Theoretical Analysis and Evaluation of NoCs with Weighted Round-Robin Arbitration

Sumit K. Mandal\textsuperscript{1}, Jie Tong\textsuperscript{1}, Raid Ayoub\textsuperscript{2}, Michael Kishinevsky\textsuperscript{2}, Ahmed Abousamra\textsuperscript{2}, Umit Y. Ogras\textsuperscript{1}
\textsuperscript{1}\text{Dept. of ECE, University of Wisconsin-Madison; \textsuperscript{2}Intel Corporation, Hillsboro, OR}

Abstract—Fast and accurate performance analysis techniques are essential in early design space exploration and pre-silicon evaluations, including software eco-system development. In particular, on-chip communication continues to play an increasingly important role as the many-core processors scale up. This paper presents the first performance analysis technique that targets networks-on-chip (NoCs) that employ weighted round-robin (WRR) arbitration. Besides fairness, WRR arbitration provides flexibility in allocating bandwidth proportionally to the importance of the traffic classes, unlike basic round-robin and priority-based arbitration. The proposed approach first estimates the effective service time of the packets in the queue due to WRR arbitration. Then, it uses the effective service time to compute the average waiting time of the packets. Next, we incorporate a decomposition technique to extend the analytical model to handle NoC of any size. The proposed approach achieves less than 5\% error while executing real applications and 10\% error under challenging synthetic traffic with different burstiness levels.

I. INTRODUCTION

Networks-on-chip continue playing a central role as many-core processors start dominating the server \cite{1, 2} and deep learning \cite{3, 5} market. As the commercial solutions scale up, the latency, area, and power consumption overheads of NoCs become increasingly crucial. Designers need analytical power-performance models to guide complex design decisions during the architecture development and implementation phases. After that, the same models are required by virtual platforms, commonly used to develop and evaluate the software ecosystem and applications \cite{6}. Hence, there is a strong demand for high-fidelity analytical techniques that accurately model fundamental aspects of industrial designs across all segments ranging from systems-on-chip to client and server systems.

NoCs can be broadly classified in terms of buffer usage as buffered and bufferless architectures \cite{7, 10}. Most early solutions adapted buffered techniques, such as wormhole and virtual-channel switching, where the packets (or their flits) are stored in intermediate routers. Area, latency, and energy consumption of buffers have later led to bufferless architectures, where the intermediate routers forward the incoming flits if they can and deflect otherwise. Bufferless NoCs save significant buffer area and enable ultra-fast, as low as single-cycle routing decisions \cite{7, 8}. Therefore, many industrial NoCs used in server and client architectures employ bufferless solutions to minimize the communication latency between the cores, last-level caches (LLC), and main memory \cite{2, 11}. These solutions give priority to the packets already in the network to enable predictable and fast communication while stalling the newly generated packets from the processing and storage nodes. However, buffer area savings and low communication latency come at the cost of the early onset of congestion. Indeed, the packets wait longer at the end nodes, and the throughput saturates faster when the NoC load increases. Moreover, all routers in the NoCs remain powered on, increasing the NoC power consumption. Therefore, there is a strong need to address these shortcomings.

Buffered NoCs with virtual channel routers have been used more commonly in academic work and most recent industry standards \cite{12, 13}. Shared buffering resources, such as input and output channels, require arbitrating among different requesters. For example, suppose that packets in different input channels request the same output channel. An arbiter needs to resolve the conflicts and grant access to one of the requesters to meet performance target. The architectures proposed to date predominantly employ basic round-robin (RR) arbiter to provide fairness to all requesters \cite{14, 15, 16}. Although the decisions are locally fair, the number of arbitrations a packet goes through grows with its path length. Hence, RR arbitration is globally unfair. More importantly, basic RR cannot provide preference to a particular input, which is typically desired since not all requests are equal. For example, data and acknowledgment packets can have higher priority than new requests to complete outstanding transactions, especially when the network is congested.

WRR arbitration provides flexibility in allocating bandwidth proportionally to the importance of the traffic classes, unlike basic round-robin and priority-based arbitration. Each requester has an assigned weight, which is a measure of its importance. A larger weight indicates that the requester is given more preference in arbitration. Due to its generality, WRR arbitration has been employed in several NoC proposals in the literature \cite{17, 19}. Indeed, WRR arbitration enables higher throughput than RR arbitration \cite{18}. Despite its potential, WRR arbitration has not been analyzed theoretically, especially for large-scale NoCs. A large body of literature has proposed performance analysis techniques for buffered and bufferless NoCs since analytical models play a crucial role in fast design space exploration and pre-silicon evaluation \cite{20, 24}. In contrast, no analytical modeling technique has been proposed to date for NoCs with WRR arbitration. A formal analysis is required to understand the behavior of NoCs with WRR arbitration. At the same time, executable performance models are needed to guide many-core processor design and enable virtual platforms for pre-silicon evaluation.

This paper presents a fast, accurate, and scalable perfor-
mance analysis technique for NoCs with WRR arbitration. To the best of our knowledge, it is the first performance analysis technique for NoCs with weighted round-robin arbitration. Furthermore, the proposed technique supports bursty core traffic observed in real applications, which is typically ignored due to its complexity. It first estimates the effective service time of the packets in the queue due to WRR arbitration. Then, it utilizes the effective service time to obtain the average waiting time of the packets. We also propose a decomposition technique to extend the analytical model for any size of NoC. Extensive experimental evaluations show that the proposed analytical model has less than 5% with real applications and 10% error with synthetic traffic having different burstiness levels congesting the NoC.

The major contributions of the work are listed below:

- A novel performance analysis technique for NoCs that employ WRR arbitration,
- A decomposition technique to obtain a scalable analytical model for NoC of any size,
- Experimental evaluations with multiple NoC configurations with different traffic scenarios showing less than 5% error for real applications.

II. RELATED WORK AND NOVEL CONTRIBUTIONS

Analytical models are required to estimate the NoC performance for fast design space exploration and pre-silicon evaluation. Multiple prior studies have proposed NoC performance analysis techniques with basic round-robin arbitration. [21, 25, 26]. Authors in [25] first construct a contention matrix between multiple flows in the NoC. Then, the average waiting time of the packets corresponding to each flow is computed. Support vector regression-based analytical model for NoCs is proposed in [21]. The analytical model proposed in [26] estimates the mean service time of the flows with RR arbitration. The estimated mean service time is used to find the average waiting time of the flows. However, none of these techniques are applicable in the presence of both bursty traffic and WRR arbitration.

Analytical modeling of round-robin arbitration has also been studied outside NoC domain [20, 27, 28]. The techniques presented in [27, 28] incorporate a polling model to approximate the effective service time of a queue in the presence of RR arbitration. However, none of these approaches are applicable when the input distribution to the queue is not geometric. A Markov chain-based analytical model is proposed in [20] to account for bursty input traffic. However, the technique is not scalable for a network of queues. Moreover, none of these techniques are applicable for discrete-time queuing systems. Since each transaction in NoC happens at discrete clock cycles, the analytical models need to incorporate discrete-time queuing systems. The major drawbacks of the prior approaches are summarized in Table I.

| Research        | Approach                  | WRR | Bursty Traffic | Scalable | Discrete Time |
|-----------------|---------------------------|-----|----------------|----------|--------------|
| Boxma et al.    | Polling model             | No  | No             | Yes      | No           |
| Wim et al.      | Extended polling model    | No  | No             | Yes      | No           |
| Wang et al.     | Markov chain              | No  | Yes            | No       | No           |
| Fischer et al.  | Heuristic                 | No  | Yes            | No       | Yes          |
| Vanlerbergee et al. | Moment generating function | No  | No             | No       | Yes          |
| **This work**   | **Queue decomposition**   | Yes | Yes            | Yes      | Yes          |

i.e., WRR, has been used in on-chip communication architectures [17, 18]. Qian et al. compute delay bounds for different channels with different weights to assign appropriate weight to each input channel of the NoC [17]. They show that WRR delivers better quality of service than NoCs with strict priority-based arbitration. Authors in [18] propose a WRR-based scheduling policy. The proposed technique assigns larger bandwidth to input channels with higher weights. It achieves higher throughput compared to round-robin arbitration. Although WRR has shown promise, no analytical modeling approach exists for NoCs with WRR to date.

This paper presents the first performance analysis technique for NoCs with WRR arbitration. It fills an essential gap since WRR can address the shortcomings of priority-based bufferless NoC architectures and the basic round-robin arbitration. Furthermore, the proposed technique supports bursty traffic observed in real applications, which is typically ignored due to its complexity. Hence, it is a vital step towards comprehending the theoretical underpinnings of NoCs with WRR arbitration and enabling their deployment in industrial designs.

III. BACKGROUND AND OVERVIEW

A. Weighted Round-Robin Arbitration

This work uses weighted round-robin arbitration in the NoC routers. The basic operating principle of WRR arbitration is illustrated in Figure 1 for three traffic classes. Packets from each class are first written to a dedicated input queue (also known as a channel). Suppose there are \( N \) input queues \( Q_1, Q_2, \ldots, Q_N \). WRR technique assigns \( Q_i \) a positive integer weight denoted as \( \omega_i \in \mathbb{Z}^+ \) for \( 1 \leq i \leq N \). The WRR arbiter serves \( Q_i \) until it becomes empty. Then, the WRR arbiter serves the subsequent queues following the same principle. After a cycle is completed, the weights are reset to their initial values, as illustrated in Figure 1, and the same arbitration cycle is repeated. In NoCs with WRR arbitration, whenever two

![Fig. 1. Illustration of weighted round-robin arbitration. 'A' is served first, 'I' last. In this example it is assumed that no new packets arrived until all prior packets (A-I) have been served.](image-url)
or more requesters compete for the same resource, they are arbitrated following WRR. WRR arbitration can be used both for arbitrating different virtual channels and different ports in the network.

B. Usage of the Proposed Performance Analysis Technique

WRR arbitration is promising for NoCs since it can tailor the communication bandwidth to different traffic classes. Furthermore, it provides end-to-end latency-fairness to different source-destination pairs, unlike basic round-robin and priority arbitration techniques. However, these capabilities come at the expense of a vast design parameter space. An n×m mesh with P-port routers has \( n \times m \times P \) tunable weights, e.g., an 8×8 2D mesh with 5-port routers would have 320 WRR weights. Due to this ample design space, the current practice is limited to assigning two weights to each router (e.g., one weight to local ports and another weight to packets already in the NoC).

The benefits of the proposed theoretical analysis are two-fold. First, it can enable accurate pre-silicon evaluations and design space exploration without time-consuming cycle-accurate simulations. Second, it can be used to find the combination of weights that optimizes the performance, i.e., to solve the optimization problem in the vast design space described in the previous paragraph. This paper focuses on constructing the proposed analysis technique and its evaluation against cycle-accurate simulation for two reasons. First, the theoretical analysis is complex, and its evaluation deserves a dedicated treatment on its own. Moreover, its application for solving optimization problems requires demonstration of its fidelity first. Multi-objective optimization of the WRR weights using the proposed model is one of our future research directions.

IV. PROPOSED METHODOLOGY AND APPROACH

Figure 2 shows a weighted round-robin arbiter with \( N \) queues. The packets that belong to traffic class-\( i \) are stored in queue \( Q_i \). The corresponding average arrival rate of class-\( i \) packets is denoted by \( \lambda_i \), as summarized in Table I. The burstiness of traffic class-\( i \) can be captured by their squared coefficient of variation of inter-arrival time, denoted by \( C_{Ai} \). Finally, the weight assigned to class-\( i \) is denoted as \( \omega_i \in \mathbb{Z}^+ \) for \( 1 \leq i \leq N \), as illustrated in Figure 2.

To provide a step-by-step derivation, Section IV-A starts with a particular case of the proposed technique tailored to the basic round-robin arbitration, i.e., all weights are set to one. Then, Section IV-B extends the formulation to weighted round-robin arbitration.

1) The first step is to compute the first two moments of the effective service time for each class-\( i \) \( (T_i, C_{Si}) \) of the fully decomposed queue nodes illustrated in Figure 2.

2) In the second step, we use the transformed effective service times \( (\tilde{T}_i, C_{Si}) \) to find the total waiting time (including the queuing delay) of each traffic class. For the clarity of notation and illustration, the derivation below assumes the original two moments of service time \( (T, C_S) \) are the same across all traffic classes.

A. Analytical Model for Basic Round-Robin Arbitration

In the basic round-robin arbitration, all weights are equal to one, i.e., \( \omega_i = 1 \), \( 1 \leq i \leq N \). Each traffic class experiences an additional delay until the arbiter serves the head packets in the other queues. The proposed technique has two steps:

\[ \text{1) The first step is to compute the first two moments of the effective service time for each class-} i \ (T_i, C_{Si}) \text{ of the fully decomposed queue nodes illustrated in Figure 2.} \]

\[ \text{2) In the second step, we use the transformed effective service times} (\tilde{T}_i, C_{Si}, 1 \leq i \leq N) \text{ to find the total waiting time (including the queuing delay) of each traffic class.} \]

For the clarity of notation and illustration, the derivation below assumes the original two moments of service time \( (T, C_S) \) are the same across all traffic classes.

\[ \text{1) Mean effective service time of RR} (\tilde{T}_i): \text{The effective service time accounts for the delay experienced by the packets at the head of each queue. This delay includes its own service time and the service time of the packet at the head of other queues since the round-robin arbiter serves them one by one. When all queues have packets, class-} i \text{ packets will be served every} N \times T \text{ cycles, i.e., they will wait for} (N - 1) \times T \text{ cycles after being served before winning the arbitration again. In general, a traffic class will only contribute to the extra service time if it has a packet waiting for service. Thus, assuming packet arrival of different classes are independent, we can express the mean effective service time as:} \]

\[ \tilde{T}_i = T + \sum_{j=1, j \neq i}^{N} p(n_j > 0 | \tilde{T}_i) p(n_j > 0 | \tilde{T}_i) T \]

\[ = T + T p(n_j > 0 | \tilde{T}_i) \sum_{j=1, j \neq i}^{N} p(n_j > 0 | \tilde{T}_i), 1 \leq i \leq N \]

where \( p(n_j > 0 | \tilde{T}_i) \) denotes the probability that the occupancy of class-\( j \) is greater than zero given a mean effective service period of class-\( i \) is \( \tilde{T}_i \), for \( 1 \leq i, j \leq N \). We approximate \( p(n_j > 0 | \tilde{T}_i) \) using Little’s law as \( \lambda_i \tilde{T}_i \) following the busy-cycle approach. Little’s law gives the expected number of class-\( j \) packets that arrive during the period of \( \tilde{T}_i \), which
Algorithm 1: Basic round-robin arbitration: Effective service time ($\hat{T}_i$) computation.

1. Input: Injection rate of each class ($\lambda$), service time ($T$), number of classes ($N$), Convergence Tolerance ($\delta$)  
2. Output: Effective service time of each class ($\hat{T}_i$)  

for $i = 1:N$ do

1. Find smaller root of the quadratic equation derived from Equation 2:  
   \[ T + T\lambda_i(\hat{T}_i) + \sum_{j=1, j \neq i}^{N} \lambda_j - \hat{T}_i = 0 \]  
   \[ \delta = \text{Tolerance, } k = 1 \]  
   while $\delta \geq \text{Tolerance}$ do

2. $\hat{T}_{i}^{k+1} \leftarrow T + T\min(1, \lambda_i, \hat{T}_i) + \sum_{j=1, j \neq i}^{N} \min(1, \lambda_j, \hat{T}_j) \]  
   $\delta = \hat{T}_{i}^{k+1} - \hat{T}_{i}^{k} \]  
   $k \leftarrow k + 1 \]  

end

end

Algorithm 1 typically takes less than ten iterations due to the bandwidth is divided between the traffic flows assuming all packets have same moments of service time. Due to this fundamental principle, the total queue occupancy $n_{sum}$ is invariant under the arbitration policy. Furthermore, this sum can be computed with near-perfect accuracy using maximum entropy-based analytical queuing model of multi-class given in [30]. One strong aspect of this model is that it can take bursty traffic that conforms to General Geometric distribution (GGeo). By leveraging this model, we can embed the capability of handling bursty traffic into our analytical model. The total queue occupancy ($n_{sum}$) can be calculated as:

\[ n_{sum} = \frac{1}{2} \sum_{i=1}^{N} \left( \rho_i (C_{Ai} - 1) + \sum_{k=1}^{N} \frac{\lambda_i}{\lambda_k} \rho_k^2 (C_{Ak} + C_S) \right) \frac{1}{1 - \sum_{k=1}^{N} \rho_k} \]  

(4)

using the parameters summarized in Table 1. We do not delve into the derivation of this equation for clarity since it is not our major focus. The detailed derivation can be found in the seminal work by Kouvatios et al. [40].

Next, we derive $n_{sum}$ using the first two moments of transformed effective service time. The occupancy of each queue $Q_i$ can be expressed using Little’s law as: $n_i = \lambda_i W_i$, where $W_i$ is the average waiting time of class-$i$. Therefore, we obtain:

\[ n_{sum} = \sum_{i=1}^{N} \lambda_i W_i \]  

(5)

The average waiting time $W_i$ consists of two components. The first one is the waiting time in the queue till the packet reaches the head of the queue ($\hat{W}_i$), i.e., until all other class-$i$ packets leave the server completely. The second component accounts for the waiting time at the head of the queue when the server is busy serving other classes (class-$j$, $j \neq i$). This component is the additional time captured by the effective service time $\hat{T}_i$, i.e., the difference between the effective and original service times: $\Delta T_i = \hat{T}_i - T$. Furthermore, the waiting $\hat{W}_i$ is expressed as a function of the residual time $\hat{T}_i$ [31]. Hence,

\[ W_i = \frac{\hat{R}_i}{1 - \lambda_i T_i} + \Delta T_i \text{, for } 1 \leq i \leq N \]  

(6)

We leverage the property that all individual mean effective residual service time of classes are equal [27]; hence we write ($R = \hat{R}_i$, $1 \leq i \leq N$). Thus, we can rewrite the total occupancy by plugging Equation 6 into Equation 5 and then solve for the residual time $R$:

\[ n_{sum} = \sum_{i=1}^{N} \lambda_i \left( R \frac{1}{1 - \lambda_i T_i} + \Delta T_i \right) \]  

\[ \Rightarrow R = \left( \frac{n_{sum}}{\sum_{i=1}^{N} \lambda_i} \right) \left( \frac{\lambda_i}{1 - \lambda_i T_i} \right)^{-1} \]  

(7)

Then, we can compute the coefficient of variation $\hat{C}_{Si}$ by substituting the residual time $R$ from Equation 7 into Equation 3.

3) Average waiting time of RR ($W_i$): Finally, we compute the mean waiting time of individual classes by plugging $\hat{T}_i$ and $\hat{C}_{Si}$ found for RR arbitration into Equation 3 and Equation 6 as:
}\[ W_i = \frac{0.5 \bar{T}_i(\bar{\rho}_i - 1 + C_{Ai} + \bar{\rho}_i \bar{C}_{Si})}{1 - \lambda_i \bar{T}_i}, \quad 1 \leq i \leq N \] (8)

B. Analytical Model for Weighted Round-Robin Arbitration

This section extends the analytical model constructed in Section IV-A to WRR using the same steps. It first describes the derivation of the first two moments of effective service time. Then, it will use these moments to find the average waiting time.

1) Mean effective service time of WRR (\( \bar{T}_i \)): In WRR, the weight of each class can be larger than or equal to one, i.e., \( \omega_i \geq 1 \), \( 1 \leq i \leq N \). To illustrate the generalization, we start with a system of two queues, where \( \omega_1 > 1 \) and \( \omega_2 = 1 \). Then, we generalize it to \( N \) queues with arbitrary weights.

Mean effective service time of class-1 with \( \omega_1 > 1 \) and \( \omega_2 = 1 \): Since \( \omega_1 > 1 \), a batch of up to \( \omega_1 \) number of class-1 packets are served without interruption. Let \( \bar{T}_i^b \) be the total service time of \( \omega_1 \) class-1 packets. When there is a class-2 packet in \( Q_2 \), the next batch of \( \omega_1 \) class-1 packet will wait. Therefore, we can express \( \bar{T}_i^b \) as:
\[ \bar{T}_i^b = \omega_i T + p(n_1 > 0|\bar{T}_i^b) p(n_2 > 0|\bar{T}_i^b) T \] (9)

Note that this equation generalizes Equation 1 to arbitrary \( \omega_1 \) for two queues. Using Little’s law and the methodology described in Section IV-A, we approximate \( \bar{T}_i^b \) as:
\[ \bar{T}_i^b = \omega_i T + \omega_1 \min(1, \lambda_1 \bar{T}_i^b) \min(1, \lambda_2 \bar{T}_i^b) \] (10)

Mean effective service time of class-2 with \( \omega_1 > 1 \) and \( \omega_2 = 1 \): The batch size of class-2 packet is one in this case since \( \omega_2 = 1 \). They need to wait in the queue for a maximum of \( \omega_1 \) class-1 packets. Therefore, \( \bar{T}_2 \) is expressed as:
\[ \bar{T}_2 = T + \sum_{i=0}^{\omega_1-1} p(n_1 > i|\bar{T}_2) p(n_2 > 0|\bar{T}_2) T \] (11)

We can approximate \( p(n_1 > i|\bar{T}_2) \) as \( \frac{1}{i+1} p(n_1 > 0|\bar{T}_2) \) to obtain:
\[ \bar{T}_2 = T + \sum_{i=0}^{\omega_1-1} \frac{1}{i+1} p(n_1 > 0|\bar{T}_2) p(n_2 > 0|\bar{T}_2) T \]
\[ = T + T\min \left(1, \sum_{i=0}^{\omega_1-1} \frac{1}{i+1} \lambda_i \bar{T}_i \right) \min(1, \lambda_2 \bar{T}_2) \] (12)

To get to the final form of Equation 12 above, we approximate the conditional probabilities based on [27].

Generalization to \( N \) queues and arbitrary weights: In general, there are \( N \) classes with weights \( \omega_i, 1 \leq i \leq N \). Consider the class-\( i \) packets with the total service time \( \bar{T}_i^b \) for a batch of \( \omega_i \) packets. The first part of the effective service time will be due to the packets of same class. This effect is captured in our two-queue illustration for class-1 packets in Equation 9. The second part of the effective service time will be due to the packets of other classes. This effect is captured in our two-queue illustration for class-2 packets in Equation 11. By combining them, we can express the generalized effective service time as:
\[ \bar{T}_i = \omega_i T + p(n_1 > 0|\bar{T}_i) \sum_{j=1,j\neq i}^{N} \left( \sum_{k=0}^{\omega_j-1} p(n_j > k|\bar{T}_i) \right) T \] (13)

Algorithm 2: Obtaining effective service time (\( \bar{T}_i \)) of weighted round-robin
1. Input: Injection rate of each class (\( \lambda \)), WRR weights (\( \omega_i \)), service time (\( T \)), number of classes (\( N \))
2. Output: Effective service time of each class (\( \bar{T}_i \))
3. for \( i = 1 \) to \( N \) do
   4. Find smaller root of the quadratic equation derived from Equation 14 for \( \bar{T}_i^b \):
   \[ \left( \bar{T}_i^b \right)^2 T \omega_i \lambda_i \sum_{j=0}^{\omega_i-1} \frac{1}{\sum_{k=0}^{k=0} \lambda_j} - \bar{T}_i^b + \omega_i T = 0 \]
   5. \( \delta = \text{Tolerance} \), \( k = 1 \)
   6. while \( \delta \geq \text{Tolerance} \) do
   7. \[ Z = \sum_{j=1}^{N} \min(1, \sum_{k=0}^{k=0} \lambda_j \bar{T}_j^b) \]
   8. \[ \bar{T}_i^b \leftarrow \omega_i T + \frac{T}{\omega_i} \min(1, \lambda_i \bar{T}_i^b) Z \]
   9. \[ k \leftarrow k + 1 \]
   10. end
   11. \( \bar{T}_i = \frac{\bar{T}_i^b}{\omega_i} \)
   12. end

Note that this equation reduces to Equation 9 for \( i = 1 \), \( N = 2 \), \( \omega_1 = 1 \), and to Equation 11 for \( i = 2 \), \( N = 2 \), \( \omega_2 = 1 \).

Finally, we can obtain the generalized effective service times by replacing \( p(n_i > 0) \) with \( \min(1, \lambda_i \bar{T}_i^b) \) as follows:
\[ \bar{T}_i = \omega_i T + \frac{T}{\omega_i} \min(1, \lambda_i \bar{T}_i^b) \sum_{j=1}^{N} \min(1, \sum_{k=0}^{k=0} \lambda_j \bar{T}_j^b) \]
\[ = \bar{T}_i^b \] (14)

Due to the non-linearity introduced by the \( \min \) operation, we compute \( \bar{T}_i \) using the iterative Algorithm 2 just like the basic round-robin case. The quadratic nature of Equation 14 enables a fast convergence within 10 iterations under 5\( \mu \)s.

2) Coefficient of variation of effective service time of WRR (\( \bar{C}_{Si} \)): In the basic round-robin case, the residual service time \( R_i \) accounts for the second-order moment of the service time, i.e., \( C_{Si} \). However, effective mean residual service time of different classes are not necessarily equal to each other for arbitrary weights, which is the case in WRR arbitration. The mean effective residual service time of class-\( i \) packets is expressed as:
\[ \bar{R}_i = 0.5 \bar{T}_i (\bar{\rho}_i - 1 + C_{Ai} + \bar{\rho}_i \bar{C}_{Si}) \] (15)

where \( \bar{T}_i \) is found from Algorithm 2 based on Equation 14.

Note that the residual service time depends on the first two moments of service time, which are the average service time \( \bar{T}_i \) and the coefficient of variation \( C_{Si} \), respectively. To approximate \( C_{Si} \) for WRR, we leverage its monotonicity decreasing behavior as a function of the corresponding weight. This behavior is expected since a larger weight corresponds to fewer arbitration stages, decreasing the effective service time variation. This observation indicates that the coefficient

1We use the same notation for RR and WRR arbitration since WRR results are generalizations of basic RR expressions.
of variation for round-robin arbitration is an upper bound of coefficient of variation for weighted round-robin arbitration. This upper bound, \( C_{Si}(RR) \) for class-\( i \), is computed using Equation 3. Given this, we first apply linear approximation to calculate the \( C_{Si} \) under WRR as \( \alpha C_{Si}(RR) \), note we use the squared value of \( \omega \) since \( C_{Si} \) is the square of coefficient of variation as defined in Table II. We add the parameter \( \alpha \) to tighten the approximation. Next, we recall Equation 4 and replace \( C_{Sk} \) by \( \alpha \frac{C_{Sk}(RR)}{\omega^2} \) to obtain:

\[
n_{sum} = \frac{1}{2} \sum_{i=1}^{N} \left( \hat{\rho}_{i}(C_{Ai} - 1) + \frac{\sum_{k=1}^{N} (\lambda_{i} - \lambda_{k})}{\lambda_{k}} \left( C_{Ak} + \alpha \frac{C_{Sk}(RR)}{\omega^2} \right) \right) \frac{1}{1 - \sum_{k=1}^{N} \hat{\rho}_{k}} \tag{16}
\]

The left hand side of the Equation 16 \( (n_{sum}) \) is obtained by applying Equation 9. Therefore, \( \alpha \) is the only unknown in Equation 16. Once we obtain \( \alpha \) (by solving Equation 16), we compute \( C_{Si} \) under WRR as \( \alpha \frac{C_{Si}(RR)}{\omega^2} \). This expression captures traffic burstiness addressing one of the significant drawbacks of the prior work.

3) **Average waiting time of WRR \( (W_i) \):** So far, we obtained first two moments of effective service time, i.e., \( \hat{T_i} \) (Equation 14) and \( \hat{C}_{Si} \). We obtain the average waiting time under WRR arbitration by plugging them to Equation 17:

\[
W_i = 0.5\hat{T_i}(\hat{\rho}_{i} - 1 + C_{Ai} + \hat{\rho}_{i}\bar{C}_{Si}) + \Delta T_i \tag{17}
\]

**C. Estimation of end-to-end latency**

The previous section described the canonical model for WRR arbitration, where all traffic classes go through a single arbiter. The packets in NoCs go through a sequence of WRR arbiters while traveling from their sources towards destinations. For example, an 8\times8 2D mesh has 64 routers with at least one arbiter per output queue in each of them. Figure 3(a) illustrates an example with two stages, where the flow that wins the first stage is routed to the second one. The major challenge in modeling multiple stages is obtaining the inter-stage distribution, which becomes the arrival flow at the subsequent stages. For instance, the coefficient of variation of arrival flow 2 in the second stage is equal to the coefficient of variation of the departure flow from stage-1 \( (C_{Di}) \) in Figure 3(a). We first find the squared co-efficient of variation of inter-departure time for each traffic class \( (C_{Di}) \):

\[
C_{Di} = \rho_{i}^2(\bar{C}_{Si} + 1)(1 - \rho_{i})C_{Ai} + \rho_{i}(1 - 2\rho_{i}) \quad \forall 1 \leq i \leq N \tag{18}
\]

Next, we apply the decomposition technique [32] to obtain the departure distribution at each stage \( (C_{Di}) \).

| Stage-1 | Stage-2 |
|---------|---------|
| \( (\lambda_1, C_{A1}) \) | \( (\lambda_1, C_{A1}) \) |
| \( (\lambda_2, C_{A2}) \) | \( (\lambda_2, C_{A2}) \) |
| \( (\lambda_3, C_{A3}) \) | \( (\lambda_3, C_{A3}) \) |
| \( (\lambda_4, C_{A4}) \) | \( (\lambda_4, C_{A4}) \) |
| \( (\lambda_5, C_{A5}) \) | \( (\lambda_5, C_{A5}) \) |
| \( (\lambda_6, C_{A6}) \) | \( (\lambda_6, C_{A6}) \) |
| \( (\lambda_7, C_{A7}) \) | \( (\lambda_7, C_{A7}) \) |
| Transformation | Transformation |

**Algorithm 3:** Estimation of end-to-end latency

1. **Input:** NoC size \( (n \times m) \), injection rates and co-efficient of variation of inter-arrival times for all classes \( (\lambda, C_A) \), Service time \( (T) \), WRR weights
2. **Output:** Average NoC \( (L) \) and source–dest. latencies \( (L_{sd}) \)
3. **Number of routers** \( N_p \) \( \leftarrow \) \( n \times m \), \( \lambda_{sum} \leftarrow 0, L_{sum} \leftarrow 0 \)
4. **for** \( s = 1:N_p \) **do**
5. **for** \( d = 1:N_p \) **do**
6. \( W_{sd} = \text{compute_waiting_time}(s,d) \)
7. \( L_{sd} = W_{sd} + \text{free_packet_delay}(s,d) \)
8. \( L_{sum} \leftarrow L_{sum} + L_{sd} \), \( \lambda_{sum} \leftarrow \lambda_{sum} + \lambda_{sd} \)
9. **end**
10. **end**
11. \( L = \frac{L_{sum}}{\lambda_{sum}} \)
12. **function** \( \text{compute_waiting_time}(s,d) \)
13. \( S \leftarrow \) The number of stages for the flow with source \( s \) and destination \( d \), \( W_{sd} \leftarrow 0 \)
14. **for** \( j = 1:S \) **do**
15. \( W_j = \text{Waiting time at stage } j \) using Equation 17
16. \( W_{sd} \leftarrow W_{sd} + W_j \)
17. \( C_D = \text{Coefficient of variation using Equation 19} \)
18. Use \( C_D \) as the \( C_A \) in the next stage
19. **end**
20. **return** \( W_{sd} \)
21. **end**

\[
C_D = \frac{\sum_{i=1}^{N} \lambda_i C_{Di}}{\sum_{i=1}^{N} \lambda_i} \tag{19}
\]

The departure distribution \( (C_D) \) is the arrival distribution to the next stage \( (C_{A2}) \). For a given source-destination pair, these calculations are performed at each arbitration stage on the path of the packets.

**Algorithm 3** presents a step-by-step procedure to obtain end-to-end latency of a given NoC with WRR arbitration and deterministic routing. The input to the algorithm is the NoC size, injection rate, and coefficient of variation of inter-arrival time of all classes, service time of the queues, and weights assigned to each arbiter. The output of the algorithm is the latency of each source-destination pair \( (L_{sd}) \) as well as the average latency \( L \).

**Algorithm 3** first computes the waiting time of each source-destination pair \( (W_{sd}) \). The function \( \text{compute_waiting_time} \) in lines 12–22 describes the procedure for this computation. The procedure starts with finding the number of stages \( (S) \) for the flow from source \( s \) to destination \( d \). At each stage, it computes the average waiting time using Equation 17 (line 16). Next, it adds the waiting time at the current state to the cumulative waiting time found so far (line 17). Then, it computes the coefficient of variation of inter-departure time.
We first evaluate the proposed analytical model under basic round-robin arbitration. Figure 4 depicts representative average end-to-end latency comparisons between analytical model and the cycle-accurate simulations for 8×1 ring and 8×8 mesh NoC. The average injection rates for both NoC configurations are varied until the NoC becomes highly congested, following geometric distribution. On average, the proposed technique incurs only 5% error for 8×1 ring and 7% error for 8×8 mesh. We also compare the proposed analytical model with a polling-based model for round-robin arbitration [28]. The polling-based model grossly overestimates the latency both for 8×1 and 8×8 NoC. A comprehensive summary for other NoC sizes and traffic patterns is presented in Section V-E. These results demonstrate that the proposed approach is accurate for difference NoC, traffic, and WRR parameters.

C. Results with Weighted Round-Robin Arbitration

The most important aspect of the proposed technique is considering WRR arbitration. Figure 5 shows the average end-to-end latency comparison of our analytical model against cycle-accurate simulations for an 8×8 mesh for three different weight configurations. The proposed analytical model incurs on average 8% error when the WRR weights associated with the traffic channels of external input to the NoC and channels connected to internal routers are 1 and 2, respectively. The average error slightly increases to 9% when the arbitration weights of the packets in the NoC increase to 3. This configuration decreases the network congestion by providing higher priority to the packets already in the NoC. Hence, the average waiting time decreases. Since there are no other analysis approaches for WRR, this section does not provide comparisons to state-of-the-art.

D. Results with Bursty Traffic and Real Applications

This section evaluates the proposed model in the presence of bursty traffic. We define probability of burstiness as \( p_{\text{burst}} \) of the general geometric distribution [30]. Increasing \( p_{\text{burst}} \) denotes increasing burstiness in the traffic. We note that no other prior work considered weighted round-robin arbitration together with bursty traffic. Figure 6 compares the end-to-end average latency results against cycle-accurate simulations for 6×6 and 8×8 mesh when the arbitration weights associated with the packets in the NoC and external packets arrivals are 3 and 1, respectively. When the burst probability is \( p_{\text{burst}} = 0.1 \) (Figure 6(a)), the average modeling errors for 6×6 and 8×8 mesh are 7% and 4%, respectively. When the burst probability increases to 0.3 (Figure 6(b)), the average errors become 6% and 5%, respectively.

Finally, we evaluate the proposed analysis technique while running applications from the SPEC CPU2017 [34] and PARSEC benchmark [35]. The applications from SPEC CPU2017 benchmarks show burstiness in the range of \( p_{\text{burst}} = 0.1 - 0.6 \).
degree of skewness, the error for moderate NoC loads ranges from 4.3% to 8.1% for all burstiness levels, WRR weight configurations, and NoC sizes. Even when the NoC is pushed to congestion, the modeling error remains under 9.0% for all configurations, except for 8 × 8 NoC with WRR weights 3 and 1. The error in this specific case is 13%; the error for 8 × 1 and 6 × 6 NoCs are 4.6% and 5.4%, respectively.

F. Execution Time of the Proposed Analysis Technique

We implemented the proposed technique in C++ to obtain end-to-end latency. The computational complexity of the proposed model is \(O(n^2 \times m^2 \times p)\), for \(n \times m\) NoC (\(n \geq m\)) and WRR arbiters with maximum \(p\) ports. The execution time as a function of NoC sizes with 3 output ports per router are summarized in Table V. We observe that even for 32 × 32 NoC, the analytical model takes only 45.07 ms to execute, while the simulation time of the 8 × 8 NoC is in the order of minutes. These results show that the proposed analysis technique is lightweight and provides four orders of magnitude speed-up compared to cycle-accurate simulations.

| NoC size | 4 × 4 | 6 × 6 | 8 × 8 | 16 × 16 | 32 × 32 |
|---------|-------|-------|-------|---------|---------|
| Exe. time (ms) | 0.56 | 1.41 | 2.25 | 13.52 | 45.07 |

VI. CONCLUSION AND FUTURE WORK

This paper presented a scalable performance analysis technique for NoCs with WRR arbitration. WRR arbitration is promising since it enables latency fairness and custom bandwidth allocation to meet the requirements of individual traffic flows, unlike basic round-robin and priority-based arbitrations. The proposed technique handles bursty traffic and provides high accuracy for both ring topologies used in client systems and large 2D mesh topologies used in servers. Extensive evaluations show the proposed approach achieves less than 5% error while executing real applications and 10% error under challenging synthetic traffic with different burstiness levels. Hence, it can be used for fast and accurate design space exploration as well as pre-silicon evaluations. One of our future directions is finding the WRR weights to meet individual traffic flows’ bandwidth and latency requirements.
