# Characterization and Modeling of Multicast Communication in Cache-Coherent Manycore Processors

Sergi Abadal[a], Raúl Martínez[b,1], Josep Solé-Pareta[a], Eduard Alarcón[a], Albert Cabellos-Aparicio[a]

[a] *NaNoNetworking Center in Catalonia (N3Cat), Universitat Politècnica de Catalunya, Barcelona, Spain*
[b] *Oracle Labs, Oracle Corporation, Vancouver, BC, Canada*

**Abstract**

The scalability of Network-on-Chip (NoC) designs has become a rising concern as we enter the manycore era. Multicast support represents a particular yet relevant case within this context, mainly due to the poor performance of NoCs in the presence of this type of traffic. Multicast techniques are typically evaluated using synthetic traffic or within a full system, which is either simplistic or costly, given the lack of realistic traffic models that distinguish between unicast and multicast flows. To bridge this gap, this paper presents a trace-based multicast traffic characterization, which explores the scaling trends of aspects such as the multicast intensity or the spatiotemporal injection distribution for different coherence schemes. This analysis is the basis upon which the concept of *multicast source prediction* is proposed, and upon which a multicast traffic model is built. Both aspects pave the way for the development and accurate evaluation of advanced NoCs in the context of manycore computing.

*Keywords:* Manycore Processors; Multicast; Broadcast; On-Chip Traffic Analysis; Network-on-Chip; Scalability;

## 1. Introduction

In the ever-changing world of microprocessor design, multicore architectures are currently the dominant trend for both conventional and high-performance computing. Chip Multiprocessors (CMPs) resulting from the interconnection of several processing cores were conceived to overcome the complexity and power scalability hurdles of processors with a single CPU; however, the scalability con-

*Email address:* abadal@ac.upc.edu (Sergi Abadal)

[1] Raúl Martínez was working for INTEL at the INTEL Barcelona Research Center when the main ideas of the paper were developed.

cerns have now migrated to facets such as memory management, programmability or the limits of parallelism as the core count increases.

Inherent parallelism limits aside, these scalability concerns are generally dependent on the architecture or programming model of choice. A long-running debate has brought up strong arguments for the adoption of two widely-known models in manycore CMPs: shared memory and message passing. Shared memory provides remarkable programmability and compatibility with legacy code. However, its scalability is arguably limited by performance and architectural complexity issues related to data consistency. On the contrary, message passing offers unique validation and hand-tuned performance benefits, which come at the cost of placing an increasingly heavy burden upon the programmer. The differences between these two extremes contrast with one common point: most of the scalability issues are tightly coupled to on-chip communication limitations. Due to this, the research focus in multiprocessors has gradually shifted from how cores compute to how cores communicate.

The limited scalability of conventional Networks-on-Chip (NoCs) in the presence of global and multicast traffic represents an important constraint to multiprocessor architects and programmers. In shared-memory multiprocessors, cache coherence is the main source of on-chip communication and is generally maintained through directory-based protocols that limit the use of multicast to the invalidation of cache blocks on a shared write. This reduces the injection of multicast traffic, yet at the cost of non-scalable area and energy overheads required to track the sharers of the data. In message passing systems, where communication is explicitly set by the programmer, the use of collective communication routines such as `MPI_Bcast` or `MPI_Allgather` is often avoided. This, however, may lower the maximum achievable performance and increase the complexity of parallel programming over message passing even further. Also, the lack of proper multicast support hampers the development of novel programming models and manycore systems that may be multicast-intensive [1].

Given that multicast and broadcast may become a critical factor guiding the design of future manycore CMPs, there is a need to understand how the characteristics of such traffic will scale with the number of cores. Providing accurate multicast traffic characterization in different scenarios would be useful for the early-stage design and evaluation of NoCs in general and multicast mechanisms in particular. However, to the best of the authors knowledge, no tools are available for the analysis and modeling of multicast traffic. Different works have characterized or modeled inter-processor communication in moderately-sized shared memory processors (see [2] and references therein), as well as in message passing clusters or supercomputers [3, 4, 5]. However, none of them has analyzed how collective communication scales for a wide set of representative applications and architectures at the CMP scale. Also, existing traffic models do not differentiate between unicast and multicast flows and only offer data for a given system size.

In this paper, we aim to address these issues in one of the contending programming models (shared memory) by performing a scalability-oriented multicast traffic characterization. Our contribution and methodology is summarized

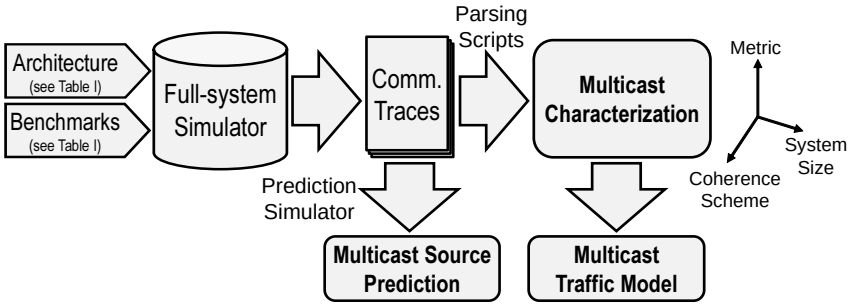



Figure 1: Simulation-based multicast traffic characterization and modeling.

in Figure 1. We first analyze traces of widely known SPLASH-2 and PARSEC benchmark applications running over a set of existing cache coherence alternatives and for different processor sizes. Since scarce data is available beyond 64 cores, we additionally perform an exploratory extension of the study up to 512 cores whenever possible. Although most of these applications were not build to scale to thousands of processors [6, 7], the analysis of its multicast traffic patterns may be still useful to extrapolate the behavior of future scalable shared-memory applications. We later employ the results of the characterization process to (1) propose and evaluate the potential of *multicast source prediction*, and (2) create a synthetic traffic generator that faithfully models multicast communication and that can be used to generate realistic mixed traffic profiles. With this, we aim to trigger further research in the area of multicast support for manycore systems.

The paper is a direct extension of previous work by the authors [2]. The original contribution is augmented as follows:

- The analysis now includes results regarding Token Coherence [9], as well as directory-based coherence with imprecise tracking up to 512-core CMPs.
- The concept of *multicast source prediction* is presented and evaluated, placing emphasis on its potential applicability at the NoC design level.
- The characterization results are used to implement a multicast traffic model and propose an appropriate synthetic traffic generator.

The remainder of this paper is as follows. In Section 2, we provide background on cache coherence and on-chip networking which may further motivate this work and be useful to understand its results. Within the same section, we present related work in NoC traffic analysis and modeling. Section 3 details the characterization methodology used to obtain multicast communication traces and to analyze them. The results of the multicast traffic characterization are presented in Section 4, which are later used in Section 6 to create and validate a multicast traffic model. Section 7 concludes the paper.

## 2. Background and Related Work

This section seeks to further motivate this work by explaining why it is necessary to analyze unicast and multicast traffic separately (Sec. 2.1); by providing background on the relation between cache coherence and the characteristics of the multicast traffic, in an attempt to both explain the importance of multicast in shared-memory manycore chips and justify the choice of protocols for analysis (Sec. 2.2); and by detailing in which aspects this paper differs from related work in multicast traffic analysis and modeling (Sec. 2.3).

### *2.1. Serving Multicast Traffic in NoCs*

The nature of on-chip interconnects has changed with the number of processors. Buses are feasible for a few cores and, therefore, all communications are inherently multicast (broadcast, in fact). As more cores are integrated within a single chip, though, the interconnect design shifts to the NoC paradigm which is point-to-point in nature. Due to this, NoCs require that multicast packets be replicated and delivered to each of the intended destinations. When dense multicasts and broadcasts go through this process, the performance of the network may suffer a severe drop due to both the delay incurred by the packet replication and the contention associated to these additional messages. The performance loss is generally proportional to the multicast intensity and to the number of destinations per message for a fixed network size.

The way packets are treated in conventional NoC has been subject of different studies. The most simple approach is to generate and inject one unicast message per destination. This process is performed at the source network interface (NIF) and, besides being highly power-inefficient, implies a potentially large serialization delay and increases contention around the transmitting node. To alleviate these effects, two in-network multicast support techniques have been explored. On the one hand, *path-based multicast* [10] relies on the transmission of a number of messages which travel around distinct regions of the chip and are replicated and delivered at the NIF of each of the destinations. On the other hand, *tree-based multicast* [11] requires the injection of a single message, which is replicated at the intermediate routers following a virtual tree until it reaches the intended destinations. In general, path-based methods are more contention-aware, whereas tree-based methods incur into lower latency.

Another way to reduce the impact of multicast communications is to improve the connectivity of the network or by using globally shared media. This can be done with traditional RC wires and buses [13] or with emerging interconnect technologies such as 3D stacking [10], nanophotonics [14] or wireless RF [1].

Regardless of the approach taken, multicast support proposals been tested either using synthetic traffic or within full-system simulations, generally assuming a fixed network size. Consequently, their impact upon the network performance is imprecise and their scalability remains largely unknown. In this paper, we aim to set the foundations of effective NoC testing for different network sizes and for a set of representative and realistic cases.

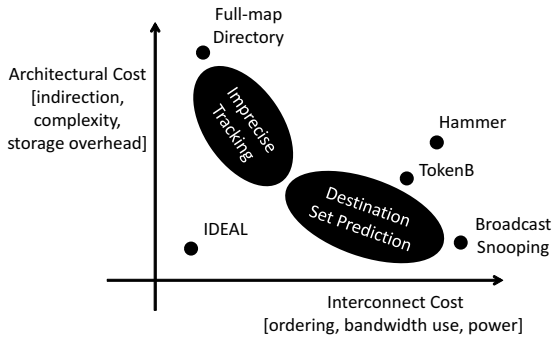



Figure 2: Representation of the cache coherence design space, inspired by the work in [15].

### *2.2. Cache Coherency and Multicast Traffic*

The modest performance of conventional NoCs in the presence of dense multicast and broadcast communication has guided the design of shared memory multiprocessors through the years, encouraging the adoption of memory architectures and cache coherency protocols that avoid such type of traffic. As a result, traditional snoopy coherence schemes gave way to directory-based coherence. The former requires an ordered broadcast mechanism; whereas the latter relies in an entity (the directory), which serves as an ordering point and coordinates, through point-to-point messages, the memory transactions among the involved processors. As shown in Figure 2, these represent the two extremes of cache coherence which trade off architectural cost against interconnect cost.

In directory-based coherence, the use of multicast is generally limited to the invalidation of cache blocks on a shared write. To send invalidation messages only to the sharers of that block, the directory needs to provide precise sharer tracking. This implies having one bit per core per each cache block to store that information. Obviously, this scheme does not scale well as storage requirements skyrocket for hundreds of cores. To relax these constraints, one can limit the number of tracked sharers and send broadcasts to invalidate heavily shared blocks. An extreme case of imprecise tracking could be the protocol implemented by AMD HyperTransport [16], which is stateless and directly broadcasts all requests. However, broadcast delivery does not need to be ordered as in snoopy protocols, and responses are still unicast. Other alternatives such as Token Coherence [9] or Destination Set Prediction [15] propose advance techniques to further limit their architectural or communication costs.

As mentioned above, the adoption of multicast-demanding methods is hampered by the performance of NoCs and their lack of ordering. Performance penalties are significant already at 64 cores and are expected to dramatically increase with the core count [11], progressively cornering the coherence solution space far from the ideal area. This further motivates the pursue of improved multicast and broadcast support in NoCs, indirectly highlighting the importance of this work as a vehicle for the evaluation of future interconnect fabrics

in realistic conditions.

*2.3. Related Work in Traffic Characterization and Modeling*

The driving motivation behind traffic characterization is the need for a cost-effective (more than using real traces) but accurate (more than using synthetic traffic) way to evaluate networks. In NoCs, traffic characterization can be performed by analyzing communication traces obtained from full-system simulations. Multiprocessor benchmarks such as SPLASH-2 [6] and PARSEC [7] are commonly used to serve this purpose in shared memory architectures. Other benchmark suites such as NAS could be employed in message passing systems more oriented to supercomputing [3].

***Traffic Characterization: Shared Memory -*** on-chip traffic generated by shared-memory architectures has been analyzed in a wide variety of settings. Our previous work in [2] contains a comprehensive view of such characterization efforts, including the seminal papers that explore the SPLASH-2 and PARSEC benchmarks [6, 7, 8]. Their main downturn, though, is that those works do not distinguish between unicast and multicast in most cases. In fact, the few existing explicit multicast analyses are thus far limited to very specific use cases. Here, instead, we perform a complete characterization of multicast traffic for different representative cases using a unified approach and, for the first time, up to 512 cores to inspect their scalability.

***Traffic Characterization: Message Passing -*** In message passing, simpler steps can be taken to analyze traffic due to the explicit nature of communication. One can obtain a traffic characterization by looking into the types of communication routines used within a given program and projecting them into a given target system. The communication routines can be point-to-point (e.g. `MPI_Send`) or collective (e.g. `MPI_Allgather`), which may directly involve multicast and broadcast patterns. By breaking down these operations into messages and aggregating all contributions, metrics such as the multicast intensity or the number of destinations per message could be obtained. In the literature, several works have analyzed the communication primitives of different parallel algorithms. In [4], collective communication patterns and their impact on scalability are analyzed. The communication characteristics of NAS benchmarks and other scientific applications have been also extensively evaluated in [3, 5] including information on collective routines.

***Traffic Modeling -*** As mentioned above, traffic analysis also enables the faithful yet simple modeling of traffic for NoC evaluation. First proposals in this regard consider that three dimensions are enough to model the injection of traffic in NoCs of different benchmarks and architectures [17]: the degree of *temporal burstiness* resulting from the widely proven self-similarity of NoC traffic, the degree of concentration in the *spatial injection distribution*, and the *hop distribution* that models the probability of a packet going through a given number of hops as a function of the NoC topology or how a given application is mapped onto the processor.

When modeling traffic using such methods, no distinction is made between unicast and multicast flows, and a unified model is used. This may be acceptable

from a behavioral perspective only if the NoC treats each multicast as a set of independent unicast packets. In contrast, both types of traffic may have to be modeled separately and then interleaved in path-based or tree-based multicast schemes, since the network behavior changes with the type of packet. Such heterogeneous approach has been adopted in works like [10], which evaluates a path-based multicast routing method using mixed traffic profiles. Our work builds upon this premise, sustaining that the same model should not be used for both profiles as they may have fundamentally different sources and, therefore, different characteristics. Actually, *we aim to improve the quality of existing models by focusing on the less-explored area of multicast characterization.*

An alternative traffic modeling methodology based upon full-system simulations is presented in [18]. Instead of relying blindly on the temporal and spatial information given by the traces, their approach uses additional information on the type of communication (e.g. purpose) to establish dependencies between messages and to determine its typology. Therefore, this method contemplates the possibility of distinguishing between multicast and the rest of traffic.

## 3. Methodology

The main objective of this work is to perform a multicast traffic analysis targeting NoC scalability. To this end, we employ the methodology summarized in Figure 1. Basically, we simulate different architectures running different benchmarks in order to obtain a set of communication traces. These traces are then parsed with the aim of extracting a set of statistics, which can further processed and graphically visualized. Such characterization can be later used to model realistic multicast traffic (see Section 6), allowing to evaluate multicast routing schemes in practical scenarios without having to resort again to full-system simulation.

It is worth noting that, even though the methodology is used here to study shared-memory systems, a similar workflow could be used to obtain the multicast traffic characteristics of message-passing applications provided that the appropriate tools are available.

### *3.1. Simulation Tools*

To capture traffic in a cache-coherent processor, a simulation of the whole memory hierarchy is required, since L1-L2 interactions are the main determinants of this traffic. Unlike in our previous work, we use two different tools to obtain the communication traces. On the one hand, we use GEM5, a widely used open-source framework for cycle-accurate full-system simulation [19]. Due to its depth and accuracy, GEM5 admits up to 64 cores with up to three levels of cache and, out of the box, includes different cache coherency schemes such as MESI, HyperTransport (HT) or TokenB (an implementation of Token Coherence). On the other hand, we used Graphite [12], an architectural simulator developed by MIT that employs more relaxed functional and performance models, allowing it to simulate large systems (hundreds and even thousands of

cores) with reasonable accuracy. Currently, Graphite models MSI coherence with a variety of directory types.

In order to obtain custom statistics and traces oriented to multicast traffic analysis, we slightly modified both simulators. In GEM5, the NIF modules now capture the time of arrival, original, destinations, type and size of each multicast packet that is to be injected to the NoC. This way, the results are independent to the multicast routing strategy. NIFs are not modeled as such in Graphite; therefore, we identify points where multicast packets are generated (i.e. invalidations) and register relevant information for each of them. Note that, due to differences in terms of simulated instruction set and internal models, we will not directly compare the output of both schemes. Instead, we will use the results from Graphite to complement the scalability analysis performed with GEM5. The main aim is to provide trends rather than exact figures beyond 64 cores; and, for that purpose, Graphite is an adequate tool despite the loss of accuracy with respect to GEM5.

### *3.2. Architecture Under Study*

Table 1 shows a summary of the simulation parameters used in the study. We assume a common tiled configuration, wherein each tile comprises a core, private instruction and data caches, and a slice of a shared L2 cache. The main variables of the study are *the multiprocessor size*, which ranges from 4 to 64 tiles in GEM5 and from 16 to 512 tiles in Graphite, and *the coherency mechanism*. We consider three main different protocols, namely, MSI/MESI, HT and TokenB mentioned in Section 2.2. In the first case, we take the number of maximum sharers per block as a variable in order to study the impact of imprecise tracking on the number of destinations per multicast.

From the network perspective, it is important to note that we consider an ideal fully connected NoC where the delay introduced by the network is fixed and independent on the source and destination(s). This approach has been proposed in [18] to minimize the impact of a specific topology or routing algorithm upon the traffic characteristics, which essentially depend on the cache coherence scheme. The resulting characterization is a base upon which models can be later obtained for a wide range of NoC configurations. Additionally, this reduces the complexity of the large-scale simulations and the impact of inaccurate network models in Graphite.

PARSEC and SPLASH-2 benchmarks are simulated on their entirety whenever possible, always using input sets large enough to scale the workload up to hundreds of cores. In all cases, statistics and traces are only collected within the region of interest, and results are only shown for those applications that were successfully executed both in GEM5 and Graphite.

## 4. Characterization Results

Next, we present the results of the multicast traffic analysis and discuss the possible implications of each explored characteristic on the NoC design.

Table 1: Simulation Parameters

| Parameter | GEM5 | Graphite |
|---|---|---|
| Number of cores | 4 - 64 | 16 - 512 |
| Coherency Protocols | MESI, HT, TokenB | MSI |
| Max Sharers Tracked | 2 - 64 | |
| Benchmarks | PARSEC, SPLASH-2 | |
| L1 Cache (I&D) | 32+32KB, 2-way, 2 cycles, 64B lines | |
| L2 Cache | 512KB/core, 8-way, 10 cycles, 64B lines | |
| NoC Topology | Fully Connected | |

### *4.1. Message Type and Size*

The architecture of a multiprocessor is the main factor that determines the methods that will trigger the transmission of multicasts. Therefore, the size of these messages can be easily inferred from it and, as further explained in [2], taken into account in the NoC design process.

In the MESI coherence protocol, multicast messages are mostly invalidations which are generated upon a write to shared data and sent to the cores that are currently sharing it. Invalidations are short control messages and account for more than 99% of the multicasts in average regardless of the system size (rarely dropping below 96%). In the rest of cases, multicasts are long data responses sent to the main memory and to a set of caches after reading an invalidated block. These replies are less much frequent and their size corresponds to the cache line size plus a given overhead. In the HT protocol, the percentage of multicast long messages is expected be even lower since not only invalidations, but all control requests, are broadcast. Actual results, however, show a similar size distribution than for MESI. Finally, in the case of TokenB, short requests represent the totality of multicast messages. These are used to collect the tokens required to perform a shared write, to request a value from a remote cache, or to avoid starvation situations by means of persistent requests.

### *4.2. Multicast Traffic Intensity*

The number of multicast messages per instruction is a measure of the multicast intensity. It is loosely dependent on the choice of NoC, and basically determined by the multiprocessor architecture, as it defines the methods that generate multicast messages; and the application, which defines the sharing structures and memory intensity.

Figure 3 plots the number of injected multicast messages per one million instructions. It is observed that applications generally become more multicast intensive as the number of cores grows: note the steep increases of particular cases such as *lu*. In comparative terms, TokenB shows the largest multicast requirements, around one order of magnitude above the requirements of HT coherence and two orders of magnitude larger than MESI. Although such increase is application-dependent and does not follow a common scaling trend, fitting methods on the average values yield a somehow logarithmic relation between


Figure 3: Number of multicast per $10^6$ instructions with MESI, HT and TokenB protocols (GEM5 traces) and the MSI protocol (Graphite traces) as a function of the processor size. Note the logarithmic scale.

multicast intensity and number of cores. The scalability of the communication-to-computation ratio, which is generally a function of the square root of the number of processors [6], may explain such tendency.

To provide hints on the evolution of this metric beyond 64 cores in directory-based settings, we used Graphite to obtain the number of multicast invalidations per instruction in MSI. The bottom plot of Figure 3 shows that such metric keeps increasing beyond 64 cores and, even though results are not directly comparable with those obtained with GEM5 due to differences in the instruction set and protocol, this confirms the increasing importance of multicast in manycore processors.

### *4.3. Number of Destinations*

The number of destinations per message is a metric that mainly depends upon the multiprocessor architecture. In cases such as HT or TokenB, the

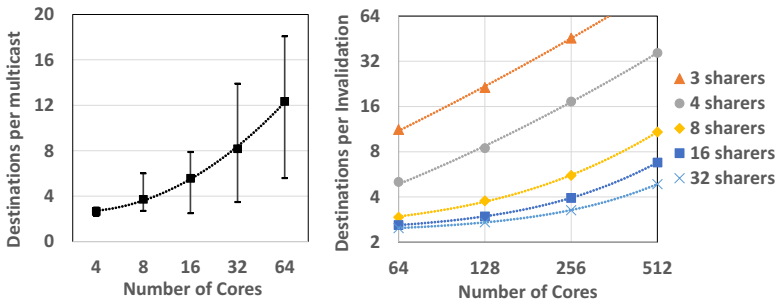


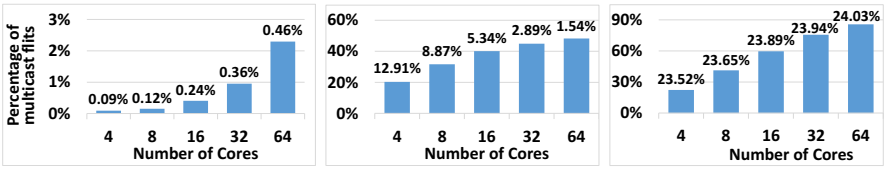

Figure 4: Number of destinations per multicast message in directory-based coherence. Left plot shows the minimum, average, and maximum of the per-application average assuming precise tracking up to 64 cores, whereas the right plot projects the number of sharers per invalidation with imprecise tracking beyond 64 cores.

coherence protocol issues a broadcast in most of the transactions. Given that a very large percentage of the multicast messages are due to coherence, the average number of destinations is around $N-1$ in a $N$-core system in those cases and regardless of the target application.

In invalidation-based protocols such as MSI/MESI, the sharing structures of each specific application play an important role in determining the number of destinations per multicast message, as it defines the number of sharers to invalidate in each shared write. Figure 4(a) represents the number of destinations per multicast message assuming perfect tracking. It can be observed that the number of destinations increases linearly with the system size, and that some 64-threaded applications involve more than 16 destinations per multicast in average.

Another aspect that defines the number of destinations is the maximum number of sharers that the directory can precisely track down. To evaluate this and to have a reference on how multicast traffic will scale beyond 64 cores, we used Graphite to obtain the number sharers per invalidation assuming different directory capacities. Results are plotted in Figure 4(b) and show that the number of destinations not only keeps increasing with the system size, but also significantly grows as the number of tracked sharers per cache block decrease. This is because an increasing number of shared variables will have to be invalidated through broadcast.

The importance of this metric lies within its impact on the overall bandwidth requirements. While the number of multicast transactions increases with the number of cores, they only represent a fraction of all the communication transactions (see labels in Figure 5). Less than 0.5% of all the transactions are multicast in MESI; in HT, the ratio is higher but it decreases sharply with the number of cores (from 12% to 1.5%); finally, broadcasts increase up to around 24% in TokenB. In contrast, as shown in Figure 5, the amount of delivered (ejected) flits belonging to multicast transactions consistently increases due to the in-network replication of each multicast message. This metric grows above




(a) MESI (b) HT (c) TokenB

Figure 5: Percentage of delivered flits due to multicast communication. Labels indicate the percentage of communication transactions that are multicast.

2% in MESI with perfect tracking, and would exceed 4% and 7% with 8 and 3 maximum sharers. In multicast-intensive schemes, the percentage of ejected flits due to multicast dramatically increases up to 50% in HT and goes beyond 85% in TokenB. This fact further encourages the employment of shared-medium NoCs, if feasible, to efficiently serve multicast traffic.

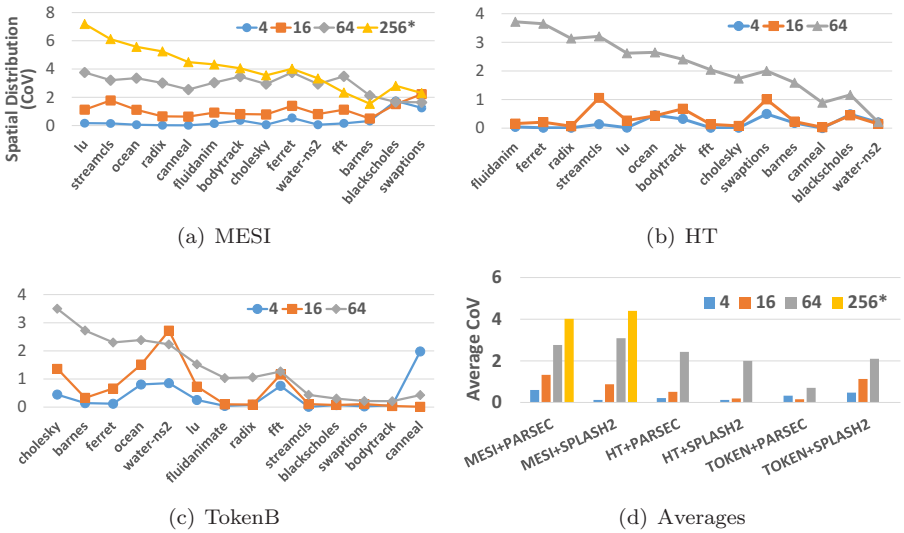
### *4.4. Spatial Distribution*

An interesting aspect to investigate is the spatial distribution of the multicast traffic injection. Results in this regard may be useful for the identification of potential hotspots and could be employed to optimize the underlying NoC by, for instance, applying priority policies. To evaluate the spatial distribution, we calculated the coefficient of variation (CoV) as $c_v = \sigma/\mu$, where $\sigma$ and $\mu$ are the standard deviation and mean of the multicasts injected by each node. We chose this metric in order to measure dispersion while filtering out the dependence of the standard deviation with the overall number of injected messages. A higher CoV means a higher concentration of the multicast injection over given cores.

Figure 6 plots the CoV of each application in the target systems, as well as the average over all the applications. The CoV grows steadily with the number of cores in an application-dependent manner: results have been in fact sorted in descending order based on the absolute growth of the imbalance. This implies that applications that appear first may yield more pronounced imbalance in manycore processors and would especially benefit from NoC designs that efficiently handle hotspot situations. From the average behavior, it is observed that MESI shows a higher imbalance in general terms. This is because applications heavily based on producer-consumer patterns rely on a few producers, which are the main sources of multicast traffic in MESI. The number of producers does not necessarily scale with the system size, aggravating the hotspot behavior. Such assumptions are not valid in HT or TokenB, since broadcast requests come from a wider base of cores. In such cases, the spatial distribution provides insight about the general memory activity of different processors: cores that frequently access to shared data will generate more broadcasts than those that do not.

### *4.5. Temporal Distribution*

In order to accurately model any kind of traffic, it is crucial to have a complete knowledge on its temporal distribution. As shown in Section 2, related




(a) MESI (b) HT

(c) TokenB (d) Averages

Figure 6: Coefficient of Variation (CoV) of the spatial injection distribution of multicasts. Applications are sorted in descending order based on the difference $\Delta$ between CoV(256)/CoV(64) and CoV(4).

works have shown that on-chip traffic is self-similar given the long-range dependency between arrivals, i.e. the generation of new messages is dependent on the delivery of prior messages. This creates memory effects and implies a given burstiness at the transmitting end, property that is widely known to have a negative impact on network performance. Provided that multicast traffic is a subset of the on-chip traffic, it is reasonable to deduce that multicasts will also exhibit self-similarity albeit not necessarily with the same intensity. In order to confirm this fact, we calculated the Hurst exponent $H$ ($0.5 < H \leq 1$) applying the RS plot method [17] on the temporal information of the full-system traces.

In light of the results of Table 2 and given that an $H$ value close to 1 denotes strong self-similarity, it can be concluded that multicast traffic is self-similar and that burstiness generally increases with the core count. Also, note that the NoC has an impact upon the value of $H$, although similar results are expected for most NoC implementations since burstiness stems from memory effects inherent to the application level. For instance, our previous work assumes a mesh NoC and yields slightly lower Hurst exponents in almost all cases. We refer the reader to [2] for more details.

### *4.6. Spatiotemporal Correlation*

Spatial and temporal analyses performed above yield two independent characterizations of the injection process: first, on the generic probability of any node transmitting and, second, on the probability of any node transmitting shortly after any other node. The potential correlation between both aspects could provide further insight on, for instance, how easy is to determine that a

Table 2: Hurst Exponents (geometric mean)

| Number of Cores | 4 | 16 | 64 |
|---|---|---|---|
| MESI | 0.8955 | 0.9152 | 0.9433 |
| HT | 0.9140 | 0.9254 | 0.9450 |
| TokenB | 0.8350 | 0.8788 | 0.9456 |

given node X will transmit shortly after a transmission of another given node Y. While correlation does not necessarily imply causality between both transmissions (the message from Y may not be triggered by the message from X), it is a highly valuable information when designing predictive strategies for NoC. We refer the reader to Section 5 for more details on NoC-related prediction.

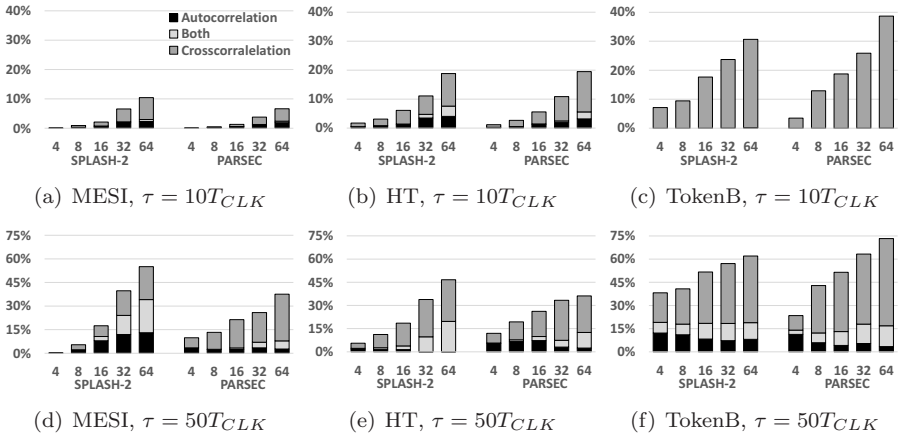
To evaluate spatiotemporal correlation, we consider transmissions separated less than a given time period $\tau$. If $X = Y$, we found a potential source of *autocorrelation*; whereas if $X \neq Y$, we are facing a case of *crosscorrelation*. The choice of $\tau$ depends on several factors as further explained in [2]. Two interesting correlation metrics can be obtained from this analysis.

First, we evaluate the *degree of correlation* of multicast transmissions. We obtain these values by marking the second transmitter of a correlated pair and, at the end of the execution, counting the number of marked transmissions. In MESI, results beyond 64 cores are approximate and given as a scalability reference. Figure 7 shows the correlation distribution of multicast transmissions assuming two different $\tau$ values. It is observed that the percentage of crosscorrelated transmissions grows with the number of cores, especially in the case of MESI and TokenB, and that autocorrelation levels are generally low. Note that these figures represent the geometric mean of all the applications and, therefore, the correlation percentages may take higher/lower values: *radix* and *canneal*, for instance, show crosscorrelation levels above the average; *bodytrack* yields lower correlation. Since a high percentage of correlated traffic could imply not only subpar NoC performance, but also a great opportunity to improve it by means of predictive strategies, results suggest that such mechanisms will gain importance as the number of cores increases. Finally, it is observed that $\tau$ impacts upon the percentage of correlated transmissions, but not much upon its scalability with respect to multiprocessor size.

Another interesting aspect to investigate is the *strength of the crosscorrelation* between any two pairs of nodes, in an attempt to quantify the predictability of the source of correlated transmissions. To this end, we define the predictability of each node X as:

$$pred_X = \frac{\max_i N_{Xi}}{\sum_i N_{Xi}} \quad i \neq X, \tag{1}$$

where $N_{XY}$ is the number of transmissions of Y correlated to X. This metric captures how predictable are the transmissions that happen shortly after a transmission by X, since a low value means that crosscorrelation is spread over a set of cores, therefore complicating the prediction (0 if transmissions are not




(a) MESI, $\tau = 10T_{CLK}$ (b) HT, $\tau = 10T_{CLK}$ (c) TokenB, $\tau = 10T_{CLK}$

(d) MESI, $\tau = 50T_{CLK}$ (e) HT, $\tau = 50T_{CLK}$ (f) TokenB, $\tau = 50T_{CLK}$

Figure 7: Degree of correlation of multicast transmissions with different $\tau$ values.

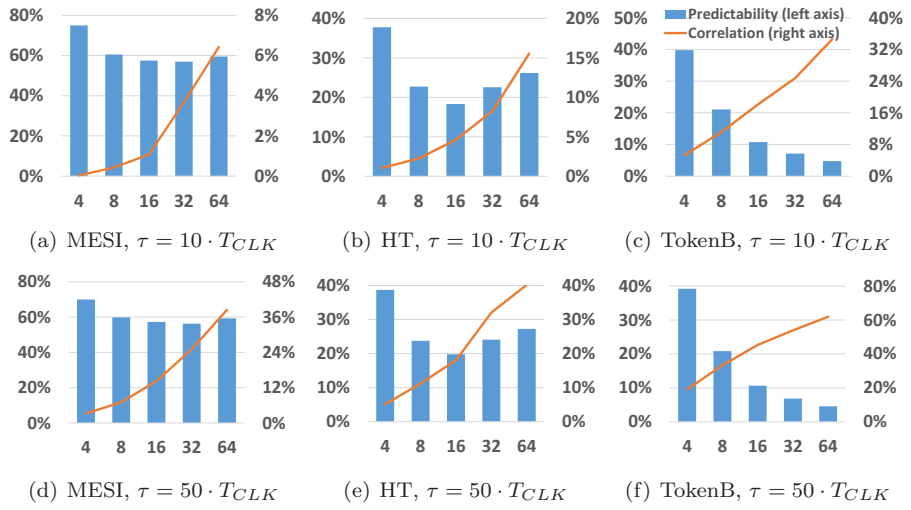
correlated). A high value indicates a strong correlation with few cores (1 if correlation is deterministic). The factor of predictability between any two pairs is calculated as the weighted average of the predictability of each core:

$$pred = \frac{\sum_{i \neq X} N_{Xi}}{\sum_i \sum_{j \neq i} N_{ij}} pred_i = \frac{\sum_i \max_{j \neq i} N_{ij}}{\sum_i \sum_{j \neq i} N_{ij}}. \quad (2)$$

Figure 8 shows the factors of predictability and correlation assuming two different $\tau$ values. Results beyond 64 cores are approximate and given as a scalability reference. It is observed that both the crosscorrelation and predictability levels increase with the number of cores in MESI and HT. The results in these cases support the hypothesis that predictive strategies have more potential in larger multiprocessors. The predictability is much higher for MESI, probably due to the clear dependence of multicast traffic with potentially predictable memory sharing patterns. In HT, the sources of multicast traffic are more varied and the use of more sophisticated predictors would be required. In TokenB, the predictability is inversely proportional to the number of cores, which suggests that the injection of multicast traffic behaves as a uniformly distributed random variable and that predictive techniques would be useless in this case. Finally, it is worth noting that increasing the observation window hardly affects the overall predictability. This could assist in the evaluation of the scope of a predictor, i.e. which events to predict.

### *4.7. Phase Behavior*

Apart from investigating self-similarity, trace-based analysis may allow the study of the different phases found in parallel applications. Phase changes affect a wide variety of metrics, including communication intensity, and recent literature demonstrates that such phase behavior is predictable [20]. Multicast communication is likely to be also influenced by the existence of phases within




(a) MESI, $\tau = 10 \cdot T_{CLK}$ (b) HT, $\tau = 10 \cdot T_{CLK}$ (c) TokenB, $\tau = 10 \cdot T_{CLK}$

(d) MESI, $\tau = 50 \cdot T_{CLK}$ (e) HT, $\tau = 50 \cdot T_{CLK}$ (f) TokenB, $\tau = 50 \cdot T_{CLK}$

Figure 8: Factor of predictability and percentage of crosscorrelation with different $\tau$ values.

an application, and, therefore, the metrics presented above could be evaluated on a per-phase basis. Our previous work [2] exemplifies this fact by showing how different multicast traffic metrics change with application phases. The results therein suggest that (1) reconfigurable multicast techniques could be of use in the on-chip scenario, and that (2) multicast prediction could improve if assisted by phase tracking techniques. Notwithstanding this, phase behavior is not taken into consideration in the remainder of this article and will be explored in future work instead.

## 5. Multicast Source Prediction

In computer architecture, prediction has been pervasively used as a tool to improve performance. The outcome of a conditional branch, the value of certain variables in memory, the sharer set of certain cache lines, or the load at a given NoC link are aspects that may show correlation in different situations due to, among other factors, the iterative nature of computer programs. Predictive systems exploit such information to optimize the processor pipeline [21], the coherence protocol [15] or the routing mechanism of the CMP [22].

In Section 4.6, we showed that multicast traffic in cache-coherent processors is highly correlated and potentially predictable. In particular, one can guess which will be the source of the next multicast message, information that could be exploited by reconfigurable NoC designs. Here, we evaluate the accuracy of two simple predictors by running the different sets of traces over a simulated environment. Then, we qualitatively discuss how NoC design could benefit from multicast source prediction.

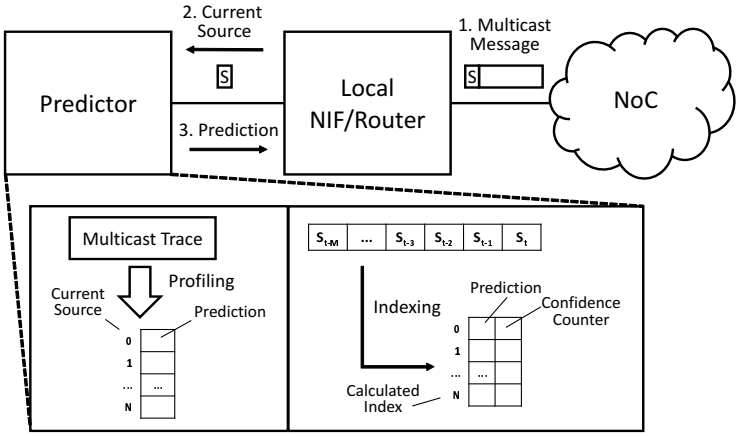



Figure 9: Multicast source prediction scheme, with the detail of a static predictor (left box) and a last value predictor (right box).

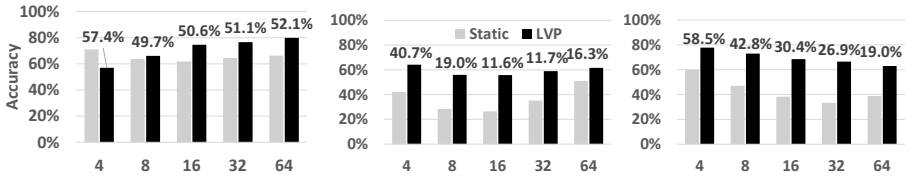
### *5.1. Implementation*

Figure 9 shows an abstract representation of the basic architecture of a NoC-juxtaposed predictive system. Basically, the predictor is attached to a local NoC component, which can be a NIF or its associated router. The communication between the NoC and the predictor is two-way: the predictor reads the packets that go through the local component to take a guess on future events and to validate previous predictions, whereas the local component reads the predictions and modifies its operation depending on a given policy.

In the case of multicast source prediction, the predictor extracts the source of each multicast packet and uses this information to guess who will be the source of the next multicast. Both the header and the prediction are kept by the predictor during a pre-defined amount of time (i.e. observation window) and then discarded. If the next multicast transmission occurs before the information is discarded, the predictor may update its table with the new source. This way, predictions over distant and potentially non-correlated events are avoided. The choice of the length of the observation window will depend on how the impact of consecutive multicasts behaves over time.

The specific implementation of the predictor will depend on the nature of the multicast correlation and on the required level of accuracy to obtain acceptable speedups. Here, we evaluate two simple and widely known predictors as depicted in Figure 9:

***Static Predictor (SP):*** This approach basically consists of the off-line profiling of code or traces that represent a given application (e.g. the results obtained in Sec. 4.6), which is later used to statically assign a prediction to each possible input. We will use the results of Section 4.6 to build the prediction table. This method, however, is highly application-dependent and does not allow to change the predictions at runtime.




(a) MESI (b) HT (c) TokenB

Figure 10: Geometric mean of the prediction accuracy in SPLASH2 and PARSEC for the SP and LVP assuming a 50-cycle observation window. Labels indicate the coverage in LVP (we assume 100% coverage in the static case).

***Last Value Predictor (LVP):*** The predictor consists of a buffer containing the last $M$ multicast sources, which indexes a table of $N$ entries where $N$ is the number of cores. In this work, we consider the most repeated source in an 8-slot buffer as the index for the next guess, giving preference to the most recent transmissions in case of a tie. This way, guesses are dynamically taken and updated at runtime. In order to further increase the accuracy, 2-bit saturating counters are associated to each entry. These counters are incremented when predictions are correct and decremented otherwise, and are only made effective if the value of the counter is '10' or '11', thus reducing the frequency of incorrect predictions.

### *5.2. Evaluation*

To provide a performance evaluation of multicast source prediction, we inject the traces to a MATLAB script that accurately simulates the aforementioned designs. For simplicity, we assume that the predictor has access to all multicast messages, which could be realizable assuming a globally shared medium [1]. The metrics used for the assessment are *coverage*, which accounts for the number of predictions made effective over the number of events of interest; and *accuracy*, which accounts for the number of correct predictions over the number of predictions cast and executed. Basically, a good predictor should achieve a high accuracy without reducing the coverage.

Figure 10 shows the coverage and accuracy of SP and LVP averaged over all the SPLASH2 and PARSEC applications for different CMP sizes and coherence protocols. We assume that the observation window is $\tau = 50 \cdot T_{CLK}$. Consistently with the results in Section 4.6, both SP and LVP show substantially better accuracy in MESI. In HT and TokenB, the use of the static predictor is highly discouraged due to its low reliability. LVP, by means of the added confidence counter, achieves accuracies over the 50% yet with decreasing prediction coverages. Note, however, that the performance of the prediction scheme strongly depends on the application: the accuracy of SP using HT is below 15% for 64-core *canneal* and above 86% for 64-core *cholesky*, to cite an example.

These figures could be improved with more sophisticated predicting schemes like two-level predictors [21]. Also, as implied in Section 4.7, phase detection

can help predictors by limiting predictions to phases in which the predictor has historically been more effective [20].

### *5.3. Possible Uses in NoC*

Thus far, prediction has been sparsely used in NoCs. The work in [23] proposes to speed up NoC routers by predicting the output port of incoming flits. Others advocate for the use of prediction to anticipate changes in the load of certain channels to improve performance or efficiency [22]. In emerging interconnect technologies, prediction has been proposed to reduce the laser power constraints of optical NoCs [24], among others. Here, we break away from this generic traffic prediction techniques and provide a qualitative discussion on designs that could exploit multicast source prediction instead.

In packet-switched NoCs, multicast source prediction could help alleviate the congestion caused by tree-based methods by timely reacting to the small bursts of flits generated by a single dense multicast. Upon predicting the generation of close multicasts, routers in the vicinities could temporarily change their priority vector or routing algorithm to avoid the potential area of congestion. Another approach would be to configure the DVFS system so that predictions are used to increase performance around the multicast source.

In NoCs based on arbitrated shared media such as nanophotonic buses [14] or wireless channels [1], arbitration will be much faster if nodes can know in advance who will transmit next. Take, for instance, a broadcast-oriented wireless NoC whose medium access control is based on a contention-based approach to minimize performance-degrading collisions [1]. By allowing only to predicted sources to transmit under given circumstances, a large percentage of collisions could be prevented, especially in communication-intensive phases of an application. This could suppose a great performance improvement over non-optimized protocols.

## 6. A Model for Multicast Traffic in Shared-Memory Processors

The results of the traffic analysis here presented can be used to create models that faithfully capture the characteristics of multicast traffic in shared-memory multiprocessors. With them, NoCs can be evaluated in realistic conditions without having to resort to lengthy traces or full-system simulations.

Pseudocode in box 1 outlines a possible implementation of a multicast traffic generator. This traffic generator is inspired by the works summarized in 2.3, which use the Hurst exponent and a hotspot factor to model the spatiotemporal distribution of on-chip traffic. We maintain these parameters and then leverage the knowledge on the destination set of multicasts to determine the number of destinations and their location. This algorithm boils down to a unicast traffic generator if the number of destinations is set to 1. Mixed profiles can be created by having two independent generators with their own parameters.

Our implementation of the multicast traffic generator is a central module virtually connected to the NIF of each tile. This module calculates which tile

**input** : $\lambda$ (load; flits/cycle), $H$ (Hurst exponent), $\sigma$ (hotspot factor), $D$ (destinations), totalMsg

Calculate spatial distribution with $\sigma$;
Calculate distribution of destinations with $D$;
**while** *numMsg* $<$ *totalMsg* **do**
  Identify source *src* using the $\sigma$ distribution;
  Identify number of destinations *Ndests* using the $D$ distribution;
  **for** *Ndests* **do**
    Randomly select one destination;
  **end**
  **if** *burst* $>$ *0* **then**
    $burst \leftarrow burst - 1$;
    $\tau \leftarrow T_{CLK}$;
  **else**
    $a_{ON} \leftarrow 3 - 2H$; $b_{ON} \leftarrow 1$;
    $burst \leftarrow pareto\_dist(a_{ON}, b_{ON})$;
    $a_{OFF} \leftarrow a_{ON}$; $b_{OFF} \leftarrow T_{CLK} b_{ON}(\lambda^{-1} - 1)$;
    $\tau \leftarrow pareto\_dist(a_{OFF}, b_{OFF})$;
  **end**
  Wait $\tau$;
  Send message through *src*;
**end**

**Algorithm 1:** Proposed multicast traffic generator.

should be sending each multicast message, to which destinations, and with which delay. The multicast message is passed to the source NIF, which treats it according to the multicast communication policies and injects it into the NoC. In the following, we provide further details on the specificities of the multicast traffic generator along with proof of its validity.

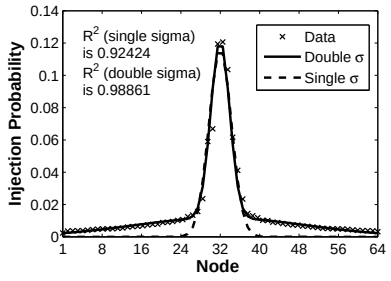
### *6.1. Source*

The work in [17] revealed that a gaussian standard deviation $\sigma$ may be enough to model the spatial distribution of the injection process in NoCs. A large value of $\sigma$ represents an rather flat, uniform distribution among all cores; whereas a small value indicates that injection of traffic follows a hotspot distribution. To obtain the $\sigma$ parameter, fitting methods are applied to the vector of injected multicasts per tile.

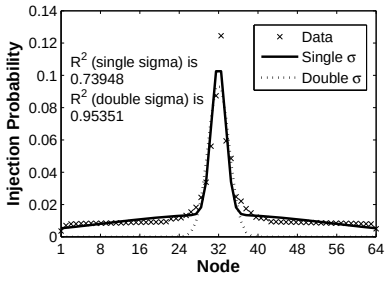
Given that multicast is a subset of all the on-chip traffic, its injection will likely follow a similar pattern. However, our simulations have revealed that using a normal distribution with a single $\sigma$ may yield inaccurate results in some applications. As illustrated in Figure 11b and 11c, some cases would benefit from a double-$\sigma$ fitting. To evaluate the confidence of both methods, we show their coefficient of determination $R^2$ averaged over all the benchmarks and for

| Coherence | Single-$\sigma$ | Double-$\sigma$ |
|---|---|---|
| MESI | 0.9556 | 0.9841 |
| HT | 0.8039 | 0.9513 |
| TokenB | 0.7551 | 0.9799 |

(a) Geometric mean of all SPLASH-2 and PARSEC benchmarks




(b) MESI/*swaptions*




(c) TokenB/*fft*

Figure 11: Coefficient of determination ($R^2$) of the spatial injection distribution, including the gaussian fitting in two relevant cases.

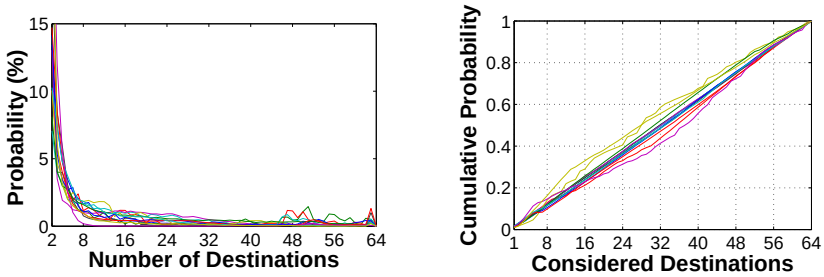
the different coherence methods in Figure 11a. A fully accurate fitting would yield $R^2 = 1$.

These results suggest that double-$\sigma$ distributions may be more appropriate to model the spatial injection distribution of multicast traffic, at the cost of slightly higher complexity. In light of Figure 11, however, one may consider using a single-$\sigma$ model plus a constant which would increase the accuracy without increasing the complexity of the model.

### *6.2. Destinations*

One of the aspects where existing NoC traffic models are not directly applicable in the context of this work is the election of the destinations of a message. In multicast traffic in general, both the number of destinations per message and the destinations themselves need to be modeled. One possibility is to model the number of destinations and then to use existing approaches to independently choose the destinations of each message. More complicated schemes could try to correlate both aspects in order to faithfully characterize certain multicast flows that go from a small set of sources to a deterministic set of destinations. Due to its simplicity, we choose the first option to model MESI multicast traffic; in HT and TokenB, the process is straightforward since almost all the generated multicast messages are broadcasts.

To model the number of destinations per message, a trivial approach would be to compute the average number of destinations for a given application. More accurate models can be obtained using fitting methods to the histogram of destinations. As shown in Figure 12a, the distribution of the number of destinations of most applications would be accurately modeled with a power function $[f(x) = a \cdot x^b + c]$ or a rational function $[f(x) = a/(x + b)]$. The coefficient of




(a) Probability distribution function of the number of destinations per multicast.

(b) Cumulative distribution function of the multicast destinations.

Figure 12: Statistical analysis of the multicast destinations in MESI (perfect tracking).

determination $R^2$ is of 0.9774 and 0.9668 in average, respectively. It is worth noting that broadcasts messages in MESI coherence represent a particular case and may need to be modeled separately here, especially when imprecise tracking is considered.

To model the destinations of each message, we consider the destination set to be independent of the source for simplicity. We collected the number of received multicast messages per NIF and performed initial modeling tests. Unlike the injection process, the spatial distribution of the received multicasts does not fit well with a gaussian distribution and exhibits a more uniform behavior instead. Figure 12b plots the cumulative distribution function (CDF) of the received multicasts, confirming that the destinations may be modeled using a uniform random variable. A linear fitting further validates this fact by achieving a coefficient of determination of 0.9964 in average.

### *6.3. Delay*

The modeling of self-similar traffic has been the subject of different studies in all areas of networking [25], including on-chip communication [17]. In light of the results shown in Section 4.5 regarding the temporal distribution of the injection of multicasts, knowledge obtained thus far can be used to model the burstiness of multicast traffic.

The most widespread method to generate self-similar traffic is through the alternate generation of ON and OFF periods. During the ON periods, the generator outputs one message per clock cycle; whereas it remains silent the rest of the time. The length of both periods follow a Pareto distribution [25], which is a heavy tailed distribution with a probability density function

$$f(x) = \frac{a \cdot b^a}{x^{a+1}} \qquad x \geq b\,. \tag{3}$$

The shape parameter $a$ is related with the Hurst exponent as

$$a_{ON} = a_{OFF} = 3 - 2H, \tag{4}$$

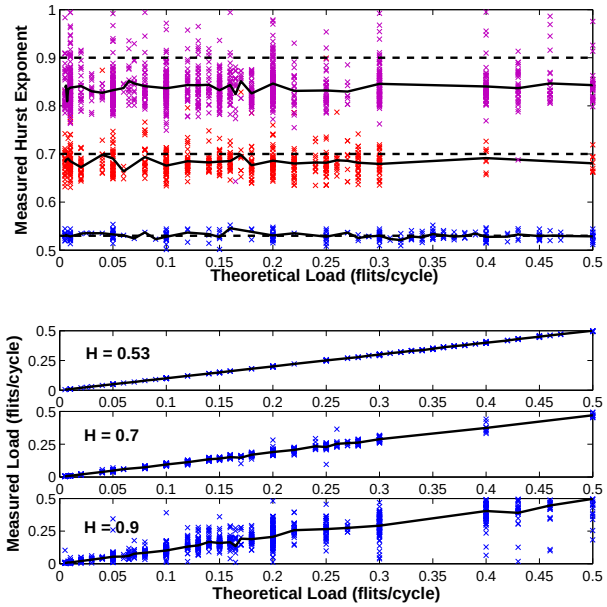



Figure 13: Measured Hurst exponent (top) and load (bottom) as functions of the input load for $H = \{0.53, 0.7, 0.9\}$. Dashed and solid lines represent the theoretical value and geometric mean of the measured values, respectively.

whereas the location parameter $b$ needs to be set at the minimum value of the distribution. In NoC environments, one can take $b_{ON}$ as the equivalent to a burst of a single multicast, whereas $b_{OFF}$ is scaled in order to fix the load to the desired $\lambda$ value, as:

$$b_{OFF} = b_{ON}(\frac{1}{\lambda} - 1). \tag{5}$$

Using this method, we successfully created synthetic traffic with the desired $H$ and $\lambda$ characteristics. To prove the validity of the approach, we generated streams of bursty traffic of 100K messages each, with $H = \{0.53, 0.7, 0.9\}$ and different loads between 0 and 0.5 flits per cycle. The traces containing the timestamps of each generated message are then analyzed to obtain the real load and Hurst exponent, and then averaged over a variable number of repetitions. Figure 13 shows the measured Hurst exponents and loads as functions of the three analyzed burstiness levels. In both figures, it is observed that results become more random as the input Hurst exponent increases. Still, both the measured load and the measured Hurst exponent increase, in average terms, proportionally to the input load and Hurst exponent, respectively. Finally, it is important to note that the average error of the measured Hurst exponent increases with the input exponent. Therefore, corrective factors need to be applied as $H$ values approach 1 and, therefore, ON and OFF periods become

large. Increasing the length of the simulations also helps to reduce this error.

## 7. Conclusions

We have analyzed the scaling trends of multicast communication in cache-coherent processors by performing a trace-based characterization of a wide set of architectures, applications and system sizes. The results point towards a sustained increase of the multicast intensity, as well as of its spatial imbalance and temporal burstiness, confirming the need for proper multicast support in many-core scenarios. To assist the evaluation of future NoCs, we proposed a simple yet accurate multicast traffic model; whereas to optimize their design, we demonstrated a consistent growth in terms of spatiotemporal correlation of multicast traffic. This trend implies an increase of the chances of correctly predicting the source of multicast transfers, as shown here both using a predictability metric and evaluating the performance of two simple predictors.

## Acknowledgment

The authors gratefully acknowledge support from INTEL through the Doctoral Student Honor Program and from the *Comissionat per a Universitats i Recerca* of the Catalan Government (Ref. 2014SGR-1427).

## References

[1] S. Abadal, B. Sheinman, O. Katz, O. Markish, D. Elad, Y. Fournier, *et al*, Broadcast-Enabled Massive Multicore Architectures: A Wireless RF Approach, IEEE MICRO 35 (5).

[2] S. Abadal, A. Mestres, R. Martínez, E. Alarcón, A. Cabellos-Aparicio, Multicast On-Chip Traffic Analysis Targeting Manycore NoC Design, in: Proceedings of the PDP '15, 2015, pp. 370–378.

[3] F. Wong, R. Martin, R. Arpaci-Dusseau, D. Culler, Architectural Requirements and Scalability of the NAS Parallel Benchmarks, in: Proceedigs of the ACM/IEEE SC '99, 1999, pp. 1–18.

[4] J. Vetter, A. Yoo, An Empirical Performance Evaluation of Scalable Scientific Applications, in: Proceedings of the ACM/IEEE SC '02, 2002, pp. 1–16.

[5] J. Shalf, S. Kamil, L. Oliker, D. Skinner, Analyzing ultra-scale application communication requirements for a reconfigurable hybrid interconnect, in: Proceedings of the ACM/IEEE SC '05, 2005, pp. 17–30.

[6] S. Woo, M. Ohara, E. Torrie, J. Singh, The SPLASH-2 programs: Characterization and methodological considerations, ACM SIGARCH Computer Architecture News 23 (2) (1995) 24–36.

[7] C. Bienia, S. Kumar, J. Singh, K. Li, The PARSEC benchmark suite: characterization and architectural implications, in: Proceedings of the PACT '08, 2008, pp. 72–81.

[8] N. Barrow-Williams, C. Fensch, S. Moore, A communication characterisation of Splash-2 and Parsec, in: Proceedings of the IISWC '09, 2009, pp. 86–97.

[9] M. Martin, Token Coherence: decoupling performance and correctness, in: Proceedings of the ISCA-30, 2003, pp. 182–193.

[10] M. Ebrahimi, M. Daneshtalab, Path-Based Partitioning Methods for 3D Networks-on-Chip with Minimal Adaptive Routing, IEEE Transactions on Computers 63 (3) (2014) 718–733.

[11] T. Krishna, L. Peh, B. Beckmann, S. K. Reinhardt, Towards the ideal on-chip fabric for 1-to-many and many-to-1 communication, in: Proceedings of the MICRO-44, 2011, pp. 71–82.

[12] J. E. Miller, H. Kasture, G. Kurian, N. Beckmann, C. Celio, J. Eastep, A. Agarwal, Graphite: A Distributed Parallel Simulator for Multicores, in: Proceedings of the HPCA-16, 2010, pp. 1–12.

[13] R. Manevich, I. Cidon, A. Kolodny, Handling global traffic in future CMP NoCs, in: Proceedings of the SLIP '12, 2012, pp. 40–47.

[14] J. Kim, K. Choi, Exploiting New Interconnect Technologies in On-Chip Communication, IEEE Journal on Emerging and Selected Topics in Circuits and Systems 2 (2) (2012) 124–136.

[15] M. M. K. Martin, P. J. Harper, D. J. Sorin, M. D. Hill, D. A. Wood, Using destination-set prediction to improve the latency/bandwidth tradeoff in shared-memory multiprocessors, in: Proceedings of the ISCA '03, 2003, pp. 206–217.

[16] P. Conway, B. Hughes, The AMD Opteron Northbridge Architecture, IEEE Micro 27 (2) (2007) 10–21.

[17] V. Soteriou, H. Wang, L. Peh, A Statistical Traffic Model for On-Chip Interconnection Networks, in: Proceedings of MASCOTS '06, 2006, pp. 104–116.

[18] M. Badr, N. Enright Jerger, SynFull : Synthetic Traffic Models Capturing Cache Coherent Behaviour, in: Proceedings of ISCA-41, 2014, pp. 109–120.

[19] N. Binkert, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, *et al*, The gem5 simulator, ACM SIGARCH Computer Architecture News 39 (2).

[20] T. Sherwood, S. Sair, B. Calder, Phase tracking and prediction, ACM SIGARCH Computer Architecture News 31 (2) (2003) 336–349.

[21] J. Hennessy, D. Patterson, Computer architecture: a quantitative approach, Morgan Kaufmann, 2012.

[22] U. Ogras, R. Marculescu, Prediction-based flow control for network-on-chip traffic, in: Proceedings of the DAC '06, 2006, pp. 839–844.

[23] H. Matsutani, M. Koibuchi, H. Amano, T. Yoshinaga, Prediction router: Yet another low latency on-chip router architecture, in: Proceedings of the HPCA '09, 2009, pp. 367–378.

[24] L. Zhou, A. K. Kodi, PROBE: Prediction-based optical bandwidth scaling for energy-efficient NoCs, in: Proceedings of the NoCs '13, 2013, pp. 1–8.

[25] W. E. Leland, M. S. Taqqu, W. Willinger, D. V. Wilson, On the self-similar nature of Ethernet traffic (extended version), IEEE/ACM Transactions on Networking 2 (1) (1994) 1–15.

## Vitae

**Sergi Abadal** is a PhD candidate at the NaNoNetworking Center in Catalonia (N3Cat), Spain, at the Universitat Politécnica de Catalunya (UPC). In 2013, he received the INTEL Doctoral Student Honor fellowship. His research interests include on-chip networking, many-core architectures, and graphene-based wireless communications. Abadal has an MSc in information and communication technologies (2011) from the UPC.

**Raúl Martínez** is a consulting member of Technical Staff at Oracle Labs. Before that, he worked at the INTEL Barcelona Research Center. His research interests include high-performance interconnects, hardware / software co-design, dynamic binary optimizations, and network on chips. Martínez has a PhD (2007) from the University of Castilla-La Mancha.

**Josep Solé-Pareta** is a full professor at the Computer Architecture Department at the UPC. He is co-founder of the CCABA and N3Cat groups also at UPC. His current research interests are in nanonetworking communications, traffic monitoring and analysis, high speed and optical networking, and energy efficient transport networks. Solé-Pareta has a PhD in computer science (1991) from the UPC.

**Eduard Alarcón** is an associate professor at the N3Cat at UPC. He has co-authored more than 250 scientific publications, and has been involved in different R&D projects within his research interests including nanocommunications, energy harvesting, and wireless energy transfer. Alarcón has a PhD in electrical engineering (2000) from the UPC.

**Albert Cabellos-Aparicio** is an associate professor at the N3Cat at UPC. He has given more than 10 invited talks and co-authored more than 70 papers within his research interests including graphene technology, nanocommunications, and software-defined networking. Cabellos-Aparicio has a PhD in computer science engineering (2008) from the UPC.