2023 Design, Automation & Test in Europe Conference (DATE 2023)

A Lightweight Congestion Control Technique for NoCs with Deflection Routing

Shruti Yadav Narayana[1], Sumit K. Mandal[2], Raid Ayoub[3], Michael Kishinevsky[3], Umit Y. Ogras[1]
[1]Dept. of ECE, University of Wisconsin-Madison; [2]Dept. of CSA, Indian Institute of Science, Bangalore, India;
[3]Intel Corporation, Hillsboro, OR

***Abstract*—Network-on-Chip (NoC) congestion builds up during heavy traffic load and leads to wasted link bandwidth, crippling the system performance. We propose a lightweight machine learning-based technique that helps predict congestion in the network by collecting features related to traffic at each destination and labelling it using a novel time reversal approach. The labelled data is used to design a low overhead and an explainable decision tree model used at runtime congestion control. Experimental evaluations with synthetic and real traffic on industrial 6×6 NoC show that the proposed approach increases fairness and memory read bandwidth by up to 114% with respect to existing congestion control technique while incurring less than 0.01% of overhead.**

I. Introduction

Systems-on-chip (SoCs) with multi-core processors use networks-on-chip (NoCs) for fast and energy-efficient communication. Bufferless NoCs, commonly used in industrial processors, store the packets only at the endpoints. Under heavy traffic, bufferless NoCs deflect packets to one of the available output ports using up NoC resources and causing back-pressure [1]. The back-pressure propagates back to the traffic sources and prevents it from injecting new packets, decreasing the throughput and overall performance.

To address the congestion problem, researchers have proposed congestion control mechanisms for industrial NoCs. These mechanisms reduce the NoC congestion and limits the packet latency in the NoC. However, the resulting throughput is lower under a heavier workload. The paper first aims to maximize and sustain the memory read/write bandwidth and second, aims to maximize the fairness between the LLC hit and miss traffic by proposing a proactive congestion control technique. The technique uses a supervised learning framework enabled by a novel time reversal technique to design a lightweight decision tree which decides whether any given sink node is likely to congest or not. Finally, the decision tree is used at runtime to control the traffic sources. Experimental evaluations with synthetic and realistic traces show that the proposed technique increases memory read bandwidth by up to 114% and the percentage of missed traffic by up to 3.1× compared to a state-of-the-art.

II. Related Work

Existing NoC congestion control techniques can be broadly classified as Global and Local. The global congestion control techniques assess the congestion status of the whole network [2, 3]. In contrast, local congestion control techniques monitor the congestion at each node. State-of-the-art industrial NoCs monitor the ingress queue sizes of each node. However, all the congestion control techniques described above are reactive. Proactive congestion control techniques for NoCs are proposed in [4]. However, industrial NoCs are priority aware and incorporates deflection routing [5]. Therefore, existing proactive congestion control techniques are not applicable to industrial NoCs. To the best of our knowledge, this work is the first proactive congestion control technique proposed for industrial NoCs with deflection routing.

This work was supported by Strategic CAD Labs, Intel Corporation, USA.

III. ML-Based Proactive Source Throttling

A. Features used for Supervised Learning

To construct the machine learning-based model, we first collect the dataset required for training. The dataset consists of features ($\mathcal{F}$) listed in Table I with corresponding labels ($\mathcal{L}$). Here $\mathcal{F} = (f^1, f^2, ..., f^N)$, where N is the number of features, $f^j \in \mathbb{R}$, $1 \leq j \leq N$ and $\mathcal{L} \in \{0, 1\}$. The features ($\mathcal{F}$) are sampled every time a packet arrives at the ingress queue at the sink. Sampling the features in both conditions (sink or bounce) enables us to monitor congestion accurately at sink node.

To capture the features accurately, we compute exponentially weighted moving average (EWMA) of each feature as:

$$\bar{f}_i^j = \alpha f_i^j + (1-\alpha)\bar{f}_{i-1}^j,\ i > 0,\ 1 \leq j \leq N \qquad (1)$$

where $\bar{f}_i^j$ denotes EWMA of the feature f^j for i^{th} packet, f_i^j denotes the original value of the feature f^j (e.g. injection rate) and the α parameter tunes the tracking accuracy and delay. The feature values are smoothened over time by computing EWMA. After deploying the machine learning model, EWMA of only the selected features are tracked at runtime. The detailed explanation of the methodology is presented in [6].

B. Training Data Collection and Decision Tree

Labeling the features: The collected features indicate NoC congestion, however, one must throttle the source before congestion. The main challenge of knowing that a packet will bounce before it's injected into the network is mitigated by using a ***novel time reversal approach*** described next. If a packet is deflected at the sink, we know that the source must have been throttled at the generation time of this packet. This sense of time enables us to go back to the generation time of the deflected packet and label the collected features.

TABLE I
List of features collected at each sink.

Injection rate to the sink queue	Total injection rate (sunk + deflected)
Co-eff. of variation of the total traffic (sunk + deflected)	Co-eff. of variation of inter-arrival time of the traffic to the sink queue
Rate of deflected packets	Mean service time of the sink queue
Co-eff. of variation of deflected packet inter-arrival time	Co-eff. of variation of sink queue inter-departure time
Occupancy	Probability that the sink queue is full
Gradient of injection rate	Gradient of queue occupancy
Gradient of total (sunk + deflected) injection rate	Gradient of probability of sink being full

978-3-9819263-7-8/DATE23/© 2023 EDAA

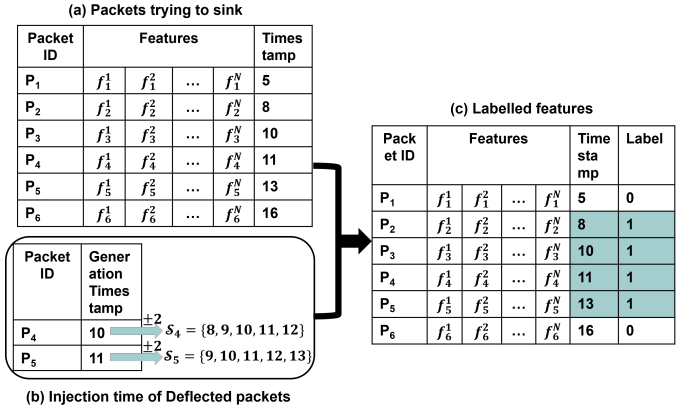

Fig. 1. An illustrative example of our proposed time reversal approach to label the features.

Figure 1(a) shows the features sampled for six packets arriving at the ingress queue for illustration. The timestamps when the deflected packets attempted to sink are sampled along with their generation timestamps. For example, Figure 1(b) has two deflected packets (P_4 and P_5) with generation timestamps (d_j in Equation 2) 10 and 11. Next, we compute a set of timestamps for each deflected packets which are within Δ cycles of the generation timestamps $\mathcal{S}_4$ and $\mathcal{S}_5$. If t_i is the timestamp of the packet P_i, where $1 \leq i \leq 5$, then we label P_i as 1 if $t_i \in \mathcal{S}_4 \cup \mathcal{S}_5$ to indicate congestion at their generation times. In general, we label (l_i) the features of the i^{th} packet arriving at the sink as:

$$l_i = \begin{cases} 1, & \text{if } (d_j - \Delta) \leq t_i \leq (d_j + \Delta) \\ 0, & \text{Otherwise} \end{cases} \quad (2)$$

where $\Delta = 2$ in this example. A label of 1 denotes that if the source sends packet to that particular sink, then it *will result in congestion* and vice versa. Therefore, all the features with timestamp within the range of Δ ($\Delta > 0$) of d_i are labelled as 1. The features of the packets with label of 1 are highlighted in Figure 1(c).

Supervised Learning: In this work, we choose binary decision tree due to its low hardware overhead. The output of the decision tree is either 0 or 1. An output of 1 denotes that there is a possibility of congestion in the near future and cores should stop injecting packets in the NoC. We observe that the decision tree obtained through supervised learning supports our idea of proactive congestion control.

IV. Experimental Evaluations

A. Experimental Setup

We use a cycle-accurate industrial NoC simulator with the NoC architecture similar to the industrial SoCs [1]. Simulation run for 600k cycles with 100k cycles warm-up period. We use a non-inclusive MESI-like cache-coherency protocol [7] with varying traffic and last-level cache (LLC) hit rates.

B. Comparison of Percentage of LLC Miss

Since our proposed technique reduces NoC congestion, more requests with LLC miss are allowed to be fetched from the memory controller. Therefore, our technique consistently results in a higher percentage of requests with LLC miss, as shown in Figure 2. In this case, the synthetic traffic is generated with a 70% hit rate. With the increasing injection rate,

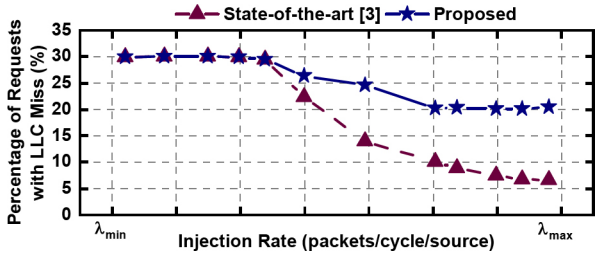

Fig. 2. Comparison of percentage of LLC miss for 70% LLC hit rate (30% LLC miss). Higher percentage of LLC miss indicates that the congestion control technique is more fair towards the miss traffic.

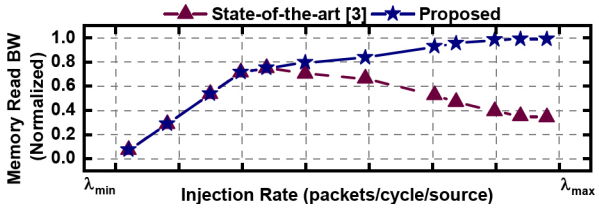

Fig. 3. Comparison of memory read bandwidth for 20% LLC hit rate. Higher memory read bandwidth indicates less NoC congestion.

the percentage of requests with LLC miss reduces, however, our proposed technique shows up to 3.1× improvement. We also observe similar improvements for other LLC hit rates.

C. Comparison of Memory Read Bandwidth

The percentage of missed packets with our proposed technique significantly increases the memory read bandwidth. Figure 2 shows the 70% LLC hit rate comparison between state-of-the-art and proposed techniques. With increasing injection rate, the memory read bandwidth decreases significantly with a state-of-the-art, however, our proposed technique keeps the memory read bandwidth at a certain level. The highest improvement seen is 190%, while on average the proposed technique achieves a 64% improvement.

V. Conclusion and Future Work

State-of-the-art NoC congestion control techniques are reactive. This paper proposes a supervised learning framework along with a time reversal technique to construct a proactive lightweight decision tree which determines if a sink is likely to be congested or not. Experimental evaluation shows that the proposed congestion control technique achieves up to 114% improvement in memory read bandwidth for realistic workloads while incurring less than 0.01% of overhead.

References

[1] Jack Doweck et al. Inside 6th-generation Intel Core: New Microarchitecture Code-named Skylake. *IEEE Micro*, (2):52–62, 2017.

[2] A-H Smai and L-E Thorelli. Global Reactive Congestion Control in Multicomputer Networks. In *Proc. of Intl. Conf. on High Perform. Comput.*, pages 179–186, 1998.

[3] Radu Marculescu, Umit Y Ogras, Li-Shiuan Peh, Natalie Enright Jerger, and Yatin Hoskote. Outstanding research problems in noc design: system, microarchitecture, and circuit perspectives. *IEEE Trans. on Computer-Aided Design of Integrated Circ. and Syst.*, 28(1):3–21, 2008.

[4] Umit Y Ogras and Radu Marculescu. Prediction-based Flow Control for Network-on-Chip Traffic. In *Proc. of DAC*, pages 839–844, 2006.

[5] Sumit K Mandal et al. Performance Analysis of Priority-aware NoCs with Deflection Routing under Traffic Congestion. In *Proc. of ICCAD*, pages 1–9, 2020.

[6] Shruti Yadav Narayana et al. Machine Learning-based Low Overhead Congestion Control Algorithm for Industrial NoCs, 2023.

[7] Mark S Papamarcos and Janak H Patel. A Low-overhead Coherence Solution for Multiprocessors with Private Cache Memories. In *Proc. of ISCA*, pages 348–354, 1984.