Multicast On-Chip Traffic Analysis Targeting Manycore NoC Design

Sergi Abadal, Albert Mestres, Eduard Alarcón
and Albert Cabellos-Aparicio
NaNoNetworking Center in Catalonia (N3Cat)
Universitat Politècnica de Catalunya (UPC)
Barcelona, Spain
Email: abadal@ac.upc.edu

Raúl Martínez
Department of Computer Architecture
Universitat Politècnica de Catalunya (UPC)
Barcelona, Spain

***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 and has been the focus of different research efforts, mainly due to the poor performance of NoCs in the presence of this increasingly important type of traffic. However, most of the proposed schemes have been evaluated using synthetic traffic or within a full system, which is either unrealistic or costly. While traffic models would allow to better assess their performance, existing proposals do not distinguish between unicast and multicast flows and often are bound to a given number of cores. In this paper, a trace-based multicast traffic characterization is presented with the aim to provide guidelines for the modeling of multicast communications in manycore settings. To this end, the scaling trends of aspects such as the multicast traffic intensity or the spatiotemporal injection distribution are analyzed. The novelty of this work resides both on its scalability-oriented approach and on the use of correlation metrics to evaluate potential prediction opportunities.**

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

I. Introduction

Dominant trends in processor design are currently pointing towards a rapid increase in the core density of multiprocessors. However, several challenges need to be addressed before this tendency translates into effective parallel performance improvements. On-chip communication is among them and is currently gaining importance within this context, to the point that the research focus is gradually shifting from how cores compute to how cores communicate.

Scaling the number of cores of a given multiprocessor architecture generally implies an increase of the communication intensity for a program sufficiently parallelized. As we reach the manycore era, the main concern is that traditional Network-on-Chip (NoC) approaches may not be able to cover the communication requirements of a multiprocessor due to various reasons. At the link level, it is expected that the power available for communication in future processors will not be enough to maintain a NoC based on electrical interconnects [1]; whereas at the network level, the performance and power of these NoCs scales poorly in the presence of selected types of traffic, e.g. global or multicast.

In order to avoid communication to become the performance bottleneck of next-generation multicore processors, numerous research endeavors have been made towards the development of scalable NoCs. On the one hand, diverse proposals seek to improve the power and performance of traditional NoCs at both the link and network levels [2]–[5]. On the other hand, considerable efforts have been directed towards overcoming the fundamental limitations of traditional resistive-capacitive (RC) wires by extending the original NoC paradigm to emerging interconnect technologies. These include the employment of vertical vias within stacked architectures [6], on-chip radio frequency (RF) transmission lines [7], nanophotonic interconnects [1], [8], and on-chip antennas enabling the concept of wireless NoC [9], [10].

In this paper, we focus on the particular case of multicast communications. In conventional NoCs, multicast messages have been originally broken down into multiple unicast packets at the network interface and served independently. Besides being highly power-inefficient, such approach significantly penalizes the network performance as implies a large serialization delay and significant levels of contention at the source [11]. This, in turn, negatively affects the multiprocessor performance [12], [13]. Such negative impact is intensified in denser networks as, first, the average number of destinations per message in multicast-demanding operations grows with the number of cores (see Section IV). Furthermore, and even though multiprocessors have been traditionally architected trying to limit the use of multicast communications, some architectural methods may require it to scale beyond a given number of cores. For instance, cache coherency protocols normally avoid multicast by storing the state of each shared variable in a directory. This increases the latency of the protocol and, depending on the storage policy, implies area and energy overheads that may not be affordable in manycore systems. Alternatives such as TokenB [14] or the implemented by AMD HyperTransport [15] avoid such possibility at the expense of incrementing the multicast requirements.

Improving the performance of multicast transactions not only implies a boost in the performance of a given set of architectures, as proved in [12], but also opens the door to new architectural innovations for manycore processors [11]. In light of this, explicit multicast support has been proposed in the literature for conventional NoCs [11]–[13], [16], [17] or considering emerging interconnect technologies [10], [18], [19]. A distiction is made in the former case between path-based multicast, wherein the message travels around the chip and is sequentially delivered to the intended destinations; and tree-based multicast, where the message is replicated at intermediate routers and delivered following a virtual tree [16]. 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.

Given the increasing importance of multicast communication, there is a need to understand how the multicast traffic requirements scale. Providing an 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. As pointed out below, existing efforts do not differentiate between unicast and multicast flows and are generally bound to a given network size. In this work, we aim to bridge this gap by presenting a characterization of the multicast traffic for a given architecture considering two different coherence protocol types. This paper is an extension of our previous work in [20] and presents new results on aspects such as the multicast traffic intensity, number of destinations or spatiotemporal distribution. Moreover, it proposes the use of correlation metrics for further characterization and points out that application phase behavior also applies to multicast traffic.

The remainder of this paper is as follows. In Section II, we summarize the related work in NoC traffic analysis and modeling. In Section III, we detail the characterization methodology used in this work. We present the results in Section IV and conclude the paper in Section V.

II. Related Work

The driving motivation behind traffic characterization is the need for a cost-effective but accurate way to evaluate networks in general and NoCs in particular. On the one hand, network simulation using real traces is the most accurate solution but traces are not always available; whereas, on the other hand, typical synthetic traffics sacrifice faithfulness at the expense of a lower computational cost. Modeling traffic based on prior characterization works lies in between these two extremes as it trades off the accuracy of trace-based simulations for the simplicity of synthetic traffic.

Table I
Literature on Traffic Characterization and Modeling

Ref.	Cores	Coherence	Benchmarks	Mcast?
[21]	32	MESI	SPLASH-2	No
[22]	8	MSI	PARSEC	No
[23]	32	MESI	SPLASH-2, PARSEC	No
[24]	40/49/64	MSI	SPEC CPU2000, SPLASH-2, PARSEC	No
[25]	16/64	MESI	OpenMP	No
[26]	8/16	MESI	TPC-C, SPECweb99, TPC-H, SPEC CPU2000	No
[27]	16/25	MSI	SPEC, MediaBench	No
[28]	49	MSI	SPLASH-2	No
[29]	16/32/64	MSI	SPLASH-2	No
[11]	16/25	SGI Origin, TokenB, HT	SPEC, TPC-H, TPC-W, SPLASH-2, MediaBench	Yes
[30]	16	HT	SPLASH-2	Yes
[12]	64	HT, TokenB	SPLASH-2, PARSEC	Yes
[31]	16	MSI	NU-MineBench, SPLASH-2	No

Traffic characterization in NoC can be performed by analyzing traces obtained from full-system simulations. Each trace contains information on the messages generated by a given application running over a particular multiprocessor architecture. Common characteristics or patterns in the traffic of different applications can be found and later used as guidelines for the optimization of NoCs. Multiprocessor benchmarks such as SPLASH-2 [21] and PARSEC [22] consist of a heterogeneous set of parallel programs, which are commonly used as target applications for evaluation purposes. Multiprogrammed and server workloads such as TPC-H or SPECweb99, which imply the simultaneous execution of various instances of a set of simple applications, are also used.

Table I contains a summary of the works that follow trace-based analysis, detailing the simulated architectures and the employed benchmarks. Note that most of these works do not explicitly distinguish between unicast and multicast communications. In [21] and [22], the focus is put on the study of the behavior of a single-level cache when running the SPLASH-2 and PARSEC benchmarks, respectively, to then provide some hints on their general communication demands. Subsequent works have investigated several characteristics of the traffic generated within a two-level cache system. A 32-core system is simulated in [23] in order to provide insight into the spatiotemporal variations of communication arising from common data sharing patterns. Spatiotemporal variations are also investigated in [24] and correlated with the network congestion they may cause. To this end, it is first demonstrated that the different program phases produce periodic patterns in the injection of traffic. The probability of having hotspots is then investigated through the inspection of the spatial variance of traffic injection. The work conducted in [31] registers the spatial pattern variations and uses this information to identify the different phases of an application. The phase behavior has been also investigated in [25], where the dominant communication flows are determined on a per-phase basis. Such results could be employed to build a hybrid network that would adapt to give priority to these flows in each stage of execution. In [26], NoC optimization is also proposed through the study of the distribution of the packet length.

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 NoC traffic of different benchmarks and architectures using one parameter per dimension [27] or one parameter per dimension and node [28]:

- *Temporal burstiness*, resulting from the widely proven self-similarity of NoC traffic and often modeled with the Hurst exponent [32]. Such property is driven by the fact that the generation of messages largely depends on how the network served prior requests. One constant Hurst exponent is proposed in [27] to capture the burstiness of traffic, while others argue that the Hurst exponent should be characterized over time in certain situations [32].
- *Spatial injection distribution*, which models whether the injection of packets is spread out or creates hotspots.
- *Hop distribution*, which models the probability of a packet going through a given number of hops. It depends on the NoC topology, the application and how the application is mapped onto the processor. Some authors demonstrate an analogy between a power law called the Rent's law and the spatial hop distribution of a NoC [29].

When modeling traffic using such method, no distinction is made between unicast and multicast flows. This may be acceptable from a behavioural perspective only in the case that multicast packets are replicated at the source network interface and treated as unicast packets. In contrast, existing models will need to consider both types of traffic separately in cases where the network behavior changes with the communication typology, i.e. path-based or tree-based multicast.

Explicit multicast analyses have been thus far limited to specific cases and oriented to motivate multicast enhancements for NoC. For instance, the percentage of multicast messages as well as the average number of destinations is given in [12] considering two coherence protocols in a 64-core system running the PARSEC and SPLASH-2 benchmarks. These are also evaluated in [11] for a 16-core system, which also provides the destination set distribution in order to prove that in some situations many of them remain unused. Finally, a breakdown of the messages generated by the HyperTransport (HT) protocol is performed over the SPLASH-2 benchmark in [30]. In spite of such efforts, traffic models that distinguish between unicast and multicast flows are not available yet.

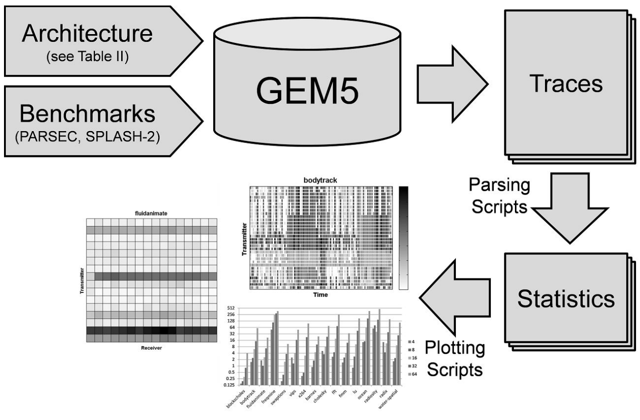

Figure 1. Simulation-based multicast traffic analysis.

III. Methodology

The main objective of this paper 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 traces (one per application and coherence protocol). 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, allowing to evaluate multicast routing schemes in practical scenarios without having to resort again to full-system simulation.

The central element of the process is the simulator. We use GEM5 [33], a widely used open-source framework for the simulation of computer system architectures both from the processor microarchitecture level and the system level. Currently, GEM5 admits up to 64 cores with up to three levels of cache and, out of the box, includes different directory-based cache coherency protocols such as MESI, MOESI or HyperTransport (HT). GEM5 also integrates packages for the modeling and simulation of the NoC that interconnects the cores. In order to obtain custom statistics and traces oriented to multicast traffic analysis, we slightly modified the network interface modules included by default. When the first flit of a multicast packet is to be injected through any network interface, its time of arrival, origin, destinations, type, and size are registered into a trace file. This way, the results are independent to the multicast routing strategy. The output of GEM5 in our case is a file containing detailed statistics on the CPU, memory system, and NoC performance, as well as a trace with the data mentioned above for each issued multicast. The latter can be processed *a posteriori* in order to obtain a variety of multicast statistics and figures, including those proposed in related works.

Figure 2 depicts the architecture considered in this paper. A tiled configuration is assumed, wherein each tile comprises a core that implements the ALPHA instruction set and

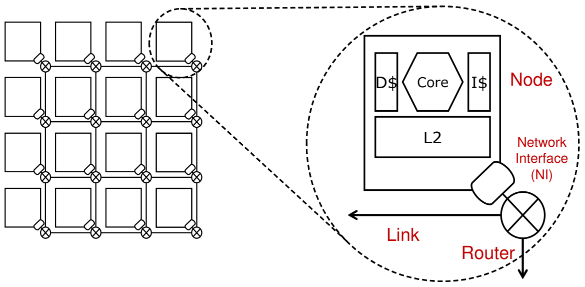

Figure 2. Schematic representation of the simulated architecture.

accounts for 32-KB instruction and data L1 caches, as well as a 512-KB bank of L2 cache and a piece of the coherence directory. The main variables are the multiprocessor size, which ranges from 4 to 64 tiles, and the coherency protocol. We run the PARSEC (*simmedium* input set) and SPLASH-2 benchmarks on their entirety whenever possible, otherwise the execution is limited to one billion instructions. Table II shows a summary of the simulation parameters used in the study from the architectural, NoC, and application perspectives.

IV. Characterization Results

Next, we present the results of the analysis in terms of multicast message type and size, traffic intensity, number of destinations, spatiotemporal distributions, spatiotemporal correlations and phase behavior. A discussion of the possible implications on the NoC design is included in each subsection.

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 nature and size of these messages can be easily inferred from it. 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, assumed to be of 8 bytes in our scenario. In some cases, e.g. read to an invalidated block, the protocol needs to multicast data responses to the main memory and to a set of caches. These replies are less much frequent and their size corresponds to the cache line size plus a given overhead (8 bytes in our case). In the HT protocol, the percentage of multicast long messages is expected be even lower since all control requests are broadcast. Actual results, however, show a similar size distribution for both coherence protocols: short messages account for more than 99% of the multicast in average. This figure is rather independent of the system size and, in fact, rarely drops below 98%.

Note that the size of messages is a parameter relevant to the NoC design. In conventional NoCs, it is important to set the data path width taking into consideration the

Table II
Simulation Parameters

Instruction Set	ALPHA
System Frequency	2 GHz
Number of cores	4, 8, 16, 32, 64
Benchmarks	PARSEC, SPLASH-2
L1 Cache (I&D)	32+32 KB, 2-way, 2 cycles
L2 Cache	512 KB/bank, 8-way, 10 cycles
Cache Line Size	64 B
Cache Inclusion	Strict
Coherency Protocols	MESI, HT
Main Memory Latency	30 ns
NoC Topology	Mesh
Link/Router Latency	1/5 cycles
Flit Size	16 B
Number of Virtual Channels (VCs)	4
Buffer Size	5 flits per VC

dominant type of traffic. This fact is exemplified in [26], which proposes to employ a dual NoC to optimize the service of both short requests and long responses, and shows substantial power and performance improvements with respect conventional designs. Therefore, in case a network plane would be devoted to serving multicast traffic, its design should be strongly oriented to short messages. In a wireless NoC, the size of a packet is also of central importance as small messages generally imply a lower performance [34].

Multicast Traffic Intensity

The number of multicast messages per instruction is a NoC-agnostic measure of the multicast intensity. It is a metric that depends both upon the multiprocessor architecture, as it defines the methods that generate multicast messages; and upon the application, which defines the sharing structures and memory intensity. Figure 3 (left) plots the number of injected multicast messages per one million instructions assuming MESI (above) and HT coherence (below). 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 *bodytrack, fft* or *lu*). In comparative terms, HT coherence has multicast requirements one order 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 logarithmic relation between multicast intensity and number of cores. Application scalability limitations may explain such tendency.

Assuming a given computation performance, it is possible to infer the number of messages per clock cycle that the NoC must support. It is important to take into consideration that architectures are scaled in order to improve performance, so that the multicast intensity will likely increase even further. For instance, our simulations yielded a 13-fold increase in the average execution speed of the target benchmarks. As we will see next, the traffic intensity is also influenced by the number of destinations per message.

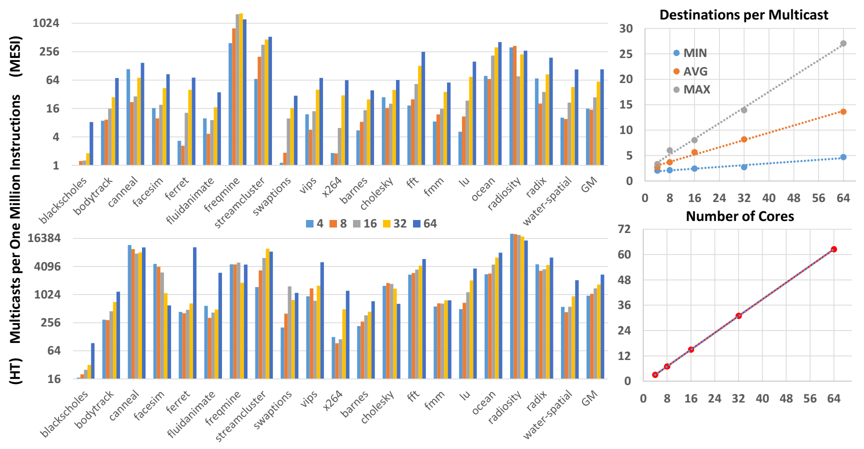

Figure 3. Number of multicast messages per 10^6 instructions and number of destinations per multicast message with MESI (above) and HT coherence (below). Note the logarithmic scale.

Number of Destinations

The number of destinations per message is a metric that firstly depends upon the multiprocessor architecture and, in some cases, on the particular sharing structures of the application being run. We evaluate the average number of destinations per multicast over the course of each considered application, to then obtain the minimum/average/maximum of these values over all the applications for different system sizes. Figure 3 (right) plots how the number of destinations per multicast scale assuming MESI (above) and HT coherence (below). In the former case, multicast are mostly invalidations to a number of destinations that depends upon number of sharers. The metric increases linearly with the number of cores in terms of minimum, maximum and average values: the application dependence is patent given the difference between these numbers. In the latter case, the coherence protocol issues on broadcast for each transaction. Provided that almost all multicast messages are due to coherence, the average number of destinations is around $N-1$ in a N-core system regardless of the target application.

The importance of this metric lies within its impact on the overall bandwidth requirements depending on the network architecture. While the number of injected multicasts increases with the number of cores, they only represent a small percentage of all the communication transactions (see labels in Figure 4). Less than 0.5% of all the transactions are multicast in MESI; the ratio is higher in HT, but it decreases sharply with the number of cores (from 12% to 1.5%) due to the associated increase of the number of unicast acknowledgements. In contrast, the amount of ejected packets consistently increases due to the multiple replication of each multicast message within the nework. Figure 4 plots the percentage of ejected flits that are due to multicast transactions, which grows above 2% in MESI and almost reaches 50% in HT. This fact further encourages

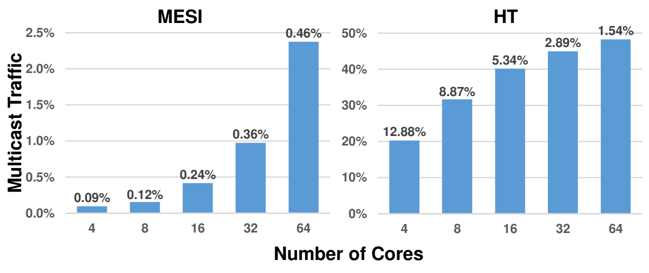

Figure 4. Percentage of ejected flits due to multicast communication. Labels indicate the percentage of injected multicast packets (without replication).

the employment of shared-medium NoCs, if feasible, to efficiently serve multicast traffic.

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. Note that while the spatial injection distribution as defined here is independent on the multicast scheme, the effects of the resulting imbalance are not (i.e., multicast hotspots are more detrimental in tree-based schemes). To evaluate the spatial distribution, we calculated the coefficient of variation (CoV) as $c_v = \sigma/\mu$, where σ and μ 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.

There are several common memory access patterns that generate traffic that can be multicast depending on the architecture [23]. In MESI, applications heavily based on producer-consumer patterns show imbalanced injection of packets as producers are the main sources of multicast traffic. Such assumption is not valid in HT, since every control operation generates a broadcast message. Instead, the spatial distribution provides insight about the memory activity of different processors: cores that frequently access to shared data will generate more broadcasts than those that do not. Therefore, a lower imbalance is expected when compared to MESI.

Figure 5 plots the CoV of each application in 4/16/64-core systems assuming MESI and HT coherence, 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, which implies that applications that appear first may yield more pronounced imbalance in manycore processors. From the average behavior, it is observed that MESI shows a higher imbalance in general terms. It is also shown that the injection is more balanced in the SPLASH-2 benchmark, yet the difference becomes negligible as the number of cores is scaled.

Temporal Distribution

In order to accurately model any kind of traffic, it is crucial to have a complete knowledge not only on its intensity and spatial distribution, but also on its temporal distribution. As shown in Section II, related works have shown that on-chip traffic is self-similar since 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. In order to confirm this fact, we calculated the Hurst exponent H $(0.5 < H \leq 1)$ applying the RS plot method [27] on the temporal information of the full-system traces. In light of the results of Table III (GM stands for Geometric Mean) 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 increases with the core count. The exponent is slightly higher in HT or running SPLASH-2 applications. Note that the NoC will have 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.

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 given node X (not any node) 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. For instance, take a NoC based upon arbitrated shared media such as nanophotonic buses or a set of wireless channels: arbitration will be much faster if nodes know in advance who will transmit next.

To evaluate spatiotemporal correlation, we consider transmissions separated less than a given time period τ. If $X = Y$, we found a potential source of *autocorrelation*; whereas if $X \neq Y$, we are facing a case of *crosscorrelation* (note that a transmission can be both auto- and crosscorrelated). The choice of τ may depend on several factors and must capture meaningful correlations. A value too low will not yield any correlation, while a value too high will dilute meaningful cases within high correlation probabilities. Two interesting correlation metrics can be obtained from this analysis.

First, we evaluate the percentages of correlation 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. Figure 6 shows the correlation distribution of multicast transmissions assuming two different τ values. It is observed that the percentage of crosscorrelated transmissions grows with the number of cores, especially in the case of MESI, and that autocorrelation levels are not significant in HT. Since a high

Table III
HURST EXPONENTS

	MESI			HT		
	4	16	64	4	16	64
blackscholes	.9628	.9497	.9473	.8394	.9640	.9853
bodytrack	.9436	.9617	.9594	.9213	.9445	.9742
canneal	.5856	.7876	.8636	.7152	.7892	.8367
facesim	.8836	.959	.9488	.9175	.9225	.9301
ferret	.8606	.8693	.9231	.9564	.9541	.9975
fluidanimate	.9624	.9459	.9514	.9211	.9053	.9817
freqmine	.7962	.9765	.9953	.8461	.9369	.9977
streamcluster	.9764	.8526	.8317	.9835	.8355	.8121
swaptions	.8812	.8638	.8983	.8832	.9376	.9375
vips	.8586	.9331	.9583	.9169	.9842	.9747
x264	.7918	.8408	.9685	.8635	.9047	.9958
GM	.8534	.9017	.9303	.8775	.9164	.9461
barnes	.9914	.9828	.9701	.9285	.9463	.9671
cholesky	.9325	.899	.9622	.9616	.9742	.9870
fft	.9207	.8972	.9616	.9798	.9886	.9774
fmm	.9782	.9684	.9655	.9393	.9565	.9754
lu	.9584	.9354	.9524	.9384	.9689	.9583
ocean	.8955	.8862	.9113	.8209	.9271	.8563
radiosity	.9451	.888	.9662	.9616	.9638	.9762
radix	.8744	.9551	.9595	.8671	.9194	.9201
water-spatial	.9677	.9542	.9614	.8453	.9426	.9681
GM	.9397	.9289	.9565	.9152	.9539	.9648

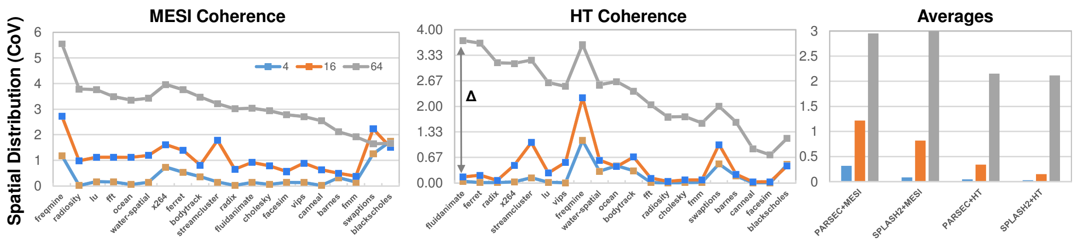
Figure 5. Coefficient of Variation (CoV) of the spatial injection distribution of multicasts for 4/16/64-core systems assuming MESI and HT coherence. Applications are sorted in descending order based on the difference Δ between CoV(64) and CoV(4).

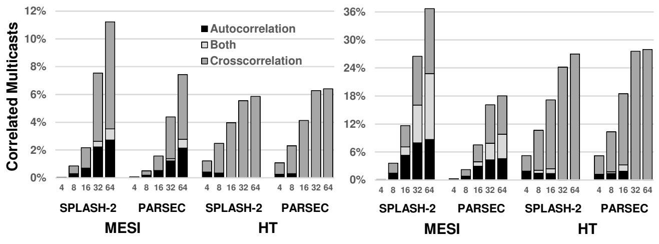
Figure 6. Percentage of correlated multicast transmissions assuming MESI or HT coherence with τ set to 10 clock cycles (left) and 50 clock cycles (right).

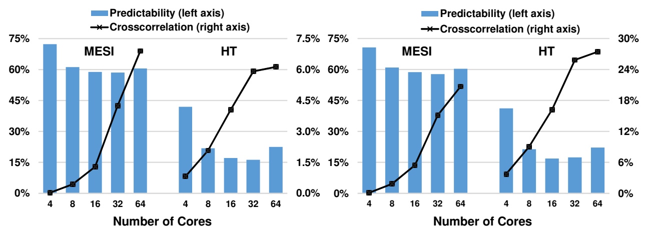
Figure 7. Factor of predictability and percentage of crosscorrelaion assuming MESI or HT coherence with τ set to 10 clock cycles (left) and 50 clock cycles (right).

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 τ impacts upon the percentage of correlated transmissions, but not much upon its scalability with respect to multiprocessor size. In future work, we aim to perform a sensitivity study of such percentage when τ changes.

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 correlated). On the contrary, a high value indicates a strong correlation with few cores (1 if transmissions by X are always correlated with one single core Y, X $\neq$ Y). The factor of predictability between any two pairs is calculated as the weighted average of the predictability of each core:

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

Note that this metric is also an estimation of the probability of correctly guessing the source of the next transmission with static prediction and observing the current transmission.

Figure 7 shows the factor of predictability assuming two different τ values. It is observed that the predictability decreases as the number of cores grows while, on the contrary, the percentage of crosscorrelation increases substantially. This supports the hypothesis that predictive strategies have more potential in larger multiprocessors. The predictability is much higher when MESI coherence is employed, probably due to the clear dependence of multicast traffic with potentially predictable memory sharing patterns in this case. In HT, the sources of multicast traffic are more varied and the use of more sophisticated predictors would be required. It is also observed that increasing the observation window from 10 to 50 clock cycles makes the crosscorrelation levels rise, yet it hardly affects the overall predictability. This analysis could assist in the evaluation of the scope of a predictor, i.e. which events to predict.

Phase Behavior

Apart from investigating self-similarity, trace-based analysis may allow the study of the different phases found in the applications. Recent literature demonstrates not only that such phases exist in applications running over shared-memory multiprocessors, but also that the characteristics

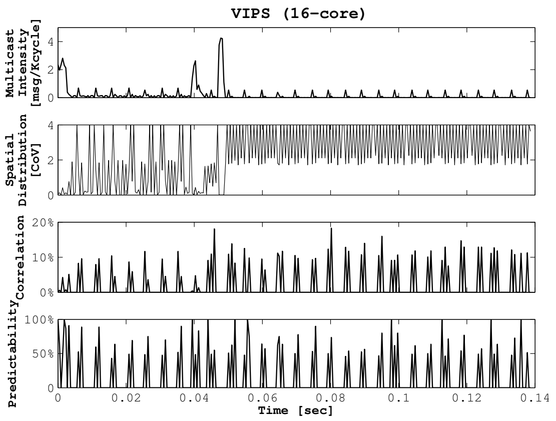

Figure 8. Temporal evolution of different multicast characterization metrics in a 16-core architecture with HT coherence running *vips*.

of these phases and the transitions between them are predictable [31], [35]. Phase changes affect a wide variety of metrics [36], including communication intensity [25]. Multicast communication is likely to be also influenced by the existence of phases within an application, and, therefore, the metrics presented above could be evaluated on a per-phase basis. The information can be used at compile-time or run-time to assist processes that may use the aforementioned metrics. For instance, a predictor used to improve NoC performance can be reconfigured on each phase change to increase its coverage or accuracy.

Rather than repeating all the analyses considering phase behavior, here we exemplify the variation of different metrics on a given case. Figure 8 plots the temporal evolution of the multicast intensity, the spatial distribution, the level of correlation and the factor of predictability of application *vips* running in a 16-core architecture with HT coherence. Two clear and differentiated sections are observed, the first one with variable spatial imbalance from the beginning of the execution until around 0.05 seconds, and the second one with high spatial imbalance from that point onwards. A periodic behavior is patent in both sections, alternating between *silent* periods and phases with moderate multicast intensity. Peaks of predictability suggest the use of predictive techniques in specific parts of the execution.

The phase behavior shown here is general to most applications and can be in principle extrapolated to different multiprocessor sizes by taking into account that, since processing work is generally distributed over the different threads, the time spent within a given phase is inversely proportional to the number of threads.

V. Conclusions

We presented a trace-based methodology for the analysis of the scaling trends of multicast communication in multiprocessors. We then used it to characterize the multicast traffic generated by the MESI and HT coherence protocols for different multiprocessor sizes. The results point towards an increase of the number of multicast messages and of the number of destinations per message as the number of cores grows. Further, multicast traffic becomes more spatially imbalanced and shows increased temporal burstiness, properties that generally degrade performance. These results confirm the need for NoCs capable of efficiently dealing with multicast messaging as we reach the manycore era and set preliminary scalability requirements for such designs. Fortunately, multicast sources show growing spatiotemporal correlations that could be exploited towards achieving this goal. In future work, our main aims will be to deeply analyze the implications of phase behavior on the characteristics of multicast traffic and to accurately model such traffic for early-stage NoC design and evaluation purposes up to a thousand cores.

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] D. A. B. Miller, “Device Requirements for Optical Interconnects to Silicon Chips,” *Proceedings of the IEEE*, vol. 97, no. 7, pp. 1166–85, 2009.

[2] R. Hesse, J. Nicholls, and N. E. Jerger, “Fine-Grained Bandwidth Adaptivity in Networks-on-Chip Using Bidirectional Channels,” in *Proceedings of the NoCS ’12*. Ieee, May 2012, pp. 132–141.

[3] F. Sibai, “A Two Dimensional Low Diameter Scalable On-Chip Network for Interconnecting Thousands of Cores,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 2, pp. 193–201, 2011.

[4] R. Manevich, I. Cidon, and A. Kolodny, “Handling global traffic in future CMP NoCs,” in *Proceedings of the SLIP ’12*, 2012, pp. 40–47.

[5] N. Abeyratne, R. Das, Q. Li, K. Sewell, B. Giridhar, R. G. Dreslinski, D. Blaauw, and T. Mudge, “Scaling towards kilo-core processors with asymmetric high-radix topologies,” in *Proceedings of the HPCA ’13*. Ieee, Feb. 2013, pp. 496–507.

[6] B. S. Feero and P. P. Pande, “Networks-on-Chip in a Three-Dimensional Environment: A Performance Evaluation,” *IEEE Transactions on Computers*, vol. 58, no. 1, pp. 32–45, Jan. 2009.

[7] E. Socher and M.-C. F. Chang, “Can RF Help CMOS Processors?” *IEEE Communications Magazine*, vol. 45, no. 8, pp. 104–111, Aug. 2007.

[8] R. G. Beausoleil, P. J. Kuekes, G. S. Snider, S.-y. Wang, and R. S. Williams, “Nanoelectronic and Nanophotonic Interconnect,” *Proceedings of the IEEE*, vol. 96, no. 2, pp. 230–247, Feb. 2008.

[9] S. Deb, A. Ganguly, P. P. Pande, B. Belzer, and D. Heo, “Wireless NoC as Interconnection Backbone for Multicore Chips: Promises and Challenges,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 2, no. 2, pp. 228–239, 2012.

[10] S. Abadal, E. Alarcón, M. C. Lemme, M. Nemirovsky, and A. Cabellos-Aparicio, “Graphene-enabled Wireless Communication for Massive Multicore Architectures,” *IEEE Communications Magazine*, vol. 51, no. 11, pp. 137–143, 2013.

[11] N. E. Jerger, L.-S. Peh, and M. Lipasti, “Virtual Circuit Tree Multicasting: A Case for On-Chip Hardware Multicast Support,” in *Proceedings of the ISCA-35*. Ieee, Jun. 2008, pp. 229–240.

[12] T. Krishna, L. Peh, B. Beckmann, and S. K. Reinhardt, “Towards the ideal on-chip fabric for 1-to-many and many-to-1 communication,” in *Proceedings of the MICRO-44*, vol. 2, 2011, pp. 71–82.

[13] T. Krishna and L.-S. Peh, “Single-Cycle Collective Communication Over A Shared Network Fabric,” in *Proceedings of the NoCS ’14*, 2014.

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

[15] P. Conway and B. Hughes, “The AMD Opteron Northbridge Architecture,” *IEEE Micro*, vol. 27, no. 2, pp. 10–21, 2007.

[16] M. Palesi and M. Daneshtalab, Eds., *Routing Algorithms in Networks-on-Chip*. Springer, 2014.

[17] S. Rodrigo, J. Flich, J. Duato, and M. Hummel, “Efficient unicast and multicast support for CMPs,” in *Proceedings of MICRO-41*. Ieee, Nov. 2008, pp. 364–375.

[18] G. Kurian, J. Miller, J. Psota, J. Eastep, J. Liu, J. Michel, L. Kimerling, and A. Agarwal, “ATAC: A 1000-Core Cache-Coherent Processor with On-Chip Optical Network,” in *Proceedings of the PACT*. ACM, 2010, pp. 477–488.

[19] M. Ebrahimi and M. Daneshtalab, “Path-Based Partitioning Methods for 3D Networks-on-Chip with Minimal Adaptive Routing,” *IEEE Transactions on Computers*, vol. 63, no. 3, pp. 718–733, 2014.

[20] S. Abadal, R. Martínez, E. Alarcón, and A. Cabellos-Aparicio, “Scalability-Oriented Multicast Traffic Characterization,” in *Proceedings of NoCS ’14*, 2014, pp. 180–181.

[21] S. Woo, M. Ohara, E. Torrie, and J. Singh, “The SPLASH-2 programs: Characterization and methodological considerations,” *ACM SIGARCH Computer Architecture News*, vol. 23, no. 2, pp. 24–36, 1995.

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

[23] N. Barrow-Williams, C. Fensch, and S. Moore, “A communication characterisation of Splash-2 and Parsec,” in *Proceedings of the IISWC ’09*, 2009, pp. 86–97.

[24] P. Gratz and S. Keckler, “Realistic workload characterization and analysis for networks-on-chip design,” in *Proceedings of the CMP-MSI ’10*, 2010.

[25] Y. Jin, E. Kim, and T. Pinkston, “Communication-Aware Globally-Coordinated On-Chip Networks,” *Parallel and Distributed Systems, IEEE Transactions on*, vol. 23, no. 2, pp. 242–254, 2012.

[26] S. Volos, C. Seiculescu, B. Grot, N. K. Pour, B. Falsafi, and G. De Micheli, “CCNoC: Specializing On-Chip Interconnects for Energy Efficiency in Cache-Coherent Servers,” in *Proceedings of the NoCS ’12*. Ieee, May 2012, pp. 67–74.

[27] V. Soteriou, H. Wang, and L. Peh, “A Statistical Traffic Model for On-Chip Interconnection Networks,” in *Proceedings of the MASCOTS ’06*. Ieee, 2006, pp. 104–116.

[28] J. Bahn and N. Bagherzadeh, “A generic traffic model for on-chip interconnection networks,” in *Proceedings of the NoCArc ’08*, 2008, pp. 22–29.

[29] W. Heirman and J. Dambre, “Rent’s rule and parallel programs: characterizing network traffic behavior,” in *Proceedings of the SLIP ’08*, 2008, pp. 87–94.

[30] M. Lodde, J. Flich, and M. E. Acacio, “Heterogeneous NoC Design for Efficient Broadcast-based Coherence Protocol Support,” in *Proceedings of the NoCS ’12*. Ieee, May 2012, pp. 59–66.

[31] Y. Zhang, B. Ozisikyilmaz, G. Memik, J. Kim, and A. Choudhary, “Analyzing the Impact of On-chip Network Traffic on Program Phases for CMPs,” in *Proceedings of the ISPASS ’09*, 2009, pp. 218–226.

[32] P. Bogdan and R. Marculescu, “Non-stationary traffic analysis and its implications on multicore platform design,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 4, pp. 508–519, 2011.

[33] N. Binkert, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, D. a. Wood, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, and T. Krishna, “The gem5 simulator,” *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, 2011.

[34] L. Kleinrock and F. Tobagi, “Packet Switching in Radio Channels: Part I–Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characteristics,” *IEEE Transactions on Communications*, vol. 23, no. 12, pp. 1400–1416, Dec. 1975.

[35] E. Perelman, M. Polito, J.-Y. Bouguet, J. Sampson, B. Calder, and C. Dulong, “Detecting phases in parallel applications on shared memory architectures,” in *Proceedings of the IPDPS ’06*, 2006, pp. 88–100.

[36] T. Sherwood, S. Sair, and B. Calder, “Phase tracking and prediction,” *ACM SIGARCH Computer Architecture News*, vol. 31, no. 2, pp. 336–349, 2003.