$$

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 $\mathcal{H} = \left\{ D | \forall i : \sum_j D_{ij} \leq q_i, \sum_j D_{ji} \leq r_i \right\}$. 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 assume that any traffic matrix in a T-Set is always equally likely (though adding weights to specific subsets can of course easily be done if needed).

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 [28]. An example of routing scheme is dimension-ordered routing (DOR) [29], 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.

$$EC(e, f, D) = \frac{\sum_{i,j} D_{ij} f_{ij}(e)}{c(e)} \quad (2)$$

When the edge congestion on e is at least 1, we will say that e is *saturated*. Further, a network is saturated if at least one edge in it is saturated. The *global congestion* for traffic matrix D using f will be obtained by taking the maximum edge congestion over all edges, that is:

$$GC(f, D) = \max_{e \in E} \{EC(e, f, D)\} \quad (3)$$

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

$$TP(f, D) = \min\{GC(f, D)^{-1}, 1\} \quad (4)$$

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*:

$$EC^{\mathcal{T}}_{CDF}(e, f, L) = \Pr\{EC(e, f, D) \leq L | D \in \mathcal{T}\} \quad (5)$$

3 T-Plots are #P-complete

We will now prove that computing the T-Plots is *#P-complete* [30], 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 typically makes it quite harder.

We will first show the #P-completeness for *edge T-Plots*, and then as well for *global T-Plots*. We refer interested readers to [31] for more formal definitions and complete proofs of all the theorems in this paper.

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.

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

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.

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 without running extensive simulations. Furthermore, we will later see that these values will be enough to provide both a Chebyshev-based deterministic bound (Section 5) and a Gaussian-based model (Section 7).

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:

$$\begin{aligned} EC^{\mathcal{P}}_{ac}(e,f) &= \frac{1}{n!}\sum_{D\in\mathcal{P}} EC(e,f,D) \\ &= \frac{1}{n!}\sum_{D\in\mathcal{P}} \frac{\sum_{ij} D_{ij} f_{ij}(e)}{c(e)} \\ &= \frac{1}{n!c(e)}\sum_{ij} f_{ij}(e) \sum_{D\in\mathcal{P}} D_{ij} \\ &= \frac{1}{nc(e)}\sum_{ij} f_{ij}(e), \end{aligned} \quad (6)$$

where 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.

Likewise, the variance is calculated using the variance formula

$$Var_{D\in\mathcal{P}}[EC(e,f,D)] = E[EC^2] - E^2[EC], \quad (7)$$

with

$$E[EC^2] = \frac{\sum_{ijkl} f_{ij}(e) f_{kl}(e) \left(\sum_{D\in\mathcal{P}} D_{ij} D_{kl}\right)}{n! c(e)^2}, \quad (8)$$

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,

$$\sum_{D\in\mathcal{P}} D_{ij} D_{kl} = \begin{cases} (n-1)! & i=k \wedge j=l \\ (n-2)! & i\neq k \wedge j\neq l \\ 0 & i=k \wedge j\neq l \\ 0 & i\neq k \wedge j=l \end{cases} \quad (9)$$

The assumptions above can be relaxed and the computation of the mean and variance can be generalized to other T-Sets [31]. 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 $k \geq 0$,

$$Pr(X \geq \mu + k\sigma) \leq \frac{1}{1+k^2} \quad (10)$$

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

$$Pr(X \geq c(e)) \leq \frac{1}{1+\left[\frac{c(e)-\mu}{\sigma}\right]^2} \quad (11)$$

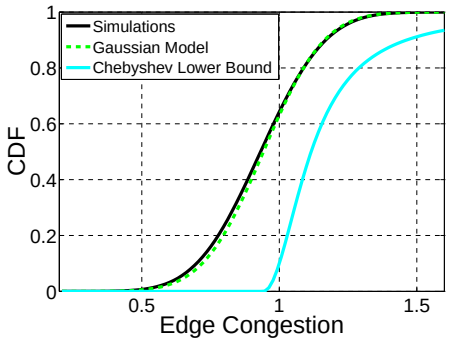
Alternatively, given a desired congestion guarantee level *G*, it is possible to calculate a capacity *c'(e)* that guarantees that at least a fraction *G* of the allowable traffic matrices would be served without saturating *e*. Transforming Equation (11), we get:

$$c'(e) = \mu + \sigma\sqrt{\frac{G}{1-G}} \quad (12)$$

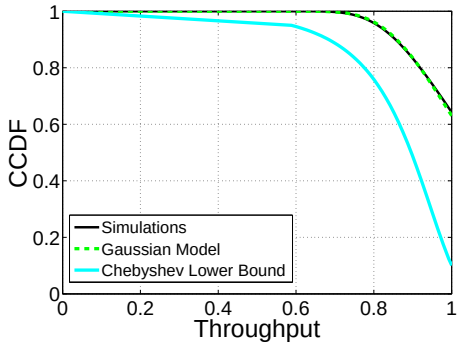
For instance, for $G = 99\%$, we need $c'(e) = \mu + 9.95\sigma$; i.e., with this edge capacity, we are guaranteed that at least 99% of the matrices can be served without saturating e.

Example – Figure 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 Figure 1, using DOR routing. It is a CDF plot, corresponding to the PDF plot of Figure 2.

As seen in Figure 3(a), the Chebyshev-based deterministic congestion guarantees are rather far below the simulated



(a) Edge congestion CDF



(b) Edge throughput CCDF

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

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. For instance, 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 Figure 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*:

$$\begin{aligned} GC_{CDF}(f,L) &= \Pr\left(\max_{e\in E}[EC(e,f,D)] \le L\right) \\ &\approx \prod_{e\in E} \Pr\left(EC(e,f,D) \le L\right) \\ &= \prod_{e\in E} EC_{CDF}(e,f,L) \end{aligned} \quad (13)$$

This edge-independent model is not always a good approximation, because matrices often 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:

$$GC_{CDF}(f,L) \approx \prod_{e\in E} \Phi\left(\frac{L-\mu(e)}{\sigma(e)}\right), \quad (14)$$

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:

$$GC_{CDF}(f,L) \le \min_{e\in E}\{EC_{CDF}(e,f,L)\}. \quad (15)$$

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:

$$\begin{aligned} Pr(GC > x) &\geq Pr(EC(e_1) > x \ \vee \ EC(e_2) > x) \\ &= Pr(EC(e_1) > x) + Pr(EC(e_2) > x) \\ &- Pr(EC(e_1) > \ x \ \wedge EC(e_2) > x) \\ &\geq Pr(EC(e_1) > x) + Pr(EC(e_2) > x) \\ &- Pr(EC(e_1) + EC(e_2) > 2x), \end{aligned} \quad (16)$$

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:

$$\begin{aligned} Pr(GC \leq x) &\leq Pr(EC(e_1) \leq x \ \wedge \ EC(e_2) \leq x) \\ &\leq EC_{CDF}(\hat{e}, 2x) \end{aligned} \quad (17)$$

A stricter global upper bound may finally be defined as the minimum of the three bounds (15), (16) and (17).

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 specific CMP mesh-based architecture, 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 an $m \times m$ mesh with DOR routing. Assume that the T-Set $\mathcal{T}$ is defined such that the processes of core i send traffic to any of the other $m^2 - 1$ 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 $D \in \mathcal{T}$,

$$\forall i \neq j, D_{ij} \sim Uniform\left(\left[0, \frac{1}{m^2 - 1}\right]\right). \quad (18)$$

Consider some edge e in the mesh. Let's denote by s the number of (source, destination) flows crossing e using DOR routing, and further denote the average, standard-deviation and maximum of the flow on edge e by μ, σ and w. By Equation (18), it is clear that the maximum flow generated by each (source, destination) pair is $\frac{1}{m^2-1}$, and therefore $w = \frac{s}{m^2-1}$. Likewise, using independence and summation rules of the expectation and variance, $\mu = \frac{w}{2}$ and $\sigma = \frac{\sqrt{s}}{\sqrt{12}(m^2-1)}$.

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 CMP architecture. While this model is not necessarily correct in all architectures, we will later use it to analyze capacity allocation schemes.

We remind that all proofs are in [31].

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

7.2 Statistical capacity allocation

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

Theorem 4 *For 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 m large enough is sufficient to guarantee full service on edge e for a fraction G of all traffic matrices.*

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:

$$C = \sum_{i=1}^{|E|} c_i = \sum_{i=1}^{|E|} \mu_i + k \sum_{i=1}^{|E|} \sigma_i, \tag{19}$$

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

$$k = \frac{C - \sum_{i=1}^{|E|} \mu_i}{\sum_{i=1}^{|E|} \sigma_i}. \tag{20}$$

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.*

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*. 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. [31] further describes the random walk procedure, and why it should intuitively converge towards the T-Plot.

9.2 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 4(a), 4(b), and 4(c), we analyze several parameters of the global congestion T-Plots for the 3x4 mesh.

First, Figure 4(a) shows the PDF of the global congestion, with the routing algorithm being either DOR or O1TURN [32]. 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). Additionally, both graphs are well fitted to Gaussians. Note that here, contrarily to all other places, we fitted the Gaussian distribution without using *a-priori* models, as we don't have an analytical model for the mean and variance of the *global* congestion. In addition, note that the maximum of many i.i.d. Gaussian random variables does *not* behave as a Gaussian random variable (it follows a Gumbel distribution [33]), and thus one must be careful with the conclusions taken from this plot.

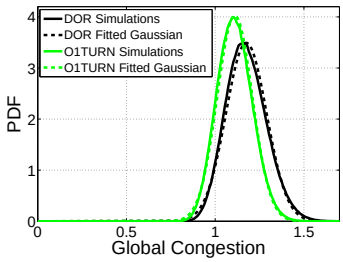
Figure 4(b) shows the CDF of the global congestion in the same network, using DOR routing. The independent-Gaussian model and the upper bound are presented in Section 6. We can see that the independent-Gaussian model assuming independent edge congestions with Gaussian distributions is rather close to 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.

Figure 4(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.

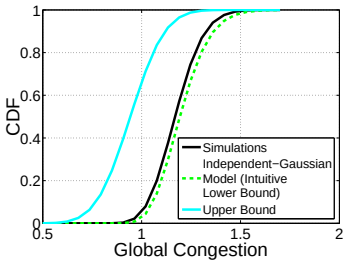
9.3 Capacity allocation algorithms

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*.

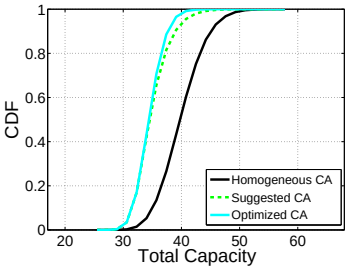
Figure 4(c) illustrates the performance of different capacity allocation (CA) algorithms on the 3×4 mesh network



(a) Global congestion PDF: DOR vs O1TURN



(b) Global congestion CDF: model and bound



(c) Global congestion CDF: CA schemes

Figure 4. Global congestion T-Plots for the 3x4 mesh

with 34 edges (presented in Figure 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 Figure 4(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 out 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 ball-walk algorithm, and is only accepted if it fares better [31]. The optimization was done using 200,000 sample matrices from $\mathcal{A}$, and the results were computed on 200,000 different matrices. We also checked that starting from different points yields the same end result.

9.4 Capacity allocation and throughput guarantee

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

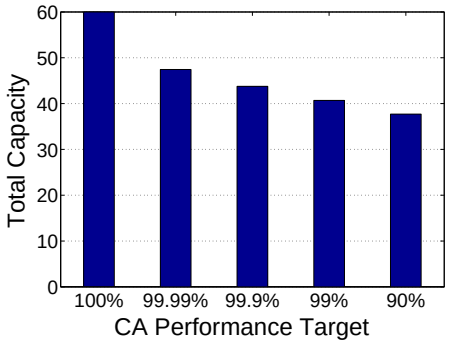
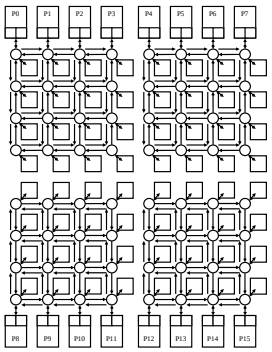


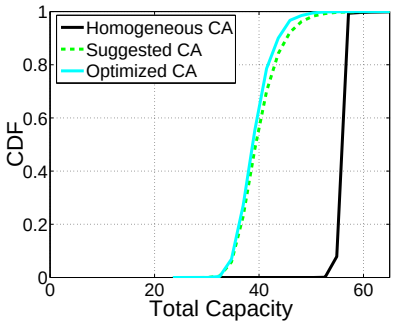
Figure 5. Total capacity required for various CA targets

Figure 5 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, 21, 22]. 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 Figure 4(c). For instance, for $G = 99.9\%$, the amount of provisioning needed is $CDF^{-1}(0.999) = 43.8$.

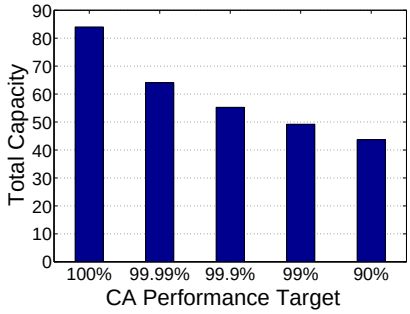
Figure 5 shows that switching from a worst-case to a statistical CA approach may save up to 37% of the total required capacity in this network, for a capacity guarantee at a 90% level. Likewise, planning for a very stringent 99.99% cutoff decreases the amount of total capacity used by 21%. As an aside, note that we didn't even compare with the naive *homogeneous* worst-case approach – such a comparison would have yielded even greater savings!



(a) NUCA architecture (based on [34] with sharing degree 4)



(b) Global congestion CDF: CA schemes



(c) Total capacity required for various CA targets

Figure 7. NUCA network: topology and performance

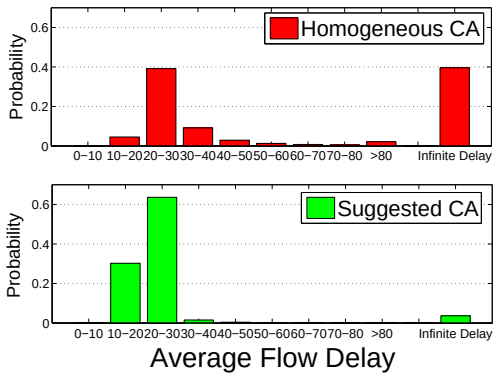


Figure 6. Average flow delay distribution over all traffic matrices, for two different CA schemes

9.5 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.

Figure 6 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.

9.6 NUCA network

Finally, we considered a different CMP architecture model based on a NUCA (non-uniform cache architecture) network. As shown in Figure 7(a) (based on [34] 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.

Figure 7(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, Figure 7(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.

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.

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

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

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

[24] 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.

[25] 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.

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

[27] 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.

[28] 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.

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

[30] 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.

[31] I. Cohen, O. Rottenstreich, and Isaac Keslassy, "Statistical approach to NoC design (extended version)," *Technical Report TR08-01*, Comnet, Technion, Israel.

[32] 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.

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

[34] 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.