Low-Cost Congestion Management in Networks-on-Chip Using Edge and In-Network Traffic Throttling

Monobrata Debnath
University of Texas at San Antonio, USA
monobrata.debnath@utsa.edu

Dimitris Konstantinou
Democritus University of Thrace, Greece
dimikons5@ee.duth.gr

Chrysostomos Nicopoulos
University Of Cyprus, Cyprus
nicopoulos@ucy.ac.cy

Giorgos Dimitrakopoulos
Democritus University of Thrace, Greece
dimitrak@ee.duth.gr

Wei-Ming Lin
University of Texas at San Antonio, USA
weiming.lin@utsa.edu

Junghee Lee
University of Texas at San Antonio, USA
junghee.lee@utsa.edu

ABSTRACT

Implementing cost effective congestion control within the Network-on-Chip (NoC) is a major design challenge. Whenever congestion awareness and/or mitigation is desired, architects typically rely on the use of adaptive routing algorithms, which aim to (intelligently) balance the traffic load throughout the NoC. Nevertheless, the hardware cost incurred by such solutions is quite considerable, since it entails the collection/propagation of traffic-related information and the provisioning of deadlock freedom guarantees. In this paper, we explore the potential of simultaneous *edge* and *in-network* traffic *throttling*, as a low-cost alternative to adaptive routing techniques. Without any reliance on adaptivity by the routing algorithm, combined throttling is demonstrated to yield better (in most cases) throughput improvements than state-of-the-art adaptive routing algorithms, but at a significantly lower cost.

CCS Concepts

•Computer systems organization → Interconnection architectures;

Keywords

Multi-Core; Network-On-Chip; Congestion management; Low-Cost architecture

1. INTRODUCTION

Over the last several years, the Network-on-Chip (NoC) paradigm has emerged as the de facto communication backbone of multi-/many-core microprocessor architectures. The insatiable desire for increased parallel processing and escalating application demands are expected to elevate the traffic load within the NoC. Poorly managed traffic bursts and uneven load distribution may lead to premature saturation of the network. This, in turn, degrades both network and system performance. Hence, protracted packet latencies due to network congestion could become a major hindrance in future many-core designs.

Traditionally, designers have relied on adaptive routing algorithms to address the issue of congestion control within the NoC [3,7]. Said algorithms improve the overall throughput of the NoC by balancing the traffic load across the network links. Current solutions essentially aim to re-distribute traffic congestion more evenly within the NoC, by sending packets through less congested routes (rather than through the shortest paths). While such approaches have proven to be effective, they incur non-negligible hardware cost. The cost is primarily due to (a) the collection and propagation of congestion information, and (b) the additional resources (virtual channel buffers) required to ensure deadlock freedom. The key observation and fundamental premise of this paper is that one can achieve equally effective load balancing and congestion distribution, through a much cheaper (in terms of hardware) avenue. It will be shown that a combination of *edge* and *in-network* traffic *throttling* can reap equally – if not more – impressive throughput improvements.

Throttling within the Network Interface Controller (NIC) – what we refer to as *edge* throttling – has already been used in prior research [2, 5, 8]. However, it does not effectively balance the load *within* the network, since edge throttling only controls the injection rate. Instead, a combination of simultaneous *edge* and *in-network* throttling will be shown to be much more effective. Toward this end, we explore the potential of a combination of edge and in-network traffic throttling as a low-cost alternative to adaptive routing algorithms. The proposed solution requires minimal additional logic to enable traffic throttling within the NIC and the NoC routers, and it can be used with simple deterministic routing algorithms.

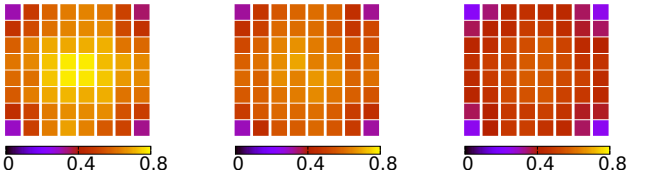
Detailed experimental results and hardware synthesis analysis demonstrate that the proposed technique achieves even better – in most cases – throughput performance than state-of-the-art adaptive routing algorithms, at a significantly lower hardware cost.

2. MOTIVATION

Adaptive routing algorithms balance the traffic load within the network by utilizing detouring paths. To avoid congestion, a packet may follow a path through low-congestion

AISTECS '17 Jan 25, 2017, Stockholm, Sweden

DOI: 10.475/123_4



(a) No congestion management (b) The GCA adaptive routing algorithm [7] (c) Edge and in-network throttling

Figure 1: The average buffer utilization rate observed in the routers of an 8×8 2D mesh NoC, with and without congestion management techniques. Lighter colors indicate higher congestion.

areas, even though the chosen path may not be the shortest path to the packet's destination. As a result, the congestion levels of the most congested routers are lowered, overall, while the congestion levels of the remaining routers are slightly increased. Hence, the congestion levels observed in the NoC are more evenly balanced by re-distributing the traffic load from highly-congested to less-congested areas.

This section aims to visually demonstrate the impact of congestion management techniques (e.g., an adaptive routing algorithm) in the network. Figure 1 shows the average buffer utilization rate in each router of an 8×8 2D mesh NoC. The utilization rate is color-coded, as indicated at the bottom of each diagram in the figure. Note that lighter colors indicate higher congestion. Synthetic one-to-many traffic [4] is used, and the injection rate is 0.41 flits/cycle/node, which is right below the onset of saturation (as also indicated in Figure 5(c)). The details of the simulation framework and all simulation parameters are described in Section 4.

Without any congestion management technique, the congestion levels in the central area of the network are very high, as shown in Figure 1(a). Consequently, the central area becomes the bottleneck that limits the overall throughput of the network. When a state-of-the-art adaptive routing algorithm – GCA [7] – is employed, congestion is more evenly distributed, as shown in Figure 1(b). Since the highest levels of congestion observed in the NoC are now lower than before, the network can sustain higher throughput. The same effect can be achieved by simultaneous edge and in-network throttling, as shown in Figure 1(c), which highlights the average buffer utilization rate under combined throttling. In fact, Figure 1(c) indicates that traffic throttling can balance congestion even more evenly than the adaptive routing algorithm in Figure 1(b).

The results in Figure 1 motivate us to study the potential of combined throttling as a *low-cost* congestion management technique, since throttling incurs much lower hardware cost than adaptive routing.

3. PROPOSED ROUTER ARCHITECTURE

Without loss of generality, the proposed traffic throttling technique employs a 2-stage pipelined speculative NoC router, in order to allow for a fair comparison with two state-of-the-art techniques [3, 7] in the experimental evaluations (those techniques also employ a 2-stage speculative router). Note, however, that the proposed throttling technique is orthogonal to the router pipeline depth; the mechanism can be applied to any router micro-architecture. The selected router architecture is appropriately modified to support both edge (i.e., in the NIC) and in-network throttling.

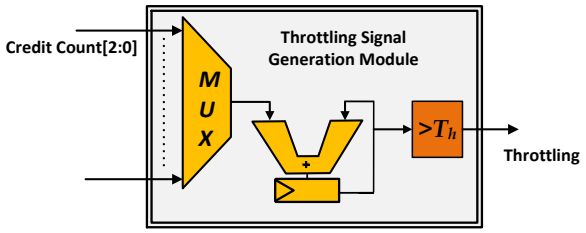


Figure 2: The hardware module that generates the traffic throttling signal in each NoC router.

3.1 Enabling traffic throttling

Traffic throttling is achieved through the use of a simple hardware module (one in each router), which generates a signal to trigger throttling of packets within the NIC and the router. The employed module – called the *Throttling Signal Generation Module* – is shown in Figure 2. The decision to trigger traffic throttling is made (at any given time) based on the current buffer utilization, as indicated by the credit counters within each router. The buffer utilization is calculated by the left portion of the hardware module, which sums the credits from all Virtual Channels (VCs) in all input ports. The buffer utilization is updated once every 40 cycles (this number is dictated by the total number of VC buffers in each router: 5 input ports × 8 VCs/port), so one adder is sufficient to accumulate the credits of all VCs in all ports. The credits of each VC buffer are accumulated cycle-by-cycle, until all 40 credit values are added. The summation result is then used in the right portion of the module, which compares the calculated buffer utilization to a pre-defined threshold value. This threshold value is empirically chosen to be 50% in this work. Thus, a buffer utilization of 50% (or higher) is used as a proxy of elevated traffic accumulating in the region.

The generated throttling signal (i.e., the output of the module in Figure 2) is sent to both the NIC and the Virtual-channel Allocation (VA) logic of the router. The throttling signal is used in conjunction with each packet's requested output port, in order to decide whether a packet should be throttled, or not. If a packet (a) is headed toward an output port connected to a zone of higher congestion (henceforth called a *throttled port*), and (b) the buffer utilization calculated in the current router is higher than a pre-defined threshold (determined by the *Throttling Signal Generation Module* described above), the packet is throttled. The methodology used to determine the particular *throttled ports* in each router is discussed in the following sub-section.

The actual throttling (delay) of the packets is achieved using the minimal masking logic depicted in Figure 3, which is placed in front of the first (local) arbitration stage of a typical separable VA module [6]. The details of how a separable VA module operates are described in [6]. The masking logic masks a request from a VC (denoted by R in Figure 3), if the throttling signal T is asserted, and if the requesting packet is headed toward a congested direction (i.e., if the packet's requested output port P is a *throttled port*, O). Moreover, to prevent starvation, a packet is not throttled more than once. To ensure this, each VC buffer maintains a single-bit flag (F), which indicates whether or not the head-of-line packet in the buffer has already been throttled.

At the NIC, whenever the throttling signal generated by the *Throttling Signal Generation Module* shown in Figure 2 is asserted, all packets within the NIC are throttled, irrespec-

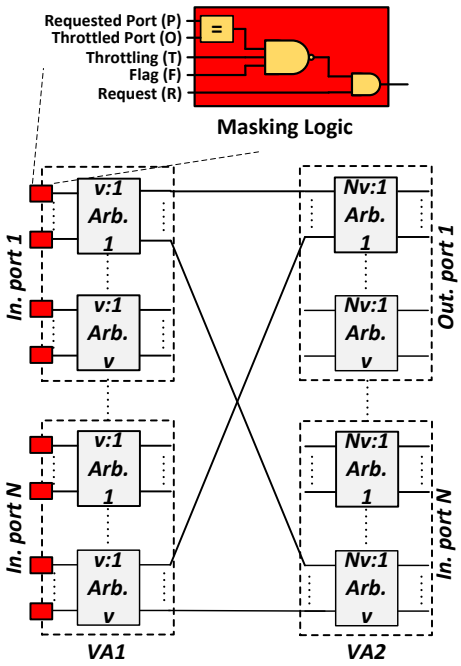


Figure 3: Packet throttling is achieved using minimal masking logic, which is placed in front of the router's Virtual-channel Allocation (VA) module. A separable VA is shown here [6].

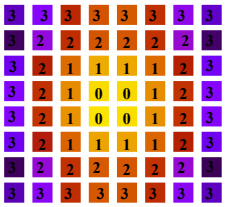


Figure 4: The routers in an 8×8 mesh NoC are grouped into distinct congestion zones. Smaller zone numbers indicate higher congestion. The zones were determined under uniform random traffic and XY routing.

tive of their destination. Similar to *in-network* throttling, each packet in the NIC is throttled at most once. Thus, *edge* (NIC) throttling is guided entirely by the buffer utilization rate.

3.2 Determining the Throttled Ports

The specific output ports to be throttled in each router (the so called *throttled ports*) are determined based on the router's location. Routers are grouped into distinct congestion zones, as shown in the example of Figure 4. These zones correspond to congestion levels observed using uniform random traffic under deterministic XY routing. The number in each square (router) in Figure 4 indicates the router's congestion zone. The zones are numbered in decreasing congestion levels, i.e., a smaller zone number indicates higher congestion. Based on this zone classification, the throttled ports in each router are the ones connected to a higher congestion zone.

In this preliminary work, the congestion zones – and, consequently, the *throttled ports* in each router – are determined

Table 1: Simulated system parameters

Network size & Topology	8×8 2D Mesh
Router Architecture	2-stage speculative
Per-Hop Latency	2 (intra-router stages) + 1 (link)
Switching Technique	Wormhole
Virtual Channels/Port	8
Flit Buffers/VC	5
Packet Length	1∼6 flits (uniformly distributed)
Warmup Cycles	10,000
Simulation Cycles	100,000

statically, based solely on the router's location in the 2D mesh. Specifically, we choose the zone classification depicted in Figure 4. Naturally, a static zone classification will not be able to effectively cope with skewed traffic patterns (e.g., random hot spots). Ideally, the *throttled ports* should be determined *dynamically*, based on the prevailing traffic conditions. Of course, such dynamicity would require exchange of congestion information with neighboring nodes.

Nevertheless, in this paper, we use a *static* setup as a proof of concept, in order to demonstrate the potential of combined traffic throttling. In Section 4, it will be demonstrated that even this static setup can yield very impressive throughput results, under several different traffic patterns. We leave the implementation of *dynamic* selection of *throttled ports* as the next step in our future work.

4. EVALUATION

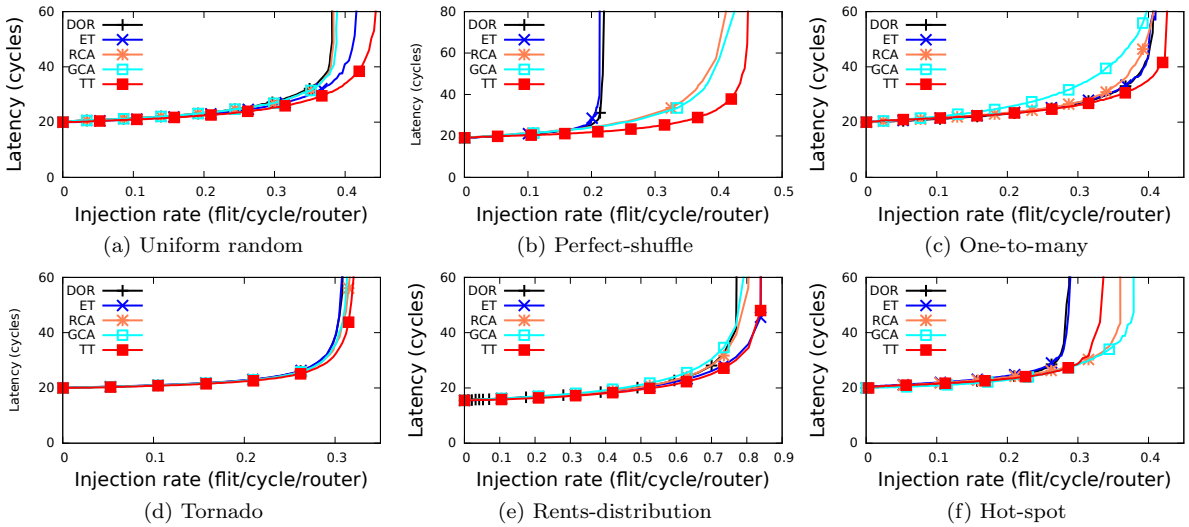
4.1 Performance Analysis

To evaluate the effectiveness of the proposed throttling technique, we employ the cycle-accurate network simulator GARNET [1], which was appropriately modified to model all techniques under investigation. The details of the simulation framework are presented in Table 1.

The proposed Traffic Throttling (TT) technique is compared against baseline XY Dimension-Order Routing (DOR), Edge-only Throttling (ET) [2], and two state-of-the-art adaptive routing algorithms: RCA [3] and GCA [7]. All solutions are evaluated using six different synthetic traffic patterns.

Figure 5 shows the load-latency graphs for the six different traffic patterns. Overall, TT is shown to exhibit excellent throughput behavior, which even outperforms the two adaptive routing algorithms (RCA and GCA), in most cases. Moreover, TT is consistently better than edge-only throttling (ET), thereby highlighting the effectiveness of *combined* throttling. The only traffic pattern that causes TT to behave worse than RCA and GCA is *hot-spot* (Figure 5(f)). This is because of the *static* selection of the *throttled ports*, which cannot identify dynamic hot-spots (as previously explained). Dynamic selection of throttled ports is expected to resolve this issue. On average, the proposed TT technique improves throughput by 26%, 25%, 8.5%, and 5.4%, as compared to DOR, ET [2], RCA [3], and GCA [7], respectively.

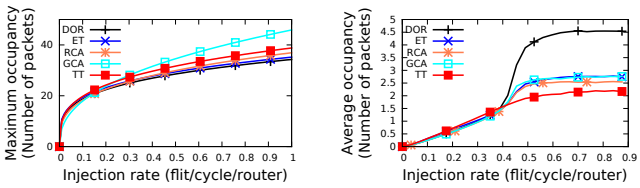
Finally, we investigate the observed buffer occupancy within the NIC, to identify whether traffic throttling necessitates the use of larger buffers. As an example, Figure 6 shows the *maximum* and *average* observed NIC buffer occupancy when the NoC operates under one-to-many traffic [4]. Note that the y-axis scale is different in the two graphs of Figure 6. The *maximum* NIC buffer occupancy under the proposed TT technique increases by a modest 4.7% near the NoC's saturation point, as compared to the baseline DOR design.



(a) Uniform random (b) Perfect-shuffle (c) One-to-many

(d) Tornado (e) Rents-distribution (f) Hot-spot

Figure 5: Latency vs. load curves for the 5 investigated designs, under various synthetic traffic patterns.



(a) Maximum (b) Average

Figure 6: (a) Maximum and (b) average observed NIC buffer occupancy under one-to-many traffic [4]. Note that the y-axis scale is different in the two graphs.

However, this small increase is observed near the NoC's saturation point, where the network is heavily loaded. In terms of *average* NIC buffer occupancy, TT is slightly worse than DOR up until saturation, beyond which TT exhibits much lower NIC buffer occupancy, due to the achieved increase in throughput. Similar trends were observed under other traffic patterns, the results of which are omitted for brevity.

4.2 Hardware Cost Analysis

Compared to the two adaptive routing algorithms (RCA and GCA), TT incurs significantly lower hardware overhead. Specifically, RCA [3] requires: (a) a sideband network for information exchange, (b) aggregation and propagation hardware, and (c) additional escape VCs to prevent deadlocks. Similarly, GCA [7] relies on additional hardware logic to extract and embed congestion information, which requires non-negligible additional storage bits in each router (in addition to extra escape VCs, which are also needed to prevent deadlocks).

On the contrary, TT utilizes minimal extra logic, with no need for a sideband network, no information exchange, and no extra VCs. The TT-driven 2-stage pipelined routers were modeled in SystemVerilog and implemented using Cadence logic synthesis and placement-and-routing tools, driven by a standard-cell library at 45 nm, and based on the configuration shown in Table 1. The obtained results reveal that the TT logic imposes a negligible 0.5% area overhead, and just a 3% delay overhead, relative to the area and delay of a 2-stage pipelined router that does not support traffic throttling (i.e., the DOR setup).

5. CONCLUSION

This paper demonstrates the potential of combined *edge* and *in-network* traffic throttling as a low-cost congestion management alternative to adaptive routing algorithms. The proposed mechanism balances the load more evenly across the NoC, by throttling (delaying) packets that are headed toward congested zones. Even when using a simplistic static selection of ports to be throttled, combined throttling is still demonstrated to achieve better – in most cases – throughput improvements than two state-of-the-art adaptive routing algorithms, but at a substantially lower cost.

6. REFERENCES

[1] N. Agarwal, T. Krishna, L. S. Peh, and N. K. Jha. Garnet: A detailed on-chip network model inside a full-system simulator. In *ISPASS '09*, 2009.

[2] E. Baydal, P. Lopez, and J. Duato. A family of mechanisms for congestion control in wormhole networks. *IEEE Transactions on Parallel and Distributed Systems*, 16(9), Sept 2005.

[3] P. Gratz, B. Grot, and S. W. Keckler. Regional congestion awareness for load balance in networks-on-chip. In *HPCA '08*, 2008.

[4] T. Krishna, L.-S. Peh, B. M. Beckmann, and S. K. Reinhardt. Towards the ideal on-chip fabric for 1-to-many and many-to-1 communication. In *MICRO '11*, pages 71–82, 2011.

[5] U. Y. Ogras and R. Marculescu. Prediction-based flow control for network-on-chip traffic. In *DAC '06*, 2006.

[6] L.-S. Peh and W. J. Dally. A delay model and speculative architecture for pipelined routers. In *HPCA '01*, 2001.

[7] M. Ramakrishna, P. V. Gratz, and A. Sprintson. Gca: Global congestion awareness for load balance in networks-on-chip. In *NoCS '13*, 2013.

[8] M. Thottethodi, A. R. Lebeck, and S. S. Mukherjee. Self-tuned congestion control for multiprocessor networks. In *HPCA '01*, 2001.