A Delay and Backlog Model for Priority-Aware Networks-on-Chip (NoC)

Baoliang Li

***Abstract*—Networks-on-Chip (NoC) is a key component for modern Chip-MultiProcessors(CMPs) and System-on-Chip (SoC). Among all the implementation alternatives, priority-aware wormhole-switched NoC is promising to meet the rigorous requirements of on-chip latency-aware communication, e.g. cache coherent protocol and multimedia applications. The end-to-end latency and buffer requirement analysis are very important for the development of real-time applications on this platform. In this paper, we propose a Real-Time Calculus (RTC) based latency and backlog analysis model to achieve this goal. Our model is topology-insensitive, it takes as input the scheduling network model and the application traffic characterization, and gives the end-to-end latency and backlog bound for each traffic flow, which enables the fast performance evaluation and buffer optimization. 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 the derived delay and backlog bound are considered.**

***Keywords*—Networks-on-Chip (NoC), priority-aware, wormhole switch, real-time calculus, delay and backlog bound**

I. Introduction

Conventional 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, higher power efficiency, and low-latency global communication, etc. As a key component of CMPs and SoC, NoC must be well designed to meet the rigorous requirements on end-to-end latency, buffer constraint, throughput and power, etc. Although various proposals have been emerged, with each focusing on improving different performance metrics of on-chip communication, most of the existing research on NoC are focusing on the improvement of average performance of on-chip wormhole-switched network, and simulation is the widely used performance evaluation method. Whereas, 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 [**?**][**?**] and multimedia application [**?**]. How to design the on-chip communication infrastructure for these applications and analyze its feasibility are of big challenges 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) [**?**] and time-triggered switch [**?**], etc. Although provide 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 real-time and non-real-time communication requirements is the most promising solution. To achieve this goal, a special scheduling policy (e.g. DifServ [**?**] or priority-aware implementation [**?**][**?**][**?**]) or flow control mechanism (e.g. [**?**][**?**]) should be implemented. For all these wormhole-switch based real-time communication proposals, a key step before their adoption as a platform of real-time applications is the analysis of the worst-case communication latency for all the real-time flows, to guarantee that the deadline of each flow can be met. In addition, an effective buffer estimation approach is also needed to optimize the buffer allocation under real-time constraint.

A reliable worst-case analysis is crucial for the application of wormhole-switched NoC, because an over optimistic estimation will lead to a wrong implementation, while an over pessimistic estimation will make the utilization of on-chip resource very low. The conventional simulation based method might not be competent for the worst-case analysis, this is because the worst-case scenarios are hard to be captured by simulation. As an alternative, the mathematical approach can establish the relationship between performance metrics and design parameters in a very short time, and giving the worst-case performance immediately. For the worst-case analysis of fixed-priority wormhole-switched on-chip networks, Flow-Level Analysis (FLA) [**?**], Link-Level Analysis (LLA) [**?**][**?**] and network calculus [**?**] based analysis are widely used. Both FLA and LLA have their roots in the classic scheduling theory, which assume that, the traffic flows are strictly periodic, and the packet transmission delay is less than the packet inter-arrival time. In addition, they both assume that, the input buffer of wormhole-switched NoC is sufficient large, so that the back-pressure caused by flow control can be ignored. Network calculus overcomes these limitations by allowing the packet arrival at arbitrary pattern and arbitrary inter-arrival time. By applying the advanced sub-additive closure operator in dioid algebra, the large buffer assumption can also be eliminated.

However, we found that, the network calculus based latency bound can be further improved, this is because, the network calculus based method [**?**] ignored the maximal service capacity of on-chip routers, which has significant impact on the output arrival curve of traffic flow served by a specific router. The over estimated output arrival curve will further leads to a looser service curve left for other flows, and the over pessimistic service curve will finally leads to a looser performance bound for the low-priority flows. We will further

demonstrate this phenomenon in Section **??**. To overcome this shortcoming of network calculus, we adopt the real-time calculus (RTC) to improve these delay bound in this paper. Compared with network calculus, the RTC use the maximal service curve and minimal arrival curve to limit the output arrival curve and left service curve, which usually leads to a tighter performance bound. The advantages of RTC will be also demonstrated in section **??**. In this paper, we build an end-to-end latency model for the wormhole-switched, credit-based flow controlled on-chip networks with RTC, and an algorithm is proposed to automatically compute the end-to-end performance bound for each flows. The main contribution of this paper is two folded: (1) we propose an end-to-end performance analysis algorithm, compared with the existing methods, e.g. LLA [**?**][**?**] and network calculus [**?**], it can gives better performance bound. The output of this algorithm can be used for the IP core mapping, task mapping, routing selection, or NoC parameters configuration; (2) we propose an RTC based buffer optimization algorithm for the application-specific NoC the reduce the buffer size under strict deadline constraint.

The rest of this paper is organized as follows: we present the existing real-time communication proposals and its related performance analysis method in Section **??**. In Section **??**, the basic assumptions on wormhole switched on-chip network and RTC theory is introduced. The detailed modeling process is presented in Section **??**, and we present the experiment results and comparison with other mathematical methods in Section **??**. Finally, we summarize our paper in Section **??**.

II. Related Work

Since introduced in 2001 [**?**], various NoC proposals have been emerged to meet different on-chip communication requirements. Different applications have different communication requirement, and the main requirements posed to NoC by on-chip applications are latency and bandwidth. To meet these demand, NoC are designed to be either either best-effort or guaranteed-service, depending on the hardware cost and application fields. The worst case analysis for these two categories is slightly different. Synchronous Data Flow (SDF) graph [**?**] and Deterministic Network Calculus [**?**] (DNC) 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, which extends applicable of the method. In [**?**], the authors build an analytical performance model with network calculus taking the various contention and flow control into consideration. This result is extended in [**?**], where the traffic splitter was proposed to support the multi-path routing polices. Another method is presented in [**?**] to compute the worst-case latency 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 latency bound in [**?**], and used for topology synthesizing of best-effort NoC in [**?**]. For more details about the mathematic modeling of best-effort NoC please refer to [**?**].

For the implementation of service-guaranteed NoC, a simple and effective solution to provide differential service for different applications is classify 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 [**?**], fixed-priority NoC [**?**] and Æthereal [**?**][**?**] etc. Although all the analytical methods mentioned above aim for the worst-case analysis of conventional NoC, when they are adopted to the priority-aware NoC, the obtained performance bound is very conservative, especially for the high priority flows. This is because, these methods do not take the priority-aware scheduling into consideration. Thus, new methods should be developed for the real-time communication analysis. In [**?**], contention tree was proposed to analyze the feasibility of real-time traffic delivered by wormhole-switched NoC. It improves the previous results , e.g. lumped link model [**?**] and dependency graph model[**?**], by allowing the concurrent link usage. This is similar as the the Link Level Analysis [**?**][**?**] (LLA), which improved the Flow-Level-Analysis (FLA) proposed in [**?**]. The main difference between FLA and LLA is that, FLA treats the entire route of a flow as a whole, while the LLA treats each link segment separately. Link-level analysis can provide much accurate results than FLA [**?**], this is because in the LLA, the latency on specific link only related with the inference other flows posed on the target flow on previous links. The main drawback of FLA and LLA is that, these method require the traffic arrival periodically, and the packet inter-arrival time must greater than the end-to-end transmission latency, and the previous research based on these two method assumes that, the buffer at each routers is sufficient large, so that the feedback caused by flow-control can be ignored. To overcome these shortcomings, network calculus [**?**] was used to analyze the worst case latency of priority-aware NoC.

But we found that, the network calculus results can be further improved if we take the maximum service curve of each router and minimum arrival curve of each flow into consideration, this is because the maximal service curve and minimum arrival curve can limit the output arrival curve, which leads to a much tighter left service curve for the low priority flow. (to explain the reason, please refer to Theorem 1.6.2 in [**?**]). To overcome this shortcoming, we adopt the Real-Time Calculus (RTC) [**?**][**?**] originally used for real-time task scheduling analysis to compute the worst-case end-to-end performance of priority-aware wormhole-switched NoC. Real-Time Calculus is extension of network calculus by integrating the maximum service curve and lower arrival curve into the network calculus theory. Due to the theoretical results is usually very tight, it has been widely used in the modeling and analysis of network processor [**?**], CAN [**?**], FlexRay [**?**][**?**] and DSP systems [**?**], etc. To ease the application of Network Calculus, an RTC toolbox [**?**] has also been implemented to support the numerical calculation.

III. Preliminaries

A. Basic Assumptions

For the detailed description of NoC architecture, please refer to [**?**][**?**]. In this paper, we consider the same network

model as in [**?**][**?**][**?**][**?**], each router has 5 pipeline stages, i.e. Buffer-Write (BW), Route Computation (RC), VC Allocation (VA), Switch Allocation (SA), Switch Traversal (ST) and Link Traversal (LT). IP core communication with each other by transmitting packets, each packet is composed of a header flit and several non-header flits optionally. Each header flit should traverse all the five stages to find a path and reserve buffer space for the follow non-header flits, and non-header flits skip the RC and VA stage since the routine and VC have been determined by header flit. To ensure the predication of packet, we assume the NoC adopts deterministic routing polices and 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, the priority-aware switch allocator 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 packet of high priority can also preempt the transmission of packets with low priority. In addition, we also assume that, the buffer depth of each VC is finite, and credit-based flow control is adopted between each router. Although we focus on the 5-stage router, our method can be easily adopted to the speculation-based router, e.g. routers with only 2 or 3 stages, the only difference is the initial delay of our model. To simplify of our analysis, we also assume that, the entire chip use synchronous design, the frequency and period of clock are f and T, respectively. Our method can also be applied to analyze Global Asynchronous Local Synchronous (GALS) NoC without any modification, because the router located in different voltage-frequency islands can communicate with a half cycle fixed-latency synchronizer [**?**].

Our model is topology independent, but to demonstrate the basic idea of our method, we take the mesh topology as an example, as shown in Fig. **??**. There are 4 traffic flows in the network, i.e. f_1, f_2, f_3 and f_4. A flow is a sequence of packets following the same transmission path with the same source and destination address. We must emphasize that, although there is only four flows in the network, it is sufficient to demonstrate our idea, 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. Denote by P_i the priority of flow f_i, in the example shown in Fig. **??**, we assume $P_1 = P_4 = 3$, $P_2 = 2$ and $P_3 = 1$, and a higher value indicates a higher priority. Our method extends the existing methods [**?**][**?**] which assumes per-flow priority, to allows multiple flows share a same priority, e.g. both f_1 and f_4 in Fig. **??** are assigned the highest priority. Flits from different flows with the same priority are served in round-robin fashion, and the flits of the same flow were served in FIFO order. 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 [**?**]. Our results can be used to determine the worst-case latency and backlog bound of each flow. If all the packets of one flow can be transmitted within their deadlines, the flow is schedulable.

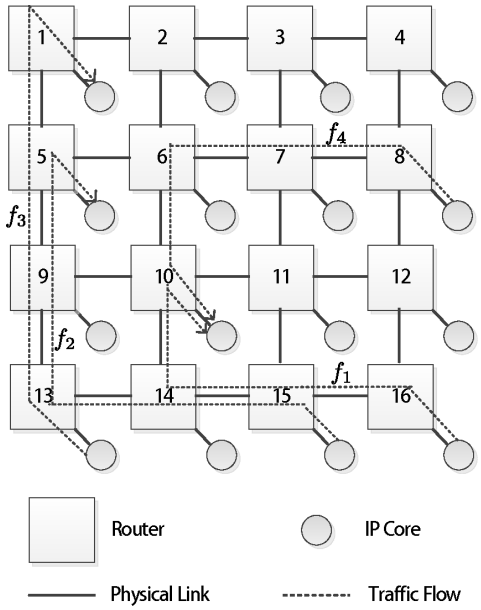


Fig. 1: A Mesh NoC with 4 Traffic Flow

B. Introduction to Real-Time Calculus

Real-time calculus is an extension of network calculus [**?**][**?**], by adding the upper service curve and lower arrival curve to describe the maximal service capacity and lower arrival rate. It is the mathematical basis of Modular Performance Analysis (MPA) [**?**] 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 [**?**][**?**].

Definition 1 (Real-Time Arrival Curve): Let $R[s,t)$ denote the number of events that arrived in 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.

From the definition, we find that the upper arrival curve (corresponding to the arrival curve of network calculus [**?**]) and lower arrival curve are used to characterize the upper and lower bound of arrival event with any interval Δ.

Definition 2 (Real-Time Service Curve): 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 α^l and upper service curve α^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 of RTC service curve, we find that the upper and lower service curve are corresponding to the maximal service curve and service curve in network calculus [**?**], respectively. Thus, the concatenation theorem (see Theorem

1.46 and Theorem 1.6.1 in [**?**]) for service curve and maximal service curve can also be applied to RTC service curve. In this paper, event is refer to the arrival of flits. In this paper, we will use the discrete time RTC arrival curve and service curve to characterize the system, this is because the minimal time unit in the wormhole-switched NoC is clock period T. 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 of this flow and leftover service curve for the other flows with the following equations:

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

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

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

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

where $\lceil \cdot \rceil$, $\lfloor \cdot \rfloor$ are the ceiling operator and flooring operator, and $\otimes$, $\oslash$, $\bar{\otimes}$, $\bar{\oslash}$ are corresponding to the min-plus convolution, de-convolution, and max-plus convolution and de-convolution [**?**], respectively.

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

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

$$Backlog(f) = V(\alpha^u_{f_i}, \beta^l_{R_1} \otimes \beta^l_{R_2} \otimes \cdots \otimes \beta^l_{R_N}). \quad (6)$$

where $H(\cdot, \cdot)$ and $V(\cdot, \cdot)$ mean the maximal vertical and horizontal deviation, respectively.

IV. Real-Time Calculus based Performance Model

Before carry out the performance analysis with RTC, we should first build the traffic and router model. In this paper, we will use the discrete time service curve and arrival curve to characterize the router and traffic, these models can be input to the RTC toolbox [**?**] to compute the end-to-end latency automatically. While applying the RTC theory to evaluation the performance of priority-aware wormhole-switched NoC, the following four aspects should be considered: (1) Only header flit need to be processed by RC and VA stage, the subsequent flits of a packet just follow the decision of header flit. We need specific mechanism to characterize the service provided to header and non-header flits in a unified way, so that we can simplify our RTC model; (2) Our model allows priority sharing between flows, thus, the leftover service curve provided to lower priority flows can only be derived after all the flows with higher priority have been serviced; (3) Due to the on-chip buffer limitation, credit-based flow control is used as a back-pressure mechanism to prevent buffer overflow. Before analysis the end-to-end performance with Eq.(**??**) and Eq.(**??**), we should first break the cyclic-dependence caused by flow control; (4) To guarantee the tightness of our theoretical bound and reduce the computation complexity, we should collapse all the collapsible subpaths. These four issues are discussed in the follow subsections.

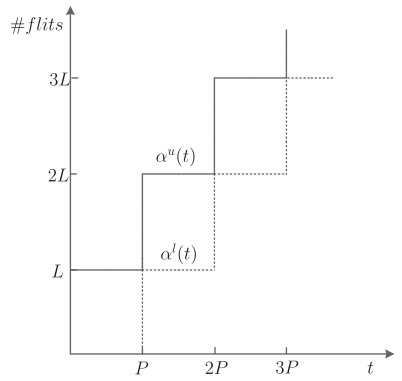


Fig. 2: Arrival Curve for Periodically Arriving Traffic

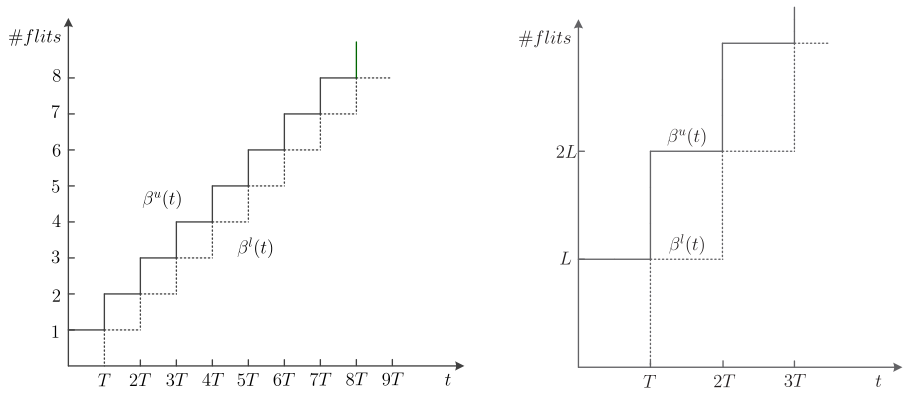
A. Traffic Model

In a CMP or SoC system interconnected with NoC, the communication between each pair of cores was realized by transmitting packets. Each packet is further divided into flits, which is the minimum transmission unit in wormhole-switched NoC. Denote the arrival curve $< \alpha^l(\Delta), \alpha^u(\Delta) >$ of flow as the minimum and maximum amount of flits within Δ time unit. Here, we provide two ways to obtain this arrival curve: (1) We can extract the arrival curve from the communication trace generated by a real CMP or SoC system with the sliding window method [**?**]. This method analyze the time series of flits by the follow way: for each window size Δ, it try to find the maximal and minimal number of arrived flits in the trace, denoted by $\alpha^l(\Delta)$ and $\alpha^u(\Delta)$, respectively. We need to mention that, the obtained arrival curve obtained by this method might not be periodic; (2) if we know the inter-arrival time of packets in a flow, we can also get the arrival curve directly. For example, suppose all the packets have the same length L and arrived periodically with period P. Then, we can obtain the service curve of this flow by simply amplifying the arrival curve for periodic event stream provided in [**?**] at y-axis with a factor L, as shown in Fig. **??**. The alert readers would notice that, the flit arrival curve obtained by this way is the same as the packet arrival curve $\mathcal{P}^L(\alpha)$. Thus, this traffic model can be used to compute the end-to-end packet latency, please refer to Theorem 1.7.1 in [**?**] for explanation and more details.

B. Router Model

While modeling the routers with RTC, we can analyze the data path stage by stage, and find the service curve for each stage. Then, the service curve provided by the router to each flow can be obtained by concatenating all the service curves of these five stages. This is significantly different with the existing network calculus based model [**?**], [**?**], where they view the entire router as a whole. The advantage of our method is that, it can be easily modified to analyze the non-standard router architecture, just by letting the traversing time of non-existing stages to be zero. Thus, we first obtain the service curve 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.



(a) Service Curve of BW, SA and ST stages (b) Service Curve of RC and VA stages

Fig. 3: Service Model for Each Pipeline Stages

Take the BW stage as an example, because this stage output one flit at each cycle T. For any time interval of length less than T, the maximum and minimal number of flits that can be seen are one and zero, respectively. 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 upper and lower service curve are shown in Fig. **??**.

(2) RC stage and VA stage: only the head flit should traverse these two stages. Suppose the packet length is L, a sophisticated solution to construct service curve for these two stages is that, since the traverse latency for the non-head flit is 0 cycle, we can view these two stages can provide service for all the L flits within T period. Then, we can get the equivalent service curve for these two stages with the same method as BW and ST stages, as shown in Fig. **??**.

(3) SA stage: each output port in the wormhole-switched NoC has a SA scheduler to schedule the switch traversal at each cycle. Thus, following the same approach as BW and ST stage, we can get the service curve provided by SA stage to all the contention flows, as shown in Fig. **??**. For the fixed-priority based scheduling policy, each switch allocator provides service for high priority flows first, and flows with the same priority will be served with Round-Robin order. The unserved flits will be imposed an additional latency T due the failure of switch arbitration.

The equivalent service curve for the entire router can be obtained by concatenating all the five service curves together. To obtain a concise notation, denote by the upper service curve and lower service curve provided by Router R_i to flow f_j are $\beta^u_{R_i,f_j}$ and $\beta^l_{R_i,f_j}$, then

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

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

The alert readers would notice that, the contention of different flows only occurs at SA stage. If we obtained the service curve provided by SA stage to the target flow f_j, we can obtain the service curve of router directly. To obtain the service curve $< \beta^l_{SA,R_i,f_j}, \beta^u_{SA,R_i,f_j} >$, we should consider the following two cases:

(a) all the flows contending with f_j at R_i have lower priorities. Then, for the synchronize router architecture, leftover service curve after serving higher priority flows is provided to f_j totally. But for an asynchronous architecture, the service curve obtained by f_j might be lower than the leftover service curve. For the worst case, flit from f_j arrives at SA stage just after a flit from lower priority gotten granted. Thus, flit from f_j has to stall for a cycle. This is refer to the inference from lower priority. At this circumstance, the final service curve obtained by flow f_j is

$$\beta^l_{SA,R_i,f_j} = \max\{0, \beta^{l'}_{SA,R_i,f_j} - 1\}. \tag{7}$$

Similarly, a higher priority flow can be blocked for a cycle by flow f_j. Thus, the upper service curve obtained by flow f_j is

$$\beta^u_{SA,R_i,f_j} = \min\{\beta^u_{SA,R_i,f_j}, \beta^{u'}_{SA,R_i,f_j} + 1\}. \tag{8}$$

(b) there exists some contention flows with the same priority as f_j, and the leftover service curve after serving flows with high priority than f_j is $< \beta^{l'}_{SA,R_i}, \beta^{u'}_{SA,R_i} >$. Denote by the set of contention flows at router R_i with the same priority as f_j as Θ_{R_i,f_j}, and let N be the number of flows in Θ_{R_i,f_j}. Then, the service curve provided to f_j is $< \lfloor \beta^l_{SA,R_i}/N \rfloor, \lceil \beta^u_{SA,R_i}/N \rceil >$, as shown in Fig. **??**, since all the flows in Θ_{R_i,f_j} got service in Round-Robin order. This is slightly different from the conventional continuous time model which is $< \beta^l_{SA,R_i}/N, \beta^u_{SA,R_i}/N >$, this is because our model is flit-level preemptive. 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.(**??**) and Eq.(**??**).

C. Upper Service Curve for Flow Controller

In this subsection, we try to break the dependence loop caused by flow control protocol and derive the equivalent service curve, especially the equivalent upper service curve, since the lower service curve has been fully discussed in the field of network calculus [**?**]. In fact, for this problem, researchers have proposed several solutions, e.g. fixed-point iteration [**?**][**?**], transformation from marked dataflow graph [**?**]. In this paper, we try to tackle the same problem with another solution. This is motivated by [**?**], where the authors

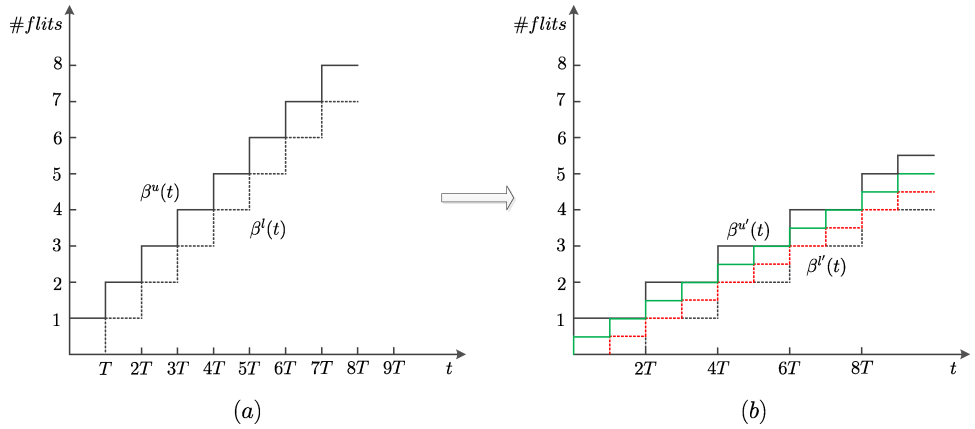


Fig. 4: Service Curve for N Flow With The Same Priority Under Round-Robin Scheduling Polices

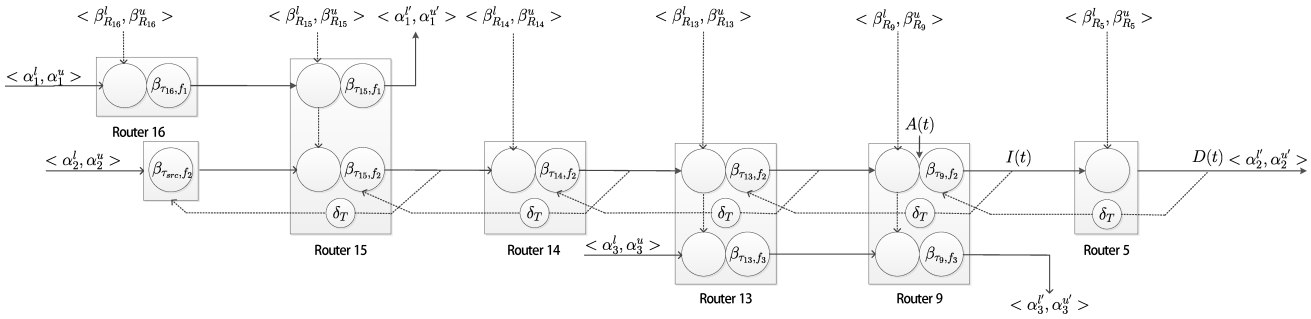


Fig. 5: Scheduling Network Model for Flow f_2

abstract the flow control as a component providing a service curve, and the equivalent service curve (corresponding to the lower service curve of RTC) of this flow controller was obtained by applying some basic dioid algebra. In this section, we follow the same procedure to derive the upper service curve of flow controller, to make the question clear, we take flow f_2 as an example to demonstrate this method. We assume that, the destination NI has sufficient large input buffer, thus there is no flow controller between the router and destination NI. But, to prevent the buffer overflow, there must be a flow controller between input NI and router. We plot the scheduling network of flow f_2 in Fig. **??**, and ignore f_4 and the flow control of other flows for simplicity. Denote by the output arrival process of router R_9, injection process and departure process of router R_5 as $A(t)$, $I(t)$ and $D(t)$, respectively. We try to derive the service curve of flow controller at router R_9, as shown in the following Theorem.

Theorem 1: Suppose each router provide an upper service curve β^u, the virtual channel depth of each router is B, and the feedback delay is T, then the flow controller provides an equivalent upper service curve β_σ^u, where

$$\beta_\sigma^u(t) = \overline{\beta^u \otimes \delta_T(t) + B}$$

and $\bar{f}$ is the sub-additive closure of f, i.e. $\bar{f} = \inf_{n\geq 0}\{f^{(n)}\}$ and $f^{(0)} = \delta_0(t)$[1].

[1] $\delta_T(t) = +\infty$ for $\forall t > T$, and 0 otherwise.

Proof: The feedback link can be represented as a network element providing upper service curve $\delta_T(t)$. Next, we applying the same idea as [**?**] to derive the upper service curve this flow controller. We know that, $I(t) = \min\{A(t), D'(t) + B\}$, where $D' = D \otimes \delta_T$. Denote by the service curve of router is β^u, thus $D(t) \leq I \otimes \beta^u(t)$. Here, we introduce a notation $\wedge$ to simplify our expression, where $a \wedge b = \min\{a, b\}$. Bring $I(t)$ and $D'(t)$ into this equation, we get

$$\begin{aligned} D(t) &\leq I \otimes \beta^u(t) \\ &\leq A \otimes \beta^u(t) \wedge (D \otimes \delta_T \otimes \beta^u(t) + B). \end{aligned}$$

By applying Theorem 4.31 in [**?**], we have

$$D \leq A \otimes \beta^u \otimes \overline{\beta^u \otimes \delta_T + B}.$$

Thus,

$$\begin{aligned} I &= A \wedge (D' + B) \\ &\leq A \wedge (D \otimes \delta_T + B) \\ &\leq A \wedge (A \otimes \beta^u \otimes \overline{\beta^u \otimes \delta_T + B} \otimes \delta_T + B) \\ &= A \otimes \delta_0 \wedge (A \otimes (\beta^u \otimes \delta_T + B) \otimes \overline{\delta_T \otimes \beta^u + B}) \\ &= A \otimes \overline{\beta^u \otimes \delta_T + B}. \end{aligned}$$

which implies that, the flow controller has an equivalent upper service curve $\overline{\beta^u \otimes \delta_T + B}$. ■

Theorem **??** derives the upper service curve of a single flow controller, and we can get the service curve of a router chain by applying Theorem **??** iteratively. As shown in Fig. **??**, the

downstream routers' service curve can affect the service curve of flow controller at upstream. Hence, we should first derive the service curve of f_2 obtained at each router with the method mentioned in Subsection **??**, and then compute the service curve of flow controller from destination to source. Service curve obtained at Router R_{15} by flow f_2 can be derived by applying Eq.(**??**) and Eq.(**??**), since f_2 is a lower priority flow at R_{15}. Router R_{13}, R_9 and R_5 provide all their service curves to f_2 because f_2 is the highest priority flow at these routers. Denote by the service curve of flow controller obtained by Theorem **??** at router R_9 is $< \beta^l_{\tau_9}, \beta^u_{\tau_9} >$, by applying the concatenation theorem, we can obtain the equivalent service curve of router R_9 are $< \beta^l_{R_9} \otimes \beta^l_{\tau_9}, \beta^u_{R_9} \otimes \beta^u_{\tau_9} >$. Then, by applying Theorem **??**, we can get the service curve of flow controller at router R_{13}, R_{14}, R_{15} and IP_{15} iteratively.

D. Longest Collapsible Sub-Path

We have build the traffic model, router model and flow control model in the previous subsection. Before we give the performance evaluation algorithm, we first consider the follow scenario. Take f_2 and f_3 in Fig. **??** as an example, they content the output link of both router R_{13} and R_9. Denote by the buffer depth of R_9 as B_9, the service curve of R_{13} and R_9 be $< \beta^l_{R_{13}}, \beta^u_{R_{13}} >$ and $< \beta^l_{R_9}, \beta^u_{R_9} >$, the service curve for flow controller of router R_{13} be $< \beta^l_{\tau_{13},f_2}, \beta^u_{\tau_{13},f_2} >$, the arrival curve of f_2 and f_3 at router R_{13} be $< \alpha^l_{R_{13},f_2}, \alpha^u_{R_{13},f_2} >$ and $< \alpha^l_{R_{13},f_2}, \alpha^u_{R_{13},f_2} >$. The question is: when B_9 is sufficiently large so that $\beta^l_{R_{13}} \otimes \beta^l_{\tau_{13},f_2} = \beta^l_{R_{13}}$ and $\beta^u_{R_{13}} \otimes \beta^u_{\tau_{13},f_2} = \beta^u_{R_{13}}$, how to derive the leftover service curve for f_3 at R_{13} and R_9, and obtain the delay and backlog bound efficiently with tighter bound?

An intuitive solution is that calculating the leftover service curve router by router: firstly, obtaining the leftover service curve of R_{13} by applying Eq.(**??**) and Eq.(**??**); secondly, deriving the output arrival curve $< \alpha^{l'}_{R_9,f_2}, \alpha^{u'}_{R_9,f_2} >$ of f_2 by applying Eq.(**??**) and Eq.(**??**); thirdly, the leftover service curve of R_9 can be easily obtained by applying Eq.(**??**) and Eq.(**??**); and finally, the leftover service curve for f_3 at R_{13} and R_9 can be easily obtained by concatenation theorem of service curve. Another solution is: we can substitute R_{13} and R_9 by a virtual router $R_{13,9}$ providing service curve $< \beta^l_{R_{13},f_2} \otimes \beta^l_{R_{13},f_2}, \beta^u_{R_{13},f_2} \otimes \beta^u_{R_{13},f_2} >$, since $\beta^l_{R_{13}} \otimes \beta^l_{\tau_{13},f_2} = \beta^l_{R_{13}}$ and $\beta^u_{R_{13}} \otimes \beta^u_{\tau_{13},f_2} = \beta^u_{R_{13}}$, as shown in Fig. **??**. Then, the leftover service curve f_3 obtained at R_{13} and R_9 can be directly obtained by applying Eq.(**??**) and Eq.(**??**). Compared with previous method, this one eliminates the calculation of intermediate arrival curve, and compute the equivalent service curve by just invoke Eq.(**??**) and Eq.(**??**) once, which is calculation efficient. We also demonstrate that, this method can achieve tighter performance bound compared with the former method, we will demonstrate this in Section **??**.

Next, we formalize this observation and propose an efficient performance calculation method. This optimization method is based on the follow concept.

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

1) all the routers in this sub-path, except the last one, satisfy the following condition:
   - $\beta^l_{R_{13}} \otimes \beta^l_{\tau_{13},f_2} = \beta^l_{R_{13}}$;
   - and $\beta^u_{R_{13}} \otimes \beta^u_{\tau_{13},f_2} = \beta^u_{R_{13}}$.
2) $P(f)$ is also the sub-path of Ω_f, where Ω_f is the set of contention flows with lower priority on this sub-path.

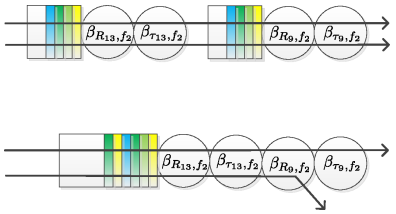


Fig. 6: Longest Collapsible Sub-Path of f_2

We can replace all the routers on $P(f)$ by a single virtual router providing service curve $< \otimes_{R_i \in P(f)} \beta^l_{R_i,f}, \otimes_{R_i \in P(f)} \beta^u_{R_i,f} >$, and the leftover service curve calculation is simplified. For the high priority flows, the process of calculating the end-to-end equivalent service curve is also a collapsing process. But, to compute the leftover service curve at some routers, we have to know the arrival curve at this router. With the aid of longest collapsible sub-path, we just only to compute the arrival curve at the starting point of each sub-path. For a longest collapsible sub-path with N routers, this method reduces $N-1$ times calculation of arrival curve and leftover service curve, which is time efficient.

E. Calculating the End-to-End Latency

After obtained the traffic model, router model and flow control model, compute the end-to-end latency is still a non-trivial task. The follow four aspects should be considered carefully: (1) We should always compute the end-to-end latency by collapse the system to a single virtual node, 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, our computation must start from higher priority flows to lower priority; (3) Before computing the leftover service curve for lower priority flows, we must ensure that all the higher priority flows has been served, in our algorithm, the label 'Calculated' is used for this purpose; (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 aspects in mind, we propose the performance evaluation algorithm, as shown in Algorithm. **??**.

Here are some explanation about our algorithm: suppose the entire NoC is represented as a directional topology graph $G : V \times E$, where V, E represent the router and link respectively. For each link $e_{i,j} \in E$ represent there is a physical channel between router R_i and router R_j, because the graph is a directional graph, $e_{i,j} \neq e_{j,i}$. The set of all the flows in the network is denoted as F, and the path set of a

flow f_i traversed is denoted as Γ_i. If there exists interference between flow f_i and f_j, $\Gamma_i \wedge \Gamma_j = \Phi$. Further, if $e_{k,l} \in \Gamma_i \wedge \Gamma_j$, it means flow f_i and f_j content with each other at router R_k. All the flows f_i has a fixed-priority P_i. Denote the routers that flow f_i passes as a set $\overline{\mathcal{R}_{f_i}}$, the arrival curve of f_i at router R_j is $< \alpha^u_{i,j}, \alpha^l_{i,j} >$, and router R_j provides a service curve $< \beta^l_{R_j,f_i}, \beta^u_{R_j,f_i} >$ for flow f_i. Suppose the buffer size at each router R_i is B_i. For all the router R_j along the routine of flow f_i, denote by the contention flows with the same priority as f_i as Θ_{R_j,f_i}, and the set with lower priority flow as Ω_{R_j,f_i}.

Algorithm 1 Calculating the End-to-End Latency

for All flows $f_i \in F$ **do**
  for All router $R_j \in \overline{\mathcal{R}_{f_i}}$ **do**
    Let $\beta^u_{i,j} = \beta^l_{i,j} = \delta_0(t)$.
  end for
end for
for For each flow $f_i \in F$, with priority order **do**
  Collapse all the possible sub-path of f_i.
  for each router $R_j \in \overline{\mathcal{R}_{f_i}}$ **do**
    if $\Theta_{R_j,f_i} \neq \emptyset$ **then**
      for each flow $f_k \in \Theta_{R_j,f_i}$ **do**
        Compute $< \beta^{l'}_{SA,R_i,f_k}, \beta^{u'}_{SA,R_i,f_k} >$;
        Compute $< \beta^l_{R_i,f_k}, \beta^u_{R_i,f_k} >$;
      end for
    end if
    if $\beta^u_{i,j} = \beta^l_{i,j} = \delta_0(t)$ **then**
      Compute $\beta_{i,k}$ with Eq.(**??**) and Eq.(**??**).
    end if
  end for
  Compute $< \beta^l_\sigma, \beta^u_\sigma >$ with Eq.(**??**).
  Compute the service curve $< \beta^l, \beta^u >$ for flow f_i.
  Compute the latency of flow f_i according to Eq.(**??**).
  Compute the backlog of flow f_i according to Eq.(**??**).
  Label the flow f_i as 'Calculated'.
  for For all the router R_j in $\overline{\mathcal{R}_{f_i}}$ **do**
    if $\Omega_{R_j,f_i} \neq \emptyset$ and all the flows in Θ_{R_j,f_i} have been labeled as 'Calculated' **then**
      Compute the arrival curve $< \alpha^l_{i,j}, \alpha^u_{i,j} >$ of flow f_i at router R_j according to Eq.(**??**) and Eq.(**??**).
      Compute the arrival curve $< \alpha^l_{k,j}, \alpha^u_{k,j} >$ at router R_j of all the flows in $\Theta R_j, f_i$ according to Eq.(**??**) and Eq.(**??**), where $f_k \in \Theta R_j, f_i$.
      Compute the aggregative arrival curve at R_j of flow f_i and flows in $\Theta f_i, j$.
      Compute the leftover service curve $< \beta^{l'}_{i,j}, \beta^{u'}_{i,j} >$ at router R_j after serving flow f_i and all flows in Θ_{R_j,f_i} according to Eq.(**??**) and Eq.(**??**).
    end if
  end for
end for

In algorithm **??**, the leftover service curve for lower priority flows should compute after all the service curve of higher priority flows have been calculated, we add the label 'Calculated' to distinguish this scenario. The overall algorithm has two embedded loops, and the computation complexity for this algorithm is $O(np)$, where n and p is the number of flows and the hop count of each flow. This algorithm is of pseudo-polynomial complexity due to the computation complexity of algorithmic min-plus convolution and sub-additive closure [**?**]. This algorithm can be easily integrated into the network calculus toolbox, such as disco[**?**] and RTC toolbox[**?**] to compute the end-to-end latency automatically.

F. Buffer Optimization

The priority-aware NoC requires the same amount of VC as the priorities, to reduce the design cost, a priority sharing techniques is proposed in [**?**]. We observed that, for the priority-aware NoC, the lower priorities, the higher locking ratios, thus leading to larger buffer size. In this section, we propose a buffer optimization algorithm in conjunction with the priority sharing techniques to reduce the cost of priority aware NoC. Our method can be applied to the application-specific NoC design. We assume the application has been mapped to the NoC, and different flows have been assigned to their corresponding priorities and deadline. Our aim is the reduce the buffer size under the constraint of deadline not be violated.

The path of flow f_i is a sequence of router, denote by the start point $start_i$, end point end_i, and for any router $curnode$ along the path, Denote by $Pre(curnode)$, and $Pre(start_i) = null$. Initially, let the buffer size of R_j reserved for f_i be $B_{j,i} = \inf\{B \geq 0 | \beta_\tau \otimes \beta_{R_i} \geq \beta_{R_i}\}$, which is the smallest buffer size which disable the flow control. After assigned the initial buffer size for each flow along the path, we optimize the buffer size by the follow way: for each flow from high priority to low priority, e.g. flow f_i, reduce the buffer size gradually from end_i to $start_i$, for each iteration, verify that whether the deadline is satisfied. Move $curnode$ to its predecessor $Pre(curnode)$ if the former reduction violates the deadline constraint. Iteration for flow f_i ends when $pre(curnode) = null$. The detailed algorithm is shown in Algorithm **??**:

Algorithm 2 Buffer Optimization Algorithm

for All flows $f_i \in F$ **do**
  Compute β_{R_j,f_i} and β_{τ_j,f_i};
  Let $B_j = \inf\{B \geq 0 | \beta_\tau \otimes \beta_{R_i} \geq \beta_{R_i}\}$;
end for
for All flows $f_i \in F$ **do**
  curnode=end_i;
  while $Pre(curnode) \neq null$ **do**
    while f_i meets its deadline **do**
      Reduce $B_{curnode,i}$ by 1;
      Recalculate the end-to-end deadline of f_i;
    end whilecurnode=Pre(curnode);
  end while
end for

The entire algorithm optimize the buffer size for each flow from high priority to low priority gradually, the entire procedure is also try to reduce the service capacity for high priority, which left more service to the flow priority flows.

The entire computation complexity is $O(np)$, where n and p are the number of flows and the number of router along the path. This complexity is pseudo-polynomial due to the end-to-end latency calculation. We can also halt the optimization when the buffer budget are met to reduce the computation procedure. Flow-Level Buffer Analysis and Link-Level Buffer Analysis are proposed to optimize the buffer size for priority aware NoC in [**?**], but all these two method assume the packet injection period is less than the end-to-end latency, which limit the feasibility of these methods. In addition, a network calculus based latency model was proposed and energy optimization based on this method was carried out with Dynamic Voltage and Frequency Scaling (DVFS) in [**?**]. Our buffer optimization algorithm can also be used to minimize the buffer consumption and chip area, since the buffer usually consumes about 30% power of entire chip. The difference between [**?**] and our method is that: for the fixed configuration and deadline, they minimize the energy consumption by adjusting the voltage and frequency, and our method try to optimize the buffer size to meet the deadline constraint.

V. Experiments

A. Simulation Setup

In this section, we will utilize the Heterogenous-Networks-on-Chip (HNoCs) [**?**], a modular open source NoC simulator, to verify our analytical model. To achieve this goal, we modify the switch allocator to support fixed-priority scheduling, and collect the maximal end-to-end packet delay at the destination IP core. We configure the clock cycle to be $2ns$, and the topology is 4×4 mesh. Suppose all the flows $f_i, (i = 1, 2, 3, 4)$ in the network are periodical sources, and the period is P_i. Denote by the packet length of flow f_i is L_i (in flits), which means that, packets from the same flow has the same length, but packets of different flows can be of different lengths. We can obtain the arrival curve according to the method introduced in subsection **??**.

B. Comparison with Other Methods and Simulation

There are lots of latency analysis models for wormhole-switched real-time communication, e.g. contention tree model [**?**], FLA model [**?**], LLA model [**?**], network calculus model [**?**], there are also lots of backlog analysis models, e.g. scheduling analysis model [**?**] and LLA model [**?**], in addition, the network calculus model proposed in [**?**] can also be used to analyze the backlog bound. In this section, we only compare our model with LLA and network calculus, as they outperform all the other existing approach.

1) Comparison with Link Level Analysis: The original LLA approach assumes each flit in a packet traverse the router with a fixed latency of 1 cycle, and can only be applicable the scenario that the deadline of each packet is less than its injection period, which limits its application. To compare with our method, we should specialize the standard 5 stages wormhole router to a single cycle router, this is achieved by letting the latency of BW, RC, VA and ST stage be zero. Thus, the service curve of entire router is the same as the service curve provided by the SA stage, which is $< \beta^l_{SA,R_i}, \beta^u_{SA,R_i} >$. We also have to suppose the VC buffers are large enough, so that we can apply LLA for the latency analysis. For this scenario, let the packet length be 1 flit, we calculate end-to-end latency for each packet with our method and LLA under different injection period. There are two flows sharing the same priority, i.e. f_1 and f_4. While analyzing the latency of f_1, we can treat f_4 as a flow with higher priority than f_1, and vice versus. the calculation result is shown in Table **??**. The author can verify these results by hand. We also need to mention that, the RTC result is obtained by considering the longest collapsible sub-path (i.e. R_{13} and R_9). If this is not take into consideration, then, the analytical result for f_1, f_2, f_3 and f_4 are $5, 6, 7, 5$, respectively.

TABLE I: Comparison with Link Level Analysis

Injection Rate	Link Level Analysis				Real Time Calculus			
	f1	f2	f3	f4	f1	f2	f3	f4
2 cycle	∞				5	8	9	5
3 cycle	∞				5	7	8	5
4 cycle	∞				5	6	6	5
5 cycle	∞				5	6	6	5
6 cycle	5	6	6	5	5	6	6	5
7 cycle	5	6	6	5	5	6	6	5

2) Comparison with Network Calculus and Simulation: In this Subsection, we present the numerical results and simulation results to demonstrate the tightness and improvement of our method. In [**?**], the authors derived the service curve for each contention flow at the same input router. But, these service curves can only be used to derive the latency of injection router, for the service curve of each flow at subsequent router, the service curve should be computed with the same procedure as [**?**]. We set the flit injection rate as 0.1 flit/cycle, buffer depth as $B = 4$ flits, and change the packet length from 1 flit to 16 flits. The simulation results and numerical results is plotted in Fig. **??**. By comparison, we find that, for flow f_1 and f_2, the network calculus method and our method get the same latency result. But, for flow f_3, we find that, RTC can derive a much tighter latency bound compared with network calculus, as shown in Fig. **??**. and the difference becomes larger when we increase the packet length. One important reason is that, the simulation might not cover the corner case during the simulation process.

Next, we explain the reason why our method outperform the network calculus based method proposed in [**?**]. As implied by Theorem 1.72 in [**?**], if we can take the maximal service curve into consideration, we can get a much tighter output arrival curve, then, the leftover service curve of the next router will be much tighter than the one calculated with [**?**]. To demonstrate this, let $B = 4$, $L = 4$ flits and $T = 2ns$, we obtained the calculated service curve for flow f_2 both with network calculus and RTC, as shown in Fig. **??**. The calculation was carried out with RTC toolbox. From Fig. **??**, we find that, the calculated service curve with RTC is much 'better' than network calculus, which explains why the RTC can produce better results.

3) Buffer Optimization Comparison: As has shown in previous subsections, the network calculus based method [**?**] can handle some circumstances that FLA and LLA can not be applied. This motivate us to optimize the buffer size of NoC

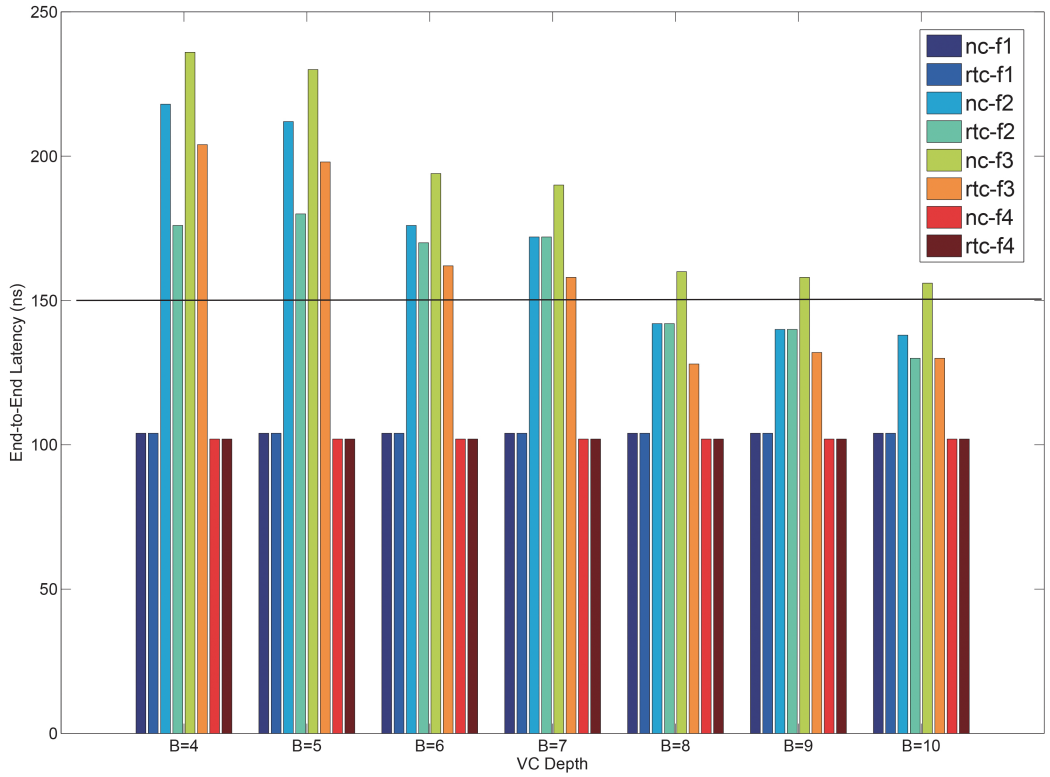


Fig. 9: end-to-end flows with different buffer size

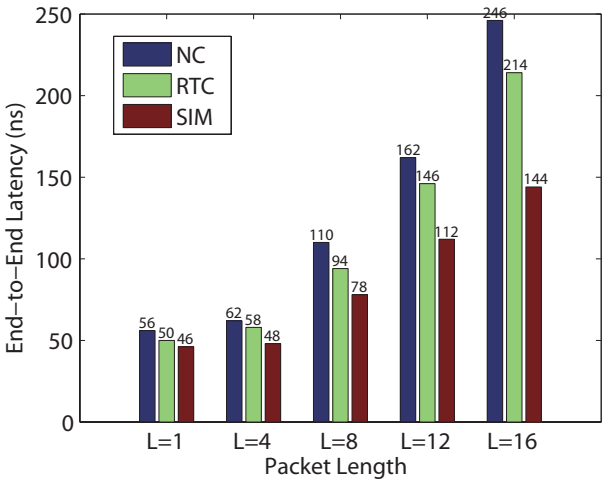


Fig. 7: Comparison with Network Calculus and Simulation

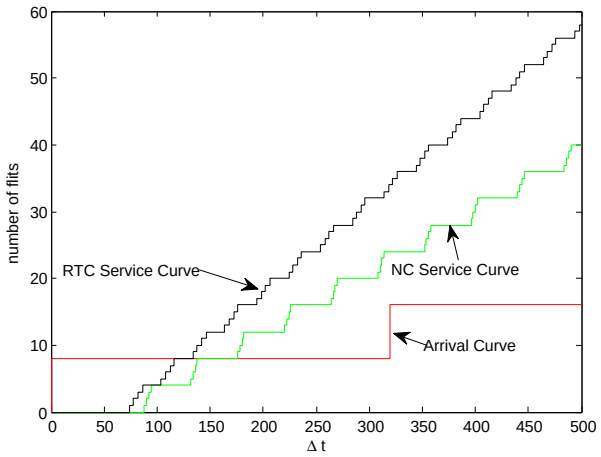


Fig. 8: Comparison of Service Curve Computed with Real-Time Calculus and Network Calculus

with our method. Let the flit injection rate be 0.1 flit/cycle, packet length $L = 16$ flits and change the VC buffer depth from 4 flits to 10 flits, we compare the theoretical bound computed with network calculus and our method, as shown in Fig. **??**. It clear that, our method outperform the network calculus based method, because it gives much tight bound for both f_2 and f_3. Further, if we set the deadline of f_2 and f_3 to 150ns, with our method, we can found that $B = 8$ flits is sufficient enough to guarantee the delay bound. While the network calculus based estimation is larger than 10 flits. We also observed from Fig. **??** that, the high priority flows (i.e. f_1 and f_4) are very insensitive to the buffer depths, which motivate us to allocate small buffers for these flows to reduce the area and power cost. Link-level analysis was extended in [**?**] to finite buffer size scenario. But, we need to mention that, the LLA approach can not be applied to optimize the buffer size, this is because the network latency is larger than the injection period for this configuration.

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. In this paper, we derived the upper service curve of flow controller with a novel method. Our contribution in this paper is two folded: First, we proposed a RTC based performance model for this kind of NoC, which outperforms the conventional scheduling theory and network calculus based methods. We also propose the concept of longest collapsible sub-path to improve the calculation accuracy and reduce the computation complexity; Second, we also propose an RTC based buffer optimization algorithm to reduce the buffer size under the constraint of deadline imposed to each flows. Our results can be applied to the mapping, routing and power optimization of NoC.

ACKNOWLEDGEMENT

The authors would thank the reviewers for their suggestions and comments, and all the experiments are carried out at VLSI Design Lab of McGill University. This research is supported by High Technology Research and Development Program of China (Grant No. 2012AA012201, 2012AA011902).