Analysis of Communication Delay Bounds for Network on Chips

Yue Qian[†], Zhonghai Lu[‡], and Wenhua Dou[†]
[†]School of Computer Science, National University of Defense Technology, China
[‡]Dept. of Electronic, Computer and Software Systems, Royal Institute of Technology, Sweden
E-mail: [†]{yueqian, douwh}@nudt.edu.cn, [‡]zhonghai@kth.se

Abstract—In network-on-chip, computing worst-case delay bound for packet delivery is crucial for designing predictable systems but yet an intractable problem due to complicated resource contention scenarios. In this paper, we present an analysis technique to derive the communication delay bound for individual flows. Based on a network contention model, this technique, which is topology independent, employs the network calculus theory to first compute the equivalent service curve for individual flows and then calculate their packet delay bound. To exemplify our method, we also present the derivation of a closed-form formula to calculate the delay bound for all-to-one gather communication. Our experimental results demonstrate the theoretical bounds are correct and tight.

I. Introduction

Quality-of-Service (QoS) has been a major concern for Network-on-Chip (NoC) since its birth around the year 2000 [2]. The reason is due to the fact that routing packets in shared resource networks inherently brings about unpredictable performance. This non-determinism does not meet the requirement of building predictable communication systems in which delay bounds must be guaranteed in any case. Apparently many applications such as multimedia, HDTV, set-top and gaming boxes have stringent requirements on communication delay bounds [11]. For example, processing 25 high definition video frames (1680 x 1024, 1920 x 1280, etc.) must be completed within 20 ms.

There are a number of approaches to address QoS for on-chip communication [2]. From a service-oriented point of view, a network may provide best-effort (BE) and guaranteed services to satisfy the requirements of different QoS provisions. To offer strict promises, a guaranteed service typically reserves resources for exclusive use. This essentially isolates interferences. For example, Time Division Multiplexing (TDM) virtual circuits (VCs) are proposed for the Æthereal [5] and Nostrum [10] NoCs. However, there are two main drawbacks. First, while such VCs provide guarantees once they are established, the setup procedure itself is non-predictable if it is done dynamically. Second, resources may often be over-reserved, leading to lower resource utilization. To make a good utilization of the shared network resources, BE networks are preferred. However, BE networks are known in achieving good average performance, but the worst case performance is extremely hard to predict. The reasons are that (1) network contention for shared resources (buffers and links) includes not only direct contention but also indirection contention. They are difficult to capture in entirety; (2) identifying the worst case is nontrivial. The worst case is in general unknown or uncertain. This hard situation leaves designers with simulation as almost the only tool to find the maximal delay by simulating various traffic scenarios. While the simulation based approach can offer the highest accuracy but can be very time-consuming. Each simulation run may take considerable time and evaluates only a single network configuration, traffic pattern, and load point. It is difficult, if not impossible, to cover all the system states [9]. In contrast, a formal-analysis-based (mathematical) method is much more efficient that it provides approximate performance numbers with a minimum amount of effort and gives insight into how different factors affect performance [9]. The accuracy of results can be rough but gives an initial and quick estimation. A calculated performance bound may be also tight enough.

In this paper, we take the formal analysis approach, aiming for deriving the worst case delay bound for individual flows, called *tagged flows*, for on-chip networks. Our assumption is that the application-specific nature of on-chip communication enables to characterize traffic with sufficient accuracy. This traffic characterization may follow the abstraction of the arrival curve in network calculus [1, 4]. The router model follows the abstraction of the service curve. We also assume a deterministic routing, which is cheap to implement and give more determinism.

For a tagged flow, we first construct its contention model. This model is a contention tree, which captures not only its contention with other *interfering flows* along its routing path but also the indirect contention experienced by the interfering flows. Based on this tree, we scan the tree to compute the output arrival curves of each branch in the contention tree. Then we derive the equivalent service curve for the tagged flow traversing the trunk of tree and calculate its delay bound. To show the usage and potential of our method, we take all-to-one gather communication, which is an important collective operation for parallel processing, as an example to derive a closed-form formula to calculate the delay bound.

The remainder of the paper is organized as follows. Section II reviews related work and summarizes our contributions. In Section III, we present the general analytical approach for calculating the delay bound for a tagged traffic flow. In Section IV, we exemplify the potential of the analysis method with the all-to-one gather collective communication. Experiments are reported in Section V. Finally we draw conclusions in Section VI.

II. Related Work

In the context of testing the feasibility of packet delivery within time constraints in wormhole networks, researchers have proposed various ways to capture network contention and then utilize scheduling theory to find and estimate the worst-case delay bounds for packets [7, 8]. In [7], the lumped link model was proposed by which the links that a packet traverses are lumped into a single link. This model does not distinguish direct and indirection contention, thus the estimated bounds are pessimistic. In [8], Lu *et al*. proposed a contention tree model, which differentiates direct and indirect contention. More importantly, the tree structure allows further differentiating trunk contention and branch contention. Both works assume deterministic routing.

In adaptively routed networks, it is a must to avoid live-lock. The best assurance of live-lock freedom is to make sure that there exists an upper bound for any packets. Studies have been carried to analytically compute packet delivery bounds. However, they consider only special cases that assume very simple traffic scenarios. For example, [3] considers only batch admissions, which inject traffic into the network once, in a deflection-routed network.

In general queuing networks, network calculus provides the means to deterministically reason about timing properties of traffic flows. Based on the powerful abstraction of *arrival curve* for traffic flows and *service curve* for network elements (routers, servers), it allows computing the worst-case delay and backlog bounds. Network calculus has been extremely successful for ensuring performance bounds in ATM, Internet with differentiated and integrated services, and other types of networks. Systematic accounts of network calculus can be found in books [1, 4].

Our work applies the network calculus theory for on-chip networks. One main reference we use is [6], where a method for computing least upper delay bound (LUDB) with leaky-bucket constrained flows and rate-latency service curves for the aggregates is presented. It derives per-flow service guarantees from per-aggregate service guarantees at a single node and consists in applying that theorem iteratively to obtain a set of end-to-end service guarantees for a flow. Then, the LUDB among all the bounds is derived from each single end-to-end service guarantee. In fact, each node is characterized by means of a lower bound on the service it offers. This method yields better bounds than those previously proposed methods [12, 13, 14].

In our approach, we also start with modeling network contention. Our model is similar to the contention tree in [8]. However, we extend this model by identifying basic contention patterns for flows. Based on these patterns, any complex contention scenario can be decomposed into the basic patterns. This divide-and-conquer approach enables to systematically study the comprehensive network contention scenarios. Furthermore, we derive the *equivalent service curve* for the tagged flow under these basic patterns, as we shall see in Section III. Moreover, we combine the network contention tree model with network calculus analysis. Consequently, this combination makes it possible to calculate delay bounds for any tagged flow in networks with an arbitrary topology. We summarize our contributions as follows:

- We identify three basic flow contention patterns. Any complex contention scenario for a tagged flow can be described by a composition of these scenarios. Furthermore, we give formal representations for calculating the equivalent service curve for the three basic patterns.
- We propose an analysis procedure to compute the delay bound for a tagged flow interfered by other flows. It contains two main steps: (1) construct a contention tree; (2) scan the tree and compute the equivalent service curve for the tagged flow.
- We present a case study on all-to-one gather communication and derive the closed-form formula to compute the delay bound.

III. The Delay Bound Analysis

We first illustrate the problem, and then detail our delay bound analysis technique.

A. The Problem Illustration

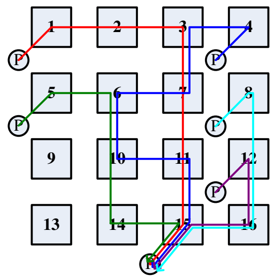

Fig. 1. A problem example.

We use an example to illustrate the problem that we address and to explain terminology used in the paper. Fig. 1 shows a network with 16 nodes, numbered from 1, 2, , 16. A node contains a core and a router. We assume deterministic routing for the network, and routers serve flows in the FIFO order. There are 5 packet flows sent from five nodes 1, 4, 8, 12 and 16 to node 15. We denote $f_{(i,j)}$ as the flow injected at router i and ejected at router j, $\alpha_{i,j}$ as arrival curve at ingress router, $\alpha^*_{(i,j)}$ as output arrival curve at egress router, and β_i as service curve of router i. We call the flow for which we shall derive its delay bound *tagged flow*, other flows that share resources with it *interfering* or *contention flows*. In this example, $f_{(1,15)}$ is the tagged flow, and $f_{(4,15)}$, $f_{(5,15)}$, $f_{(8,15)}$ and $f_{(12,15)}$ are interfering flows.

B. The Router Service Model

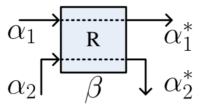

Fig. 2. Two flows multiplexing in a router.

Fig. 2 shows a lossless router serving two flows, *tagged* and *contention*, in the FIFO order. Assume the router guarantees a service curve β to both flows. The tagged and contention flow have α_1, α_2 as arrival curve, respectively. We define $\epsilon(\beta, \alpha_2)$ as the *equivalent service curve* [4, 6] to the tagged flow, where $\epsilon(.,.)$ is a function to compute the equivalent service curve. Thus, according to [4], the output arrival curve of tagged flow can be derived easily as $\alpha_1^* = \alpha_1 \oslash \epsilon(\beta, \alpha_2)$, and its delay bound is $h(\alpha_1, \epsilon(\beta, \alpha_2))$, where $h(.,.)$ is the function to compute the maximum horizontal distance between the arrival curve and the service curve.

C. Interference Patterns and their Analytical Models

Apparently, network flow contention scenarios are diverse and complicated. A tagged flow directly contends with interfering flows. Also, interfering flows may contend with each other and then contend with the tagged flow again. This indirect contention may in turn influence the performance of the tagged flow. To decompose a complex contention scenario, we identify three primitive contention patterns.

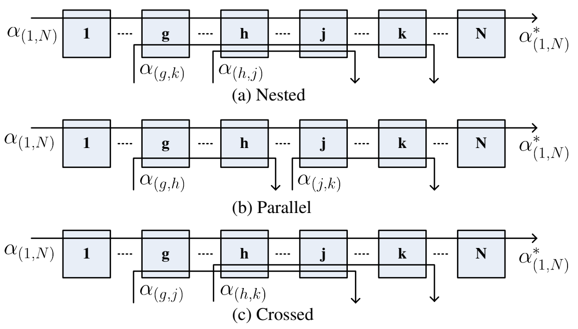

Fig. 3. The three basic contention patterns for a tagged flow.

Suppose that a tagged flow $f_{(1,N)}$ traverses a tandem of N routers from source to destination, and is multiplexed with contention flows. The contention scenarios the tagged flow may experience can be classified into three patterns, *Nested*, *Parallel* and *Crossed*, as shown in Fig. 3. In the following, we analyze the three scenarios and derive their basic analytical models with focus on the derivation of service curve the tandem provides. Once obtaining the tandem service curve, the output arrival curve and delay bound can be derived.

Scenario I: Nested contention flows. As illustrated in Fig. 3(a), contention flow $f_{(h,j)}$ is nested in $f_{(g,k)}$, $1 \leq g \leq h \leq j \leq k \leq N$. The sub-tandem (h,j) serves the aggregation of $f_{(1,N)}$, $f_{(g,k)}$ and $f_{(h,j)}$ with service curve $\bigotimes_{i=h}^{j}\beta_i$, if excluding $f_{(h,j)}$, we can get the equivalent service curve for $f_{(1,N)}$ and $f_{(g,k)}$ as $\epsilon_{(h,j)} = \epsilon(\bigotimes_{i=h}^{j}\beta_i, \alpha_{(h,j)})$. Then the service curve of sub-tandem (g,k) for the aggregation of $f_{(1,N)}$ and $f_{(g,k)}$ is computed as $(\bigotimes_{i=g}^{h-1}\beta_i) \otimes \epsilon_{(h,j)} \otimes (\bigotimes_{i=j+1}^{k}\beta_i)$, thus we can derive the equivalent service curve $\epsilon_{(g,k)}$ for $f_{(1,N)}$ as $\epsilon\Big((\bigotimes_{i=g}^{h-1}\beta_i) \otimes \epsilon_{(h,j)} \otimes (\bigotimes_{i=j+1}^{k}\beta_i), \alpha_{(g,k)}\Big)$. Therefore the tandem $(1,N)$ serves $f_{(1,N)}$ with the service curve $\beta_{(1,N)} = (\bigotimes_{i=1}^{g-1}\beta_i) \otimes \epsilon_{(g,k)} \otimes (\bigotimes_{i=k+1}^{N}\beta_i)$.

Scenario II: Parallel contention flows. As shown in Fig. 3(b), contention flows $f_{(h,j)}$ and $f_{(g,k)}$ are independent. Similarly to Scenario I, we can get the equivalent service curve of sub-tandem (g,h) and (j,k) for $f_{(1,N)}$ respectively, as $\epsilon_{(g,h)} = \epsilon(\bigotimes_{i=g}^{h}\beta_i, \alpha_{(g,h)})$ and $\epsilon_{(j,k)} = \epsilon(\bigotimes_{i=j}^{k}\beta_i, \alpha_{(j,k)})$. Hence the service curve of tandem $(1,N)$ for $f_{(1,N)}$ can be calculated as $\beta_{(1,N)} = (\bigotimes_{i=1}^{g-1}\beta_i) \otimes \epsilon_{(g,h)} \otimes (\bigotimes_{i=h+1}^{j-1}\beta_i) \otimes \epsilon_{(j,k)} \otimes (\bigotimes_{i=k+1}^{N}\beta_i)$.

Scenario III: Crossed contention flows. As shown in Fig. 3(c), contention flow $f_{(g,j)}$ is crossed with $f_{(h,k)}$. We can see there are two cross points produced by these two flows, one at the ingress of router h and the other at the egress of router j. If we cut $f_{(g,j)}$ arbitrarily at the first cross point, i.e. at ingress of router h, $f_{(g,j)}$ will be split into two flows $f_{(g,h-1)}$ and $f_{(h,j)}$, as shown in Fig. 4. Then the problem is strictly transformed to the combination of scenario I and II that $f_{(g,h-1)}$ is separate and $f_{(h,j)}$ is nested in $f_{(h,k)}$. Apparently the arrival curve $\alpha_{(g,h-1)}$ of $f_{(g,h-1)}$ equals to $\alpha_{(g,j)}$ and the arrival curve $\alpha_{(h,j)}$ of $f_{(h,j)}$ equals to $\alpha^*_{(g,h-1)}$. To compute $\alpha^*_{(g,h-1)}$, we need to get the arrival curve of $f_{(1,N)}$ at the ingress of sub-tandem $(g, h-1)$ which equals to $\alpha^*_{(1,g-1)}$ as $\alpha_{(1,N)} \oslash (\bigotimes_{i=1}^{g-1}\beta_i)$. Then the equivalent service curve of sub-tandem $(g, h-1)$ for $f_{(g,h-1)}$ is derived as $\epsilon(\bigotimes_{i=g}^{h-1}\beta_i, \alpha^*_{(1,g-1)})$. Thus we obtain the output arrival curve of $f_{(g,h-1)}$ as $\alpha^*_{(g,h-1)} = \alpha_{(g,h-1)} \oslash \epsilon(\bigotimes_{i=g}^{h-1}\beta_i, \alpha^*_{(1,g-1)})$. With $\alpha_{(g,h-1)}$, $\alpha_{(h,j)}$ and $\alpha_{(h,k)}$ obtained, we can calculate the tandem service curve for $f_{(1,N)}$ directly as $\beta_{(1,N)} = (\bigotimes_{i=1}^{g-1}\beta_i) \otimes \epsilon_{(g,h-1)} \otimes \epsilon_{(h,k)} \otimes (\bigotimes_{i=k+1}^{N}\beta_i)$.

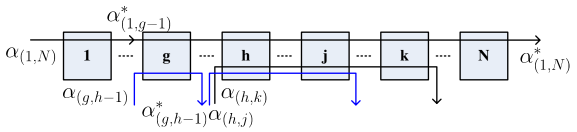

Fig. 4. Transform the crossed flows to non-crossed flows.

D. The Analysis Procedure

In Section III-C, we have derived three basic analytical models for analyzing the delay bound for the tagged flow multiplexed with two contention flows in three different scenarios. For complex scenarios with more than two contention flows in the tandem, the problem can be split into the basic models and expressed with a contention tree. The analysis procedure is detailed as follows:

Step 1: Construct a contention tree to model the network contentions produced by interfering flows;

- **Step 1.1**: Let the tandem traversed by the tagged flow be the trunk;
- **Step 1.2**: Let the tandems traversed by the interfering flows before reaching the trunk node be branches, which may also have its own sub-branches;

Step 2: Scan the contention tree and compute all the output arrival curves of flows traversing the branches using the basic analytical models iteratively;

Step 3: Compute the equivalent service curve for the trunk flow (tagged flow) and derive the delay bound.

We use the example in Fig. 1 to explain this procedure. In Fig. 1, the tagged flow $f_{(1,15)}$ is multiplexed with $f_{(4,15)}$, $f_{(5,15)}$, $f_{(8,15)}$ and $f_{(12,15)}$. We construct a contention tree shown in Fig. 5 to model the flow contentions, where the tandem $(1,15)$ traversed by $f_{(1,15)}$ composes the trunk with the ingress router 1 up to egress router 15 and the tandems traversed by contention flows before reaching the trunk nodes compose the branches. At router 15, $f_{(5,15)}$ is first injected into the trunk and the sub-tandem traversed by $f_{(5,15)}$ builds

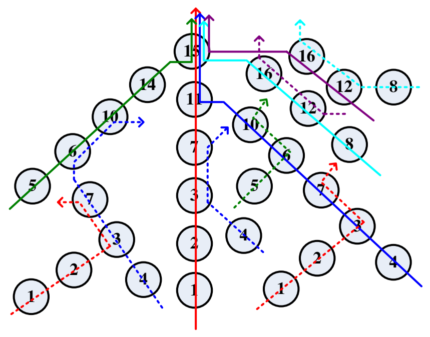

Fig. 5. The contention tree for the tagged flow $f_{(1,15)}$.

the first branch. Branch $(5, 14)$ has sub-branch $(4, 7)$ produced by its own contention flow $f_{(4,15)}$ as indicated by the dashed blue line in Fig. 5. Also $f_{(1,15)}$ contends with $f_{(4,15)}$ at the sub-tandem $(3, 7)$ that builds the sub sub-branch $(1, 2)$ as the dashed red line indicates for sub-branch $(4, 7)$.

If a tandem containing a sequence of nodes, at the node where one flow is multiplexed with the traffic coming form the upstream node, we treat the flows in the tandem as an aggregate flow. For instance, $f_{(8,15)}$ and $f_{(12,15)}$ injected into router 15 produce two branches $(8, 16)$ and $(12, 16)$, where branch $(12, 16)$ has sub-branch $(8, 8)$. These two branches are composed of the same nodes. Thus, we treat $f_{(8,15)}$ and $f_{(12,15)}$ as an aggregate flow $(f_{(8,15)}, f_{(12,15)})$ and reserve only one branch $(8, 16)$ for optimization.

If flows coming from different branches injected into and ejected out a tandem at the same ingress and egress nodes, we treat the flows at the ingress node as an aggregate flow. For instance, since both $f_{(5,15)}$ and the aggregate flow $(f_{(8,15)}, f_{(12,15)})$ are injected into router 15 and ejected out router 15 to the core, they can further compose into a larger aggregate flow $(f_{(5,15)}, f_{(8,15)}, f_{(12,15)})$ for the trunk.

$f_{(4,15)}$ contends with $f_{(1,15)}$ twice. It is first injected into the trunk at router 3 and ejected out at router 7, and then injected at router 11 again. Hence two branches extend at router 3 and 11 in the trunk.

To derive the equivalent service curve for trunk flow $f_{(1,15)}$, we scan the contention tree in Fig. 5 using the Depth-First-Search to first compute the output arrival curves of $f_{(1,15)}$ traversing the sub sub-branches $(1, 2)$ for sub-branch $(4, 7)$, then the output arrival curve of $f_{(4,15)}$ traversing the sub-branches $(4, 7)$ for branch $(5, 14)$, thus the output arrival curve of $f_{(5,15)}$ traversing branch $(5, 14)$ for trunk $(1, 15)$ is derived using the basic analytical models. Analogously, the output arrival curves of $f_{(4,15)}$ traversing branch $(4, 10)$ and $(4, 4)$ for trunk $(1, 15)$ are derived. We compute the output arrival curve of the aggregate flow $(f_{(8,15)}, f_{(12,15)})$.

After all arrival curves of injected flows are obtained, we then compute the trunk service curve for $f_{(1,15)}$ and thus delay bound for $f_{(1,15)}$ can be derived.

We have designed two algorithms, one for constructing the contention tree and the other for computing the equivalent service curves of branches and trunk. Due to space limitation, they are not presented here.

IV. All-to-One Delay Bound

To demonstrate the potential of our analysis method, we study an all-to-one communication case for which we derive closed-form formula to calculate the delay bound.

A. All-to-one Gather Communication on a Mesh NoC

We consider all-to-one gather communication, which is one of common collective operations for parallel applications, on a $m \times n$ mesh NoC with the XY routing. We assume a homogeneous NoC where all cores and routers have the same capacity. To simplify our discussions, we assume that all flows from the cores are constrained by an affine arrival curve $\alpha_{r,b}(t) = rt + b$ and all routers provide a rate-latency service curve $\beta_{R,T}(t) = R(t - T)^+$. As there coexist $m \times n - 1$ flows simultaneously, $(mn - 1)r \leq R$, meaning that the sum of the rates for the flows must be less than or equal to the router's service rate.

Fig. 6(a) shows an all-to-one gather communication in a 4×4 NoC with the gathering node $(3, 3)$. Since the NoC is a mesh, we naturally use 2D coordinates (x, y) to represent a node. We also define a labeling function $l(x, y) = n(x - 1) + y$ to mark the nodes with a sequence of numbers. Let node (x, y) be the gathering node, we use $f_{(n(i-1)+j, n(x-1)+y)}$ to represent the flow from node (i, j) to node (x, y). Note this is consistent with the labeling convention for flows in Section III. By the labeling function, nodes $(1, 1)$ and $(3, 3)$ are labeled as 1 and 11, respectively. Fig. 6(b) depicts the contention tree for flow $f_{(1,11)}$ considering aggregate flows in the branches.

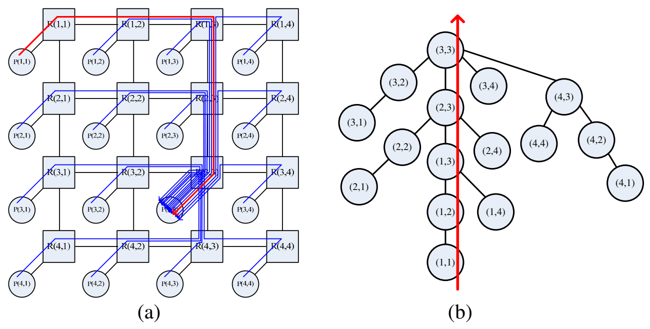

Fig. 6. All-to-one gather communication in 4×4 NoC and the contention tree for $f_{(1,11)}$.

B. Closed-form Formula for Delay Bound

Without loss of generality, we set $f_{(1,n(x-1)+y)}$ as the tagged flow and (x, y) as the gathering node. The contention tree for $f_{(1,n(x-1)+y)}$ can be built similarly to Fig. 6(b). Next we scan the contention tree to compute the arrival curves of all injected aggregate flows coming from branches to the trunk. After that we can calculate the equivalent service curve of the trunk for the tagged flow. Finally, the delay bound for the tagged flow can be derived.

While scanning the tree, we classify the branches into three types and derive the formulas for computing the output arrival curves of the aggregate flows injected to the trunk.

For branches $(i, 1) \rightarrow (i, y - 1)$, $i = 2, \ldots, m$ and $(i, n) \rightarrow (i, y-1)$, $i = 1, \ldots, m$, shown in Fig. 7, we use Theorem 4.15 in [6] to compute the output arrival curve of the aggregate flow at egress node $(i, y - 1)$ and $(i, y + 1)$ as follows.

At node $(i, y - 1)$, $\alpha^*_{(i,y-1)} = \alpha_{r^*_1, b^*_1}$ with

$$\begin{cases} r^*_1 = (y - 1)r \\ b^*_1 = (y - 1)(b + \frac{y}{2}rT), \end{cases}$$

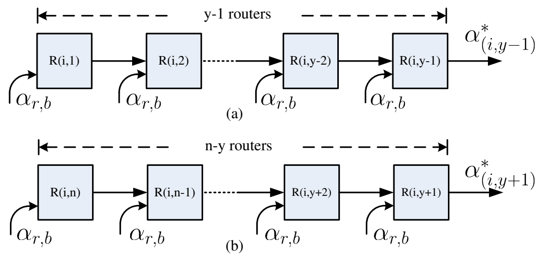

Fig. 7. Output arrival curves for the aggregate flow at egress node $(i, y-1)$ and $(i, y+1)$.

and at node $(i, y+1)$, $\alpha^*_{(i,y+1)} = \alpha_{r^*_2,b^*_2}$ with

$$\begin{cases} r^*_2 = (n-y)r \\ b^*_2 = (n-y)(b + \dfrac{n-y+1}{2}rT). \end{cases}$$

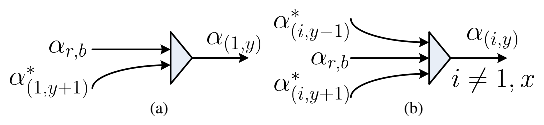

Fig. 8. Input arrival curves for the aggregate flow at ingress node (i, y).

At trunk node $(1, j)$, $j = 2, \ldots, y-1$, only one flow with arrival curve $\alpha_{r,b}$ is newly arriving. At trunk node $(1, y)$, the aggregate flow is the sum of traffic coming from the local core and a branch as depicted in Fig. 8(a), for that $\alpha_{(1,y)} = \alpha_{r_1,b_1} = \alpha_{r,b} + \alpha^*_{(1,y+1)}$ and the parameters r_1 and b_1 is derived below. At node (i, y), $i = 2, \ldots, x-1, x+1, \ldots, m$, the newly injected traffic is composed of three incoming flows such as $\alpha^*_{(i,y-1)}, \alpha_{r,b}, \alpha^*_{(i,y+1)}$ as illustrated in Fig. 8(b), thus the aggregate flow's arrival curve is $\alpha_{(i,y)} = \alpha_{r_2,b_2} = \alpha^*_{(i,y-1)} + \alpha_{r,b} + \alpha^*_{(i,y+1)}$ with parameters as follows.

At trunk node $(1, y)$, $\alpha_{(1,y)} = \alpha_{r_1,b_1} = \alpha_{r,b} + \alpha^*_{(1,y+1)}$ with

$$\begin{cases} r_1 = (n-y+1)r \\ b_1 = (n-y+1)(b + \dfrac{n-y}{2}rT), \end{cases} \tag{1}$$

and the input arrival curve at node (i, y) is $\alpha_{(i,y)} = \alpha_{r_2,b_2} = \alpha^*_{(i,y-1)} + \alpha_{r,b} + \alpha^*_{(i,y+1)}$ with

$$\begin{cases} r_2 = nr \\ b_2 = nb + \dfrac{n^2 + 2y^2 - 2ny + n - 2y}{2}rT. \end{cases} \tag{2}$$

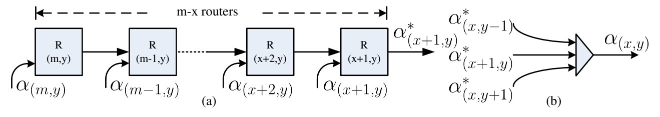

Fig. 9. Output arrival curves for the aggregate flow at egress node $(x+1, y)$ and input arrival curve at trunk node (x, y).

Finally, at trunk node (x, y), the newly injected flow is coming from three branches with the arrival curve of $\alpha^*_{(x,y-1)}, \alpha^*_{(x+1,y)}, \alpha^*_{(x,y+1)}$ respectively. The branch $(m, y) \rightarrow (x+1, y)$ has sub-branches at each branch node. Fig. 9(a) indicates the flows aggregated at the branch and the output at $(x+1, y)$. The corresponding output arrival curve is $\alpha^*_{(x+1,y)} = \alpha_{r^*_3,b^*_3}$, where

$$\begin{cases} r^*_3 = (m-x)r_2 \\ b^*_3 = (m-x)(b_2 + \dfrac{m-x+1}{2}r_2T). \end{cases}$$

Hence we get the aggregated branch arrival curve $\alpha_{(x,y)}$, $\alpha_{(x,y)} = \alpha_{r_3,b_3} = \alpha^*_{(x,y-1)} + \alpha^*_{(x+1,y)} + \alpha^*_{(x,y+1)}$ with

$$\begin{cases} r_3 = r^*_1 + r^*_2 + r^*_3 \\ b_3 = b^*_1 + b^*_2 + b^*_3. \end{cases} \tag{3}$$

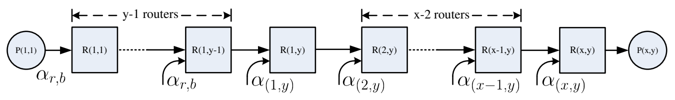

Fig. 10. All aggregate flows multiplexed with the tagged flow in the trunk.

Now we derive all branch arrival curves for the trunk. As shown in Fig. 10, the contention flows are nested. We can apply the basic analytical model of Scenario I iteratively to derive the equivalent service curve for the tagged flow. Then we get Formula (4) for computing the delay bound $\bar{D}$, which has been proved to be the least upper delay bound [6].

$$\begin{aligned} \bar{D} = & \sum_{i=1}^{x+y-1} T + \\ & \sum_{j=1}^{y-1} \frac{b}{R^{x+y-j} \cdot \prod_{k=1}^{y-1-j} \frac{1}{R+r} \cdot \frac{1}{R+r_1} \cdot \prod_{l=1}^{x-2} \frac{1}{R+r_2} \cdot \frac{1}{R+r_3}} + \\ & \frac{b_1}{R^x \cdot \prod_{l=1}^{x-2} \frac{1}{R+r_2} \cdot \frac{1}{R+r_3}} + \\ & \sum_{j=1}^{x-2} \frac{b_2}{R^{x-j} \cdot \prod_{k=1}^{x-2-j} \frac{1}{R+r_2} \cdot \frac{1}{R+r_3}} + \frac{b_3}{R}. \end{aligned} \tag{4}$$

V. Experiment

We design experiments to verify our analytical approach. We use the all-to-one gather communication. Since we have derived the delay bound formula for all-to-one communication in Section IV, we conduct simulations to compare observed maximum delays in simulations with calculated bounds.

We consider a 4×4 mesh NoC with different gathering node $(3,3)$ and $(4,4)$ and a 5×5 mesh NoC with gathering node $(3,3)$, $(4,4)$ and $(5,5)$. The delay bound for the flow generated by node $(1,1)$ to the gathering node is calculated and simulated. Each flow generated by the core is constrained by the arrival curve of $\alpha_{r,b}(t) = 1t + 4$ where the generating rate $r = 1$ packet/cycle and the burst $b = 4$ packets, and each router guarantees the service curve of $\beta_{R,T}(t) = 25(t-3)^+$, where the serving rate $R = 25$ packets/cycle and the latency $T = 3$ cycles.

We build a NoC simulator that each core generates a Poisson stream with rate $\lambda = 1.1$ packets/cycle, which then passes through a traffic shaper with leaky rate $r = 1$ packet/cycle and bucket size $b = 4$ packets. We construct the router by rate-latency server as $\beta_{25,3}$ with big enough buffer size. The delay experienced by a packet of the tagged flow is recorded when it reaches the gathering node. We assume zero link delay in experiments.

We run every simulation for 1E+9 cycles and get the calculated delay bounds using Eq. (4) for different settings. The calculated bounds and experimental maximum delays of all packets are listed in Table I. We denote calculated delay bound and simulated maximum delay as $\bar{D}$ and $D_{\max}$, respectively. For all results, the unit for delay is *cycle*.

NoC	4×4 Mesh		5×5 Mesh		
Tagged Flow	$(1,1) \rightarrow (3,3)$	$(1,1) \rightarrow (4,4)$	$(1,1) \rightarrow (3,3)$	$(1,1) \rightarrow (4,4)$	$(1,1) \rightarrow (5,5)$
$\bar{D}$	19.97	25.85	25.43	30.36	36.37
$D_{\max}$	19	23	23	29	34

TABLE I
CALCULATED BOUNDS AND OBSERVED MAXIMUM DELAYS.

From Table I we can see that the calculated delays bound the experimental maximum delays tightly. It also shows that the delay of the same tagged flow becomes larger when the NoC mesh size increases. When the gathering node is nearer to the network edge from $(3,3)$ to $(4,4)$ and later to $(5,5)$, the maximum delay and the delay bound increase since the overall distance from sources to the gathering node increases.

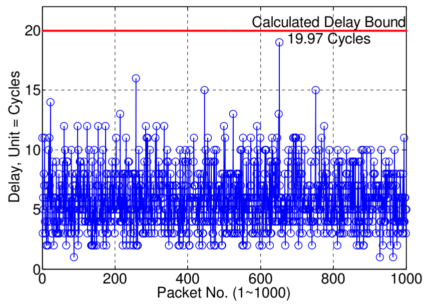

Fig. 11. Calculated bound and observed maximum delay for flow $(1,1) \rightarrow (3,3)$ in 4×4 mesh NoC.

We also plot a sequence of 1000 observed delays including the maximum delay for tagged flow $(1,1) \rightarrow (3,3)$ in 4×4 mesh NoC in Fig. 11. The blue circle indicates the delay experienced by a packet and the red line represents the calculated delay bound. We can see the simulated delays are totally constrained by the calculated delay bound and the bound is tight. The calculated bound is 19.97 cycles, and the observed maximum delay is 19 cycles.

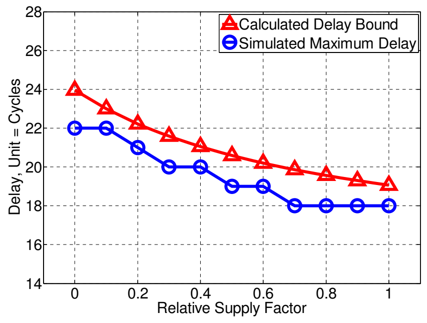

Fig. 12. Delays under different relative supply factors.

We also consider the relationship between the delay and the service capacity of the routers. Assume the routers have a service rate $R = (mn-1) \cdot r \cdot (1+\eta)$, i.e., η is a relative supply factor to express the extra service capability of the routers. Fig. 12 reports the calculated delay bounds and simulated maximum delays for the 4×4 mesh NoC with $r = 1$ packet/cycle, $b = 4$ packets, $T = 3$ cycles and η varying form 0 to 1 with step 0.1. As can be seen, the simulated maximum delays are bounded by the calculated delay bound. As η increases, the maximum delay decreases but the decrement is becoming smaller. This indicates that increasing the service rate can reduce the delay bound until the delay approaches the propagation delay.

VI. CONCLUSION

Application exerts stringent requirements on on-chip networks to ensure performance bounds even under worst cases. In this work, we present an analysis method to compute the delay bound for tagged flows. This method is the consequence of synergistically combining the network contention tree model with network calculus. It assumes deterministic routing but is topology independent. Thus it can be applied to a variety of networks with a regular or irregular topology. To exemplify the potential of our technique, derivation of a closed-form formula for computing the delay bound for all-to-one communication is detailed. Our experiments demonstrate the correctness and tightness of the calculated bounds compared with simulated results. We conclude that derivation of the delay bound for traffic flows in best-effort routed NoCs can be well conducted.

Our work has followed the network calculus theory. This theory allows computing buffer bounds in routers as well. We shall dig out this in the future. As we have considered feed-forward networks in this work, investigation on feedback networks is another dimension to carry out. This would lead to capture back pressures for given buffer sizes.

REFERENCES

[1] C. Chang, "Performance Guarantees in Communication Networks," *Springer-Verlag*, 2000.
[2] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of Network-on-chip," *ACM Computing Survey*, vol. 38(1), 2006.
[3] J. T. Brassil and R. L. Cruz, "Bounds on Maximum Delay in Networks with Deflection Routing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 6(7), pp. 724–732, 1995.
[4] J.-Y. Le Boudec and P. Thiran, "Network Calculus-A Theory of Deterministic Queuing Systems for the Internet," *Springer-Verlag*, vol. 2050, 2004.
[5] K. Goossens, J. Dielissen, and A. Rădulescu, "The Æthereal Network on Chip: Concepts, Architectures, and Implementations," *IEEE Design and Test of Computers*, vol. 22(5), pp. 21–31, 2005.
[6] L. Lenzini, L. Martorini, E. Mingozzi, and G. Stea, "Tight End-to-End Per-Flow Delay Bounds in FIFO Multiplexing Sink-Tree Networks," *Perform. Eval.*, vol. 63(9-10), pp. 956–987, 2006.
[7] S. Balakrishnan and F. Ozguner, "A Priority-Driven Flow Control Mechanism for Real-Time Traffic in Multiprocessor Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 09(7), pp. 664–678, 1998.
[8] Z. Lu, A. Jantsch, and I. Sander, "Feasibility Analysis of Messages for On-Chip Networks Using Wormhole Routing," *Proc. of Asia South Pacific Design Automation*, 2005.
[9] W. J. Dally and B. Towles, "Principles and Practices of Interconnection Networks," *Morgan Kaufmann*, 2004.
[10] Z. Lu and A. Jantsch, "Slot Allocation for TDM Virtual-Circuit Configuration for Network-on-Chip," *ICCAD'07*, San Jose, CA, USA, Nov. 2007.
[11] J. Ostermann et. al, "Video Coding with H.264/AVC: Tools, Performance and Complexity," *IEEE Circuits and Systems Magazine*, vol. 4(1), pp. 7–28, 2004.
[12] A. Charny and J.-Y. Le Boudec, "Delay bounds in a network with aggregate scheduling," in: *Proc. of QoFIS00*, Berlin, Germany, 2526 September 2000, pp. 1-13.
[13] M. Fidler and V. Sander, "A parameter based admission control for differentiated services networks," *Elsevier Computer Networks*, vol. 44(1), pp. 463–479, 2004.
[14] J.B. Schmitt and F.A. Zdarsky, "The DISCO network calculator – A toolbox for worst case analysis," in: *Proc. of VALUETOOLS06*, ACM, Pisa, Italy, 2006, October.