Composable Worst-Case Delay Bound Analysis Using Network Calculus

Yanchen Long, Zhonghai Lu, *Member, IEEE,* and Haibin Shen

***Abstract*—Performance analysis is playing an indispensable role in design and evaluation for on-chip networks. In former studies, the end-to-end delay bound is calculated by the Equivalent Service Curve method based on network calculus when resource sharing happens. However, in this paper, we propose a composable method to get the bound. This method uses the aggregated Local Arrival Curve to get the local delay bound first, then calculates the end-to-end bound by summing up local bounds. This method solves the scalability problem and largely decreases the computation complexity compared with the former method.**

***Index Terms*—Delay bound, network calculus, composable method, local arrival curve (LAC).**

I. Introduction

As the number of IP cores and devices increases rapidly following Moore's Law, the on-chip interconnection plays a critical role in manycore systems. Quality-of-Service (QoS) in on-chip interconnection has been a major concern since it is indispensable in providing a predictable system. Real-time applications running on SoCs, such as audio/video delivery, voice conversation and HDTV, require firm or soft performance guarantees. Typically, there are two ways to ensure performance: simulation approach and mathematical approach. Simulation is widely used by designers, but extremely time-consuming, and might not be able to cover all corner cases. Thus, it is necessary to compute a formal analytical delay bound, giving the designers more insights into what factors affect the worst case.

In previous works, one of the analytical approaches for delay guarantee is given by Network Calculus (NC) [1] [3], which gives an upper bound for the worst-case delay. The difficulty in getting a tight bound is due to a variety of resource-sharing scenarios. Competitions over links and buffers add to the unpredictability of end-to-end delays. The NC theory deals resource-sharing models with the Equivalent Service Curve (ESC) method. For tag flow, first the equivalent service it can get from each node is derived, then all the local equivalent services along its traversal path are convoluted up [3]. The ESC method shows great advantage in dividing a complicated system into one-flow-one-server sub-systems in view of each flow. However, the drawbacks cannot be ignored. Getting local ESC needs heavy calculation. Also the ESC method is not scalable. Thus, a new approach for computing the end-to-end delay bound in resource-sharing networks is needed.

In this paper, we propose a composable method in NC for resource-sharing networks, which is named as aggregated Local Arrival Curve (LAC) method. The LAC method is based on the very basic definition of arrival curve, as well as the relationship between local delay bounds and end-to-end delay bound in [3]. The new method has three main contributions. First, it carries out much less computation than the ESC method does. Second, it is superior to the ESC method in terms of scalability and composability, in that the results for each node are re-usable as long as the input and output remain the same. Finally, the results by the LAC method can be also tight.

Manuscript received November 2, 2016; revised March 15, 2017 and May 30, 2017; accepted Jun 09, 2017. (*Corresponding author: Zhonghai Lu*)

Y. Long and H. Shen are with the VLSI Institute, Zhejiang University, Hangzhou, 310027 China (e-mail: yanchen@kth.se).

Z. Lu is with the School of Information and Communication Technology, KTH Royal Institute of Technology (e-mail: zhonghai@kth.se).

The rest of this paper is organized as follows. In Section II we review the related works and explain our motivation. Section III describes and discusses the problems of previous ESC method by two illustrative examples. Section IV details the proposed LAC method, revisits the case studies again using the new method, and compares it with the ESC method. Section V discusses the experimental results. Finally, we conclude in Section VI.

II. Related Work

NC is a theory for performance guarantee analysis, based on highly abstract modeling of traffic's Arrival Curve (AC) and server's Service Curve (SC). NC was originally proposed for communication networks. Thanks to the works by numerous researchers, now it is also widely used in embedded systems, SoCs and on-chip interconnections.

The AC model which is characterized as (b, r) standing for the injected traffic's burstiness and rate was defined in [1]. Cruz first put forward a calculus method to obtain the delay bounds. The definition of SC was later proposed in [2], which described the service by its lower bound. In [3], Boudec and Thiran systematically summarized the previous works, in particular, in relation to min-plus and max-plus algebra. In Chang's [4], he related NC to a filtering theory which further extended the theory.

Modeling resource-sharing scenarios has long been a difficult issue in NC. Chang [4] deducted the delay bound for aggregating scheduling. In [3], specific bounds for different multiplexing nodes were given. In [5], Qian et al. applied the theory to on-chip networks. They proposed three basic interference models and stated that all resource-sharing cases can be grouped into the three solo models or their combinations. They showed the ESC analysis technique for each model, and calculated the per-flow delay bound by ESC. This method shows its advantage especially when considering specific arbitration policies [7]. In [8], tightness study and evaluation of ESC method are discussed. Though ESC results can sometimes reach 100% tightness, there are many cases that ESC gives very pessimistic results.

Real-time calculus extends NC for realtime systems [10]. Uniformed framework and modular performance analysis were proposed in [11] and [12], which decomposed the system into individual nodes and analyzed each node separately. Still, when calculating the end-to-end bound, the Input Arrival Curve (IAC), Output Arrival Curve (OAC), or the concatenation of services for each event stream, must be calculated. Other real-time analysis metrics, like worst-contention delay [19], Real-Time Bound for High-Bandwidth/Low-Latency traffic (RTB-HB/RTB-LL) [20], were applied to NoCs as well.

Another topic for performance analysis for NoCs is about the average performance estimation. In [13] the authors proposed a router model under Poisson traffic arrival assumption, and derived a closed-form formula for average latency per flow. [17] gave a Support Vector Regression-based learning approach for NoC's average latency prediction. Bogdan et al. proposed a new traffic model based on mean field approach, which described the traffic on multicore systems by a series of interwoven time scales defined by several exponents instead of the traditional single exponent expressions in [14]. In [16] the authors modeled the traffic for adaptive wormhole-switched

hypercube and torus networks by the Compound Poisson Process to capture the properties of the bursty and batch arrival traffic. A new non-stationary fractal queuing model was raised in [15] for both the core workloads and on-chip traffic.

III. ESC Method for Delay Bound Analysis

A. An Illustrative Example

In a typical 4x4 mesh on-chip network in Fig.1, we assume that the on-chip routers deploy virtual channel flow control policy and XY routing. All buffers are FIFOs. We consider all-to-one traffic pattern where all flows originate from their own source nodes to the same sink node N_{14}. Take the flow with longest path, i.e. flow from Node N_0 to N_{14}, as tag flow f_0. All routers are work-conserving service nodes using Weighted Round Robin (WRR) arbitration. Other notations and definitions are explained in Table I. Flow i confroms to a linear AC γ_{r_i,b_i}. All nodes provide a latency-rate service $\beta_{R,T} = \beta_{1,0}$ [18] for each output channel.

We first consider the case that only f_0, f_1 and f_2 are in the network. Then we extend it to a 4-flow interference case with f_3 joining the network at node N_4. As we can see in the zoom-in picture in Fig.1, when 2 flows meet each other the first time (e.g. f_0 and f_1 meet at N_1, f_{01} and f_2 meet at N_2), they only compete for link bandwidth due to the virtual channel assumption. Once they merge together, they share both the input buffer and link bandwidth for all down stream nodes (e.g. f_0 and f_1 share one input buffer and the link bandwidth of $N_{2,6,10,14}$).

TABLE I: Notations and Definitions

Notation	Definition
$f_i/f_{i...j}$	ith flow/ aggregate of ith and jth flow
$\alpha_i/\alpha_{i...j}$	IAC of ith flow/ aggregate of ith and jth flow
$\alpha_i^*/\alpha_{i...j}^*$	OAC of ith flow/ aggregate of ith and jth flow
$\gamma_{r,b}$	AC with burstiness b and rate r. $\gamma_{r,b} = rt + b$, t denotes time
β^{N_j}	SC at Node N_j
$\hat{\beta}_i^{N_j}$	ESC of flow f_i at Node N_j
$\beta_{R,T}$	Latency-rate service curve with rate R and latency T $\beta_{R,T} = R[t-T]^+ = \begin{cases} R(t-T), t>0 \\ 0, otherwise \end{cases}$, t denotes time
$D_i^{N_j}$	Local delay bound of flow f_i at Node N_j
$\overline{D}_i$	End-to-end delay bound of flow f_i
$h(\alpha,\beta)$	Horizontal distance of curve α and β
$\phi_i/\phi_{i...j}$	Service allocation in time to flow f_i/ aggregated flow f_i and f_j
$w_i/w_{i...j}$	Service weight of flow f_i/ aggregated flow f_i and f_j $w_{i...j} = \frac{\sum_{m=i}^{m=j}\phi_m}{\sum_{\forall f_i at N_j}\phi_i}$
$\otimes$	Min-plus convolution. $f \otimes g(t) = \inf_{0\le s\le t}\{f(t-s)+g(s)\}$
$\oslash$	Min-plus deconvolution. $f \oslash g(t) = \sup_{u\ge 0}\{f(u+t)-g(u)\}$

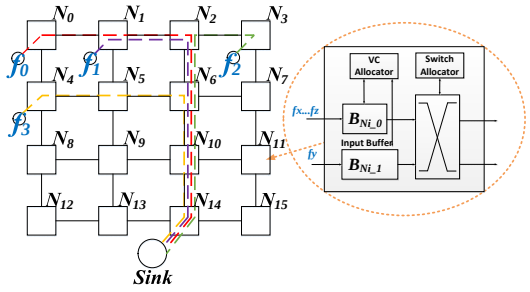

Fig. 1: A 4x4 mesh with interference flows

B. Equivalent Service Curve (ESC) Method

1) General Analysis Steps

a) Build up the NC contention model from the micro-architecture. Extract the AC for each injection flow and SC for each network node.

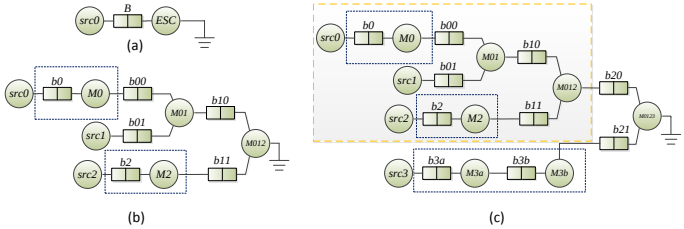

Fig. 2: NC analysis graphs

b) Calculate the ESC for tag flow f_0 at each node, denoted as $\hat{\beta}_0^{N_0}$, $\hat{\beta}_0^{N_1}$, ..., $\hat{\beta}_0^{N_j}$.

c) Calculate the end-to-end ESC for tag flow using concatenation theory in [3], $\hat{\beta}_0 = \hat{\beta}_0^{N_1} \otimes \hat{\beta}_0^{N_2} \otimes ... \otimes \hat{\beta}_0^{N_j}$.

d) Calculate the end-to-end delay bound, $\overline{D}_0 = h(\alpha_0, \hat{\beta}_0)$.

2) Case Study I: Applying the ESC analysis steps above, we solve Case I in which only f_1 and f_2 are interference flows, as depicted in Fig.1. To obtain an ESC for f_0 at each node, we could apply the left-over service property [6], e.g. at N_1, $\hat{\beta}_0^{N_1} = \beta^{N_1} - \alpha_1 = \beta_{1-r_1, \frac{b_1}{1-r_1}}$, $\hat{\beta}_1^{N_1} = \beta^{N_1} - \alpha_0 = \beta_{1-r_0, \frac{b_0}{1-r_0}}$. On the other hand, due to the property of WRR, another method based on link bandwidth partition is applied in [7], called isolation property. It describes the minimum guaranteed service one flow can get ($w_i R^{N_j}$) and the maximum delay it can meet ($\sum \phi_{m\neq i}$). So, at N_1, $\hat{\beta}_0^{N_1} = \beta_{w_0,\phi_1}^{N_1}$, $\hat{\beta}_1^{N_1} = \beta_{w_1,\phi_0}^{N_1}$. Both methods can be used for calculating ESCs. We will use isolation property as an example in all following cases.

a) Get an NC interference model shown in Fig.2(b).

b) At N_1, we get ESC for f_0: $\hat{\beta}_0^{N_1} = \beta_{w_0,\phi_1}$.

c) At N_2, because $\beta^{N_2} = \beta^{N_6} = \beta^{N_{10}} = \beta^{N_{14}} = \beta_{1,0}$, we first concatenate N_2, N_6, N_{10} and N_{14} together by concatenation theorem in [3]: $\beta^{N_2} \otimes \beta^{N_6} \otimes \beta^{N_{10}} \otimes \beta^{N_{14}} = \beta_{1,0}$. The ESC for aggregated flow f_{01}: $\hat{\beta}_{01}^{N_2} = \beta_{w_{01},\phi_2}$.

To get $\hat{\beta}_0^{N_2}$ from it, there is no other way but taking away the IAC of f_1 at N_2 ($\alpha_1^{N_2}$), i.e. the OAC of f_1 at N_1 ($\alpha_1^{*N_1}$). Thus, we have to go back to the upstream node N_1 and get ESC of f_1: $\hat{\beta}_1^{N_1} = \beta_{w_1,\phi_0}$. By the deconvolution of IAC and ESC of f_1 at N_1: $\alpha_1^{N_2} = \alpha_1^{*N_1} = \alpha_1 \oslash \hat{\beta}_1^{N_1}$. The ESC for f_0 at N_2 is:

$$\hat{\beta}_0^{N_2} = \hat{\beta}_{01}^{N_2} - \alpha_1^{N_2} = \beta_{w_{01}-r_1, \frac{b_1 + r_1\phi_0 + w_{01}\phi_2}{w_{01}-r_1}}$$

d) The end-to-end ESC for f_0 is:

$$\hat{\beta}_0 = \hat{\beta}_0^{N_1} \otimes \hat{\beta}_0^{N_2} = \beta_{w_0,\phi_1} \otimes \beta_{w_{01}-r_1, \frac{b_1 + r_1\phi_0 + w_{01}\phi_2}{w_{01}-r_1}}$$

e) The end-to-end delay bound for f_0 is:

$$\overline{D}_0 = h(\alpha_0, \hat{\beta}_0) = \max(\phi_1, \frac{b_1 + r_1\phi_0 + w_{01}\phi_2}{w_{01} - r_1}) + \frac{b_0}{\min(w_0, w_{01} - r_1)}$$

3) Case Study II: Now we extend the interference model to a 4-flow-3-node pattern, with newly joined interference flow f_3 in Fig.1. The NC analysis graph is given in Fig.2(c). Going through the analysis steps, we get the end-to-end delay bound for f_0:

$$\overline{D}_0' = \max(\phi_1, \frac{b_1 + r_1\phi_0 + w_{01}\phi_2}{w_{01} - r_1}, \frac{b_1 + b_2 + r_1\phi_0 + r_2\phi_{01} + w_{012}\phi_3 + r_1\frac{b_0 + r_0\phi_1 + w_{01}\phi_2}{w_{01}-r_0}}{w_{012} - r_1 - r_2}) + \frac{b_0}{\min(w_0, w_{01} - r_1)}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2017.2729283, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

C. Discussions and Conundrum of ESC Method

The advantage of ESC method is obvious. No matter how complicated the system is, the ESC method can finally turn it into an equivalent system for each flow going through one equivalent end-to-end service, as shown in Fig.2(a).

However, according to Case II, even a 4-flow merging scenario can exhibit such a tedious calculation procedure and complicated result by the ESC method. From Sec.IV-C we can see that, the complexity of computation and the final closed-form formula increases sharply. This is because when flows merge together and move to the next hop, they share not only the link bandwidth, but also the input buffer. In previous works of NC analysis, the only way to break buffer sharing multiplexing is applying the left-over service property in [3]. That requires every interference flow's ESCs and OACs at all traversing nodes, which cause heavy iterative convolution/deconvolution operations and may result in loose upper bounds as well. Another problem of ESC method is composability issue. Further discussions on ESC's composability, tightness and computation cost, will be given in Sec.IV-C.

IV. A Composable Analysis Method

The proposed Local Arrival Curve (LAC) analysis does not integrate the whole system into a one-arrival-one-service model. Instead, it breaks the whole contention model at each node where interference happens, gets the local delay bound stage by stage, and finally sums them up.

A. General Analysis Steps

a) Build up the NC contention model from the micro-architecture. Extract AC for each injection flow and SC for each network node.

b) Calculate the local delay bound for the aggregate flow which contains tag flow f_0 at each node N_j, $D_0^{N_j} = h(\alpha_{01\ldots m}^{N_j}, \hat{\beta}_{01\ldots m}^{N_j})$.

In our method, we use a simplified method to calculate the IAC for f_0 at each node. Once two flows aggregate, for their OACs, the inequation $\alpha_0^* + \alpha_1^* \geq \alpha_0^*$ always holds true. So, by the definition of arrival curve in [3], $\alpha_0^* + \alpha_1^*$ is also an arrival curve for f_0. Therefore, at each node, after a new flow joins, we update the OAC of f_0 as $\alpha_{01\ldots m}^* = \sum \alpha_m^*$

Accordingly, the SC for this aggregate flow is $\hat{\beta}_{01\ldots m}^{N_i}$.

c) Calculate the end-to-end delay bound, $\overline{D}_0 = \sum_i D_0^{N_i}$

B. Case Study by LAC

1) Case Study I:

a) The same NC model as ESC in Fig.2(b).

b) At N_1, by isolation property, we get the SC for f_0 and f_1 respectively: $\hat{\beta}_0^{N_1} = \beta_{w_0,\phi_1}$, $\hat{\beta}_1^{N_1} = \beta_{w_1,\phi_0}$. Therefore, the local delay bound at N_1 is $D_0^{N_1} = h(\alpha_0, \hat{\beta}_0^{N_1}) = h(\gamma_{r_0,b_0}, \beta_{w_0,\phi_1})$. Now we update the OAC of tag flow as:

$$\begin{aligned}\alpha_{01}^{*N_1} =& \alpha_0^{*N_1} + \alpha_1^{*N_1}\\ =& \gamma_{r_0,b_0} \oslash \beta_{w_0,\phi_1} + \gamma_{r_1,b_1} \oslash \beta_{w_1,\phi_0}\\ =& \gamma_{(r_0+r_1),(b_0+b_1+r_0\phi_1+r_1\phi_0)}\end{aligned}$$

c) At N_2, the SCs for f_{01} and f_2 are $\hat{\beta}_{01}^{N_2} = \beta_{w_{01},\phi_2}$ and $\hat{\beta}_2^{N_2} = \beta_{w_2,\phi_{01}}$. Since the updated IAC of the tag flow at N_2 equals its updated OAC at upstream node N_1, the local delay bound at N_2 is:

$$D_{01}^{N_2} = h(\alpha_{01}^{N_2}, \hat{\beta}_{01}^{N_2}) = h(\gamma_{(r_0+r_1),(b_0+b_1+r_0\phi_1+r_1\phi_0)}, \beta_{w_{01},\phi_2})$$

d) The end-to-end delay bound for f_0 is:

$$\overline{D}_0 = D_0^{N_1} + D_{01}^{N_2} = \phi_1 + \phi_2 + \frac{b_0}{w_0} + \frac{b_0+b_1+r_0\phi_1+r_1\phi_0}{w_{01}}$$

2) Case Study II: Comparing Fig.2 (b) and (c), it is shown that the NC analysis graph before N_6 in Fig.2(c) is exactly the same as that in Fig.2(b). Therefore, by the composability property of LAC method, we can re-use the result in Case I directly and calculate only $D_{012}^{N_6}$: $\overline{D}_0' = \overline{D}_0 + D_{012}^{N_6}$. Since we have:

$$D_{012}^{N_6} = h(\alpha_{012}^{N_6}, \hat{\beta}_{012}^{N_6}) = h((\alpha_{01}^{N_2} \oslash \hat{\beta}_{01}^{N_2} + \alpha_2 \oslash \hat{\beta}_2^{N_2}), \beta_{w_{012},\phi_3})$$

We got the end-to-end delay bound:

$$\begin{aligned}\overline{D}_0' =& \overline{D}_0 + D_{012}^{N_6}\\ =& \phi_1 + \phi_2 + \phi_3 + \frac{b_0}{w_0} + \frac{b_0+b_1+r_0\phi_1+r_1\phi_0}{w_{01}}\\ &+ \frac{b_0+b_1+b_2+r_0(\phi_1+\phi_2)+r_1(\phi_0+\phi_2)+r_2\phi_{01}}{w_{012}}\end{aligned}$$

C. Comparisons of ESC and LAC

We call the LAC method an arrival-curve-centric approach since the key idea is to treat the newly aggregated flow enrolling the tag flow as the updated tag flow, while the ESC is a service-curve-centric approach. By comparing the closed-form formula given above, we also compare these two methods in composability, tightness and computation complexity.

For composability, imagine another flow f_4 traversing the network at $\{N_2, N_6, N_{10}, N_{14}\}$ in sequence in Fig.1. By the ESC method, ESCs for all other flows after N_2 cannot be re-used, because the ESC for f_{01} and f_2 change, so the OAC and IAC for each flow also change in all downstream nodes, e.g. $\alpha_2^{*N_2} = \alpha_2 \oslash \beta_2^{N_2}$, where $\beta_2^{N_2}$ changes when f_4 joins. However, by the LAC method, after f_4 joins, the local service for the aggregated flow at N_1 (f_{01}) and N_6 (f_{0124}) can be reused. This does not add any computation cost to calculate the local delay bound for each node.

As a measurement of the quality of each method, comparison of their tightness (defined in Sec.V-B) is also necessary. Both methods can introduce pessimistic prediction on the delay bounds. The LAC method takes aggregate flow's delay bound as the tag flow's bound. As proved in [3], $\overline{D} < \sum_j D^{N_j}$, the compositional bound is also an end-to-end delay bound for the tag flow, but results in looser bound.

Why the delay bound by ESC method can be loose is more implicit. Suppose that Case I deploys a general blind arbitration policy, $\hat{\beta}_0 = \hat{\beta}_0^{N_1} \otimes \hat{\beta}_0^{N_2} = (\beta^{N_1} - \alpha_1) \otimes ((\beta^{N_2} - \alpha_2) - \alpha_1 \oslash (\beta^{N_1} - \alpha_0))$. The ESC method gives an equivalent end-to-end SC to f_0. Looking at the items on both sides of $\otimes$, we suppose that $(\beta^{N_1} - \alpha_1)$ represents the exact service f_0 gets at N_1, then $(\beta^{N_1} - \alpha_0)$ is actually an under-estimated service for f_1, because when f_0 meets the worst-case service means the service is mostly allocated to f_1. This makes $\alpha_1 \oslash (\beta^{N_1} - \alpha_0)$ an over-estimation for the output arrival process of f_1. In turn, $((\beta^{N_2} - \alpha_2) - \alpha_1 \oslash (\beta^{N_1} - \alpha_0))$ becomes an under-estimated service for f_0, and finally makes the end-to-end ESC $\hat{\beta}_0$ under-estimated. So the end-to-end delay bound is over-estimated, which means a looser delay bound.

The influence of LAC and ESC methods to their tightness has no correlation to each other. While the tightness of LAC is affected by local worst cases of different merging nodes, the tightness of ESC method is influenced by competing flows merging at the same node. There is no general conclusion as to which method is superior to the other in terms of tightness. This can be observed in the experiments.

The main advantage of the proposed LAC method is lower computation complexity. Consider a larger network. Assume the tag flow traverses M multiplexing nodes along its path, and there are at most k flows contending with the tag flow at each node. Before Node N_j there are x flows already aggregated with f_0, and y flows already aggregated with f_0 before Node N_{j-1}.

For ESC method, the heaviest computation part is calculating $\hat{\beta}_0^{N_j}$. $\hat{\beta}_0^{N_j} = \hat{\beta}_{01...x}^{N_j} - \alpha_1^{N_{j-1}} \oslash \hat{\beta}_1^{N_{j-1}} - \alpha_2^{N_{j-1}} \oslash \hat{\beta}_2^{N_{j-1}} - ... - \alpha_x \oslash \hat{\beta}_x^{N_{j-1}}$, and for each $\hat{\beta}_m^{N_{j-1}} = \hat{\beta}_{01...x}^{N_{j-1}} - \sum_{i \neq m} \alpha_i^{N_{j-2}} \oslash \hat{\beta}_i^{N_{j-2}}$. So the calculation of $\hat{\beta}_0^{N_j}$ is an iterative one which needs $M-1$ rounds. Thus the complexity of ESC computation is $O(k^{M-1})$.

For LAC method, the heaviest part is calculating $\alpha_{01...x}^{N_j}$. $\alpha_{01...x}^{N_j} = \alpha_{01...x}^{*N_{j-1}} = \alpha_{01...y}^{N_{j-1}} \oslash \beta_{01...y}^{N_{j-1}} + \sum_{i=y+1}^{i=x} \alpha_i^{N_{j-1}} \oslash \hat{\beta}_i^{N_{j-1}}$. Note that $\alpha_{01...y}^{N_{j-1}}$ is already calculated at the last node, so it can be re-used directly without extra iterative calculation. All $\alpha_i^{N_{j-1}}$ and $\hat{\beta}_i^{N_{j-1}}$ are given, or can be derived within constant time. As deconvolution operation is carried out at most k times at each node, and in total there are M nodes. Thus, the complexity is $O(Mk)$. The linear complexity to k also means that the LAC method is scalable. It is faster than ESC method, and other methods like RTB-HB/RTB-LL in [20] with quadratic complexity.

We can also exhibit LAC method's lower complexity by an example of a 10×10 NoC with all-to-one traffic pattern, in which each node injects a flow travelling to the same destination, the bottom right node, using XY-routing. To get the end-to-end delay bound of the tag flow (the one injected in the upper left node), we need $(1+2+...+9) \times 10 + (10+20+...+90) = 900$ min-plus deconvolution operations applying the ESC method while merely $(1+2\times 8)\times 10 + (2+3\times 8) = 196$ operations are needed by our LAC method.

V. Experimental Results

A. Simulation Platform and Settings

In the experiments, we calculate and evaluate the end-to-end delay bound of a specific flow in the network with pre-determined settings such as traffic pattern, routing, etc. All experiments are based on NC graph given in Fig.2(b), which depicts the pre-determined network in view of flows and nodes. We build up a cycle-accurate simulation platform in SystemC. The (b,r) injection model [1] is with average injection rate $r \in (0,1]$ packet/cycle and burstiness $b \in [1,16]$ packet(s). According to [9], the most bursty traffic for a single flow is a periodic ON-OFF injection. However, as the number of flows grows, applying ON-OFF injection for all flows does not necessarily produce the heaviest congestion to the tag flow. Here we implement a revised ON-OFF injection, in which the peak injection rate is in $[0.33,1]$ and maximum packets sent during one ON period is in $[\frac{b-r}{1-r}, \frac{b-3r}{1-3r}]$. We use the revised ON-OFF injection for tag flow and Leaky Bucket Shaper injection (with injection rate r_i and token number b_i) for interference flows, to create most bursty flows.

All servers are work-conserving latency-rate servers, described as $\beta_{1,0}$. For WRR arbiters, weights $\phi_i \in [1,8]$. Besides, the constrain $\sum r_i < 1$ for all flows passing through each node N_j, must be met.

However, even if the NC theory gives each case an upper bound, whether the worst case can occur in a particular simulation is a reachability problem. The simulation time suffers from an exponential growth according to the different configurations of traffic and services' parameters. In [8] tightness evaluation is formulated as constrained optimization problems, with traffic and services' parameters generated as the candidate of next movement and tightness is transformed into energy function $E(s)$. The well-defined problem enables a heuristic searching method using Adaptive Simulated Annealing (ASA) algorithm to guide the input configuration process for the worst case. We introduce the ASA algorithm in our experiments to avoid exhausting simulations and accelerate the simulation process.

As defined in [5], we use tightness $\xi = \frac{D_{sim}}{D_{cal}}$ to measure how much the observed worst-case delay in simulation can reach its theoretical bound. Purpose of measuring ξ is multi-folded. Firstly, validate the correctness of LAC method. Secondly, measure the calculated bound's quality. Thirdly, find out under what simulation configuration can a worst case happen.

B. Results

In the whole exploration space for all traffic and services' configurations, we observe no tightness greater than 100%. This confirms that the delay bound obtained by LAC method is an upper bound for all cases. We can also observe it from Fig.3 to Fig.5, where all ξs are bounded by 100%. Thus, the correctness of LAC method is empirically validated.

Fig.3 shows the comparisons between LAC and ESC method. For all lines, tag flow is $f_0 = 6 + r_0 t$. We can see that the LAC method can deliver a tight theoretical bound which could be over 95% tight. Though, in this case, the LAC method does not give 100% tight bound as those of ESC method, it is not always less tight. The two methods outperform each other in different scenarios. As the blue lines illustrate, when the interference flows are relatively busy ($f_1 = 7+0.3t$, $f_2 = 15+0.4t$), LAC can give a 10% better result than the ESC method. When the interference flows are relatively idle ($f_1 = 7+0.14t$, $f_2 = 15+0.08t$), the red lines show that ESC delivers tighter calculation bounds. These validate the statements we made in Sec.IV-C.

Fig.3 also shows how tag flow's injection rate r_0 impacts tightness. When r_0 is getting closer to the rate of $\hat{\beta}_0^{N_i}$, in other words, when f_0 uses most of its allocated bandwidth, tightness may increase. Note that the trends for both LAC and ESC are almost the same. Because when a busy flow f_1 merging together with f_0 proceeds to the next hop, together with the newly joined flow f_2 (also busy), they make the shared buffer very congested, which gives f_0 more chances to meet the local bound predicted by both LAC and ESC.

We also found that different WRR weight allocation values can generate a variety of interference behaviors, then influence tightness. Thus, we use ϕ_i as evaluating variables in Fig.4 and Fig.5. From numerous experiments, we found that tightness is more sensitive to the allocation at the first aggregating node than at all the other nodes. So, in Fig.4 and Fig.5, the X-axis variable is set to $r_0 : w_0$, where the latter one gives the ratio between ϕ_0 and $\phi_0 + \phi_1$. However, the downstream nodes' allocation weights are also considered, represented in different colors in the figures. Fig.4 shows a busy tag flow $f_0 = 5+0.5t$ while Fig.5 has less busy one $f_0 = 5+0.05t$. Two interference flows $f_1 = 7+0.15t$ and $f_2 = 14+0.15t$ remain the same. In Fig.4 we have $r_0 : r_1 = 10:3$ and $(r_0+r_1) : r_2 = 13:1$, while in Fig.5 $r_0 : r_1 = 1:3$ and $(r_0+r_1) : r_2 = 4:3$.

The good results in Fig.4 appear mostly in the region $(r_0 : w_0) \in [0.5,1]$ when the tag flow is occupying half to whole bandwidth allocated to it. The long-tail phenomenon in both figures, together with the poor performance given in yellow and pink dots, shows that the unbalanced weight allocation would bring about lower tightness, in other words, much less congestion. This would be a helpful hint for designers when they intend to guarantee the service for certain applications.

With the blue dots and red triangles in Fig.4, and blue dots and green stars in Fig.5 performing better than the other lines, we can also conclude that weight allocation $(\phi_{01} : \phi_2)$ which is close to the injection rate's ratio $((r_0+r_1) : r_2)$, also contributes to higher tightness. Further, different weight allocations with the same ratio, such as 1:1 and 8:8, have different results. This reminds us that not only the weight allocation ratio, but also the absolute value impacts the results, because this can influence how much one flow can accumulate its burstiness at the input buffer of each arbitration point.

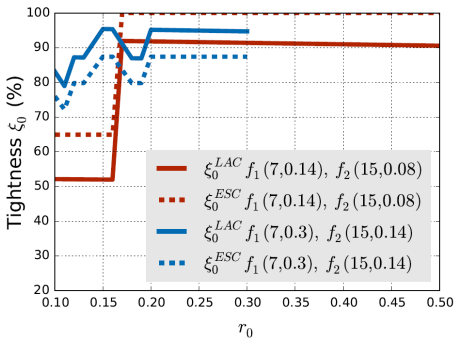

Fig. 3: LAC vs. ESC

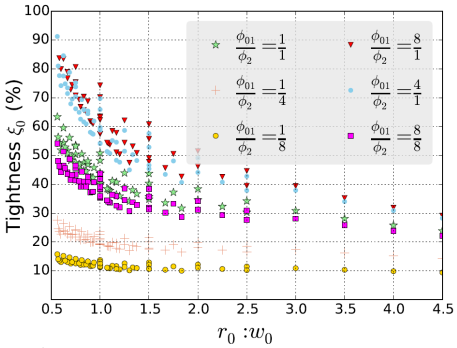

Fig. 4: Influence of different weights (1)

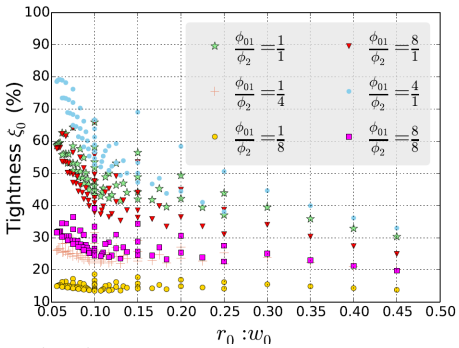

Fig. 5: Influence of different weights (2)

VI. Conclusion

The worst-case delay bound analysis for multi-flow-multi-node interference model has long been a difficult problem for researchers. In previous studies, the NC based analysis uses the ESC method to calculate the end-to-end delay bound for those models. However, this ESC method introduces heavy computation cost, and is neither composable nor scalable.

In this paper, we propose a composable analysis approach called LAC method. This method first partitions the system into smaller ones according to the multiplexing nodes' positions, calculates the local delay bound based on the aggregate arrival curve and service curve, then sums up the local results to finally get the end-to-end delay bound. By this method, the computation complexity largely decreases, and the results are reusable when the network scales up. By case studies and experiments, the LAC method is evaluated against the ESC method. Future works will mainly focus on improvement of tightness for results from the LAC method, which may be achieved by developing more accurate traffic and service models.

References

[1] R. L. Cruz, "A calculus for network delay, Part I: Network elements in isolation; Part II: Network analysis," *IEEE Trans. on Information Theory*, vol. 37, no. 1, pp. 114-131, 1991.

[2] A. Rajeev, R. L. Cruz, O. Clayton and R. Rajendran, "Performance bounds for flow control protocols," *IEEE/ACM Trans. on Networking*, vol. 7, no. 3, pp. 310-323, 1999.

[3] J. L. Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*, online ed. Springer-Verlag, 2004.

[4] C. S. Chang, "Stability, queue length, and delay of deterministic and stochastic queueing networks," *IEEE Trans. on Automatic Control*, vol. 39, no. 5, pp. 913-931, 1994.

[5] Y. Qian, Z. Lu and W. Dou, "Analysis of Worst-case Delay Bounds for Best-effort Communication in Wormhole Networks on Chip," in *Proc. of ACM/IEEE NOCS*, CA, USA, 2009, pp. 44-53.

[6] Y. Qian, Z. Lu and W. Dou, "Analysis of Worst-Case Delay Bounds for On-Chip Packet-Switching Networks," *IEEE Trans. on Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 5, pp. 802-815, 2010.

[7] Y. Qian, Z. Lu and W. Dou, "QoS Scheduling for NoCs: Strict Priority Queueing versus Weighted Round Robin," in *Proc. of ICCD*, Amsterdam, Netherlands, 2010, pp. 52-59.

[8] X. Zhao and Z. Lu, "Heuristics-Aided Tightness Evaluation of Analytical Bounds in Networks-on-Chip," in *IEEE Trans. on Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 6, pp. 986-999, 2015.

[9] Z. Lu, M. Millberg, A. Jantsch, A. Bruce, P. VanderWolf and T. Henriksson, "Flow regulation for on-chip communication," in *Proc. of DATE*, Dresden, Germany, 2009, pp. 578-581.

[10] L. Thiele, S. Chakraborty and M. Naedele, "Real-time calculus for scheduling hard real-time systems," in *Proc. of ISCAS*, Geneva, Switzerland, 2000, pp. 101-104.

[11] S. Chakraborty, S. Knzli and L.Thiele, "A general framework for analysing system properties in platform-based embedded system designs," in *Proc. of DATE*, Munich, Germany, 2003, pp. 190-195.

[12] 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*, vol. 8, no. 6, pp. 649-667, 2006.

[13] U. Y. Ogras, P. Bogdan and R. Marculescu, "An Analytical Approach for Network-on-Chip Performance Analysis," in *IEEE Trans. on Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 12, pp. 2001-2013, 2010.

[14] P. Bogdan and R. Marculescu, "Workload characterization and its impact on multicore platform design," in *Proc. of CODES/ISSS*, Scottsdale, AZ, USA, 2010, pp. 231-240.

[15] P. Bogdan, "Mathematical Modeling and Control of Multifractal Workloads for Data-Center-on-a-Chip Optimization," in *Proc. of ACM/IEEE NOCS*, Vancouver, BC, Canada, 2015, pp. 21:1-21:8.

[16] Y. Wu, G. Min, M. Ould-Khaoua, H. Yin, and L. Wang, "Analytical modelling of networks in multi-computer systems under bursty and batch arrival traffic," in *The Journal of Supercomputing*, vol. 51, no. 2, pp. 115-130, 2010.

[17] Z. Qian, D. Juan, P. Bogdan, C. Tsui, D. Marculescu, and R. Marculescu, "A Support Vector Regression (SVR)-Based Latency Model for Network-on-Chip (NoC) Architectures," in *IEEE Trans. on Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 3, pp. 471-484, 2016.

[18] S. Dimitrios and A. Varma, "Latency-rate servers: a general model for analysis of traffic scheduling algorithms," in *IEEE/ACM Trans. on Networking*, vol. 6, no.5, pp. 611-624, 1998.

[19] M. Panic, C. Hernandez, E. Quinones, J. Abella, and F. J. Cazorla, "Modeling High-Performance Wormhole NoCs for Critical Real-Time Embedded Systems," in *Proc. of IEEE RTAS*, Vienna, Austria, 2016, pp. 1-12.

[20] 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," in *IEEE Trans. on Computers*, vol. 62, no. 3, pp. 452-467, 2013.