Delay Analysis and Buffer Sizing for Priority-Aware Networks-on-Chip (NoC)

Baoliang Li, Zeljko Zilic, Wenhua Dou,

Abstract—Among all the implementation alternatives of Networks-on-Chip (NoC), priority-aware wormhole-switched NoC is promising to meet both the worst-case and average-case performance requirements of on-chip communication. The worst-case end-to-end delay and buffer requirement analysis are very important for the development of real-time applications on this platform. In this paper, we first build a Real-Time Calculus (RTC) based performance model for the priority-aware wormhole-switched NoC. Then, we propose an end-to-end delay analysis algorithm and a buffer sizing algorithm based this model. The latency analysis algorithm can give much tighter delay bound than the existing deterministic network calculus method, since it take the maximum service capacity and minimum arrival traffic into consideration. The buffer sizing algorithm try to reduce the buffer space required for each flow without violating the deadline constraint, which improves the existing backlog bound obtained by link-level buffer-space analysis method. Our algorithms are topology-independent, taking as input the network topology graph and the traffic characterization, our algorithms can give the end-to-end delay bound and buffer requirement for each traffic flow. Our model enables the fast performance evaluation and buffer allocation of priority-aware wormhole-switched NoC, which can be used for application mapping, routing selection and power reduction, etc. Experiment results demonstrate the effectiveness and tightness of our model. In addition, further comparisons with other theoretical models also indicates that, our method outperforms the existing methods while the tightness of delay and backlog bounds are considered.

Keywords—Networks-on-Chip (NoC), priority-aware, real-time calculus, delay bound, buffer optimization

I. INTRODUCTION

The conventional on-chip interconnection paradigms, e.g. bus, ring and point-to-point links, are not able to meet the strict and complex communication requirements of modern large scale Chip-MultiProcessors (CMPs) and System-on-Chip (SoC). As an alternative, Networks-on-Chip (NoC) is proposed to provide better scalability and higher power efficiency. As a key component of CMPs and SoC, NoC must be well designed to meet the rigorous requirements on performance, power and design cost. Although various proposals have emerged, each focusing on improving different performance metrics of on-chip interconnects, e.g. end-to-end latency, throughput and power, most of the existing research on NoC are focusing on the improvement of average performance, and simulation is the most widely used performance evaluation method. Whereas,

Baoliang Li and Wenhua Dou are with the College of Computer Science, National University of Defense Technology, Changsha 410073, P.R. China

Zeljko Zilic are with Department of Electrical & Computer Engineering, McGill University, Montreal H3A-2A7, Quebec, Canada

there also exists lots of on-chip applications, which are sensitive to the worst-case or real-time communication performance of NoC, e.g. cache coherent protocol [1] and multimedia application [2]. How to design the on-chip communication infrastructure for these applications and analyze its feasibility are of a big challenge for the researchers.

To meet the rigorous Quality-of-Service (QoS) requirement, various special hardware implementations have been proposed, e.g. Time-Division Multiplexing-Access (TDMA) [3], circuit-switch [4] and time-triggered switch [5]. Although providing strict real-time communication guarantee, the average performance and resource utilization of these proposals are very poor. In contrast, wormhole-switched NoC is widely used in on-chip network due to its simplicity and high-efficiency. Thus, providing real-time communication support on the conventional wormhole-switched NoC to meet both average-case and worst-case communication requirements is the most promising solution. To achieve this goal, a special scheduling policy (e.g. DifServ [6] or priority-aware implementation [7][8][9] or flow control mechanism (e.g. [10][11]) should be integrated into the conventional wormhole-switched NoC. For all these real-time communication proposals based on wormhole-switched NoC, a key step before their adoption as the platform of real-time applications is the analysis of the worst-case communication delay for all the real-time flows to guarantee the satisfaction of deadline constraint. An effective buffer analysis approach is also needed to optimize the buffer allocation under real-time constraint, since buffer usually consumes about 46% power [12] and occupies 30% area [13] of entire router.

A creditable and accurate worst-case performance analysis is crucial for the application of wormhole-switched NoC in real-time communication, since an overoptimistic estimation will lead to the violation of deadline, while an overpessimistic estimation will make the utilization of on-chip resource very low. The conventional simulation-based evaluation method is not competent for the worst-case performance analysis, because the worst-case scenarios are hard to be captured by simulation. As an alternative, the analytical method can establish the relationship between performance metrics and design parameters in a very short time, and give the worst-case performance immediately. For the worst-case analysis of fixed-priority wormhole-switched on-chip networks, Flow-Level Analysis (FLA) [7], Link-Level Analysis (LLA) [14][15] and Deterministic Network Calculus (DNC) [16] have been successfully applied to analyze the end-to-end delay and backlog. Both FLA and LLA have their roots in the classic scheduling theory, which assume that the traffic flows are strictly periodic,

and the buffer size of wormhole-switched NoC is sufficient large, so that the back-pressure caused by flow control can be ignored. In addition, they simplify the router model by restricting the per-hop latency to be one cycle, which is complying with the conventional router implementation.

Deterministic network calculus based method [16] overcomes these limitations by applying the advanced operators and properties of DNC. However, we found that the DNC based performance bound in [16] can be further improved if we take the maximal service curve of on-chip routers and minimal arrival curve of traffic into consideration. These two curves can be used to improve the tightness of output arrival curve of current router and leftover service curve of the downstream routers. The improved leftover service curve further tightens the obtained performance bounds for low-priority flows (please refer to Theorem 1.6.2 in [17] for more details). Motivated by this observation, we adopt the Real-Time Calculus (RTC) theory [18] originally used for the real-time analysis of task scheduling to build an end-to-end performance model for the wormhole-switched NoC with credit-based flow control. Real-time calculus integrates the minimum arrival curve and maximum service curve into DNC theory to characterize more detailed information about the traffic and service process. Compared with the DNC model in [16], our model can significantly improves the tightness of performance bounds, we will further explain the reason and demonstrate the improvement in Section V. The main contribution of this paper is twofold: (1) We propose an end-to-end delay analysis algorithm for the priority-aware wormhole-switched NoC based on the RTC model. The output of this algorithm can be used as the reference of IP core mapping, task mapping, routing selection, or NoC parameters configuration, etc. (2) We also propose an RTC based buffer sizing algorithm for the priority-aware wormhole-switched NoC to improve the buffer allocation under deadline constraint. This is a significant improvement of previously proposed FLA and LLA based buffer optimization method [15], which can be used to minimize the power consumption and chip area.

The rest of this paper is organized as follows: we present the existing real-time communication proposals and its related performance analysis methods in Section II. In Section III, the basic assumptions on priority-aware wormhole-switched NoC and a brief introduction to RTC theory is presented. The detailed modeling process are presented in Section IV, where we also propose the end-to-end delay analysis algorithm and buffer sizing algorithm. We present the experimental results and comparison with other analytical methods in Section V. Finally, we summarize our paper in Section VI.

II. RELATED WORK

Since first introduced in 2001 [19], various NoC proposals have emerged to meet different on-chip communication requirements. The main requirements posed to NoC by on-chip applications are latency and bandwidth. To meet these demands, NoC are designed to be either best-effort or guaranteed-service, depending on the hardware cost and application requirements. Best-effort NoC can make better use of the on-chip shared resource, but it does not necessarily provide any performance guarantee for the applications. To provide guaranteed-service for different applications, a simple and effective solution is classifying these applications into several service classes, each with different priorities, and the network provides services according to the priority of each class. Representative implementations of this idea include QNoC [20], fixed-priority NoC [7] and Æthereal [3] etc. The performance evaluation method for both best-effort and guaranteed service NoC include the average analysis and worst-case analysis. For the average analysis, simulation- and probability-based methods hold the dominant position for both of these two categories. However, for the worst-case analysis, simulation is not competent due to the difficulty in covering all the corner cases. The analytical worst case analysis for these two categories is also slightly different.

Synchronous Data Flow (SDF) graph [21] and DNC [22] have been presented to model the worst case performance bound of best-effort NoC. The former method assumes the traffic flow to be periodical, and the latter one eliminates this constraint to allow the traffic to be arbitrary patterns. In [22], the authors build an analytical performance model with DNC taking the various contention and flow control into consideration. This result is further extended in [23], where the traffic splitter is proposed to support the multi-path routing polices. Another method is presented in [24] to compute the worst-case delay for conventional wormhole switched network, and a real-time Wormhole Channel Feasibility Checking (WCFC) algorithm is proposed. This research is further extended to calculate the bandwidth and delay bounds in [25], and used for topology synthesis of best-effort NoC in [26].

In [27], contention tree model was proposed to analyze the feasibility of real-time traffic delivered by priority-aware wormhole-switched NoC. It improves the previous results, e.g. lumped link model [11] and dependency graph model [8], by allowing the concurrent link usage. This is similar as the the LLA [14], which improves the FLA proposed in [7] by treating each link segment separately. A comprehensive comparison among FLA, contention tree model [27], lumped link model [11] and dependency graph model [8] can be found in [28], in which several defects of previous works [27][11][8] are illustrated and the advantages of FLA are highlighted. Two buffer sizing method based on FLA and LLA, i.e. Flow-Level Buffer-space Analysis (FLBA) and Link-Level Buffer-space Analysis (LLBA), are proposed in [15] to estimate the buffer size of priority-aware wormhole-switched NoC. The main drawback of FLA and LLA is that, both these two methods assume the traffic arrives periodically and the router has single cycle latency, which are of significant simplification to the realistic traffic pattern and router implementation. In addition, the FLBA and LLBA can only compute the minimum backlog bound at each router which does not trigger the flow control. In fact, we can further reduce the buffer size as long as the the flow control does not cause the violation of deadline.

On the other hand, although the DNC based performance model for best-effort NoC proposed in [22] can be applied to the analysis of priority-aware wormhole-switched NoC, the obtained performance bounds is very conservative, especially

for the high priority flows. This is because, it does not take the priority-aware scheduling into consideration. To overcome this limitation, a revised DNC performance model is proposed to analyze the worst case delay of priority-aware wormhole-switched NoC in [16]. But we found that, the DNC method in [16] can be further improved if we take the maximum service curve of each router and minimum arrival curve of each flow into consideration. Motivated by this observation, we adopt the RTC theory [18] to build the worst-case performance model for the priority-aware wormhole-switched NoC. Real-time calculus is the extension of DNC theory [17] by integrating the maximum service curve and minimal arrival curve, which has been widely used in the modeling and analysis of network processor [29], CAN [30], FlexRay [31] and DSP systems [32], etc. To ease the application of RTC, a real-time calculus toolbox [33] has also been implemented to support the numerical calculation.

III. PRELIMINARIES

A. Basic Assumptions

In this paper, we consider the same network model as the priority-aware wormhole-switched NoC proposed in [9][7][11][14]. Each router has the same number of input and output port, and each input port has sufficient number of FIFO buffer, i.e. Virtual Channel (VC), to accommodate all the incoming traffic of different priority levels. The allocation of VC is determined by the VC allocator. The buffer depth of each VC is finite, and credit-based flow control is adopted between each router. To ensure the predicable transmission of message, we assume that, a deterministic routine computation module is used to determine the output port of each communication message. Crossbar is used to switch traffic from input ports to the output ports, and the switch operation is determined by the switch allocator. The switch allocator is priority-aware: if multiple flits from different input ports or different VCs of the same input port contend for the same output port, it will only grant the flit with highest priority. Flits from a lower priority can transmit a flit if and only if there are no flits from higher priority in the input buffer or the flits with higher priority are self-blocked due to the insufficiency of VC buffer at downstream router.

The router micro-architecture considered in this paper has standard 5 pipeline stages, i.e. Buffer-Write (BW), Route Computation (RC), VC Allocation (VA), Switch Allocation (SA), Switch Traversal (ST) and Link Traversal (LT). For the detailed description about the implementation and functionality of this standard router micro-architecture, please refer to [34]. Although we focus on the standard 5-stage router, our method can be easily modified to support other router micro-architecture, e.g. single-cycle router and speculation-based router. We will demonstrate the adoption of our model in a single-cycle router in subsection V-A. Communication message is broken input packets, and each packet is composed of one head flit, one tail flit and several body flits. Each head flit should traverse all the five stages to find a path and reserve VC for the follow non-head flits. Non-head flits skip the RC and VA stage since the routine and VC have
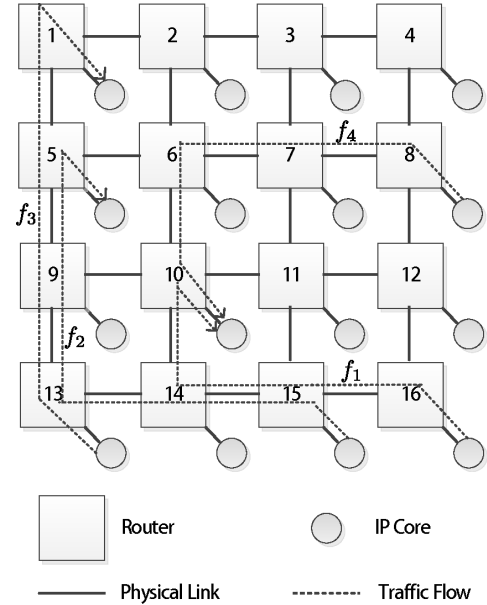

Fig. 1: Mesh topology with 4 real-time traffic flows

been determined by head flit. Router resource and control information reserved for a packet will be released only after the tail flit of this packet has been departed from the router. An additional priority field in the head flit is required for the routers to scheduling multiple contending flows according to their priority. To simplify of our analysis, we also assume that the entire chip is synchronous, with clock frequency f and period T. Our method can also be applied to analyze Global Asynchronous Local Synchronous (GALS) NoC with little modification, because the routers located in different voltage-frequency islands can be synchronized with a half cycle synchronizer [35], corresponding to a fixed-latency element in DNC theory.

Our performance model is topology independent, but to demonstrate the basic idea of our method, we take the mesh topology shown in Fig. 1 as an example throughout this paper. The router in mesh topology has five ports, corresponding to the four cardinal directions (West, East, North and South) and the Network Interface (NI), which connects to the local Intellectual Property (IP) core. There are 4 traffic flows in Fig. 1, i.e. f_1, f_2, f_3 and f_4. A flow is a sequence of packets with the same transmission path, source address and destination address. The path of a traffic flow is defined as a router chain started from ingress router and ended at egress router. We must emphasize that, although there is only four flows in the network, it is sufficient to demonstrate our the idea of our method, and our method can handle more traffic flows efficiently. To ensure the low latency transmission for real-time traffic flows, we have to assign higher priorities to these flows. As the minimum transmission unit in NoC is flit and a higher priority packet can preempt the transmission of a lower priority packet, the NoC architecture considered in this paper is flit-level preemptive [24]. Our method extends the existing methods in [14][16] to allowing multiple flows share the same priority. Flits from different flows with the same priority are

served in round-robin order.

B. Introduction to Real-Time Calculus

Real-time calculus [18] is a theoretic extension of DNC [17], by adding the upper service curve and lower arrival curve to describe the maximal service capacity of processing elements and minimal arrival rate of events. It is the mathematical basis of Modular Performance Analysis (MPA) [36] technique used for real-time task scheduling. Here we only present the definition of RTC arrival curve and service curve, for more details about this theory, please refer to [18].

Definition 1 (Real-Time Arrival Curve [18]): Let $R[s, t]$ denote the number of events that arrived within the time interval $[s, t]$. The lower bound and upper bound on $R[s, t]$ is called the lower arrival curve α^l and upper arrival curve α^u which satisfy

$$\alpha^l(t - s) \leq R[s, t] \leq \alpha^u(t - s), \forall s < t$$

where $\alpha^l(0) = \alpha^u(0) = 0$. The RTC arrival curve for R is denoted as $< \alpha^l, \alpha^u >$ for short.

Definition 2 (Real-Time Service Curve [18]): Let $S[s, t]$ denote the total number of events that can be processed by the system in the time interval $[s, t]$. The lower bound and upper bound on $S[s, t]$ is called the lower service curve β^l and upper service curve β^u which satisfy

$$\beta^l(t - s) \leq S[s, t] \leq \beta^u(t - s), \forall s < t$$

where $\beta^l(0) = \beta^u(0) = 0$. The RTC service curve for S is denoted as $< \beta^l, \beta^u >$ for short.

From the definition, we find that the upper arrival curve and lower service curve of RTC theory are corresponding to the arrival curve and service curve of DNC theory [17], respectively. Similarly, the upper service curve of RTC theory is identical to the maximum service curve in DNC theory. Without explicit stated, we will use these two terms interchangeably. In addition, the concatenation theorems for service curve (see Theorem 1.46 in [17]) and maximal service curve (see Theorem 1.6.1 in [17]) are also applicable to the RTC service curve.

In this paper, we will utilize the discrete time RTC arrival curve and service curve to characterize the arrived traffic and service capacity, since the minimal time unit in the wormhole-switched NoC is clock period T. Events in arrival curve and service curve refer to the arrival and service of flits, respectively. If we obtain the arrival curve $< \alpha^l, \alpha^u >$ of a specific traffic flow and the service curve $< \beta^l, \beta^u >$ provided to this flow, we can get the output arrival curve $< \alpha^{l'}, \alpha^{u'} >$ of this flow and leftover service curve $< \beta^{l'}, \beta^{u'} >$ for the other flows with the following equations [18]:

$$\alpha^{l'} = \min\{(\alpha^l \oslash \beta^u) \otimes \beta^l, \beta^l\} \quad (1)$$

$$\alpha^{u'} = \min\{(\alpha^u \otimes \beta^u) \oslash \beta^l, \beta^u\} \quad (2)$$

$$\beta^{l'} = (\beta^l - \alpha^u) \bar{\otimes} 0 \quad (3)$$

$$\beta^{u'} = \max\{(\beta^u - \alpha^l) \bar{\oslash} 0, 0\} \quad (4)$$

where $\otimes$, $\oslash$, $\bar{\otimes}$, $\bar{\oslash}$ are corresponding to the min-plus convolution, de-convolution, and max-plus convolution and de-convolution [17], respectively.

After we obtained the arrival curve $< \alpha^l_{f_i}, \alpha^u_{f_i} >$ of traffic flow f_i and the service curve $< \beta^l_{R_j, f_i}, \beta^u_{R_j, f_i} >$ provided by each router R_j to flow f_i on its path, we can obtain the end-to-end delay bound by the following equation,

$$Delay(f_i) = H(\alpha^u_{f_i}, \beta^l_{R_1, f_i} \otimes \beta^l_{R_2, f_i} \otimes \cdots \otimes \beta^l_{R_N, f_i}), \quad (5)$$

where operators $H(\cdot, \cdot)$ means the maximal horizontal deviation between the two operands.

IV. DELAY ANALYSIS AND BUFFER SIZING

In this section, we first build an RTC based performance model for the priority-aware wormhole-switched NoC. This model consists of two parts, i.e. traffic model and service model. The traffic model utilizes the RTC arrival curve to describe the flit arrival process of each flow in the priority-aware NoC, and we will introduce two methods to obtain the arrival curve in subsection IV-A. The service model is slightly complicated than the traffic model. While constructing the service model for priority-aware wormhole-switched NoC, the follow four issues should be considered: (1) Only the head flit need to be processed by RC and VA stage, the subsequent flits of a packet just follow the data-path built by head flit. To simplify our RTC model, we need specific mechanism to characterize the service provided to head and non-head flits in a unified way. (2) Our model extends the existing approach [14][16] by allowing priority sharing between flows. Thus, the leftover service curve provided to lower-priority flows can only be derived after the service curves of all the high-priority flows have been calculated. (3) Due to the limited on-chip buffer, credit-based flow control is used as a back-pressure mechanism to prevent buffer overflow. Before analysis the end-to-end delay bound with Eq.(5), we should first break the cyclic-dependence caused by flow control. (4) To guarantee the tightness of our theoretical performance bounds and simplify the numerical computation, we should apply the concatenation theorems as far as possible. But, this concatenation process becomes complicated when the credit-based flow control is taken into consideration. We will discuss the first two issues in subsection IV-B, and the last two issues are discussed in subsection IV-C and subsection IV-D, respectively. Based on the proposed performance model, we propose an end-to-end delay analysis algorithm and buffer sizing algorithm, as presented in subsection IV-E and subsection IV-F, respectively.

A. Traffic Model

The communication in a priority-aware wormhole-switched NoC is realized by transmitting packets, and the packet is further divided into flits, which is the minimum transmission unit in wormhole-switched NoC. Denote by $< \alpha^l(\Delta), \alpha^u(\Delta) >$ the flit arrival curve of a flow, namely, the minimum and maximum number of flits can be seen within any time window of width Δ. We can extract the flit arrival curve from the synthetic traffic or communication trace with the sliding window method [18]. For each window size Δ, this method tries to find the
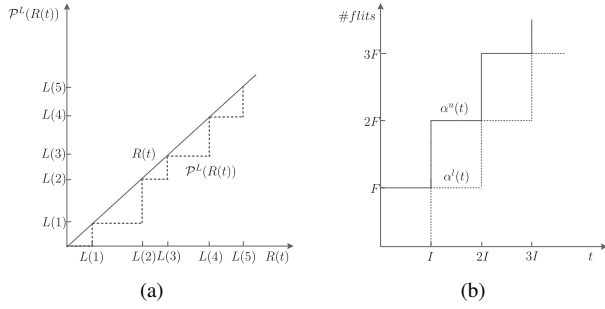
Fig. 2: Traffic model. (a) Definition of $\mathcal{P}^L(R(t))$. Cumulative arrival function $R(t)$ and the L-packetized cumulative arrival function $\mathcal{P}^L(R(t))$ are represented by the dotted line and solid line, respectively. (b) Real-time calculus arrival curve for periodically arrived traffic with period I and packet length F. The solid line and dotted line represent the upper arrival curve and lower arrival curve.
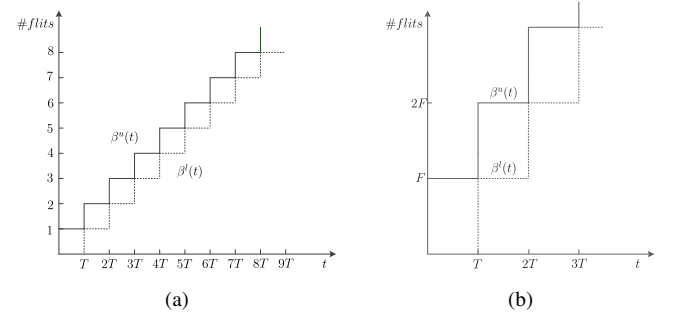

Fig. 3: Service model for each pipeline stages. The solid lines and dotted lines represent the upper service curves and lower service curves, respectively. (a) Service curve of BW, SA and ST stages; (b) Service curve of RC and VA stages.

maximal and minimal number of arrived flits (corresponding to $\alpha^l(\Delta)$ and $\alpha^u(\Delta)$) by analyzing the time series of flits. The synthetic traffic or communication trace is not necessarily to be periodic, since the RTC theory makes no assumption on the periodicity of the traffic. But, the obtained flit arrival curve can only be applied to compute the worst-case performance bound at the flit level. To compute the worst-case performance bound at packet level, this arrival curve must be L-packetized [17]. Denote by $L(n)$ the cumulative packet length (in flits) of the first n packets in a flow, $R(t)$ the cumulative arrived flits by time t. Then, the L-packetizer operator $\mathcal{P}^l(\cdot)$ is defined as $\mathcal{P}^L(R(t)) = \sup_{n\in\mathcal{N}}\{L(n)1_{L(n)\leq R(t)}\}^1$. Intuitively, $\mathcal{P}^l(\cdot)$ can be interpreted as the largest cumulative packet length contained in $R(t)$, as shown in Fig. 2a. Denote by L_{max} the maximum packet length (in flits) of a flow, according to the basic properties of L-packetizer [17], we have

$$R(t) - l_{max} \leq \mathcal{P}^L(R(t)) < R(t)$$

and

$$R(s) - l_{max} \leq \mathcal{P}^L(R(s)) < R(s)$$

which indicate

$$\alpha(t-s)^l - l_{max} < \mathcal{P}^L(R(t)) - \mathcal{P}^L(R(s)) < \alpha(t-s)^u + l_{max}.$$

The inequalities above indicate that, if a flow has a flit arrival curve $<\alpha^l(\Delta), \alpha^u(\Delta)>$, the L-packetized flow has a packet arrival curve $<\alpha^l(\Delta)-l_{max}1_{\{\Delta>0\}}, \alpha^u(\Delta)+l_{max}1_{\{\Delta>0\}}>$. We can also directly obtain the L-packetized arrival curve instead of transformation from flit arrival curve for some special cases. For example, suppose all the packets in a flow have the same length F and arrived periodically with period I. We can obtain the flit arrival curve of this flow by applying the sliding window method, as shown in Fig. 2b. The obtained flit arrival curve is equivalent to the L-packetized arrival curve $\mathcal{P}^L(\alpha)$, since $\mathcal{P}^L(R(t)) = R(t)$ and $\mathcal{P}^L(\alpha(t)) = \alpha(t)$.

B. Service Model

While modeling the service capacity of routers with RTC, we can analyze the data-path of a flow stage-by-stage. After obtained the service curve for each stage, the service curve provided by the router to the flow can be obtained by concatenating all the service curves of these five stages. This is significantly different with the existing DNC based model [22], [16], where they treat the entire router as a whole and designate a Latency-Rate (LR) service curve for simplicity. The advantage of our method is that, it can be easily modified to characterize the non-standard router micro-architectures, by simply letting the service curve of non-existed stages to be burst delay function $\delta_0(t)^2$. Next, we try to obtain the service curves of all these 5 stages:

(1) BW stage and ST stage: all the flits within a traffic flow will traverse these two stages, and experience a fixed delay T. Take the BW stage as an example, for any time interval of length less than T, the maximum and minimal number of flits can be served are one and zero, respectively, since it outputs one flit at each cycle T. Similarly, for any time interval of length greater than T, the maximum and minimal number of flits that can be seen are two and one, respectively. The resulted service curves, i.e. $<\beta^l_{BW}, \beta^u_{BW}>$ and $<\beta^l_{ST}, \beta^u_{ST}>$, are shown in Fig. 3a.

(2) RC stage and VA stage: the traverse latency of head flit and non-head flits at these two stages are T and 0, respectively. A sophisticated solution to construct a unified service curve for head flit and non-head flits at these two stages is that, we can view these two stages can service an entire packet within T period. Suppose the packet length is F, the equivalent service curve for these two stages, i.e. $<\beta^l_{RC}, \beta^u_{RC}>$ and $<\beta^l_{VA}, \beta^u_{VA}>$, can be easily obtained by amplifying the service curve of Fig. 3a in y-axis with a factor F, as shown in Fig. 3b.

(3) SA stage: each output port in the wormhole-switched NoC has a switch allocator to schedule the switch traversal among all the contending flows at each clock cycle. Thus, following the same approach as BW and ST stage, we can get the service curves provided by the switch allocators at each output port to all the contending flows, as shown in Fig. 3a. We

---

[1] $\mathcal{N}$ is the set of natural numbers and $1_{\{val\}}$ is the indicator function, $1_{\{val\}} = 1$ if and only if val is true.

[2] $\delta_{val}(t) = +\infty$ if $t > val$, and 0 otherwise.

also define the notation $< \beta^l_{SA,R^p_i}, \beta^u_{SA,R^p_i} >$ to distinguish the service curve of switch allocator at specific output port from the others. For the mesh topology, the port indicator p can be concreted with W (West), E (East), S (South) and N (North) or L (Local).

The alert readers would notice that, the contention of different flows only occurs at SA stage. For the fixed-priority based scheduling policy, switch allocators provide service for high priority flows first, and flows with the same priority will be served with Round-Robin order. The unserved flows will be imposed an additional latency T due the failure of switch arbitration. Thus, if we obtained the service curve provided by SA stage to flow f_j, we can obtain the service curve of router directly.

Denote by $< \beta^l_{SA,R^p_i,f_j}, \beta^u_{SA,R^p_i,f_j} >$ the service curve provided to flow f_j by SA stage of router R_i, the equivalent service curve of router R_i provided to f_j, i.e. $< \beta^l_{R^p_i,f_j}, \beta^u_{R^p_i,f_j} >$, can be obtained by concatenating the service curves of all the 5 stages together:

$$\beta^l_{R^p_i,f_j} = \beta^l_{BW} \otimes \beta^l_{RC} \otimes \beta^l_{VA} \otimes \beta^l_{SA,R^p_i,f_j} \otimes \beta^l_{ST},$$

$$\beta^u_{R^p_i,f_j} = \beta^u_{BW} \otimes \beta^u_{RC} \otimes \beta^u_{VA} \otimes \beta^u_{SA,R^p_i,f_j} \otimes \beta^u_{ST}.$$

Suppose the total service curve provided by a switch allocator is $< \beta^l_{SA,R^p_i}, \beta^u_{SA,R^p_i} >$ and the leftover service curve after serving the flows with higher priority than f_j is $< \beta^{l'}_{SA,R^p_i}, \beta^{u'}_{SA,R^p_i} >$, in order to obtain the service curve $< \beta^l_{SA,R^p_i,f_j}, \beta^u_{SA,R^p_i,f_j} >$, we should consider the following two cases:

(a) all the flows contending with f_j at R_i have lower priorities. For the synchronize router architecture, flow f_j obtain the total leftover service curve $< \beta^{l'}_{SA,R^p_i}, \beta^{u'}_{SA,R^p_i} >$.

(b) there exists some contention flows with the same priority as f_j. Denote by Θ_{R_i,f_j} the set of contention flows at router R_i with the same priority as f_j, and let N_{R_i,f_j} be the number of flows in Θ_{R_i,f_j}. Since all the flows in Θ_{R_i,f_j} got serviced in Round-Robin order, the service curve provided to f_j is $< \lfloor \beta^{l'}_{SA,R^p_i}/(N_{R_i,f_j}+1) \rfloor, \lceil \beta^{u'}_{SA,R^p_i}/(N_{R_i,f_j}+1) \rceil >$, where $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$ are the ceiling operator and flooring operator, respectively. After serving by all the flows in Θ_{R_i,f_j}, the leftover service capacity for low priority flows can be obtained by applying Eq.(3) and Eq.(4).

C. Upper Service Curve of Flow Controller

Credit-based flow control introduces cyclic-dependence between the adjacent routers, and leads to the self-blocking within a flow due to the insufficiency of buffer space at the downstream router. To make the discussion concrete, we take flow f_2 as an example and utilize the scheduling network model [18] in RTC theory to visualize the credit-based flow control and complex relationship among f_2 and the other flows, as shown in Fig. 4. We ignore flow f_4 and the flow control of other flows for clarity. We also assume that, the destination IP core can consume the ejected flits immediately, thus there is no flow control between the ejection router and destination NI. But, to prevent the buffer overflow, the flow control between input NI and injection router is necessary.

The flow control mechanism introduces feedback control loop between adjacent routers, preventing us from deriving the performance bound directly even after we have obtained the service curve reserved at each router for the target flow. Historically, this issue is addressed by fixed-point iteration [37] or transformation from marked dataflow graph [38]. In this paper, we try to tackle the same problem with another solution. This is motivated by [22], where the authors abstract the flow control as a network element (called flow controller) providing a service curve β_τ (corresponding to the lower service curve in RTC theory). Then, this service curve is obtained by applying some basic properties in DNC theory [17].

In this subsection, we follow the similar procedure as [22] to derive the upper service curve for the flow controller, as summarized in Theorem 1. The obtained upper service curve together with the lower service curve derived in [22] enable us to break the control loop caused by flow control and build a comprehensive performance model with RTC.

Theorem 1: Suppose the router provides a RTC service curve $< \beta^l, \beta^u >$, the buffer size and credit feedback delay are denoted as B and σ, respectively. Then, the flow controller provides an equivalent RTC service curve $< \overline{\beta^l \otimes \delta_\sigma(t) + B}, \overline{\beta^u \otimes \delta_\sigma(t) + B} >$, where $\bar{f}$ is the subadditive closure of f [17].

Proof: The lower service curve has been derived in [22]. In the rest of this proof, we will take the flow control between R_9 and R_5 in Fig. 4 as an example to derive the upper service curve. Denote the amount of injected and departed flits at R_5 by time t as $I(t)$ and $D(t)$, and the amount of flits served by R_9 by time t is denoted as $A(t)$. The feedback link can be represented as a network element providing upper service curve $\delta_\sigma(t)$.

According to the basic property of credit-based flow control, we have

$$I(t) = \min\{A(t), D'(t) + B\}$$

where $D' = D \otimes \delta_\sigma$.

Based on the equivalent definition of upper service curve[3], we have

$$D(t) \leq I \otimes \beta^u_{R^L_5,f_2}(t).$$

Bring $I(t)$ and $D'(t)$ into this equality, we get

$$
\begin{aligned}
D(t) & \leq & I \otimes \beta^u_{R^L_5,f_2}(t) \\
& \leq & \min\{A \otimes \beta^u(t), D \otimes \delta_\sigma \otimes \beta^u_{R^L_5,f_2}(t) + B\}.
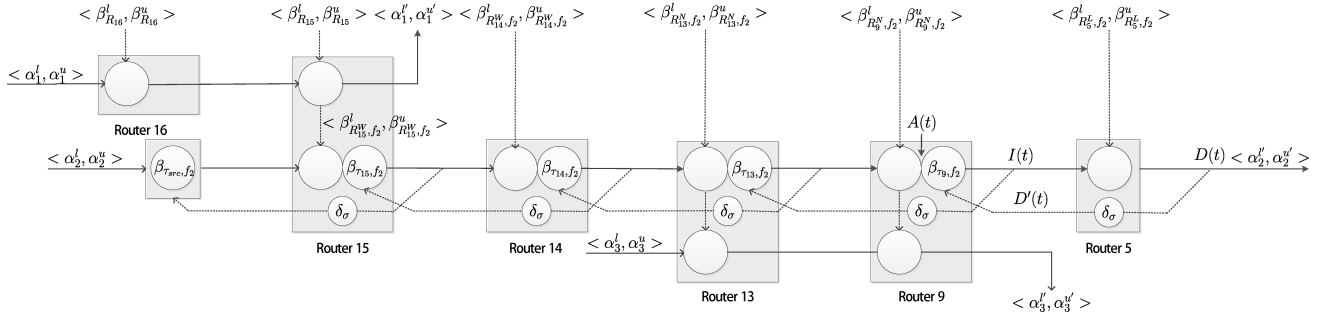\end{aligned}
$$

By applying Theorem 4.31 in [17], we have

$$D \leq A \otimes \beta^u \otimes \overline{\beta^u_{R^L_5,f_2} \otimes \delta_\sigma + B}.$$

Thus,

$$
\begin{aligned}
I & = & \min\{A, D' + B\} \\
& \leq & \min\{A, D \otimes \delta_\sigma + B\} \\
& \leq & \min\{A, A \otimes \beta^u_{R^L_5,f_2} \otimes \overline{\beta^u_{R^L_5,f_2} \otimes \delta_\sigma + B} \otimes \delta_\sigma + B\} \\
& = & \min\{A \otimes \delta_\sigma, A \otimes \overline{\delta_\sigma \otimes \beta^u_{R^L_5,f_2} + B}\} \\
& = & A \otimes \min\{\delta_\sigma, \overline{\beta^u_{R^L_5,f_2} \otimes \delta_\sigma + B}\} \\
& = & A \otimes \overline{\beta^u_{R^L_5,f_2} \otimes \delta_\sigma + B}.
\end{aligned}
$$

[3] please refer to definition 1.6.1 in [17] for more details.

Fig. 4: Scheduling network model for flow f_2

which implies that, the flow controller between R_9 and R_5 provides an equivalent upper service curve $\overline{\beta_{R_5^L,f_2}^u} \otimes \delta_\sigma + B$. Similarly, for any router providing upper service curve β^u, the corresponding flow controller has an equivalent upper service curve $\beta_\tau^u(t) = \overline{\beta^u} \otimes \delta_\sigma(t) + B$. ■

Theorem 1 derives the upper service curve of a single flow controller, and we can get the service curves of all the flow controllers along the router chain of all the flows by applying Theorem 1 iteratively. As shown in Fig. 4, the service curve of downstream routers can affect the service curve of flow controller at upstream. Hence, we should compute the service curves of flow controller from ejection router to injection router. Take flow f_2 as an example, we should first derive the service curves of f_2 obtained at each router with the method provided in Subsection IV-B. Service curve provided by router R_{15} to flow f_2 can be derived by applying Eq.(3) and Eq.(4), since f_2 is a lower priority flow at R_{15}. Router R_{13}, R_9 and R_5 provide all their service capacity to f_2 because f_2 is the highest priority flow at these routers. Then, the service curve of flow controller at router R_9 can be obtained by applying Theorem 1, which is $< \beta_{\tau_9}^l, \beta_{\tau_9}^u >$. By applying the concatenation theorem, we can obtain the equivalent service curve provided to f_2 by router R_9, which is $< \beta_{R_9^N,f_2}^l \otimes \beta_{\tau_9}^l, \beta_{R_9^N,f_2}^u \otimes \beta_{\tau_9}^u >$. Follow the same procedure, we can get the service curve of flow controller at router R_{13}, R_{14}, R_{15} and the source NI that connected to R_{15} iteratively.

D. Collapsible Sub-Path

To this end, we have build the traffic model, service model and flow control model. Before giving the delay analysis algorithm, we first consider the follow scenario. Flow f_2 and f_3 in Fig. 1 contend the output link at both router R_{13} and R_9. Suppose $P_2 > P_3$ and denote by $B_{R_9^N,f_2}$ the buffer size of router R_9 allocated to flow f_2, $< \beta_{R_{13}^N,f_2}^l, \beta_{R_{13}^N,f_2}^u >$ and $< \beta_{R_9^N,f_2}^l, \beta_{R_9^N,f_2}^u >$ the service curve provided to flow f_2 by router R_{13} and R_9, $< \beta_{\tau_{13},f_2}^l, \beta_{\tau_{13},f_2}^u >$ the service curve for flow controller of router R_{13}, $< \alpha_{R_{13},f_2}^l, \alpha_{R_{13},f_2}^u >$ and $< \alpha_{R_{13},f_3}^l, \alpha_{R_{13},f_3}^u >$ the arrival curve of f_2 and f_3 at router R_{13}. The question is: when $B_{R_9^N,f_2}$ is sufficiently large so that $\beta_{R_{13}^N,f_2}^l \otimes \beta_{\tau_{13},f_2}^l = \beta_{R_{13}^N,f_2}^l$ and $\beta_{R_{13}^N,f_2}^u \otimes \beta_{\tau_{13},f_2}^u = \beta_{R_{13}^N,f_2}^u$, how to derive the leftover service curve for f_3 at R_{13} and R_9, and obtain a tight performance bound of f_3 efficiently?
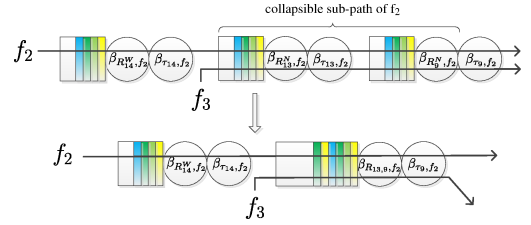
Fig. 5: Collapsible sub-path of f_2. Since the flow control between R_{13} and R_9 can be ignored, they are replaced by a single virtual router $R_{13,9}$.

An intuitive solution is: Firstly, obtaining the leftover service curve of R_{13} by applying Eq.(3) and Eq.(4); Secondly, deriving the output arrival curve $< \alpha_{R_9,f_2}^{l'}, \alpha_{R_9,f_2}^{u'} >$ of f_2 by applying Eq.(1) and Eq.(2); Thirdly, the leftover service curve of R_9 can be easily obtained by applying Eq.(3) and Eq.(4); And finally, the leftover service curve for f_3 at R_{13} and R_9 can be easily obtained by concatenating these two service curves. Another solution is: we can substitute R_{13} and R_9 by a virtual router $R_{13,9}$ providing service curve $< \beta_{R_9^N,f_2}^l \otimes \beta_{R_{13}^N,f_2}^l, \beta_{R_9^P,f_2}^u \otimes \beta_{R_{13}^N,f_2}^u >$, since $B_{R_9^N,f_2}$ is sufficiently large and the flow control between R_{13} and R_9 can be ignored, as shown in Fig. 5. Then, the service curve of f_3 obtained at R_{13} and R_9 can be directly obtained by applying Eq.(3) and Eq.(4). Compared with previous router-by-router calculation method, it eliminates the calculation of intermediate arrival curve, and compute the equivalent service curve by just invoke Eq.(3) and Eq.(4) once, which is of calculating efficiency. We formalize this observation and propose the concept of Collapsible Sub-Path (CSP) to simplify the calculation and improve the tightness of performance bounds.

Definition 3 (Collapsible Sub-Path): The collapsible sub-path of flow f, denoted by $CSP(f)$, is a sub-path of f satisfying the follow conditions:

1) all the routers R_i on this sub-path, except the last one, all satisfy $\beta_{R_i^P,f}^l \otimes \beta_{\tau_i,f}^l = \beta_{R_i^P,f}^l$ and $\beta_{R_i^P,f}^u \otimes \beta_{\tau_i,f}^u = \beta_{R_i^P,f}^u$.
2) $CSP(f)$ is also a sub-path of all the flows in Ω_f, where Ω_f is the set of contention flows with lower priorities on this sub-path.

For the high-priority flows, the process of calculating the end-to-end equivalent service curve with concatenation theo-

rem is also a collapsing process. But, to compute the leftover service curve for low-priority flows at some routers, we have to know the arrival curve of high-priority flows at these routers. As implied previously, instead of the router-by-router calculation, we can leverage the concept of CSP and replace all the routers on $CSP(f)$ with a single virtual router providing service curve $< \otimes_{R_i \in CSP(f)} \beta^l_{R_i^p, f}, \otimes_{R_i \in CSP(f)} \beta^u_{R_i^p, f} >$. For a CSP with n ($n \geq 2$) routers, this method reduces $n - 1$ times calculation of arrival curve and leftover service curve. The leftover service curve calculation is greatly simplified, and the accuracy of performance bound is also improved due to the well-known "Pay-Burst-Only-Once" phenomenon in DNC theory [17].

E. End-to-End Delay Analysis

After obtained the traffic model, service model and flow control model, compute the end-to-end delay is still a non-trivial task. The follow four issues should be considered carefully: (1) We should always collapse all the CSP to a single virtual node before computing the end-to-end delay, as the hop-by-hop computation will lead to a looser bound; (2) In the fixed-priority flit-level preemptive NoC, only the leftover service curve can be used by the low-priority flows, thus, the computation process must start from high-priority flows to low-priority flows; (3) Our performance model supports priority-sharing, before computing the leftover service curve for lower priority flows, we must ensure that all the flows with the same priority have been calculated; (4) The computed service curve for flow controller can only be applicable for specific flow, and we should compute these curves for each flows. Keeping these four issues in mind, we propose the end-to-end delay analysis algorithm, as shown in Algorithm 1.

Suppose the entire NoC is represented as a directional topology graph $G : V \times E$, where V and E represent the set of routers and links, respectively. For each link $e_{i,j} \in E$ represent there is a physical channel between router R_i and router R_j, $e_{i,j} \neq e_{j,i}$ since the topology graph is a directional graph. The set of all the flows in the network is denoted as $\mathcal{F}$, and each flow $f_i \in \mathcal{F}$ has a fixed-priority P_i and deadline D_i. The set of routers a flow f_i traversed is denoted as $\mathcal{R}_i$, and the set of links a flow f_i traversed is denoted as Γ_i. There exists interference between flow f_i and f_j, if and only if $\Gamma_i \wedge \Gamma_j \neq \Phi$. Denote by $< \alpha^l_{f_i}, \alpha^u_{f_i} >$ the arrival curve of flow f_i at the ingress port, $< \alpha^l_{R_j, f_i}, \alpha^u_{R_j, f_i} >$ the arrival curve of f_i at router R_j, and $< \beta^{l'}_{SA, R_j^p}, \beta^{u'}_{SA, R_j^p} >$ the leftover service curve of switch allocator at output port p (Initially, $\beta^{l'}_{SA, R_j^p} = \beta^l_{SA, R_j^p}$ and $\beta^{u'}_{SA, R_j^p} = \beta^u_{SA, R_j^p}$). The path of flow f_i is started from ingress router (denoted by $start_i$) to egress router (started by end_i). For all the router R_j along the path of flow f_i, denote the set of contending flows at R_j sharing the same priority with f_i as Θ_{R_j, f_i}, the set of contending flows at R_j with lower priorities as Ω_{R_j, f_i}, and the buffer size as $B_{R_j^p, f_i}$. In addition, let N_{R_j, f_i} be the number of flows in Θ_{R_j, f_i}.

The proposed algorithm calculates the end-to-end delay from high-priority flows to low-priority flows. For each iteration, this algorithm performs the following four steps: (1)

Algorithm 1 End-to-End Delay Analysis Algorithm

Require: Network Topology Graph and Flow Specification
Ensure: Worst-case End-to-End Delay for all the flows
1: **for** each flow $f_i \in \mathcal{F}$ with priority order **do**
2:    $\beta^l_\tau = \delta_0(t)$; $\beta^u_\tau = \delta_0(t)$;
3:    **for** each router $R_j \in \mathcal{R}_i$ from end_i to $start_i$ **do**
4:      **if** $\Theta_{R_j, f_i} \neq \emptyset$ **then**
5:         $\beta^l_{R_j^p, f_i} = \beta^l_{BW} \otimes \beta^l_{RC} \otimes \beta^l_{VA} \otimes \lfloor \frac{\beta^{l'}_{SA, R_j^p}}{N_{R_j, f_i} + 1} \rfloor \otimes \beta^l_{ST}$;
6:         $\beta^u_{R_j^p, f_i} = \beta^u_{BW} \otimes \beta^u_{RC} \otimes \beta^u_{VA} \otimes \lceil \frac{\beta^{u'}_{SA, R_j^p}}{N_{R_j, f_i} + 1} \rceil \otimes \beta^u_{ST}$;
7:      **else**
8:         $\beta^l_{R_j^p, f_i} = \beta^l_{BW} \otimes \beta^l_{RC} \otimes \beta^l_{VA} \otimes \beta^{l'}_{SA, R_j^p} \otimes \beta^l_{ST}$;
9:         $\beta^u_{R_j^p, f_i} = \beta^u_{BW} \otimes \beta^u_{RC} \otimes \beta^u_{VA} \otimes \beta^{u'}_{SA, R_j^p} \otimes \beta^u_{ST}$;
10:     **end if**
11:     $\beta^l_{\tau_j, f_i}(t) = \beta^l_\tau$; $\beta^u_{\tau_j, f_i}(t) = \beta^u_\tau$;
12:     $\beta^l_\tau = \overline{\beta^l_{R_j^p, f_i} \otimes \beta^l_\tau \otimes \delta_\sigma(t) + B_{R_j^p, f_i}}$;
13:     $\beta^u_\tau = \overline{\beta^u_{R_j^p, f_i} \otimes \beta^u_\tau \otimes \delta_\sigma(t) + B_{R_j^p, f_i}}$;
14:    **end for**
15:    $\beta^l_{\tau src, f_i} = \beta^l_\tau$; $\beta^u_{\tau src, f_i} = \beta^u_\tau$;
16:    $\beta_{f_i} = \beta^l_{\tau src, f_i} \otimes (\underset{R_k \in \mathcal{R}_{f_i}}{\otimes} (\beta^l_{R_k, f_i} \otimes \beta^l_{\tau_k, f_i}))$;
17:    $Delay(f_i) = H(\alpha^u_{f_i}, \beta_{f_i})$;
18:    $\beta^l_{f_i} = \beta^l_{\tau src, f_i}$; $\beta^u_{f_i} = \beta^u_{\tau src, f_i}$;
19:    Collapse $CSP(f_i)$ and update $\Theta_{R_j, \cdot}$, $\Omega_{R_j, \cdot}$ and $\mathcal{R}.$;
20:    **for** $\forall R_j \in \mathcal{R}_{f_i}$ from $start_i$ to end_i **do**
21:      **if** $\Omega_{R_j, f_i} \neq \emptyset$ **then**
22:         $\beta^l = \beta^l_{f_i} \otimes \beta^l_{BW} \otimes \beta^l_{RC} \otimes \beta^l_{VA}$;
23:         $\beta^u = \beta^u_{f_i} \otimes \beta^u_{BW} \otimes \beta^u_{RC} \otimes \beta^u_{VA}$;
24:         $\alpha^l_{R_j, f_i} = \min\{(\alpha^l_{f_i} \oslash \beta^u) \otimes \beta^l, \beta^l\}$;
25:         $\alpha^u_{R_j, f_i} = \min\{(\alpha^u_{f_i} \otimes \beta^u) \oslash \beta^l, \beta^u\}$;
26:         **if** $\forall f_k \in \Theta_{R_j, f_i}$ have been calculated **then**
27:            $\alpha^l_{R_j, f_i} = \alpha^l_{R_j, f_i} + \sum_{f_k \in \Theta_{R_j, f_i}} \alpha^l_{R_j, f_k}$;
28:            $\alpha^u_{R_j, f_i} = \alpha^u_{R_j, f_i} + \sum_{f_k \in \Theta_{R_j, f_i}} \alpha^u_{R_j, f_k}$;
29:            $\beta^{l'}_{SA, R_j^p} = (\beta^{l'}_{SA, R_j^p} - \alpha^u_{R_j, f_i}) \bar{\otimes} 0$;
30:            $\beta^{u'}_{SA, R_j^p} = \max\{(\beta^{u'}_{SA, R_j^p} - \alpha^l_{R_j, f_i}) \bar{\oslash} 0, 0\}$;
31:         **end if**
32:      **end if**
33:     $\beta^l_{f_i} = \beta^l_{f_i} \otimes \beta^l_{R_j^p, f_i}$; $\beta^u_{f_i} = \beta^u_{f_i} \otimes \beta^u_{R_j^p, f_i}$;
34:    **end for**
35: **end for**

Calculating the service curve provided by router (lines 4-10) and flow controller (lines 11-12); (2) Identifying and collapsing all the CSP on the path of a flow to improve the analyzing accuracy (line 14); (3) Computing the worst-case end-to-end delay (line 15); (4) Calculating the leftover service curve for low-priority flows (lines 16-32). We also need to emphasize that, the leftover service curve for lower priority flows must be computed after all the service curve of higher priority flows have been calculated, as implied by line 25 in Algorithm 1. The overall algorithm has two-level embedded loops, and the computation complexity for this algorithm is $O(HN)$, where N and H is the number of flows and the hop

count of each flow. This algorithm can be easily integrated into the RTC toolbox [33] to compute the end-to-end delay bound automatically.

F. Buffer Sizing

The priority-aware wormhole-switched NoC [7] requires the same amount of VC as the priorities to prevent priority inversion [11], which refers to the blocking of high-priority flows when the low-priority flows occupy all the VCs. To reduce the buffer area and power consumption, priority sharing [39] and buffer optimization [15] techniques are proposed. However, the two backlog bounds derived in [15] is the minimum buffer size that does not trigger the flow control, which can be further reduced as long as the constraint of deadline is not be violated. Suppose the applications have been mapped onto the NoC, and each flow f_i has been assigned to their corresponding priority P_i and deadline D_i. Following the same notations as Algorithm 1, we propose the buffer sizing algorithm to further reduce the buffer size, as shown in Algorithm 2. It tries to reduce the buffer size for each flow from high-priority to low-priority gradually. For each iteration, it performs the following four steps: (1) Calculating the service curve provided by router (lines 3-9); (2) Calculate the minimum buffer size that can avoid flow control (lines 10-13); (3) Reduce the initial buffer size gradually as long as the constraint of deadline is not violated (lines 15-26); (4) Calculating the leftover service curve for low-priority flows (lines 27-43). This algorithm can be implemented in RTC toolbox [33] to optimize the buffer size automatically. The entire computation complexity is $O(HN)$, where N and H are the number of flows and the hop count of each flow.

A significant difference with Algorithm 1 is that, this algorithm does not identify and collapse the CSP, because the reduction of buffer size introduces flow control between adjacent routers, which prevents the flits progressing in the network and leads to a larger end-to-end delay. The amount of cycles that a packet can be stalled in the network without violating the deadline is defined as 'slack'. Our buffer sizing algorithm can reduce the buffer size iteratively as long as slack is greater than zero, which can be used to reduce the router area and power consumption. But, it is significantly different from the DNC based slack optimization in [40]. In [40], the energy optimization is realized by adjusting the voltage, frequency and link bandwidth of on-chip routers for the fixed configuration and deadline. In contrast, our method try to optimize the buffer size under the deadline constraint, and the buffer reduction directly leads to the area and power saving. In addition, our algorithm can be used in conjunction with the priority sharing techniques [39] to optimize the buffer size of priority-aware wormhole-switched NoC .

V. Experiments

In this section, we validate the correctness and tightness of our performance model by comparison with other analytical methods and simulation. Several analytical methods exist for the delay analysis of priority-aware NoC, examples include contention tree model [27], lumped link model [11],

---

Algorithm 2 Buffer Sizing Algorithm
___
Require: Network Topology Graph and Flow Specification
Ensure: Optimized Buffer Size
1: **for** each flow $f_i \in \mathcal{F}$ with priority order **do**
2: 　**for** each router $R_j \in \mathcal{R}_i$ **do**
3: 　　**if** $\Theta_{R_j,f_i} \neq \emptyset$ **then**
4: 　　　$\beta^l_{R^p_j,f_i} = \beta^l_{BW} \otimes \beta^l_{RC} \otimes \beta^l_{VA} \otimes \lfloor \frac{\beta^{l'}_{SA,R^p_j}}{N_{R_j,f_i}+1} \rfloor \otimes \beta^l_{ST}$;
5: 　　　$\beta^u_{R^p_j,f_i} = \beta^u_{BW} \otimes \beta^u_{RC} \otimes \beta^u_{VA} \otimes \lceil \frac{\beta^{u'}_{SA,R^p_j}}{N_{R_j,f_i}+1} \rceil \otimes \beta^u_{ST}$;
6: 　　**else**
7: 　　　$\beta^l_{R^p_j,f_i} = \beta^l_{BW} \otimes \beta^l_{RC} \otimes \beta^l_{VA} \otimes \beta^{l'}_{SA,R^p_j} \otimes \beta^l_{ST}$;
8: 　　　$\beta^u_{R^p_j,f_i} = \beta^u_{BW} \otimes \beta^u_{RC} \otimes \beta^u_{VA} \otimes \beta^{u'}_{SA,R^p_j} \otimes \beta^u_{ST}$;
9: 　　**end if**
10: 　　$B^l = \inf\{B | \beta^l_{R^p_j,f_i} \otimes \overline{\beta^l_{R^p_j,f_i} \otimes \delta_\sigma(t)} + B \geq \beta^l_{R^p_j,f_i}\}$;
11: 　　$B^u = \inf\{B | \beta^u_{R^p_j,f_i} \otimes \overline{\beta^u_{R^p_j,f_i} \otimes \delta_\sigma(t)} + B \geq \beta^u_{R^p_j,f_i}\}$;
12: 　　$B_{R^p_j,f_i} = \max\{B^l, B^u\}$;
13: 　　$\beta^l_{\tau_j,f_i} = \delta_0(t); \beta^u_{\tau_j,f_i} = \delta_0(t)$;
14: 　**end for**
15: 　$\beta^l_{\tau_{src},f_i} = \beta^l_\tau; \beta^u_{\tau_{src},f_i} = \beta^u_\tau$;
16: 　**for** each router $R_j \in \mathcal{R}_i$ from end_i to $start_i$ **do**
17: 　　$\beta_{f_i} = \beta^l_{\tau_{src},f_i} \otimes \bigotimes_{R_k \in \mathcal{R}_{f_i}} (\beta^l_{R^p_k,f_i} \otimes \beta^l_{\tau_k,f_i})$;
18: 　　$Delay(f_i) = H(\alpha^u_{f_i}, \beta_{f_i})$;
19: 　　**while** $Delay(f_i) \leq D_i$ and $B_{R^p_j,f_i} > 1$ **do**
20: 　　　$B_{R^p_j,f_i} = B_{R^p_j,f_i} - 1$;
21: 　　　Update $\beta^l_{\tau_k,f_i}$ for all R_k from R_j to $start_i$;
22: 　　　$\beta_{f_i} = \beta^l_{\tau_{src},f_i} \otimes \bigotimes_{R_k \in \mathcal{R}_{f_i}} (\beta^l_{R^p_k,f_i} \otimes \beta^l_{\tau_k,f_i})$;
23: 　　　$Delay(f_i) = H(\alpha^u_{f_i}, \beta_{f_i})$;
24: 　　**end while**
25: 　　**if** $Delay(f_i) > D_i$ **then**
26: 　　　$B_{R^p_j,f_i} = B_{R^p_j,f_i} + 1$;
27: 　　　Update $< \beta^l_{\tau_j,f_i}, \beta^u_{\tau_j,f_i} >$;
28: 　　**end if**
29: 　**end for**
30: 　$\beta^l_{f_i} = \beta^l_{\tau_{src},f_i}; \beta^u_{f_i} = \beta^u_{\tau_{src},f_i}$;
31: 　**for** $\forall R_j \in \mathcal{R}_{f_i}$ from $start_i$ to end_i **do**
32: 　　**if** $\Omega_{R_j,f_i} \neq \emptyset$ **then**
33: 　　　$\beta^l = \beta^l_{f_i} \otimes \beta^l_{BW} \otimes \beta^l_{RC} \otimes \beta^l_{VA}$;
34: 　　　$\beta^u = \beta^u_{f_i} \otimes \beta^u_{BW} \otimes \beta^u_{RC} \otimes \beta^u_{VA}$;
35: 　　　$\alpha^l_{R_j,f_i} = \min\{(\alpha^l_{f_i} \oslash \beta^u) \otimes \beta^l, \beta^l\}$;
36: 　　　$\alpha^u_{R_j,f_i} = \min\{(\alpha^u_{f_i} \otimes \beta^u) \oslash \beta^l, \beta^u\}$;
37: 　　　**if** $\forall f_k \in \Theta_{R_j,f_i}$ have been calculated **then**
38: 　　　　$\alpha^l_{R_j,f_i} = \alpha^l_{R_j,f_i} + \sum_{f_k \in \Theta_{R_j,f_i}} \alpha^l_{R_j,f_k}$;
39: 　　　　$\alpha^u_{R_j,f_i} = \alpha^u_{R_j,f_i} + \sum_{f_k \in \Theta_{R_j,f_i}} \alpha^u_{R_j,f_k}$;
40: 　　　　$\beta^{l'}_{SA,R^p_j} = (\beta^{l'}_{SA,R^p_j} - \alpha^u_{R_j,f_i}) \bar{\oslash} 0$;
41: 　　　　$\beta^{u'}_{SA,R^p_j} = \max\{(\beta^{u'}_{SA,R^p_j} - \alpha^l_{R_j,f_i}) \bar{\oslash} 0, 0\}$;
42: 　　　**end if**
43: 　　**end if**
44: 　　$\beta^l_{f_i} = \beta^l_{f_i} \otimes \beta^l_{R^p_j,f_i}; \beta^u_{f_i} = \beta^u_{f_i} \otimes \beta^u_{R^p_j,f_i}$;
45: 　**end for**
46: **end for**

dependency graph model [8], FLA [7], LLA [14] and DNC [16], etc. There are also extensive research on the buffer sizing of priority-aware NoC, representative works include shaping delay analysis [41] and LLBA [15]. Among all these analytical methods, LLA [14][15] and DNC [16] based model outperform the others when the tightness of delay and backlog bound are considered. Thus, we will only perform comparison with LLA and DNC, as presented in subsection V-A and subsection V-B, respectively.

A. Comparison with Link Level Analysis

The network topology and flows are shown in Fig. 1. There are four flows (i.e. f_1, f_2, f_3 and f_4) in the network, with different priority $P_4 > P_1 > P_2 > P_3$. Suppose the packet length of flow f_i is F_i (in flits) and the injection period is I_i. We perform the comparison on a set of periodical traffic due to the restriction of LLA method [14][15], and the traffic jitter for all the flows are set to be zero. To ease the analysis, LLA supposes the number of bits in a flit is the same as the physical channel width, and the latency of a flit traverses a router is 1 cycle. To compare with LLA, our model for the standard 5 stages wormhole router should be specialized to a single cycle router, this is achieved by letting the service curve of BW, RC, VA and ST stage be burst delay function $\delta_0(t)$. Under this condition, the service curve of entire router is the same as the service curve provided by the SA stage, which is $< \beta^l_{SA,R^p_i}, \beta^u_{SA,R^p_i} >$. In addition, the LLA method does not consider the self-blocking caused by flow control. Thus, we set the credit feedback delay $\sigma = 0$ cycle in our model for a fair comparison. Next, we compare the end-to-end delay and buffer requirement computed with LLA and our model.

1) End-to-End Delay: For this scenario, we suppose the VC buffer is large enough so that we can ignore the back-pressure caused by flow control and directly apply LLA for the delay analysis. Suppose all the flows have the same injection period I_i and packet length F_i, we examine the end-to-end delay of the four flows in Fig. 1 under different packet length F_i and injection period I_i, the calculated result is shown in Table I. The RTC result is obtained by collapsing the CSP of f_2 (i.e. R_{13} and R_9). Each quaternion in the table is corresponding to the worst-case delay of f_1, f_2, f_3 and f_4 (in cycles). The blanking items corresponding to LLA columns indicate that the worst-case delay of a flow is greater than its injection period which is unable to be analyzed with LLA [14][15], and the blanking items corresponding to RTC columns indicate that the network is unstable because the injection rate exceeds the processing capability of the network. As shown in Table I, the RTC method is applicable to these scenarios which can not be analyzed by LLA and give the worst-case delay. In addition, we also observed from the table that the RTC result is as tight as that of LLA except for the scenarios that the worst-case delay of a flow is close to the injection period, e.g. $F_i = 1$, $I_i = 6$ in Table I. The root source of these exceptions is due to the fact that RTC theory we used in this paper is a count-based algebra approach [42], which ignores the state information of entire network.

TABLE I: Delay comparison with link level analysis

I_i	$F_i = 1$		$F_i = 2$		$F_i = 4$	
	RTC	LLA	RTC	LLA	RTC	LLA
3	7,7,8,4	—	—	—	—	—
4	6,6,6,4	—	—	—	—	—
5	6,6,6,4	—	9,10,12,5	—	—	—
6	5,6,6,4	5,6,5,4	9,8,9,5	—	—	—
7	5,6,5,4	5,6,5,4	9,8,9,5	—	—	—
8	5,6,5,4	5,6,5,4	7,8,9,5	7,8,7,5	—	—
9	5,6,5,4	5,6,5,4	7,8,9,5	7,8,7,5	15,16,19,7	—
10	5,6,5,4	5,6,5,4	7,8,7,5	7,8,7,5	15,12,15,7	—
11	5,6,5,4	5,6,5,4	7,8,7,5	7,8,7,5	15,12,15,7	—
12	5,6,5,4	5,6,5,4	7,8,7,5	7,8,7,5	11,12,15,7	—
13	5,6,5,4	5,6,5,4	7,8,7,5	7,8,7,5	11,12,15,7	—
14	5,6,5,4	5,6,5,4	7,8,7,5	7,8,7,5	11,12,15,7	—
15	5,6,5,4	5,6,5,4	7,8,7,5	7,8,7,5	11,12,15,7	11,12,11,7
16	5,6,5,4	5,6,5,4	7,8,7,5	7,8,7,5	11,12,11,7	11,12,11,7
17	5,6,5,4	5,6,5,4	7,8,7,5	7,8,7,5	11,12,11,7	11,12,11,7

2) Buffer Sizing: The LLBA method [15] can only give the required buffer size at each VC to prevent flow control, because the flow control can lead to chain blocking in the network, which complicates the analysis with LLBA. Our RTC-based performance model can model the flow control behavior in a natural way, because we abstract the flow control mechanism with the wide-sense service curve. By taking the flow control into consideration, our buffer sizing algorithm can be utilized to reduce the buffer size calculated with LLBA method further as long as the deadline constraint is not violated. We take a simple example to demonstrate the strength of our method. Suppose all the flows have the same packet injection period $I_i = 20$ cycles and packet length $F_i = 4$ flits ($i = 1, 2, 3, 4$), we can get the end-to-end delay with LLA method for the four flows, which are (11,12,11,7) cycles. We can also get the buffer size reserved at each router for the four flows with LLBA method, which are $(1, 1, 1, 4)$, $(4, 1, 1, 1, 1)$, $(4, 4, 1, 1)$ and $(1, 1, 1, 1)$, respectively. For the same configuration, if we change the deadline constraint from 15 cycles to 27 cycles, the buffer required for each flow can be further reduced with our buffer sizing algorithm, as shown in Table II. Take the deadline $D_i = 27$ as an example, the calculated buffer size is 58.62% smaller than the total buffer size calculated with the LLBA method.

TABLE II: buffer requirement computed with RTC model

Deadline	f_1	f_2	f_3	f_4	Total
(15,15,15,15)	(1,1,1,2)	(2,1,1,1,1)	(4,4,1,1)	(1,1,1,1)	25
(17,17,17,17)	(1,1,1,2)	(2,1,1,1,1)	(3,3,1,1)	(1,1,1,1)	23
(19,19,19,19)	(1,1,1,2)	(2,1,1,1,1)	(2,2,1,1)	(1,1,1,1)	22
(21,21,21,21)	(1,1,1,2)	(1,1,1,1,1)	(2,2,1,1)	(1,1,1,1)	20
(23,23,23,23)	(1,1,1,1)	(1,1,1,1,1)	(2,2,1,1)	(1,1,1,1)	19
(25,25,25,25)	(1,1,1,1)	(1,1,1,1,1)	(2,2,1,1)	(1,1,1,1)	19
(27,27,27,27)	(1,1,1,1)	(1,1,1,1,1)	(1,1,1,1)	(1,1,1,1)	17

B. Comparison with Network Calculus

In this subsection, we present the numerical results to demonstrate the improvement of our method over DNC method proposed in [16]. We use the periodical traffic as an example to make the comparison. Denote by I_i the injection period, F_i the packet length, the arrival curve can be easily
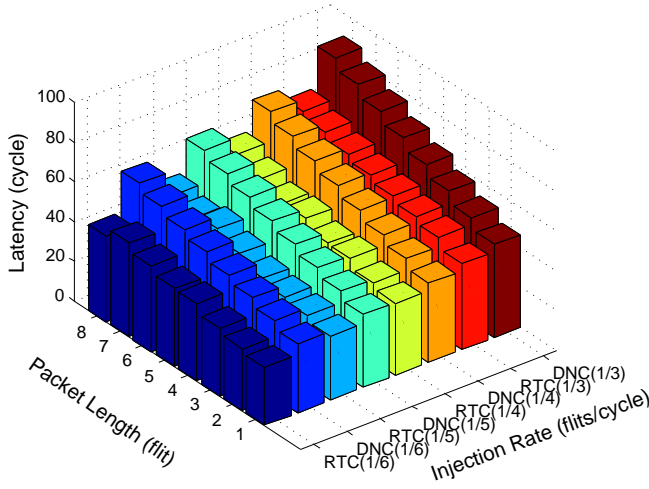
Fig. 6: Comparison with network calculus

Fig. 7: Service curve computed with RTC and DNC

obtained according to the method introduced in subsection IV-A. Suppose the priority of each flow in Fig. 1 satisfy $P_4 = P_1 > P_2 > P_3$, the buffer depth at each router as $B = 15$ flits, credit feedback delay $\sigma = 0$ cycle. Define injection rate $V_i = F_i/I_i$, we change V_i ($i = 1, 2, 3, 4$) from $1/3$ flits/cycle to $1/6$ flits/cycle and packet length from 1 flit to 8 flits, the end-to-end delay of flow f_3 calculated with DNC and our method are ploted in Fig. 6. By comparison, we find that our method can derive a much tighter delay bound than the DNC method proposed in [16]. The rooted cause for this improvement lies in the fact that our method utilizes the maximum service curve to limit the output arrival curve, and further leads to a tighter leftover service curve for the low-priority flows. To demonstrate this, for the same configuration, we plot the calculated service curve for flow f_2 both with DNC and RTC in Fig. 7. The numerical calculation was carried out with RTC toolbox. From Fig. 7, we find that the calculated service curve with DNC indeed underestimates the service curve of f_2, and this underestimation directly leads to the looser delay bound when compared with our method.

C. Comparison with Simulation

The correctness of our delay analysis algorithm and buffer sizing algorithm is verified by simulation. We modified the switch allocator of a modular open source NoC simulator, Heterogenous Networks-on-Chip (HNoCs) [43], to support fixed-priority scheduling, and collect the maximal end-to-end packet delay at each destination IP core.

VI. CONCLUSION

The priority-aware wormhole-switched NoC is a promising platform for the on-chip real-time communication if the worst-case performance can be accurately analyzed and guaranteed. Simulation is not competent for this purpose because it is difficult to cover all the corner cases. In this paper, we propose an RTC based performance model to achieve this goal. We first build the traffic model and service model for this NoC,
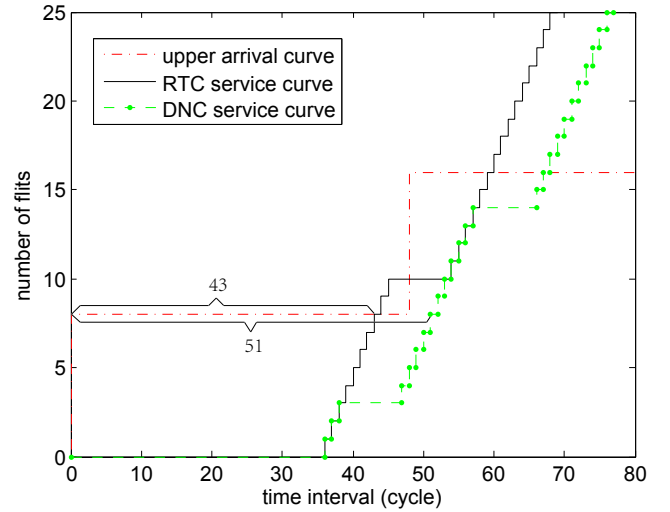
and propose a novel method to derive the upper service curve of credit-based flow control. Compared with the FLA and LLA method which assume the router to be single cycle and free of flow control, our performance model is more general and comprehensive. Then, based on the proposed RTC model, we proposed an end-to-end delay analysis algorithm and a buffer sizing algorithm. The delay analysis algorithm can be implemented to compute the end-to-end delay for each flow automatically, and verify whether all these flows meet their deadline under this configuration. To improve the calculated delay bound, we also proposed the concept of collapsible sub-path. The proposed buffer sizing algorithm can optimize the buffer size from high-priority flows to low-priority flows. It can also be implemented to perform the buffer reduction automatically under the constraint of deadline. Compared with the DNC based performance model, our model can give tighter performance bound, because the RTC-based model take the maximum service curve and minimum arrival curve into consideration. Experimental results also illustrated that, our method indeed outperforms the conventional analytical methods, e.g. LLA and DNC, when the tightness of performance bound are considered. Our results can be applied to the mapping, routing and power reduction of NoC.

REFERENCES

[1] E. Bolotin, Z. Guz, I. Cidon, R. Ginosar, and A. Kolodny. The power of priority: Noc based distributed cache coherency. In *Networks-on-Chip (NoCs), 2007. First International Symposium on*, pages 117–126, May 2007.

[2] Jörn Ostermann, Jan Bormans, Peter List, Detlev Marpe, Matthias Narroschke, Fernando Pereira, Thomas Stockhammer, and Thomas Wedi. Video coding with h. 264/avc: tools, performance, and complexity. *Circuits and Systems magazine, IEEE*, 4(1):7–28, 2004.

[3] Kees Goossens, John Dielissen, and Andrei Rădulescu. The Æthereal network on chip: Concepts, architectures, and implementations. *IEEE Design & Test of Computers*, 22(5):414–421, Sep.-Oct. 2005.

[4] Shaoteng Liu, A. Jantsch, and Zhonghai Lu. Analysis and evaluation of circuit switched noc and packet switched noc. In *Digital System Design (DSD), 2013 Euromicro Conference on*, pages 21–28, Sept 2013.

[5] C. Paukovits and H. Kopetz. Concepts of switching in the time-triggered network-on-chip. In *Embedded and Real-Time Computing Systems and Applications, 2008. RTCSA '08. 14th IEEE International Conference on*, pages 120–129, 2008.

[6] M.D. Harmanci, N.P. Escudero, Y. Leblebici, and P. Ienne. Providing qos to connection-less packet-switched noc by implementing diffserv functionalities. In *System-on-Chip, 2004. International Symposium on*, pages 37–40, Nov 2004.

[7] Zheng Shi and Alan Burns. Real-time communication analysis for on-chip networks with wormhole switching. In *Proceedings of the Second ACM/IEEE International Symposium on Networks-on-Chip (NoCs), 2008*, pages 161–170, Washington, DC, USA, 2008. IEEE Computer Society.

[8] Byungjae Kim, Jong Kim, SungJe Hong, and Sunggu Lee. A real-time communication method for wormhole switching networks. In *Parallel Processing, 1998. International Conference on*, pages 527–534, 1998.

[9] S.L. Hary and F. Ozguner. Feasibility test for real-time communication using wormhole routing. *Computers and Digital Techniques, IEE Proceedings -*, 144(5):273–278, 1997.

[10] Jong-Pyng Li and Matt W. Mutka. Real-time virtual channel flow control. *Journal of Parallel and Distributed Computing*, 32(1):49 – 65, 1996.

[11] S. Balakrishnan and F. Ozguner. A priority-driven flow control mechanism for real-time traffic in multiprocessor networks. *Parallel and Distributed Systems, IEEE Transactions on*, 9(7):664–678, 1998.

[12] Partha Kundu. On-die interconnects for next generation cmps. In *2006 Workshop on On- and Off-Chip Interconnection Networks for Multicore Systems*, Stanford, CA, USA, December 2006.

[13] G. Michelogiannakis, D. Sanchez, W.J. Dally, and C. Kozyrakis. Evaluating bufferless flow control for on-chip networks. In *Networks-on-Chip (NOCS), 2010 Fourth ACM/IEEE International Symposium on*, pages 9–16, May 2010.

[14] Hany Kashif, Hiren D. Patel, and Sebastian Fischmeister. Using link-level latency analysis for path selection for real-time communication on nocs. In *Proceedings of the Asia South Pacific Design Automation Conference (ASPDAC)*, pages 499–504, Sydney, Australia, February 2012.

[15] Hany Kashif and Hiren Patel. Bounding buffer space requirements for real-time priority-aware networks. In *Proceedings of the Asia South Pacific Design Automation Conference (ASPDAC)*, SunTec, Singapore, January 2014.

[16] Yue Qian, Zhonghai Lu, and Qiang Dou. Qos scheduling for nocs: Strict priority queueing versus weighted round robin. In *Computer Design (ICCD), 2010 IEEE International Conference on*, pages 52–59, Oct 2010.

[17] Jean-Yves Le Boudec and Patrick Thiran. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer-Verlag, Berlin, Heidelberg, 2001.

[18] S. Chakraborty, S. Kunzli, and L. Thiele. A general framework for analysing system properties in platform-based embedded system designs. In *Design, Automation and Test in Europe Conference and Exhibition, 2003*, pages 190–195, 2003.

[19] W. J. Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. In *Proceedings of the 38th Design Automation Conference*, pages 684–689, Jun. 2001.

[20] Evgeny Bolotin, Israel Cidon, Ran Ginosar, and Avinoam Kolodny. QNoC: QoS architecture and design process for network on chip. *Journal of Systems Architecture, Special Issue on Networks on Chip*, 50(2-3):105–128, Feb. 2004.

[21] P. Poplavko, T. Basten, M. Bekooij, J. van Meerbergen, and B. Mesman. Task-level timing models for guaranteed performance in multiprocessor networks-on-chip. In *Proceedings of the 2003 international conference on Compilers, architecture and synthesis for embedded systems*, pages 63–72. ACM, 2003.

[22] Yue Qian, Zhonghai Lu, and Wenhua Dou. Analysis of worst-case delay bounds for best-effort communication in wormhole networks on chip. In *Networks-on-Chip (NoCs), 2009. 3rd ACM/IEEE International Symposium on*, pages 44–53, May 2009.

[23] Gaoming Du, Cunqiang Zhang, Zhonghai Lu, Alberto Saggio, and Minglun Gao. Worst-case performance analysis of 2-d mesh nocs using multi-path minimal routing. In *Proceedings of the Eighth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, CODES+ISSS '12, pages 123–132, New York, NY, USA, 2012. ACM.

[24] Sunggu Lee. Real-time wormhole channels. *J. Parallel Distrib. Comput.*, 63(3):299–311, March 2003.

[25] D. Rahmati, S. Murali, L. Benini, F. Angiolini, G. De Micheli, and H. Sarbazi-Azad. Computing accurate performance bounds for best effort networks-on-chip. *Computers, IEEE Transactions on*, 62(3):452–467, 2013.

[26] Ciprian Seiculescu, Dara Rahmati, Srinivasan Murali, Luca Benini, Giovanni De Micheli, and Hamid Sarbazi-Azad. Designing Best Effort Networks-on-Chip to Meet Hard Latency Constraints. *ACM Transactions on Embedded Computing Systems*, 12(4), 2013.

[27] Zhonghai Lu, Axel Jantsch, and Ingo Sander. Feasibility analysis of messages for on-chip networks using wormhole routing. In *Proceedings of the Asia and South Pacific Design Automation Conference*, pages 960–964, Shanghai, China, Jan. 2005.

[28] Zheng Shi. *Real-Time Communication Services for Networks on Chip*. PhD thesis, The University of York, 2009.

[29] M. Gries, C. Kulkarni, C. Sauer, and K. Keutzer. Comparing analytical modeling with simulation for network processors: a case study. In *Design, Automation and Test in Europe Conference and Exhibition, 2003*, pages 256–261 suppl., 2003.

[30] D.B. Chokshi and P. Bhaduri. Modeling fixed priority non-preemptive scheduling with real-time calculus. In *Embedded and Real-Time Computing Systems and Applications, 2008. RTCSA '08. 14th IEEE International Conference on*, pages 387–392, 2008.

[31] Andrei Hagiescu, Unmesh D. Bordoloi, Samarjit Chakraborty, Prahladavaradan Sampath, P. Vignesh V. Ganesan, and S. Ramesh. Performance analysis of flexray-based ecu networks. In *Proceedings of the 44th Annual Design Automation Conference*, DAC '07, pages 284–289, New York, NY, USA, 2007. ACM.

[32] Lothar Thiele, Ernesto Wandeler, and Samarjit Chakraborty. Performance analysis of multiprocessor dsps: a stream-oriented component model. *Signal Processing Magazine, IEEE*, 22(3):38–46, 2005.

[33] Ernesto Wandeler and Lothar Thiele. Real-time calculus (rtc) toolbox. http://www.mpa.ethz.ch/Rtctoolbox, 2006.

[34] N.E. Jerger and L.S. Peh. On-chip networks. *Synthesis Lectures on Computer Architecture*, 4(1):1–141, 2009.

[35] W.J. Dally and S.G. Tell. The even/odd synchronizer: A fast, all-digital, periodic synchronizer. In *Asynchronous Circuits and Systems (ASYNC), 2010 IEEE Symposium on*, pages 75–84, May 2010.

[36] E. Wandeler, L. Thiele, M. Verhoef, and P. Lieverse. System architecture evaluation using modular performance analysis: a case study. *INTERNATIONAL JOURNAL ON SOFTWARE TOOLS FOR TECHNOLOGY TRANSFER (STTT)*, 8(6):649–667, 2006.

[37] Henrik Schiøler, Jan Jakob Jessen, Jens Dalsgaard Nielsen, and Kim Guldstrand Larsen. Network calculus for real time analysis of embedded systems with cyclic task dependencies. In *Proc. 20th International Conference on Computers and Their Applications, CATA 2005*, pages 326–332, 2005.

[38] Lothar Thiele and Nikolay Stoimenov. Modular performance analysis of cyclic dataflow graphs. In *Proceedings of the Seventh ACM International Conference on Embedded Software*, EMSOFT '09, pages 127–136, New York, NY, USA, 2009. ACM.

[39] Zheng Shi and A. Burns. Real-time communication analysis with a priority share policy in on-chip networks. In *Real-Time Systems, 2009. ECRTS '09. 21st Euromicro Conference on*, pages 3–12, July 2009.

[40] Jia Zhan, N. Stoimenov, Jin Ouyang, L. Thiele, V. Narayanan, and Yuan Xie. Designing energy-efficient noc for real-time embedded systems through slack optimization. In *Design Automation Conference (DAC), 2013 50th ACM / EDAC / IEEE*, pages 1–6, May 2013.

[41] S. Manolache, P. Eles, and Zebo Peng. Buffer space optimisation with communication synthesis and traffic shaping for nocs. In *Design, Automation and Test in Europe, 2006. DATE '06. Proceedings*, volume 1, pages 1–6, March 2006.

[42] L. T. X. Phan, S. Chakraborty, and P. S. Thiagarajan. *A Multi-Mode Real-Time Calculus*, pages 59–69. Real-Time Systems Symposium - Proceedings. Ieee Computer Soc, Los Alamitos, 2008.

[43] Y. Ben-Itzhak, E. Zahavi, I. Cidon, and a. kolodny. Hnocs: Modular open-source simulator for heterogeneous nocs. In *Embedded Computer*

Systems (SAMOS), 2012 International Conference on, pages 51–57, July 2012.