A

Least Upper Delay Bound for VBR Flows in Networks-on-Chip with Virtual Channels

FAHIMEH JAFARI, KTH Royal Institute of Technology, Sweden
ZHONGHAI LU, KTH Royal Institute of Technology, Sweden
AXEL JANTSCH, Vienna University of Technology, Austria

Real-time applications such as multimedia and gaming require stringent performance guarantees, usually enforced by a tight upper bound on the maximum end-to-end delay. For FIFO multiplexed on-chip packet switched networks we consider worst-case delay bounds for *Variable Bit-Rate (VBR)* flows with aggregate scheduling, which schedules multiple flows as an aggregate flow. VBR Flows are characterized by a maximum transfer size (L), peak rate (p), burstiness (σ), and average sustainable rate (ρ). Based on network calculus, we present and prove theorems to derive per-flow end-to-end *Equivalent Service Curves (ESC)* which are in turn used for computing *Least Upper Delay Bounds (LUDBs)* of individual flows. In a realistic case study we find that the end-to-end delay bound is up to 46.9% more accurate than the case without considering the traffic peak behavior. Likewise, results also show similar improvements for synthetic traffic patterns. The proposed methodology is implemented in C++ and has low run-time complexity, enabling quick evaluation for large and complex SoCs.

Categories and Subject Descriptors: C.4 [**Performance of Systems**]: Modeling techniques

General Terms: Design, Performance

Additional Key Words and Phrases: Network-on-chip (NoC), performance evaluation, network calculus, worst-case delay bound, FIFO multiplexing

ACM Reference Format:
Jafari, F., Lu, Z., and Jantsch, A. 2015. Least Upper Delay Bound for VBR Flows in Networks-on-Chip with Virtual Channels. ACM Trans. Des. Autom. Electron. Syst. V, N, Article A (January YYYY), 34 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

In networks-on-chip, resources like wires, buffers, and switches are shared among multiple communication flows to provide cost efficiency. At the same time many applications have real-time requirements and, consequently, delay and throughput constraints on the communication. To guarantee maximum delay and minimum throughput for one given communication flow, the interference in the shared resources from other flows has to be analyzed and bounded. We assume that all traffic can be well characterized as flows and scheduled as aggregate which means multiple flows are scheduled as an aggregate flow. For a given flow, we study the maximum interference of all other flows based on the network calculus theory [Le Boudec et al. 2004].

Author's addresses: F. Jafari and Z. Lu, Department of Electronic and Computer Systems, School of Information and Communication Technology, KTH Royal Institute of Technology, Stockholm, Sweden; email: {fjafari, zhonghai}@kth.se; A. Jantsch, Vienna University of Technology, Vienna, Austria; email: jantsch@ict.tuwien.ac.at.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

In network calculus, flows are characterized as *arrival curves* and the service offered to flows by a network element such as a link or a switch is abstracted as *service curve*. Since the network contention for shared resources includes not only direct contention but also indirect contention, predicting the worst-case performance is extremely hard.

To calculate the accurate delay bound per flow, the main problem is to obtain the *end-to-end Equivalent Service Curve (ESC)* and internal output arrival curves of individual flows in an arbitrary network of servers in terms of the latencies of the individual schedulers in the network. Since the required theorems for calculating performance metrics of VBR traffic transmitted in the FIFO order and scheduled as aggregate have not been represented so far, we [Jafari et al. 2011; Jafari et al. 2012] have defined and proved them based on network calculus [Chang 2000; Le Boudec et al. 2004]. In [Jafari et al. 2011], we proposed and proved the required theorem for deriving the output characterization of VBR traffic under the defined system model to have exact vision about output metrics used for obtaining performance bounds. In [Jafari et al. 2012], the required theorems for computing end-to-end ESC and end-to-end delay bound are defined and proved. Moreover, we presented a simple example to show how the proposed theorems can be used in the network. The method presented in [Jafari et al. 2012] only considers direct contentions of a tagged flow. In this paper, we use the proposed theorems [Jafari et al. 2011; Jafari et al. 2012] to present a formal approach for performance analysis modeling both direct and indirect contentions.

VBR is a class of traffic in which the rate can vary significantly from time to time, containing bursts. Real-time compressed voice and video and time-sensitive bursty data traffic are examples of VBR traffic. Real-time VBR flows can be characterized by a set of four parameters, (L, p, σ, ρ), where L is the maximum transfer size, p peak rate, σ burstiness, and ρ average sustainable rate [Le Boudec et al. 2004]. For instance, in a NoC with a link data width of 32 *bits*, frequency of 500 *MHz*. This means a link bandwidth of 16 *Gbits*$/s$ (32 *bits*$\times 500$ *MHz*). An HDTV video stream can be characterized with $L = 32$ *bits*, $p = 16$ *Gbits*$/s$, $\sigma = 960$ *Kbits*, $\rho = 76$ *Mbits*$/s$. Our assumption is that the application-specific nature of the network enables to characterize traffic with sufficient accuracy.

For an individual flow, called a *tagged flow*, we first consider resource sharing scenarios (channel sharing, buffer sharing, and channel&buffer sharing) in the routers and then build analysis models for different resource sharing components. We assume that the routers employ round robin scheduling to share the link bandwidth. Based on these models, we can derive the intra-router ESC for an individual flow. To consider the contention which a flow may experience along its routing path, we present a recursive algorithm to classify and analyze flow interference patterns. The algorithm uses the proposed theorems to analyze the effect of contention flows on the tagged flow. Based on this algorithm, we derive the end-to-end ESC and then Least Upper Delay Bound (LUDB) for a tagged flow under the mentioned system model. To show the potential of our method, we experiment three case studies to derive delay bounds and compare them with simulation results. It is worth mentioning that the paper does not deal with the back-pressure, but calculates the buffer size thresholds to make sure the back-pressure does not occur in the network.

The remainder of this paper is organized as follows. Section 2 gives an account of related works. In Section 3, we introduce the basics of network calculus. Section 4 discusses the underlying system model and notations in our analysis. Section 5 is devoted to the theorems required for computation of performance metrics. We present our formal method for the performance analysis and computation of LUDB in Section 6. Numerical results are reported in Section 7. Finally, Section 8 gives the conclusions and highlight directions for future work.

2. RELATED WORK

Recently, NoC designers have a great deal of interest in the development of analytical performance models [Bakhouya et al. 2011]. Ogras et al. [2005] give a unified representation of NoC architectures and applications and consider some major research problems in the design area. As represented in this work, most of research problems need to analytically analyze and evaluate performance metrics in the network. we [Kiasari et al. 2013] have surveyed four popular mathematical formalisms -*dataflow analysis*, *schedulability analysis*, *queueing theory*, and *network calculus*- along with their applications in NoCs. Also, we have reviewed strengths and weaknesses of each technique and its suitability for a specific purpose.

Dataflow analysis is a deterministic approach based on graph theory. As an example, Hansson et al. [2008] present a model using a cyclo-static dataflow graph for buffer dimensioning for NoC applications. In dataflow analysis, it is assumed that the pattern of communication among cores and switches are deterministic and predefined. Dataflow analysis must be used with restricted models such as DDF and CSDF to capture dynamic behavior. In other words, the expressiveness is typically traded off against analyzability and implementation efficiency in this formalism.

Schedulability analysis is an analytical approach for investigating the timing properties in real-time systems. It gets a set of tasks, their worst-case execution time, and a scheduling policy as inputs and determines whether these tasks can be scheduled such that deadline misses never occur. One example of this approach in NoCs is presented by Shi and Burns [2008]. Schedulability analysis uses simpler event models compared to the other mathematical formalisms and consequently the performance model is easily extracted with less accuracy.

The proposed models by Lee [2003] and Rahmati et al. [2009 ; 2013] are inspired by schedulability analysis. Lee [2003] presents a worst-case analysis model for real-time communication and also proposes a feasibility test algorithm for a simplex virtual circuit in wormhole networks. This work is extended by Rahmati et al. [2009] towards NoCs, computing real-time bounds for high bandwidth traffic. They also extend the model [2013] to provide more detailed switch models and consider virtual channels and variable buffer lengths. The key advantage of these methods is that they compute the worst-case bounds with low time complexity without any special hardware support, but the main limitation is that they do not leverage the input arrival patterns, which it leads to over approximations of the performance analysis.

Most of the current works use *queuing theory*-based approaches. For example, Moadeli et al. [2007] analyze the traffic behavior in a NoC with the spidergon topology and wormhole routing and then present a queuing-theory-based analytical model for evaluating the average message latency in the network. Ben-Itzhak et al. [2011] propose an analytical model for deriving average end-to-end delay in a heterogeneous wormhole based NoC with heterogeneous traffic patterns, non-uniform link capacities and a variable number of virtual channels per link. Queuing approaches often use probability distributions like Poisson to model traffic in the network while Poisson distribution used in queuing model is not appropriate for characterizing traffic patterns in NoC applications because it is not able to model all significant features in this network. Queuing theory generally evaluate average quantities of metrics in an equilibrium state and characterizing the transient behavior is a very difficult problem. An approach for addressing this problem is suggested by Bogdan and Marculescu [2007]. Authors in this work proposed a statistical physics-inspired framework to model the information flow and buffers behavior in NoCs. They analyze the traffic dynamics in NoCs and effectively capture the nonstationary effects of the system workload. In following up to this work, Bogdan et al. [2010] proposed QuaLe model based on statistical physics that

can account for nonstationary observed in packet arrival processes. They also investigated the impact of packet injection rate and the data packet sizes on the multifractal spectrum of NoC traffic.

Network calculus is a mathematical framework for deriving worst-case bounds on maximum latency, backlog, and minimum throughput in network-based systems. It is able to model all traffic patterns with bounds defined by arrival curves. In this respect, designers can capture some dynamic features of the network based on shapes of the traffic flows [Bakhouya et al. 2011]. Network calculus can also abstract many scheduling algorithms and arrival classes at single queue with multiplexed arrival flows, by service curves. The service curves through a network can be convolved as a single service curve. Hence a multi-node network analysis can be simplified to a single-node analysis. Regarding these two features, network calculus can analyze many scheduling algorithms and arrival classes over a multi-node network in a uniform framework while classical queuing theory separately models different combination of them [Ciucu et al. 2012]. The probabilistic version of (deterministic) network calculus is stochastic network calculus. In some networks, such as wireless networks, the service offered by a communication channel may vary randomly over time due to channel contention and impairment. Such networks can only provide stochastic services and guarantees. For example, Rizk and Fidler [2012] use stochastic network calculus to derive per-flow end-to-end performance bounds in a network of tandem queues under open-loop fBm cross traffic which is a model for self-similar and long-range dependent aggregate Internet traffic. Since we employ deterministic network calculus, in the rest of our paper, network calculus refers to the deterministic type. A network calculus-based methodology [Bakhouya et al. 2011] analyzes and evaluates performance and cost metrics, such as latency, energy consumption, and area requirements in on-chip interconnects. Authors in this paper compare 2D mesh, spidergon, and WK-recursive topologies using a given traffic pattern and show that WK-recursive outperforms mesh and spidergon in all considered metrics. The proposed model in this paper is simple without considering virtual channel effects and modeling all interferences between flows sharing a resource in the network. Moreover, the model does not investigate the peak behavior of flows which leads to less accurate bounds while we consider performance analysis for VBR traffic in on-chip networks employing aggregate resource management.

The performance evaluation of real-time services in networks employing aggregate scheduling is particularly challenging because of its complexity. Aggregate scheduling arises in many cases. In addition to NoC, for example, it can also be applied for obtaining scalability in large-size networks. The Differentiated Service (DiffServ) [Blake et al. 1998] is an example of an architecture based on aggregate scheduling in the Internet. Despite the research efforts, few results have appeared on this subject. A survey on the subject can be found in [Bennett et al. 2002]. Charny et al. [2000] consider a closed-form delay bound for a generic network configuration under the fluid model assumption. It is also extended by Jiang [2002] to consider packetization effects. However, these works can derive bounds only for small utilization factors in a generic network configuration.

Martin et al. [2006 ; 2003] and Bauer et al. [2010] employ Trajectory Approach (TA) to compute end-to-end delay bounds in FIFO systems. The Trajectory Approach computes all the possible trajectories of a system under constraints and then takes maximum end-to-end delays on them. Bauer et al. [2010] compare Network Calculus and the Trajectory approaches on a real avionics AFDX configuration and shows that The Trajectory approach computes upper bounds which are tighter than the upper bounds computed by the network calculus one. However, they derive delay bounds by summing per-node bounds, expectedly not arriving at tight bounds but reported as being at least close under practical conditions.

The computation of delay bounds through network calculus in feed-forward networks under arbitrary multiplexing has already been addressed in different lectures [Schmitt et al. 2008; Kiefer et al. 2010; Bouillard et al. 2010]. One of these works [Bouillard et al. 2010] describes the first algorithm which can compute the worst-case end-to-end delay for a given flow for any feed-forward network under blind multiplexing, with concave arrival curves and convex service curves. Since the problem is intrinsically difficult (NP-hard), the authors show that in some cases, like tandem networks with cross-traffic interfering along intervals of servers, the complexity becomes polynomial. Then, the approach is refined [Bouillard et al. 2011] in order to take into account fixed priorities. Bouillard et al. study networks with a fixed priority service policy which means each flow is assigned a fixed priority and try to take into account the pay multiplexing only once (PMOO) phenomenon. This stream of works deal with networks of arbitrary multiplexing also known as general or blind multiplexing, which means no assumption is made about the service policy while by assuming an explicit multiplexing scheme like FIFO, tighter bounds can be obtained.

A related stream of works [Lenzini et al. 2006; Lenzini et al. 2008; Bisti et al. 2010] propose a methodology which calculates delay bounds in tandem networks of rate-latency nodes traversed by leaky bucket shaped flows. They also introduce a software tool, called DEBORAH, which implements algorithms employed in their methodology to compute delay bounds. These works consider servers in tandem or sink trees, while our proposed method computes end-to-end delay in a generic topology of NoC. Moreover, these works investigate computing delay bounds only for average behavior of flows and they do not consider peak behavior, which results in less accurate bounds.

Boyer [2010] tries to model shaping for an end-to-end delay where each server is shared by two flows. An applicative token bucket $\gamma_{r,b}$ is shaped by the bit-rate of the link λ_R, leading to a two-slopes affine arrival curve which this arrival curve is similar to one we consider for double leaky buckets. The paper investigates a simple topology, a sequence of rate-latency servers, each one shared by two flows with a FIFO policy, and a simple case of nested contentions. Moreover, authors state that their modeling is incomplete: when computing the worst-case traversal time of a flow, they model only the shaping on the considering flow, not on the interfering ones (leading to the title 'half-modeling of shaping') In this paper, we investigate both nested and crossed contentions in general to model all flows (even interfering ones) with complex interferences in on-chip networks.

All aforementioned works in the subject of aggregate resource management compute delay bounds in various network infrastructures but not on-chip networks. As regards to NoC architecture, analytical models are very close to the reality of the system. For instance, a router in on-chip networks can be modeled in pure hardware which means the micro-architecture is feasible for analysis. Therefore, network calculus can provide the analysis more accurate in on-chip networks.

Qian et al. [2010] present analytical models for traffic flows under strict priority queueing and weighted round robin scheduling in on-chip networks. They then derive per-flow end-to-end delay bounds using these models. Like most of mentioned works, the proposed model by Qian et al. [2010] does not deal with peak behavior of flows, which results in less accurate bounds. The proposed method in this paper considers performance analysis for VBR traffic characterized by (L, p, σ, ρ) in on-chip networks employing aggregate resource management. As such, our method achieves more accurate delay bounds.

3. NETWORK CALCULUS BACKGROUND

Network calculus is a mathematical framework to derive worst case bounds and analyze performance guarantees in networks. This paper uses Traffic SPECification

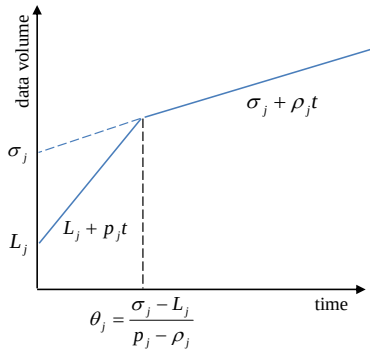

Fig. 1. Arrival curve of flow f_j with TSPEC $(L_j, p_j, \sigma_j, \rho_j)$.

(TSPEC) [Wroclawski 1997] to model the average and peak characteristics of flow f_j as *arrival curve* $\alpha_j(t) = min(L_j + p_j t, \sigma_j + \rho_j t)$ in which L_j is the maximum transfer size, p_j the peak rate $(p_j \geq \rho_j)$, σ_j the burstiness $(\sigma_j \geq L_j)$, and ρ_j the average (sustainable) rate. We denote it as $f_j \propto (L_j, p_j, \sigma_j, \rho_j)$. As shown in Figure 1, $\theta_j = (\sigma_j - L_j)/(p_j - \rho_j)$ and $\alpha_j(t) = L_j + p_j t$ if $t \leq \theta_j$; $\alpha_j(t) = \sigma_j + \rho_j t$, otherwise.

In this paper, we also consider a class of curves, namely pseudoaffine curves [Lenzini et al. 2006], which is a multiple affine curve shifted to the right and given by $\beta = \delta_T \otimes [\otimes_{1 \leq x \leq n} \gamma_{\sigma_x, \rho_x}]$. In fact, a pseudoaffine curve represents the service received by single flows in tandems of FIFO multiplexing rate-latency nodes. Due to concave affine curves, it can be rewritten as $\beta = \delta_T \otimes [\wedge_{1 \leq x \leq n} \gamma_{\sigma_x, \rho_x}]$, where the non-negative term T is denoted as *offset*, and the affine curves between square brackets as leaky-bucket stages. It is clear that a rate-latency service curve is in fact pseudoaffine, since it can be expressed as $\beta = \delta_T \otimes \gamma_{0,R}$.

Given arrival curve α and service curve β, the delay is bounded by the horizontal deviation between the arrival and service curves.

4. SYSTEM MODEL AND NOTATIONS

As depicted in Figure 2, we consider an NoC architecture in which every node contains a router and a core which performs its own computational, storage or I/O processing functionality, and is equipped with a Network Interface (NI). As you can see in the figure, buffers are arranged to construct VCs in each input channel. To characterize flows based on their defined TSPEC, we assume unbuffered leaky bucket controllers (regulators) which do not buffer the packets, but stall the traffic producers or IPs [Jafari et al. 2010].

Assumptions in this work are listed as follows:

— The NoC architecture can have different topologies.
— Packets have fixed length and traverse the network in a best-effort fashion with virtual-cut-through switching technique using a deadlock-free deterministic routing.
— Routers have only input buffers and VCs.
— Buffers are bounded and the network is lossless.
— The router can have multiple VCs per in-port. VC allocation is deterministic and each VC receives an aggregate service.
— All traffic is the part of TSPEC flows $f = TSPEC(L, p, \sigma, \rho)$ at the entry into the network.
— In each node that guarantees to serve the flow a pseudo affine service curve $\beta = \delta_T \otimes \gamma_{\sigma_x, \rho_x}$, it is assumed that $\rho \leq \rho_x$ and $p \geq \rho_x$.

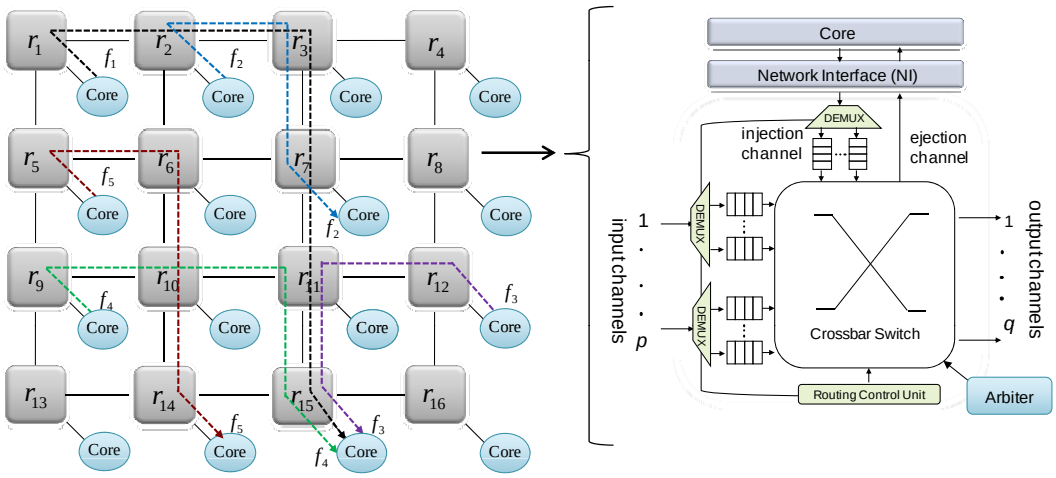

Fig. 2. An example of an NoC with 16 nodes and 5 flows along with the structure of a single node.

— Flows are classified into a pre-specified number of aggregates.
— Traffic of each aggregate is buffered and transmitted in the FIFO order, denoted as FIFO multiplexing.
— Different aggregates are buffered separately and each aggregate is guaranteed a rate-latency service curve.
— We use a concrete policy, in this case, round-robin arbitration, to support the assumption on rate-latency service curve. Indeed, it can use some other arbitration policies as well. We also assume a fixed word length of L_w in all of flows.
— The peak rate is limited by the hardware. It is always 1 *flit*/*cycle*.

NoC designers can obtain per flow end-to-end delay bound in NoC architectures by the proposed method in this paper under the mentioned assumptions.

Most of assumptions in this paper have been widely used by some previous models [Qian et al. 2009; Jafari et al. 2010]. The system model in this paper is more general than the mentioned models [Qian et al. 2009; Jafari et al. 2010] because they consider a Constant Bit Rate (CBR) flow in NoCs, defined by (σ, ρ) which is a special case of TSPEC. Furthermore, we have relaxed a significant limitation of the previous analytical model [Jafari et al. 2010] which presumes the number of VCs for each PC is the same as the number of flows passing through that channel.

We use an example depicted in Figure 2 to explain terminology used in the paper. The figure shows a network with 16 nodes numbered from $1, 2, ..., 16$ connected by links. There are 5 flows in the example denoted as $f_1, ..., f_5$. Multiple flows share the same buffer and channel in the router are scheduled as a flow called aggregate flow. For instance, $f_{\{1,2\}}$ in router 3 is an aggregate flow. A *tagged flow* is the flow that we shall derive its delay bound and other flows that share resources with the tagged flow are *contention flows*. In this example, f_1 is the tagged flow, and f_2, f_3, and f_4 are contention flows. Notations in the paper are listed in Table I.

We use sub-index "(f_i, r_j)" for notations to indicate that they are related to flow f_i in router r_j. For example, $\alpha_{(f_1,r_2)}$ denotes the arrival curve of flow f_1 in router r_2. We also employ sub-index "(s_i, r_j)" to state notations are related to f_{s_i} in router r_j. In this case, f_{s_i} can be one flow or an aggregate flow. For instance, $\beta_{(\{1,2,3\},r_2)}$ indicates the service curve of aggregate flow $f_{\{1,2,3\}}$ in router r_2.

Table I. The list of notations

f_i	Flow i
α_i	The arrival curve of f_i
α_i^*	The output arrival curve of f_i
L_i	The maximum transfer size of f_i *(flits)*
p_i	The peak rate of f_i *(flits/cycle)*
σ_i	The burstiness of f_i *(flits)*
ρ_i	The average rate of f_i *(flits/cycle)*
$Src(i)$	The source node of f_i
r_j	Router j
β_j	The service curve of r_j
R	The minimum service rate in a rate-latency service curve
T^l	The maximum processing latency of the arbiter in the router *(cycles)*
T^{HoL}	The maximum waiting time in the FIFO queue of the router *(cycles)*
T^{Total}	The total processing delay which comes from contention flows and equals to the sum of T^l and T^{HoL} *(cycles)*
D_{router}	Time spent for packet routing decision *(cycles)*
L_w	The word length in the flow *(flits)*
C	The channel capacity *(flits/cycle)*
ρ_x	The minimum service rate in a pseudo affine service curve
CF_t	The set of contention flows of tagged flow f_t in the network
s_i	The set of joint flows in an aggregate flow (when the number of elements of s_i is equal to 1, there is only a single flow)
f_{s_i}	An aggregate flow of s_i
$\lvert s_i \rvert$	The cardinality of set s_i, which is a measure of the "number of elements of the set"
$S = \{s_i\}$	A set of s_i's in a tandem of routers
s^m	A set which has the maximum cardinality between the sets in S. $s^m = \{s_x \lvert \lvert s_x \rvert = max(\lvert s_i \rvert); \forall s_i \in S\}$
f_{s^m}	The flow related to s^m
r^m	The router related to s^m
β^m	The service curve related to s^m
$F^B_{(s_i,r_j)}$	The set of flows which share the same buffer in router r_j with flow f_{s_i}
$\left\lvert V_{(s_i,r_j)} \right\rvert$	The number of virtual channels that passing flows from them share the same channel of router r_j with flow f_{s_i}
$F_{(VC_k,PC_i,r_j)}$	The set of flows passing through VC_k in physical channel PC_i of router r_j

5. PROPOSED THEOREMS

In this section, we review the earlier proposed theorems [Jafari et al. 2011; Jafari et al. 2012] which are required for analyzing performance of VBR flows in a FIFO multiplexing network.

We first represent a theorem for computing delay bound as follows.

Theorem 1. *(Delay Bound) Let β be a pseudo affine curve, with offset T and n leaky-bucket stage γ_{σ_x,ρ_x}, $\quad 1 \leq x \leq n$, this means we have:*

$$\beta = \delta_T \otimes [\otimes_{1\leq x\leq n}\gamma_{\sigma_x,\rho_x}] = \delta_T \otimes [\wedge_{1\leq x\leq n}\gamma_{\sigma_x,\rho_x}]$$

and let $\alpha = min(L + pt, \sigma + \rho t) = \gamma_{L,p} \wedge \gamma_{\sigma,\rho}$. If $\rho^*_\beta \geq \rho$ ($\rho^*_\beta = min_{1\leq x\leq n}\rho_x$), then the maximum delay for the flow is bounded by

$$h(\alpha, \beta) = T + \left[\vee_{1\leq x\leq n} \frac{L - \sigma_x + \theta\,(p - \rho_x)^+}{\rho_x}\right]^+ \tag{1}$$

PROOF. We have proved it in [Jafari et al. 2012]. See Appendix A.

In the rest of the paper, we apply Theorem 1 on the end-to-end ESC to calculate LUDB for a tagged flow. Due to our proposed method in Section 6, to obtain the end-to-end ESC, we should able to subtract contention flows from a service curve. To this end, we propose Proposition 1 and Theorem 2. In Proposition 1, we derive ESC with FIFO multiplexing where service curve is a pseudo affine curve. We then use Corollary 1 which is an immediate consequence of Proposition 1 to propose Theorem 2. This theorem is employed for deriving ESC in the underlying system model.

In Proposition 1 and Theorem 2, we obtain ESC with FIFO multiplexing under different assumptions.

Proposition 1. *(Equivalent Service Curve) Let β be a pseudo affine curve, with offset T and n leaky-bucket stage $\gamma_{\sigma_x,\rho_x}, \quad 1 \leq x \leq n$, this means we have:*

$$\beta = \delta_T \otimes [\otimes_{1\leq x\leq n}\gamma_{\sigma_x,\rho_x}] = \delta_T \otimes [\wedge_{1\leq x\leq n}\gamma_{\sigma_x,\rho_x}]$$

and let $\alpha = min(L + pt, \sigma + \rho t) = \gamma_{L,p} \wedge \gamma_{\sigma,\rho}$. If $\rho^*_\beta \geq \rho$ ($\rho^*_\beta = min_{1\leq x\leq n}\rho_x$) and $p \geq \rho^\circ_\beta$ ($\rho^\circ_\beta = max_{1\leq x\leq n}\rho_x$), then the ESC obtained by subtracting arrival curve α, $\{\beta^{eq}(\alpha,\tau), \quad \tau = h(\alpha,\beta)\} \equiv \beta^{eq}(\alpha)$, with

$$\beta^{eq}(\alpha) = \delta_{T+\vee_{1\leq i\leq n}\left[\frac{L-\sigma_i+\theta(p-\rho_i)^+}{\rho_i}\right]^+ +\theta} \otimes \Big[\otimes_{1\leq x\leq n}\Big[\gamma_{\rho_x\left\{\vee_{1\leq i\leq n}\left[\frac{L-\sigma_i+\theta(p-\rho_i)^+}{\rho_i}\right]^+ - \frac{\sigma-\sigma_x-(\rho_x-\rho)\theta}{\rho_x}\right\},\rho_x-\rho}\Big]\Big] \tag{2}$$

PROOF. We have proved it in [Jafari et al. 2012]. See Appendix B.

The following corollary is an immediate consequence.

Corollary 1. *Let $\beta = \delta_T \otimes \gamma_{\sigma_x,\rho_x}$ be a pseudo affine curve, with offset T and one leaky-bucket stage γ_{σ_x,ρ_x}, and let $\alpha = min(L + pt, \sigma + \rho t) = \gamma_{L,p} \wedge \gamma_{\sigma,\rho}$. If $\rho_x \geq \rho$ and $p \geq \rho_x$, then the ESC obtained by subtracting arrival curve α, β^{eq}*

$$\beta^{eq} = \delta_{T+\left[\frac{L-\sigma_x+\theta(p-\rho_x)^+}{\rho_x}\right]^+ +\theta} \otimes \gamma_{0,\rho_x-\rho} \tag{3}$$

PROOF. We can easily obtain this corollary by applying Proposition 1 for service curve β when $n = 1$.

We can specifically capitalize on Corollary 1 to obtain a parametric expression for the ESC of a tagged flow passing through a rate-latency node. We assume the number of flows passing through this node is $K+1$. Therefore, for computing equivalent service curve for the tagged flow, we should subtract the arrival curves of other K flows. It can be calculated by iteratively applying Corollary 1 for K times. Without loss of generality, we presume that the tagged flow is flow $K + 1$. We now present following theorem:

Theorem 2. *(Equivalent Service Curve for Rate-Latency Service Curve with $K + 1$ Flows) Consider one node with a rate-latency service curve $\beta_{R,T} = \delta_T \otimes \gamma_{0,R}$. Let $\alpha_i = min(L_i+p_it, \sigma_i+\rho_it) = \gamma_{L_i,p_i} \wedge \gamma_{\sigma_i,\rho_i}$ be arrival curve of flow i and $p_i \geq R-\sum_{(j=1;j\neq i)}^{K+1}\rho_j$, where $1 \leq i \leq K + 1$ and $K + 1$ is the number of flows passing through that node as*

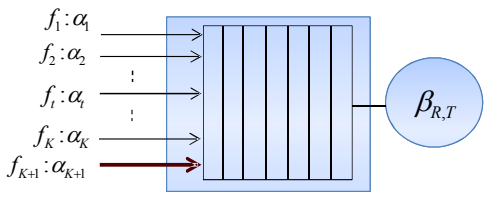

Fig. 3. Computation of equivalent service curve for flow $K+1$ in a rate-latency node.

shown in Figure 3. Assuming $\sum_{j=1}^{K+1}\rho_j \leq link\ rate$*, where* C *is the link rate, the ESC for flow* $K+1$ *in the node, obtained by subtracting* K *arrival curves, is:*

$$\beta_{K+1}^{eq} = \delta_{T+\sum_{i=1}^{K}\left(\left[\frac{L_i+\theta_i(p_i-R+\sum_{j=1}^{i-1}\rho_j)^+}{R-\sum_{j=1}^{i-1}\rho_j}\right]^+ +\theta_i\right)} \otimes \gamma_{0,R-\sum_{j=1}^{K}\rho_j} \tag{4}$$

PROOF. We have proved it in [Jafari et al. 2012]. See Appendix C.

Theorem 3 states how output arrival curve of a VBR flow in a FIFO multiplexing node can be calculated.

Theorem 3. *(Output Arrival Curve with FIFO) Consider a VBR flow, with TSPEC* (L,p,ρ,σ)*, served in a node that guarantees to the flow a pseudo affine service curve* $\beta=\delta_T\otimes\gamma_{\sigma_x,\rho_x}$*. The output arrival curve* α^* *given by:*

$$\alpha^* = \begin{cases} \theta > T & \gamma_{(p\wedge\rho_x)T+\theta(p-\rho_x)^+ +L-\sigma_x,p\wedge\rho_x} \\ & \wedge\gamma_{\sigma-\sigma_x+\rho T,\rho} \\ \theta \leq T & \gamma_{\sigma-\sigma_x+\rho T,\rho} \end{cases} \tag{5}$$

PROOF. We have proved it in [Jafari et al. 2011]. See Appendix D.

We apply this theorem to calculate internal output arrival curves. For instance, in Section 6.2, we obtain the output arrival curve of a crossed flow when it is split into two nested flows.

6. FORMAL METHOD FOR LUDB DERIVATION

We have presented and proved the required theorems for deriving LUDB for VBR flows in on-chip networks based on aggregate scheduling with multiple virtual channels. As mentioned before, to calculate LUDB per flow, we should first obtain the end-to-end ESC which the tandem of routers provides to the flow. For calculating the end-to-end ESC, we propose two following steps:

- **Step 1:** Intra-router ESC
- **Step 2:** Inter-router ESC

In the first step, we consider resource sharing scenarios in the routers and then build analysis models for different resource sharing components. Based on these models, we can derive the intra-router ESC for an individual flow. In the second step, we consider the contention which a flow may experience along its routing path. Therefore, we present recursive algorithm End-to-End ESC to classify and analyze resource sharing models and flow interference patterns. Based on this algorithm, we can derive the end-to-end ESC for a tagged flow passing through the tandem of routers.

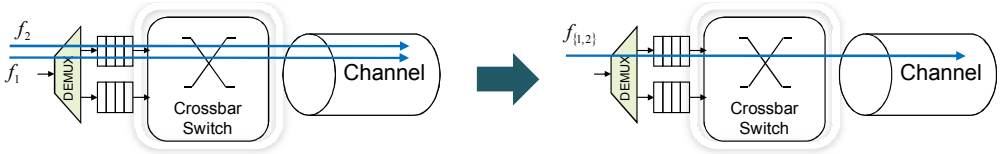

Fig. 4. An example of channel&buffer sharing.

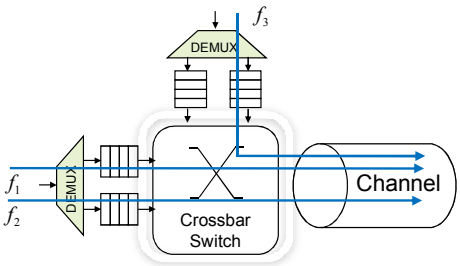

Fig. 5. An example of a channel sharing three flows.

6.1. Step1: Intra-router ESC

To compute intra-router ESC for a tagged flow, it is necessary to investigate resource sharing. At each router, we identify three types of resource sharing, namely, *channel sharing*, *buffer sharing*, and *channel&buffer sharing*. *Channel sharing* means that multiple flows share the same out-port and thus the output channel bandwidth. *Buffer sharing* means that multiple flows share the same buffer but not channel. In *channel&buffer sharing*, multiple flows share both buffers and channels. They are scheduled as a flow called aggregate flow.

6.1.1. Channel&Buffer Sharing. Figure 4 depicts an example of flows sharing both channel and buffer in the router. As shown in the figure, we consider these flows as an aggregate flow. When an aggregate flow includes the tagged flow, it is called as *tagged aggregate flow*. In this respect, we calculate intra-router ESC for the tagged aggregate flow in the router instead of the tagged flow. In Section 6.2, we show how ESC of the tagged flow is extracted from the ESC of the tagged aggregate flow by removing contention flows one by one. For simplicity, in the rest of the paper, "tagged flow" refers to both tagged flow and tagged aggregate flow.

6.1.2. Channel Sharing. Figure 5 depicts a channel shared between three flows f_1, f_2, and f_3. Since the arbitration policy determines how much the flows influence each other, it has to be known. We assume that, while serving multiple flows, the routers employ round robin scheduling to share the channel bandwidth. Assuming a fixed word length of L_w in all of flows, round robin arbitration means that each flow f_{s_i} in router r_j gets at least a $\frac{C}{|V_{(s_i,r_j)}|}$ of the channel bandwidth, where C is the channel capacity and $|V_{(s_i,r_j)}|$ the number of virtual channels that passing flows from them share the same channel of router r_j with flow f_{s_i}. A flow may get more if other flows use less, but we now know a worst-case lower bound on the bandwidth. Round robin arbitration has good isolation properties because the minimum bandwidth for each flow does not depend on properties of the other flows.

Since network calculus uses the abstraction of service curve to model a network element processing traffic flows [Le Boudec et al. 2004], we can also model a round

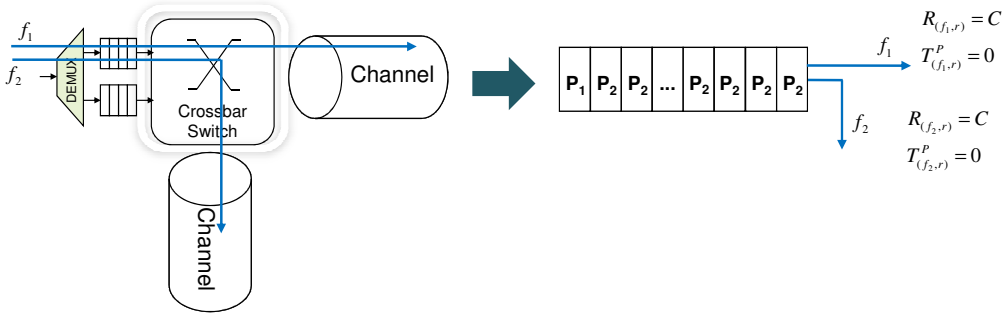

Fig. 6. An example of a buffer sharing two flows.

robin arbiter in router r_j for flow f_{s_i} as a rate-latency server [Gebali et al. 2009] that its function is as $\beta_{(s_i,r_j)} = R_{(s_i,r_j)}(t - T^l_{(s_i,r_j)})^+$, where $R_{(s_i,r_j)}$ is the minimum service rate and $T^l_{(s_i,r_j)}$ is the maximum processing latency of the arbiter in router r_j for flow f_{s_i}. $R_{(s_i,r_j)}$ and $T^l_{(s_i,r_j)}$ are defined as follows:

$$R_{(s_i,r_j)} = \frac{C}{\left|V_{(s_i,r_j)}\right|} \tag{6}$$

$$T^l_{(s_i,r_j)} = \left(\left|V_{(s_i,r_j)}\right| - 1\right) \times \left(\frac{L_w}{C} + D_{router}\right) \tag{7}$$

where D_{router} is the delay for packet routing decision in a router.

As mentioned in Section 5, a rate-latency service curve is in fact a pseudoaffine. Therefore, $\beta_{(s_i,r_j)}$ can be expressed as $\delta_{\left(\left|V_{(s_i,r_j)}\right|-1\right)\times\left(\frac{L_w}{C}+D_{router}\right)} \otimes \gamma_{0,\frac{C}{\left|V_{(s_i,r_j)}\right|}}$. Assuming f_1 is the tagged flow in the example, $\beta_{(f_1,r)} = \delta_{2\times\left(\frac{L_w}{C}+D_{router}\right)} \otimes \gamma_{0,\frac{C}{3}}$.

6.1.3. Buffer Sharing. Figure 6 shows a buffer shared between two flows f_1 and f_2. In this type of sharing, in addition to maximum processing latency for link sharing, T^l, we introduce the head-of-Line delay for a tagged flow as below:

Head-of-Line delay (HoL): Given a flow comes at time t in a router, the maximum waiting time in the FIFO queue would be in time $t + T^{HoL}$.

Therefore, the total processing delay which comes from contention flows for tagged flow f_{s_i} in router r_j, $T^{Total}_{(s_i,r_j)}$, is equal to $T^l + T^{HoL}$

We assume f_1 in Figure 6 is the tagged flow. According to Equation (7), $T^l_{(f_1,r)} = 0$. From the figure, it is clear that $T^{HoL}_{(f_1,r)}$ is equal to the maximum delay for passing packets of flow f_2 in the buffer. According to the network calculus theory [Le Boudec et al. 2004], the maximum delay for flow f_j is bounded by Equation (8).

$$\bar{D}_{(f_j,r)} = T^l_{(f_j,r)} + \frac{L_j + \theta_j(p_j - R_{(f_j,r)})^+}{R(f_j,r)} \tag{8}$$

Therefore, we formulate $T^{HoL}_{(f_1,r)}$ as follows:

$$T^{HoL}_{(f_1,r)} = T^l_{(f_2,r)} - \theta_2 + \frac{L_2 + \theta_2 p_2}{R_{(f_2,r)}} \tag{9}$$

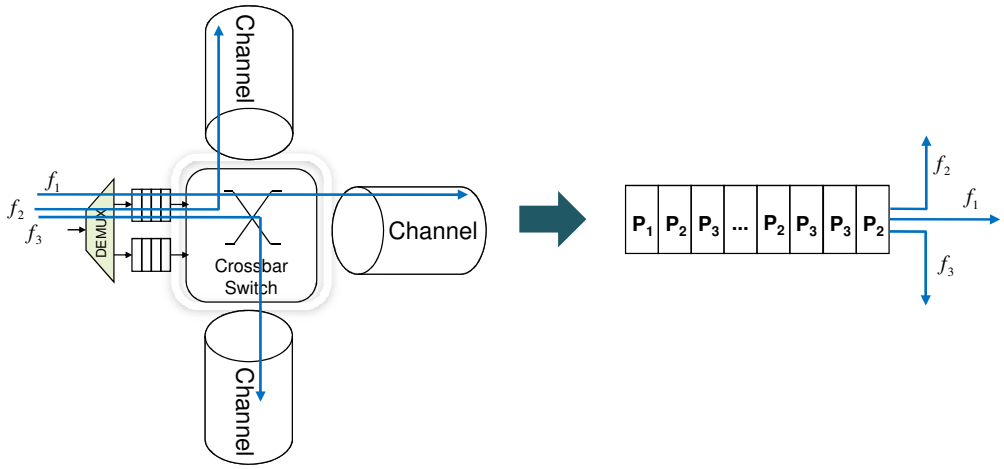

Fig. 7. An example of a buffer sharing three flows.

If there is more than one flow sharing the buffer with the tagged flow as shown in Figure 7, HoL delay for tagged flow f_{s_i} in router r_j is given by

$$T^{HoL}_{(s_i,r_j)} = \sum_{\forall f_c \in F^B_{(s_i,r_j)}} T^{HoL(f_c)}_{(s_i,r_j)} \tag{10}$$

where $F^B_{(s_i,r_j)}$ is the set of flows which share the same buffer in router r_j with tagged flow f_{s_i}. $T^{HoL(f_c)}_{(s_i,r_j)}$ is calculated as follows.

$$T^{HoL(f_c)}_{(s_i,r_j)} = T^{l}_{(f_c,r)} - \theta_c + \frac{L_c + \theta_c p_c}{R_{(f_c,r)}} \tag{11}$$

Therefore router r_j can serve flow f_{s_i} by curve $\beta_{(s_i,r_j)} = \delta_{T^{Total}_{(s_i,r_j)}} \otimes \gamma_{0,R_{(s_i,r_j)}}$, where $T^{Total}_{(s_i,r_j)} = T^{HoL}_{(s_i,r_j)} + T^{l}_{(s_i,r_j)}$ and $R_{(s_i,r_j)}$ is calculated by Equation (6).

We analyze the buffer space threshold for each VC based on traffic specifications of flows passing through that VC, and also interference between them. The buffer space threshold for virtual channel VC_k in physical channel PC_i of router r_j is given as below:

$$B_{(VC_k,PC_i,r_j)} = \sum_{\forall f_c \in F_{(VC_k,PC_i,r_j)}} \left(\sigma_c + \rho_c T^p_{(f_c,r_j)} + \left(\theta - T^p_{(f_c,r_j)}\right)^+ \left[\left(p_c - R_{(f_c,r_j)}\right)^+ - p_c + \rho_c \right] \right) \tag{12}$$

where $F_{(VC_k,PC_i,r_j)}$ is the set of flows passing through VC_k in physical channel PC_i of router r_j.

6.2. Step2: Inter-router ESC

We have analyzed and modeled three kinds of sharing to compute the intra-router ESC. After analyzing per-router resource sharing (intra-ESC), the effects of buffer sharing and channel sharing on tagged flow have been considered and we can view an

analysis model which keeps only channel&buffer sharing for tagged flow. This model is called *aggregate analysis model*. For example, suppose that a tagged flow f_1 traverses a tandem of routers, and is multiplexed with contention flows as depicted in Figure 8(a). After analyzing intra-router ESC, aggregate analysis model is shown as 8(b). In this model, $\beta_{(s_i,r_j)}$ indicates that the service curve is related to flow f_{s_i} in router r_j. For instance, $\beta_{(\{1,2\},r_3)}$ is the service curve of flow $f_{\{1,2\}}$ in router r_3. $f_{\{1,2\}}$ indicates to a flow aggregated by flows f_1 and f_2. A set of s_i's in a tandem of routers is denoted as $S = \{s_i\}$. For example, in Figure 8(b), $S = \{\{1\}, \{1,2,3\}, \{1,2\}, \{1\}\}$.

Now, we consider aggregate analysis model to recognize interference patterns and remove contention flows one by one. A tagged flow directly contends with contention flows. Also, contention flows may contend with each other and then contend with the tagged flow again. To consider inter-ESC in the aggregate analysis model, we decompose a complex contention scenario to two basic contention patterns, namely, *Nested* and *Crossed*.Figures 8, 9, 10, and 11 illustrate examples of different kinds of nested contentions and an example of crossed contention is shown in Figure 12. In the following, we will describe these examples with more details.

We use the algebra of sets to recognize the contention scenarios. To facilitate our discussion, we define convenient notations by the example in Figure 8(b). In the example, the tandem of servers is as $\{\beta_{(\{1\},r_1)}, \beta_{(\{1,2,3\},r_2)}, \beta_{(\{1,2\},r_3)}, \beta_{(\{1\},r_4)}\}$ and $S = \{s_i\} = \{\{1\}, \{1,2,3\}, \{1,2\}, \{1\}\}$. We define $s^m = \{s_x \big| |s_x| = max\left(|s_i|\right); \forall s_i \in S\}$, where $|s_x|$ is the cardinality (the number of elements) of set s_x. The service curve, flow, and router related to s^m are denoted as f_{s^m}, β^m, and r^m, respectively. Thus, in Figure 8(b), $s^m = \{1,2,3\}$, $f_{s^m} = f_{\{1,2,3\}}$, $r^m = r_2$, and $\beta^m = \beta_{(\{1,2,3\},r_2)}$.

We denote the service curve placed before β^m on the aggregate analysis model by β^{Prev} and related aggregate flow and router as $f_{s^{Prev}}$ and r^{Prev}, respectively. Notation β^{Next} indicates to the service curve placed after β^m, as well. Therefore, due to $\beta^m = \beta_{(\{1,2,3\},r_2)}$ in Figure 8(b), $\beta^{Prev} = \beta_{(\{1\},r_1)}$, $s^{Prev} = \{1\}$, $f_{s^{Prev}} = f_{\{1\}}$, $r^{Prev} = r_1$, $\beta^{Next} = \beta_{(\{1,2\},r_3)}$, $s^{Next} = \{1,2\}$, $f_{s^{Next}} = f_{\{1,2\}}$, $r^{Next} = r_3$.

Contention recognition procedure in an aggregate analysis model can be generalized as following steps:

(1) Find $s^m = \{s_x \big| |s_x| = max\left(|s_i|\right); \forall s_i \in S\}$.
(2) if $s^{Prev} \subset s^{Next}$ then the contention is *Nested*; –Remove $f_{s^m - (s^m \cap s^{Prev})}$ from β^m.
(3) if $s^{Next} \subset s^{Prev}$ then the contention is *Nested*; –Remove $f_{s^m - (s^m \cap s^{Next})}$ from β^m
(4) else
  (a) if $s^{Prev} \subset s^m$ and $s^{Next} \not\subset s^m$ then the contention is *Nested*;
    — Remove $f_{s^m - (s^m \cap s^{Prev})}$ from β^m.
  (b) if $s^{Next} \subset s^m$ and $s^{Prev} \not\subset s^m$ then the contention is *Nested*;
    — Remove $f_{s^m - (s^m \cap s^{Next})}$ from β^m
  (c) else, it is *Crossed*.
    — The problem is strictly transformed to the combination of two nested flows

To remove a contention flow from a service curve and derive the new service curve from that, we apply the proposed corollary 1 in Section 5.

When s^m is not unique, each of them can be selected. In this paper, we choose the first one from the left side in the aggregate analysis network.

In the case of $s^{Next} = s^{Prev}$, there are two possibilities:

(1) $s^{Next} = s^{Prev} \neq s^m$: Since $s^{Next} \subset s^{Prev}$ and $s^{Prev} \subset s^{Next}$, the contention is nested as previously described in contention recognition steps.

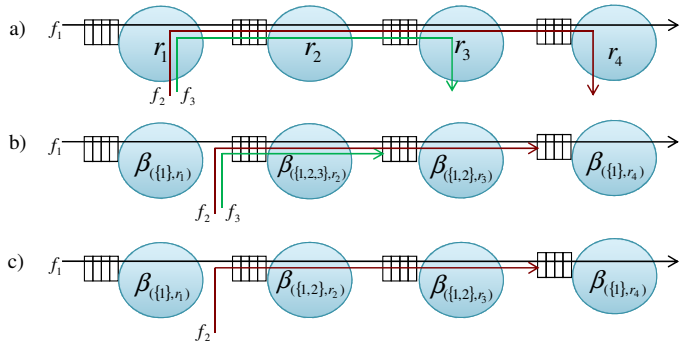

Fig. 8. Analysis for the first type of nested flows.

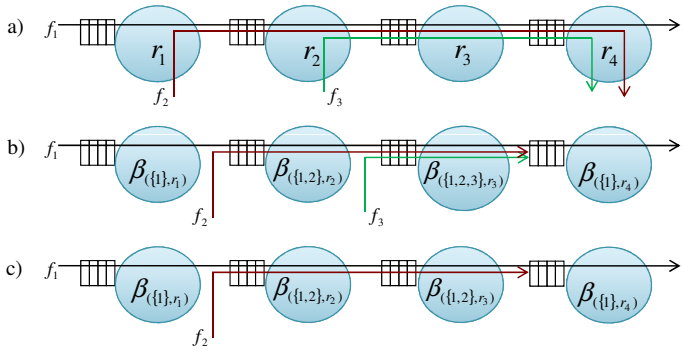

Fig. 9. Analysis for the second type of nested flows.

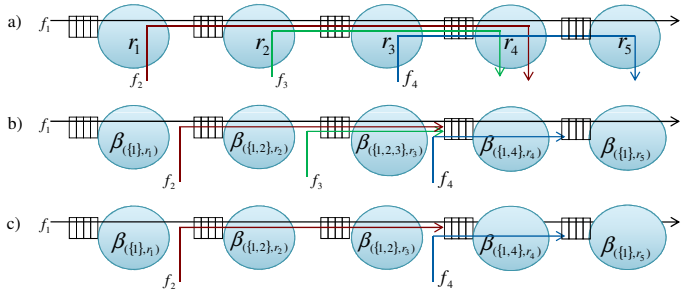

Fig. 10. Analysis for the third type of nested flows.

(2) $s^{Next} = s^{Prev} = s^m$: In this case, three nodes s^{Next}, s^{Prev}, and s^m should be combined as a single server by applying the theorem of *concatenation of network elements* [Le Boudec et al. 2004]. It will be discussed in Section 6.3.

In the following, we give examples for various contention patterns.

6.2.1. Nested Flows. Four different types of nested contention are exemplified as Figures 8, 9, 10, and 11. Flow f_3 is nested in flow f_2 in Figures 8, 9, and 10 and it is also nested in flow f_4 in Figure 11.

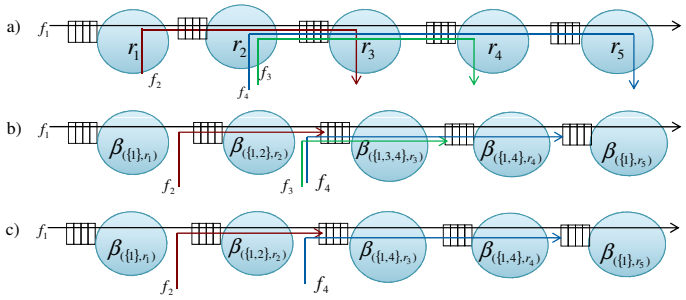

Fig. 11. Analysis for the fourth type of nested flows.

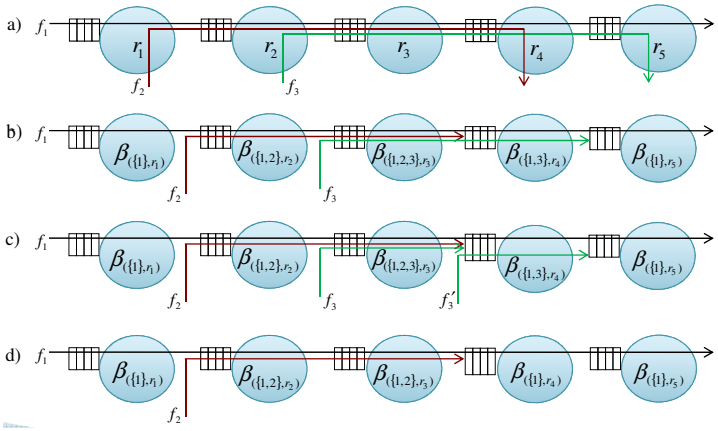

Fig. 12. Analysis for crossed Flows.

— Figure 8(b) shows the first type of nested flows after applying intra-ESC, in which $s^m = \{1,2,3\}$, $s^{Prev} = \{1\}$, and $s^{Next} = \{1,2\}$. In this case, $s^{Prev} \subset s^{Next}$ and due to step 2 of contention recognition procedure, we remove flow $f_{\{1,2,3\}-(\{1,2,3\}\cap\{1,2\})} = f_{\{3\}}$ from $\beta_{(\{1,2,3\},r_2)}$ and derive $\beta_{(\{1,2\},r_2)}$, as depicted in Figure 8(c).
— The second type of nested flows in the aggregate analysis model is depicted in Figure 9. Due to Figure 9(b), $s^m = \{1,2,3\}$, $s^{Prev} = \{1,2\}$, and $s^{Next} = \{1\}$. In this case, $s^{Next} \subset s^{Prev}$ and flow $f_{\{1,2,3\}-(\{1,2,3\}\cap\{1,2\})} = f_{\{3\}}$ is eliminated from $\beta_{(\{1,2,3\},r_3)}$ regarding step 3 of contention recognition procedure. Figure 9(c) shows aggregate analysis model after removing f_3.
— Figure 10 shows an example of the third type of nested contention. Based on aggregate analysis model depicted in Figure 10(b), $s^m = \{1,2,3\}$, $s^{Prev} = \{1,2\}$, and $s^{Next} = \{1,4\}$. Since $s^{Next} \not\subset s^{Prev}$, $s^{Prev} \not\subset s^{Next}$, $s^{Prev} \subset s^m$, and $s^{Next} \not\subset s^m$, due to step 4.a) of contention recognition procedure, the case is nested contention and flow $f_{\{1,2,3\}-(\{1,2,3\}\cap\{1,2\})} = f_{\{3\}}$ is removed from $\beta_{(\{1,2,3\},r_3)}$, as shown in Figure 10(c).
— Figure 11 shows a type of nested contention related to step 4.b) of contention recognition procedure. Due to Figure 11(b), $s^m = \{1,3,4\}$, $s^{Prev} = \{1,2\}$, and $s^{Next} = \{1,4\}$. Since $s^{Next} \not\subset s^{Prev}$, $s^{Prev} \not\subset s^{Next}$, $s^{Next} \subset s^m$, and $s^{Prev} \not\subset s^m$, it is a nested contention and Figure 11(c) shows that flow $f_{\{1,3,4\}-(\{1,3,4\}\cap\{1,4\})} = f_{\{3\}}$ is eliminated from $\beta_{(\{1,3,4\},r_3)}$.

6.2.2. Crossed Flows. Figure 12 shows contention flow f_2 crossed with f_3. Regarding Figure 12(b), $s^m = \{1,2,3\}$, $s^{Prev} = \{1,2\}$, and $s^{Next} = \{1,3\}$. Since s^{Prev} is not a

subset of s^{Next}, and vice versa and also both of them are a subset of s^m, due to step 4.c) of contention recognition procedure, this case is a crossed contention. There are two cross points, one between r_2 and r_3 and the other between r_3 and r_4. We cut f_3 at the second cross point, i.e., at the ingress of r_4, f_3 will be split into two flows, f_3 and $\acute{f}_3$, as shown in Figure 12(c). Then the problem is strictly transformed to the combination of nested flows such that f_3 is nested in flow f_2 and $\acute{f}_3$ in f_1. It is clear that the arrival curve $\alpha_{(f_3,r_3)}$ equals to α_3 and the arrival curve $\alpha_{(\acute{f}_3,r_3)}$ equals to $\alpha^*_{(f_3,r_3)}$. To compute $\alpha^*_{(f_3,r_3)}$, we need to get the ESC of r_3 for f_3, $\beta_{(f_3,r_3)}$. Then, we calculate the output arrival curve of f_3 as $\alpha^*_{(f_3,r_3)} = \alpha_{(f_3,r_3)} \oslash \beta_{(f_3,r_3)}$ by applying the proposed Theorem 3 in Section 5. Now, nested flows f_3 and $\acute{f}_3$ can be removed from the tandem as shown in Figure 12(d).

6.3. End-to-end ESC

We show a high-level analysis flow for deriving the end-to-end ESC in Figure 13 and then present end-to-end ESC algorithm along with more details and one example.

To calculate end-to-end ESC, we first obtain intra-router ESC for the tagged flow in each router. Then we use the theorem of *concatenation of network elements* [Le Boudec et al. 2004] to model nodes sequentially connected and each is offering a service curve on the same aggregate flows $\beta_{(s_i,r_j)}$, $j = 1, 2, ..., n$ as a single server as follows:

$$\beta_{(s_i,r_{1,2,...,n})} = \beta_{(s_i,r_1)} \otimes \beta_{(s_i,r_2)} \otimes ... \otimes \beta_{(s_i,r_n)}$$

In the next step, we calculate inter-router ESC by applying contention recognition stages and removing contention flows as described in Section 6.2. After that, the concatenation theorem is applied again to find more equivalent servers and reduce the number of service curves. For instance, after removing contention flow f_3 in Figure 8(c), the service curve of sub-tandem $\{r_2, r_3\}$ for aggregate flow $f_{\{1,2\}}$ is computed as $\beta_{(\{1,2\},r_{2,3})} = \beta_{(\{1,2\},r_2)} \otimes \beta_{(\{1,2\},r_3)}$. If we repeat contention recognition steps, the next contention flow is f_2 nested in f_1. If we similarly remove it from $\beta_{(\{1,2\},r_{2,3})}$ and calculate convolution $\beta_{(\{1\},r_{1,2,3})} = \beta_{(\{1\},r_1)} \otimes \beta_{(\{1\},r_{2,3})}$, the end-to-end ESC of tagged flow f_1 is obtained.

Algorithm 1 explains the procedure of calculating end-to-end ESC with more details.

— *Joining node*: In Lines $2-8$, the algorithm checks if source node of a contention flow f_i is one of the nodes along the tagged flow's path or not. If it is not, this means that we should calculate input TSPEC of the contention flow f_i in the point joined to the tagged flow's route (point A in Figure 14 when f_1 is the tagged flow). We obtain this point by function $JoiningPoint(f_i)$ and call it *joining node*.

We give an example in Figure 15 to show how to derive an aggregate analysis model and obtain end-to-end ESC by following the proposed algorithm.

Assuming the tagged flow is f_1, line 1 of the algorithm finds CF_t which is $\{f_2, f_3, f_4\}$ in the example.

— **Loop 1 in the algorithm (Lines $2-8$):** In Lines $3-4$, the algorithm obtains *joining node* for each contention flow which its source node is not one of the nodes along the tandem. Then, end-to-end ESC of flow f_j from the source node to joining node has been derived by recursively calling $ESC(f_j, Src(j), joiningnode)$ in Line 5. Line 6 computes output arrival curve which is input arrival curve to the joining node and input TSPEC is extracted from that. In the example of Figure 15(a), all source nodes of contention flows are in the tagged flow's route and lines $4-6$ are skipped for them.

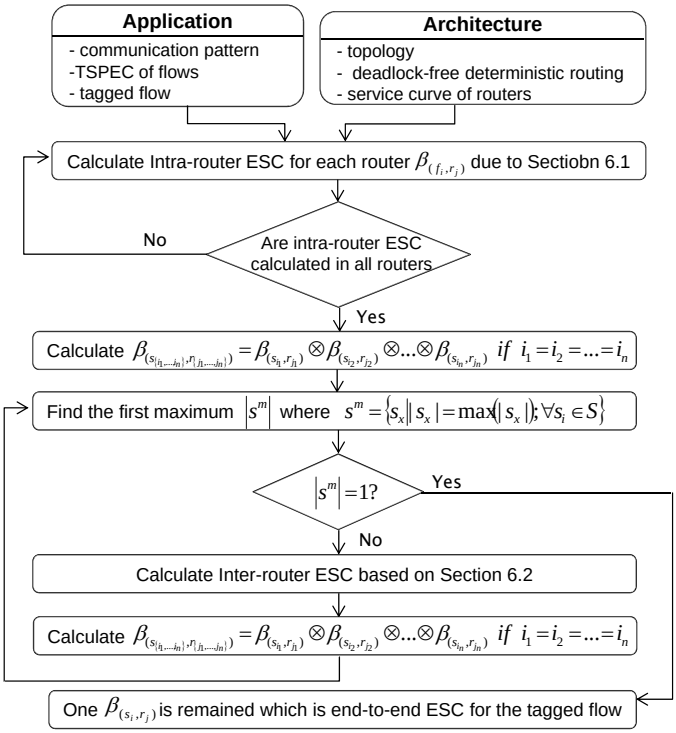

Fig. 13. End-to-end ESC analysis flow.

Line 9 obtains intra-router ESC for the tagged flow due to Section 6.1. Figure 15(b) shows the aggregate analysis model for the example. Due to line 10, $\beta_{(\{1,2\},r_{3,4})} = \beta_{(\{1,2\},r_3)} \otimes \beta_{(\{1,2\},r_4)}$. Figure 15(c) depicts the example in this step. Regarding line 11, $s^m = \{1,2,3\}$.

— **Loop 2 in the algorithm (Lines** $12-32$**):** In Lines $13-29$, we consider different contention scenarios along the route using the algebra of sets. In this step, we intend to remove contention flows one by one due to their effects on the tagged flow as

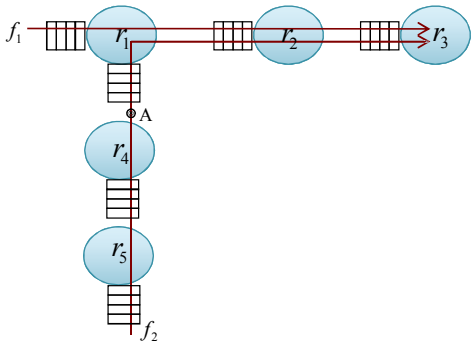

Fig. 14. The example of *joining point*.

Algorithm 1 end-to-end ESC

1: Find the set of contention flows of tagged flow f_t, denoted by CF_t
2: **for** $\forall j \in CF_t$ **do**
3: **if** $Src(j) \notin Path(t)$ **then**
4: Find $joiningnode = JoiningPoint(f_j)$
5: Calculate $X = ESC(f_j, Src(j), joiningnode)$
6: $\alpha_j = \alpha_j \oslash X$
7: **end if**
8: **end for**
9: Calculate intra-router ESC based on Section 6.1.
10: Calculate $\beta_{(s_{i_1},r_{j_1})} \otimes \beta_{(s_{i_2},r_{j_2})} \otimes ... \otimes \beta_{(s_{i_n},r_{j_n})}$ ***if*** $i_1 = i_2 = ... = i_n$.
11: Find $s^m = \{s_x || s_x| = max(|s_i|); \forall s_i \in S\}$.
12: **repeat**
13: **if** $s^{Prev} \subset s^{Next}$ **then**
14: Remove $f_{s^m - (s^m \cap s^{Next})}$ from β^m
15: **else if** $s^{Next} \subset s^{Prev}$ **then**
16: Remove $f_{s^m - (s^m \cap s^{Prev})}$ from β^m.
17: **else**
18: **if** $s^{Prev} \subset s^m$ and $s^{Next} \not\subset s^m$ **then**
19: Remove $f_{s^m - (s^m \cap s^{Prev})}$ from β^m
20: **else if** $s^{Next} \subset s^m$ and $s^{Prev} \not\subset s^m$ **then**
21: Remove $f_{s^m - (s^m \cap s^{Next})}$ from β^m.
22: **else**
23: Find $joiningnode = JoiningPoint(f_{(s^m - s^{Prev})})$.
24: Calculate $X = ESC(f_{(s^m - s^{Prev})}, joiningnode, r^{Next})$.
25: $\acute{\alpha}_{(s^m - s^{Prev})} = \alpha_{(s^m - s^{Prev})} \oslash X$
26: Remove $f_{(s^m - s^{Prev})}$ from β^m.
27: Remove $\acute{f}_{(s^m - s^{Prev})}$ from β^{Next}.
28: **end if**
29: **end if**
30: Calculate $\beta_{(s_{i_1},r_{j_1})} \otimes \beta_{(s_{i_2},r_{j_2})} \otimes ... \otimes \beta_{(s_{i_n},r_{j_n})}$ ***if*** $i_1 = i_2 = ... = i_n$.
31: Find s^m.
32: **until** $|s^m| \neq 1$
33: **return** end-to-end ESC for tagged flow f_t

mentioned in Section 6.2. Lines $13-21$ consider nested contentions and lines $22-28$ crossed one.

— **Nested contention in the example:** From Figure 15(c), $s^m = \{1,2,3\}$, $s^{Prev} = \{1\}$, and $s^{Next} = \{1,2\}$. Since $s^{Prev} \subset s^{Next}$, due to line 13, flow $f_{\{1,2,3\}-(\{1,2,3\}\cap\{1,2\})} = f_3$ is removed from $\beta_{(\{1,2,3\},r_2)}$ as shown in Figure 15(d).

Lines $30-31$ are the same as lines $10-11$ which calculate concatenation of the nodes on the same aggregate flows and then obtain new s^m, which result in $\beta_{(\{1,2\},r_{2,3,4})} = \beta_{(\{1,2\},r_2)} \otimes \beta_{(\{1,2\},r_{3,4})}$, and $s^m = \{1,2,4\}$ (Figure 15(e)).

— **Crossed contention in the example:** If we repeat contention recognition steps in Loop 2, the next contention in the example is crossed. From Figure 15(e), $s^m = \{1,2,4\}$, $s^{Prev} = \{1,2\}$, and $s^{Next} = \{1,4\}$. Since neither $s^{Prev} \subset s^{Next}$ nor $s^{Next} \subset s^{Prev}$ and also either $s^{Next} \subset s^m$ and $s^{Prev} \subset s^m$, it goes to the else part (lines $22-28$) of the algorithm. As shown in Figure 15(e), contention flow f_2 is crossed with

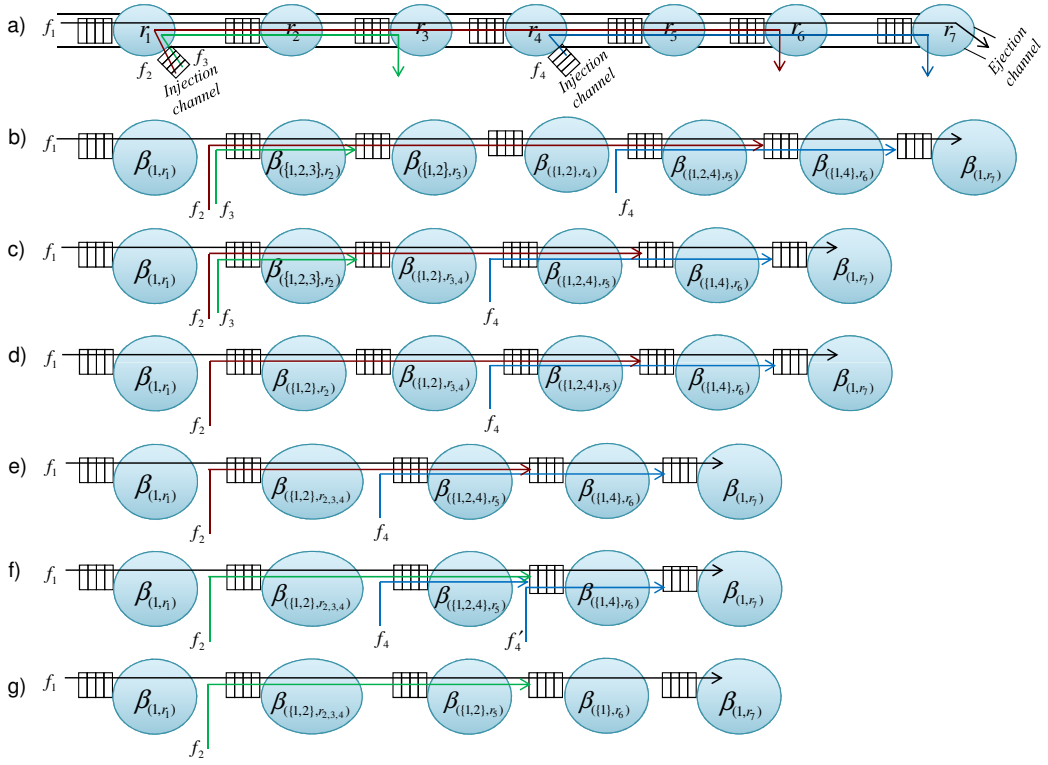

Fig. 15. An example of end-to-end ESC computation.

f_4. There are two cross points, one between $r_{2,3,4}$ and r_5 and the other between r_5 and r_6. Regarding the algorithm, we cut f_4 at the second cross point, i.e., at the ingress of r_6, f_4 will be split into two flows, f_4 and $\acute{f}_4$, as shown in Figure 15(f). Then, the problem is transformed to the combination of two nested scenarios. Apparently the arrival curve $\alpha_{\acute{f}_4}$ of $\acute{f}_4$ is equal to $\alpha^*_{f_4}$ of f_4. To compute $\alpha^*_{f_4}$, we need to get the ESC of f_4 from r_5 to r_6, which is derived regarding lines 23 and 24. Then, line 25 calculates output arrival curve $\alpha^*_{f_4}$ ($\alpha_{\acute{f}_4}$) by applying the proposed Theorem 3 in Section 5. Then, f_4 and $\acute{f}_4$ are removed from r_5 and r_6 due to lines 26 and 27, respectively, as shown in Figure 15(g).

Therefore, according to lines $30-31$, $\beta_{(\{1,2\},r_{2,3,4,5})} = \beta_{(\{1,2\},r_{2,3,4})} \otimes \beta_{(\{1,2\},r_5)}$, $\beta_{(\{1\},r_{6,7})} = \beta_{(\{1\},r_6)} \otimes \beta_{(\{1\},r_7)}$, and $s^m = \{1,2\}$. We similarly repeat contention recognition and convolution steps until $|s^m| \neq 1$. When $|s^m| = 1$, the end-to-end ESC of tagged flow f_1 is obtained.

6.4. LUDB Derivation

To compute the delay bound for a flow passing a series of nodes, one simple way is to calculate the summation of delay bounds at each node. However, this results in a loose total delay bound. To tighten the worst-case delay bound along the network, the end-to-end service curve of the flow is used as stated in corollary *Pay Bursts Only Once* [Le Boudec et al. 2004]. Hence, we first calculate the end-to-end ESC of the tagged flow based on the proposed algorithm and then obtain LUDB according to Theorem 1. We have implemented algorithms employed in our methodology.

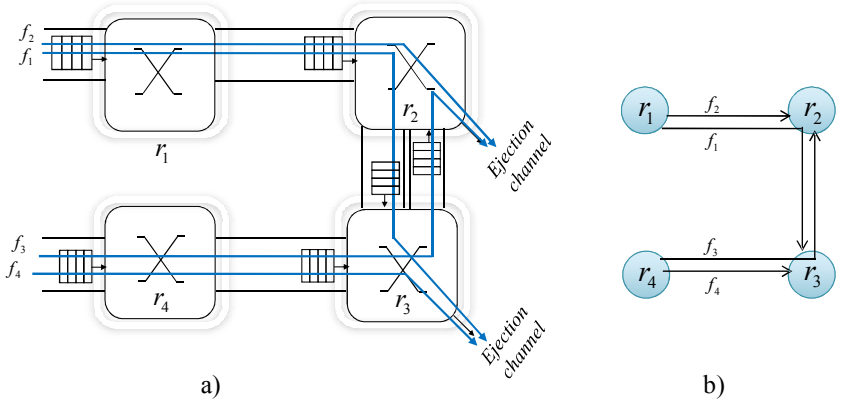

Fig. 16. A synthetic example.

7. EXPERIMENTS

7.1. Experimental Setup

To evaluate the capability of our method, we applied it to a synthetic traffic pattern and a realistic one. Throughout the experiments, we assume an SoC with 500 MHz frequency in which packets traverse the network using the XY routing algorithm. Flows follow TSPEC, $f_i \propto (L_i, p_i, \sigma_i, \rho_i)$, and each node guarantees the service curve of $\beta_{R,T}(t) = \delta_T \otimes \gamma_{0,R}$, where the serving rate R is C *flit/cycle* and the latency T, $\frac{L_w}{C} + D_{router}$ *cycle*. We have implemented the proposed analytical model in C++ to automate analysis steps.

7.2. Synthetic Traffic Pattern

We synthesize a simple traffic pattern as shown in Figure 16 to follow the analytical approach step by step and derive numerical results. The figure depicts a network with 4 flows and 4 routers which serve flows in the FIFO order. f_1 is the tagged flow and f_2 and f_4 are contention flows.

7.2.1. ***Computation of the end-to-end equivalent service curve.***

Step 1: We first calculate the intra-router ESC for the tagged flow in each node. Then, we can model a flow passing through a series of routers as a series of concatenated pseudoaffine servers.

It is worth mentioning that TSPEC of each flow f_j mentioned above is the TSPEC of the input flow to its source node, for example $f_2 \propto (L_2, p_2, \sigma_2, \rho_2)$ which means $\rho_{(f_2,r_1)} = \rho_2$ and other characteristics can be obtained as well.

— ***In router*** r_1***:*** From Equation (6) and (7), the ESC for aggregate flow $f_{\{1,2\}}$ in node 1 is given by:

$$\beta_{(f_{\{1,2\}},r_1)} = \delta_0 \otimes \gamma_{0,C}. \tag{13}$$

— ***In router*** r_2***:*** $F^B_{(f_1,r_2)} = \{f_2\}$ and due to Equation (6) and (7), $R_{(f_1,r_2)} = C$ and $T^l_{(f_1,r_2)} = 0$. Furthermore, $T^{Total}_{(f_1,r_2)} = T^l_{(f_1,r_2)} + T^{HoL}_{(f_1,r_2)}$ and regarding to Equation (10) and (11), $T^{HoL}_{(f_1,r_2)} = max_{\forall f_c \in F^B_{(f_1,r_2)}} \left(T^{HoL(f_c)}_{(f_1,r_2)}\right) = T^{HoL(f_2)}_{(f_1,r_2)}$ where $T^{HoL(f_2)}_{(f_1,r_2)}$ is calculated as follows:

$$T^{HoL(f_2)}_{(f_1,r_2)} = T^{l}_{(f_2,r_2)} - \theta_{(f_2,r_2)} + \frac{L_{(f_2,r_2)} + \theta_{(f_2,r_2)} p_{(f_2,r_2)}}{R_{(f_2,r_2)}} \tag{14}$$

where $R_{(f_2,r_2)} = \frac{C}{2}$, $T^{l}_{(f_2,r_2)} = \frac{L_w}{C} + D_{router}$, because two VCs (one transmits f_2 and the other f_3) are sharing the ejection channel of router r_2. In Equation (14), we should obtain TSPEC of input flow f_2 to r_2 which is TSPEC of output flow f_2 from r_1. Since TSPEC is derived from arrival curve, we obtain arrival curve of output flow f_2 from r_1 by applying the proposed Theorem 3 in Section 5. We assumed $\theta_i \leq T_{(f_i,r_j)}$ *for* $\forall f_i$ ***passing through*** $\forall r_j$. Thus, $\alpha^{*}_{(f_2,r_1)} = \alpha_{(f_2,r_2)} = \gamma_{\sigma_{(f_2,r_1)}+\rho_{(f_2,r_1)}T_{(f_2,r_1)},\rho_{(f_2,r_1)}}$ where $\rho_{(f_2,r_1)} = \rho_2$ and $\sigma_{(f_2,r_1)} = \sigma_2$. In this respect, we can say $\alpha_{(f_2,r_2)} = \gamma_{\sigma_2+\rho_2 T_{(f_2,r_1)},\rho_2}$. For deriving $T_{(f_2,r_1)}$, we should first obtain ESC for flow f_2 in router r_1, $\beta_{(f_2,r_1)}$, as follows.
From Equation (13), $\beta_{(f_{\{1,2\}},r_1)} = \delta_0 \otimes \gamma_{0,C}$. We then remove f_1 from aggregate flow $f_{\{1,2\}}$ according to Corollary 1 in Section 5, $\beta_{(f_2,r_1)}$ is given by:

$$\beta_{(f_2,r_1)} = \delta_{\left[\frac{L_1+\theta_1(p_1-C)^+}{C}\right]^+ + \theta_1} \otimes \gamma_{0,C-\rho_1} = \delta_{\frac{L_1+\theta_1 p_1}{C}} \otimes \gamma_{0,C-\rho_1} \tag{15}$$

In this respect $T_{(f_2,r_1)} = \frac{L_1+\theta_1 p_1}{C}$, and $\alpha_{(f_2,r_2)} = \gamma_{\sigma_2+\frac{\rho_2(L_1+\theta_1 p_1)}{C},\rho_2}$ which means $\sigma_{(f_2,r_2)} = \sigma_2 + \frac{\rho_2(L_1+\theta_1 p_1)}{C}$, $\rho_{(f_2,r_2)} = \rho_2$, $L_{(f_2,r_2)} = L_{(f_2,r_1)} = L_2$, and $p_{(f_2,r_2)} = p_{(f_2,r_1)} = p_2$. Therefore, Equation (14) is rewritten as below:

$$T^{HoL(f_2)}_{(f_1,r_2)} = \frac{L_w}{C} + D_{router} - \theta_{(f_2,r_2)} + \frac{L_2 + \theta_{(f_2,r_2)} p_2}{\frac{C}{2}} \tag{16}$$

where $\theta_{(f_2,r_2)} = \frac{\sigma_{(f_2,r_2)} - L_{(f_2,r_2)}}{p_{(f_2,r_2)} - \rho_{(f_2,r_2)}} = \frac{\sigma_2 C + \rho_2 L_1 + \rho_2 \theta_1 p_1 - L_2 C}{C(p_2 - \rho_2)}$.
As mentioned before, $R_{(f_1,r_2)} = C$, $T^{l}_{(f_1,r_2)} = 0$, and $T^{Total}_{(f_1,r_2)} = T^{l}_{(f_1,r_2)} + T^{HoL}_{(f_1,r_2)}$. Therefore, the ESC for tagged flow f_1 in router 2 is given by:

$$\beta_{(f_{\{1\}},r_2)} = \delta_{0+T^{HoL(f_2)}_{(f_1,r_2)}} \otimes \gamma_{0,C}. \tag{17}$$

— ***In router*** r_3***:*** Since VC of f_1 is sharing the ejection channel of r_3 with VC of f_4, due to Equation (6) and (7), $R_{(f_1,r_3)} = \frac{C}{2}$ and $T^{l}_{(f_1,r_3)} = \frac{L_w}{C} + D_{router}$. Thus, the ESC for tagged flow f_1 in router 3 is given by:

$$\beta_{(f_{\{1\}},r_3)} = \delta_{\left(\frac{L_w}{C} + D_{router}\right)} \otimes \gamma_{0,\frac{C}{2}}. \tag{18}$$

Step 2: Now, we are able to compute per-flow ESC provided by the tandem of routers the tagged flow passes. Figure 17 depicts different steps of computing end-to-end ESC for tagged flow f_1. After calculating intra-router ESC as mentioned in Step 1, we have an *aggregate analysis model* as shown in Figure 17(b). Since we have investigated the effect of flow f_2 on tagged flow f_1 in router r_2, when we calculated $\beta_{(f_1,r_2)}$ in step 1, f_2 is removed from r_2 in Figure 17(b). Similarly, f_3 and f_4 are eliminated from r_2 and r_3, respectively. We then obtain end-to-end ESC for tagged flow f_1 by following Algorithm 1. Due to the algorithm, $\beta_{(\{1\},r_{2,3})}$ in Figure 17(c) is calculated as $\beta_{(\{1\},r_2)} \otimes \beta_{(\{1\},r_3)}$.

We use the theorem of *Concatenation of network elements* [Le Boudec et al. 2004]. Given are two nodes sequentially connected and each is offering a latency-rate service

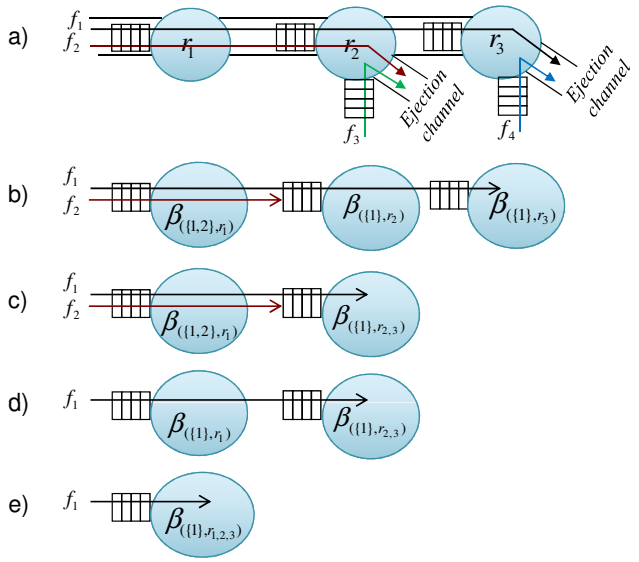

Fig. 17. Analysis steps for the example in Figure 15.

curve $\beta_{R_i,T_i},\ i = 1 and 2$. These nodes can be represented as a single latency-rate server as follows:

$$\beta_{R_1,T_1} \otimes \beta_{R_2,T_2} = \beta_{min(R_1,R_2),T_1+T_2} \tag{19}$$

Therefore, $\beta_{(\{1\},r_{2,3})}$ is given by:

$$\beta_{(\{1\},r_{2,3})} = \delta_{\frac{L_w}{C}+D_{router}+T^{HoL(f_2)}_{(f_1,r_2)}} \otimes \gamma_{0,\frac{C}{2}}. \tag{20}$$

In Figure 17(c), $s^m = \{1,2\}$, $s^{Prev} = \{\}$, and $s^{Next} = \{1\}$. The algorithm then removes flow f_2 from aggregate flow $f_{\{1,2\}}$ in router r_1. To this end, we apply the proposed corollary 1 to obtain ESC $\beta_{(\{1\},r_1)}$ by subtracting arrival curve of α_2 from $\beta_{(\{1,2\},r_1)}$, as follows:

$$\beta_{(\{1\},r_1)} = \delta_{\frac{L_2+\theta_2(p_2-C)^+}{C}+\theta_2} \otimes \gamma_{0,C-\rho_2} \tag{21}$$

Figure 17(c) depicts the example after removing arrival curve of flow f_2 from $\beta_{(\{1,2\},r_1)}$. Now, end-to-end ESC can be calculated by:

$$\begin{aligned}\beta_{(\{1\},r_{1,2,3})} &= \beta^{eq}_{f_1} = \beta_{(\{1\},r_1)} \otimes \beta_{(\{1\},r_{2,3})} \\ &= \delta_{\frac{L_w+L_2+\theta_2(p_2-C)^+}{C}+D_{router}+\theta_2+T^{HoL(f_2)}_{(f_1,r_2)}} \otimes \gamma_{0,min\left(\frac{C}{2},C-\rho_2\right)}\end{aligned} \tag{22}$$

Suppose that flows follow TSPEC, $f_1 \propto (1,1,8,0.128)$, $f_2 \propto (1,1,2,0.032)$, $f_3 \propto (1,1,2,0.008)$, and $f_4 \propto (1,1,4,0.128)$. Therefore, θ_j is computed for each flow f_j as $\theta_1 = (\sigma_1 - L_1)/(p_1 - \rho_1) = (8-1)/(1-0.128) = 8.027$, $\theta_2 = 1.033$, $\theta_3 = 1.008$, and $\theta_4 = 3.44$. Also, we assume serving rate $C = 1$ *flit/cycle*, $L_w = 1$ *flit*, and $D_{router} = 1$ *cycle*. We then replace the variables in Equation (22) by numbers as follows:

$$\beta_{f_1}^{eq} = \delta_{9.363} \otimes \gamma_{0,0.5} \tag{23}$$

7.2.2. ***Computation of LUDB***.

According to Theorem 1 and Equation (22), the maximum delay for flow f_1 is bounded by

$$h(\alpha_1, \beta_{f_1}^{eq}) = \left\lceil \frac{L_w + L_2 + \theta_2(p_2 - C)^+}{C} + D_{router} + \theta_2 + T_{(f_1,r_2)}^{HoL(f_2)} + \frac{L_1 + \theta_1 \left(p_1 - min\left(\frac{C}{2}, C - \rho_2\right)\right)^+}{min\left(\frac{C}{2}, C - \rho_2\right)} \right\rceil \tag{24}$$

If we substitute the values into variables in the above mentioned equation, $h(\alpha_1, \beta_{f_1}^{eg})$ is equal to $\lceil 19.39 \rceil = 20$.

In what follows, we consider the accuracy of our proposed analytical method through the BookSim simulator [Jiang et al. 2013] and then compare it with the methods without considering the traffic peak rate behavior [Lenzini et al. 2006].

7.2.3. ***Computation of Buffer Size Thresholds***.

As routers are assumed to be input-buffered, we derive buffer size threshold for each input channel in each router by following Eq. (12). In the example of Figure 16, we have assumed one VC per each PC. Therefore, buffer size thresholds are calculated and presented as Table II.

Table II. Buffer size thresholds in the case study with synthetic traffic pattern

	Injection Channel	*Western Channel*	*Northern Channel*	*Eastern Channel*	*Southern Channel*
Router 1	6	–	–	–	–
Router 2	–	11	–	–	3
Router 3	–	8	8	–	–
Router 4	6	–	–	–	–

The buffers size thresholds marked by "-" are not used by flows and thus not relevant for the threshold calculation.

The value of buffer size thresholds per channel depends on the traffic load on that channel which is affected by the number of flows passing through the channel, their traffic specifications, and the contention between them.

It is worth mentioning that while the calculated delay is an upper bound, the calculated buffer size threshold gives the lower bound of a buffer size to avoid back-pressure and buffer overflow. Therefore, the buffer sizes in simulations are set to be equal to or larger than the corresponding calculated buffer thresholds. To go into more details, Table III shows the delay bounds derived from both analytical model and simulation results for the tagged flow f_1 versus different values of the buffer size. As it is assumed that all routers has the same buffer space, the buffer size in the table should be equal to or larger than the maximum calculated space threshold in order to no buffer space threshold has been violated. Due to Table II, the maximum space threshold in this example is equal to 11 *flits*. As it can been seen from Table III, when the buffer size is equal to or larger than 11 *flits*, the delay bound calculated by the simulator is fixed and very close to the analytical result. Otherwise, the simulation results cannot be compared with the analytical results because the back-pressure and buffer overflow may happen and in turn the delay bound calculated from the model becomes invalid.

Table III. End-to-end delay bound for tagged flow f_1 under different buffer sizes

Buffer Size (*flits*)	14	13	12	11	8	6	4	3	2
Simulation Results (*cycles*)	19	19	19	19	19	19	19	17	16
Analytical Model Results (*cycles*)	20	20	20	20	20	20	20	20	20

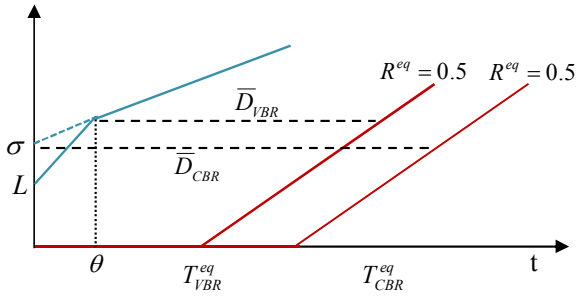

Fig. 18. Comparing $\bar{D}_{VBR}$ and $\bar{D}_{CBR}$ with the same equivalent service curve.

7.2.4. ***Simulation Result.***

We investigate the accuracy of the proposed analytical model through BookSim simulator which is a cycle-accurate simulator [Jiang et al. 2013]. The simulation uses the same assumptions as the analytical model. We have considered a 2×2 mesh on-chip interconnect as shown in Figure 16 and input-buffered routers with 12 *flits* in each input channel. It takes 1 clock cycle to pass a flit within a router and 1 clock cycle to transmit a flit over wires between neighboring routers. We also consider the XY routing algorithm to route the data packets among cores.

Simulation result shows that worst-case delay for tagged flow f_1 in the previously mentioned system is equal to 19 *cycles*, which is below the LUBD of 20 *cycles*, predicted by our model.

We also change the value of σ_2 from 2 to 4 to consider more experiments. The LUDB calculated by our analytical model for tagged flow f_1 is equal to 24 *cycles* and the result from the simulation is also 24 *cycles*, again below the analytical LUDB.

7.2.5. ***Comparison.***

If we use (σ, ρ) instead of TSPEC, each flow j would be constrained by arrival curve $\alpha_j = \sigma_j + \rho_j t = \gamma_{\sigma_j,\rho_j}$. Therefore, flows in the example are represented as $f_1 \propto (8, 0.128)$, $f_2 \propto (2, 0.032)$, $f_3 \propto (2, 0.008)$, and $f_4 \propto (4, 0.128)$. We then follow the stages of computing individual delay bounds for a tagged flow as stated before. For this purpose, we can easily revise our proposed theorems for (σ, ρ) flows by substituting σ and ρ into L and p, respectively, in all formulas. We can also apply the method presented in [Lenzini et al. 2006]. With both approaches, the same value for $h(\alpha_1, \beta^{eq}_{f_1})$ is achieved and equals to 26. Thus, our proposed method which calculates $\bar{D}_{VBR}$ has 23% improvement on the accuracy of the delay bound than the method with CBR flows ($\bar{D}_{CBR}$).

To analyze delay sensitivity, Table IV depicts LUDB for tagged flow f_1 in a network with CBR flows ($\bar{D}_{CBR}$) and also VBR flows ($\bar{D}_{VBR}$) versus the different values of service rate R, along with values for the end-to-end ESC parameterized by $R^{eq}_{f_1}$, $T^{eq}_{f_1,CBR}$ and $T^{eq}_{f_1,VBR}$. From this table, it is clear that the end-to-end equivalent service rate, $R^{eq}_{f_1}$, is decreasing by reducing R, while the end-to-end processing delays and delay bounds are increasing as well. Also, it is worth mentioning that the improvement percentage (η) decreases because of reduction of $R^{eq}_{f_1}$ and increase of $T^{eq}_{f_1,CBR}$ and $T^{eq}_{f_1,VBR}$.

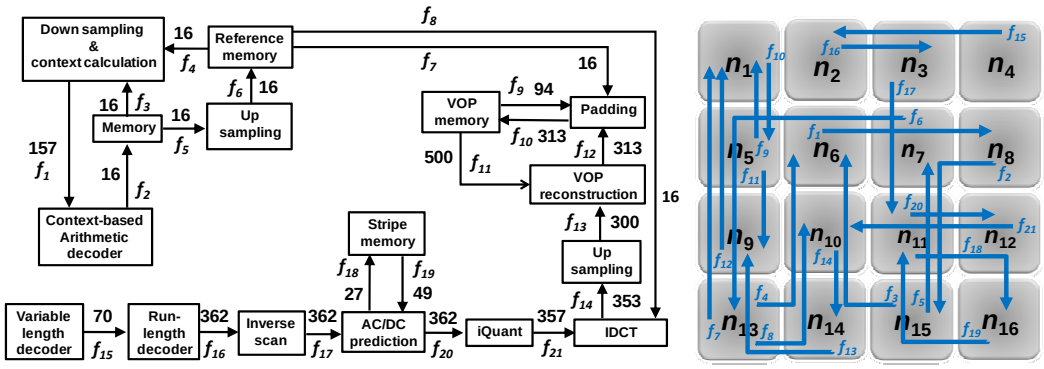

Fig. 19. VOPD Application

This is due to the relation between these parameters which we will elaborate it in the following.

Figure 18 shows $\bar{D}_{CBR}$ and $\bar{D}_{VBR}$ for $R^{eq} = 0.5$ where $p \geq R^{eq}$ and the end-to-end ESCs are in the form of $\delta_{T^{eq}} \otimes \gamma_{0,R^{eq}}$. According to the network calculus theory [Le Boudec et al. 2004], $\bar{D}_{CBR} = T^{eq}_{CBR} + \frac{\sigma}{R^{eq}}$ and with Theorem 1, $\bar{D}_{VBR} = T^{eq}_{VBR} + \frac{L+\theta(p-R^{eq})}{R^{eq}}$. η is calculated as follows.

$$\eta = \frac{\bar{D}_{CBR} - \bar{D}_{VBR}}{\bar{D}_{CBR}} = \frac{\sigma - L - p\theta + R^{eq}\left(T^{eq}_{CBR} - T^{eq}_{VBR} + \theta\right)}{\sigma + T^{eq}_{CBR} R^{eq}} \tag{25}$$

To analyze the behavior of η, we compute the derivative of function η in terms of R^{eq} as follows:

$$\frac{d\eta}{dR^{eq}} = \frac{T^{eq}_{CBR}\left(L + p\theta\right) - \sigma\left(T^{eq}_{VBR} - \theta\right)}{\left(\sigma + T^{eq}_{CBR} R^{eq}\right)^2}$$

From Figure 18, it is obvious that $L + p\theta \geq \sigma$ and $T^{eq}_{CBR} \geq T^{eq}_{VBR} - \theta$ which results $T^{eq}_{CBR}\left(L + p\theta\right) - \sigma\left(T^{eq}_{VBR} - \theta\right) \geq 0$. Thus, $\frac{d\eta}{dR^{eq}} \geq 0$ and η is an increasing function in terms of R^{eq} which means that when R^{eq} increases or decreases, η shows the same behavior as R^{eq}.

Table V shows $R^{eq}_{f_1}$, $T^{eq}_{f_1,CBR}$ and $T^{eq}_{f_1,VBR}$, $\bar{D}_{CBR}$, and $\bar{D}_{VBR}$ for tagged flow f_1 versus the different values of processing delay T. From this table, it can be seen that the end-to-end processing delays and delay bounds are decreasing by reducing T.

Table IV. End-to-end delay comparison for tagged flow f_1 under different service rates

	$R_1 = 1$	$R_2 = 0.7$	$R_3 = 0.5$
$R^{eq}_{f_1}$	0.5	0.35	0.25
$T^{eq}_{f_1,CBR}$	10	13.428	18
$T^{eq}_{f_1,VBR}$	9.363	13.326	18.951
$\bar{D}_{f_1,CBR}$	26	37	50
$\bar{D}_{f_1,VBR}$	20	32	48
η_{f_1}	23%	13.5%	4%

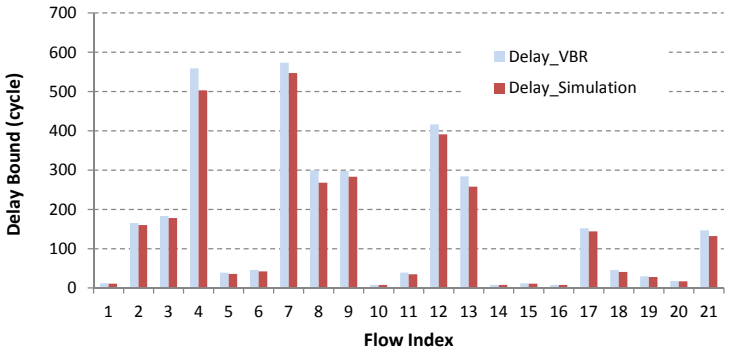

Fig. 20. Comparison of delay bounds from the proposed model and simulator for VOPD application

7.3. Realistic Traffic Pattern

We consider a real-time multimedia application with a random mapping to the tiles of a 4×4 mesh on-chip network. Figure 19 shows the task graph and flow mapping of a Video Object Plane Decoder (VOPD) [Bertozzi et al. 2002] in which each block corresponds to an IP and the numbers near the edges represent the bandwidth (in MBytes/sec) of the data transfer, for a 30 frames/sec MPEG-4 movie with 1920×1088 resolution [Van der Tol et al. 2002]. There are 21 communication flows which characterized by TSPEC. We assume L_i and p_i for all flows are the same and equal to $1\ flit$ and $1\ flit/cycle$, respectively. ρ_i is determined in $flits/cycle$ due to associated bandwidth with flow f_i in Figure 19 and also, σ_i varies between 8 and $128\ flits$ for different flows.

We derive delay bounds from the proposed analytical model, $\bar{D}_{f_i,VBR}$, and BookSim simulator, $\bar{D}_{f_i,Sim}$ for the whole set of flows in Figure 20. In order to have a better insight about the proposed model, for each obtained delay bound, the relative error with respect to simulation result is calculated. The calculations show that the maximum and average relative errors are about 12.1% and 6.8%, respectively, which confirm the accuracy of the proposed model.

As can be observed from Figure 20, a flows may have larger (like f_7) or smaller (like f_{14}) worst-case delay bound than the other flows, which depends on its traffic specification (TSPEC) and the situation of that flow in the network. For example, if the worst-case delay bound of a particular flow is too large, 1) the flow is probably more limited by its TSPEC parameters for injecting to the network, 2) the flow may have a longer path from its source to destination, or 3) the flow may have more contentions (both direct and indirect) with other flows along its path.

Figure 21 compares the results of applying our analytical model, $\bar{D}_{f_i,VBR}$, and the method with CBR flows, $\bar{D}_{f_i,CBR}$. As you can see in this figure, the proposed model in this paper is more accurate than the method without considering the traffic peak

Table V. End-to-end delay comparison for tagged flow f_1 under different processing delay

	$T_1 = 10$	$T_2 = 2$	$T_3 = 1$	$T_4 = 0.5$	$T_5 = 0.1$
$R^{eq}_{f_1}$	0.5	0.5	0.5	0.5	0.5
$T^{eq}_{f_1,CBR}$	26	10	8	7	6.2
$T^{eq}_{f_1,VBR}$	25.363	9.363	7.363	6.363	5.563
$\bar{D}_{f_1,CBR}$	42	26	24	23	23
$\bar{D}_{f_1,VBR}$	39	20	18	17	16
η_{f_1}	7.1%	23%	25%	26.1%	30.4%

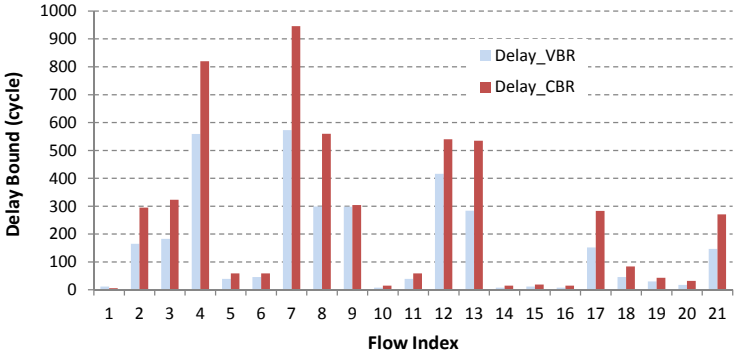

Fig. 21. Comparing $\bar{D}_{VBR}$ and $\bar{D}_{CBR}$ for VOPD application

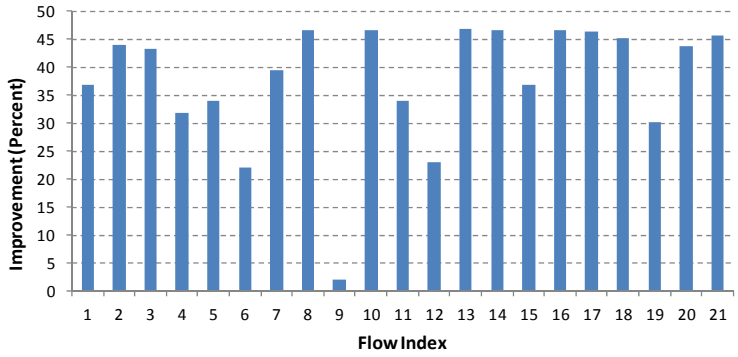

Fig. 22. Improvement percentage of $\bar{D}_{VBR}$ than $\bar{D}_{CBR}$ for VOPD application

behavior. Figure 22 presents improvement percentage for each flow f_i, η_{f_i}, as defined in Eq. 25 to show the effectiveness of our model. Compared to previous models with two parameters, the proposed method improves the accuracy of the delay bounds up to 46.9% and more than 37% on average over all flows.

Table VI presents buffer size threshold of input channels used by flows due to Eq. 12.

Table VI. Buffer size thresholds for VOPD application

$B_{(r_1,I)}=1$	$B_{(r_5,I)}=17$	$B_{(r_7,W)}=1$	$B_{(r_9,S)}=259$	$B_{(r_{12},I)}=1$	$B_{(r_{14},N)}=1$
$B_{(r_1,S)}=275$	$B_{(r_5,N)}=1$	$B_{(r_7,N)}=68$	$B_{(r_{10},I)}=1$	$B_{(r_{12},W)}=4$	$B_{(r_{14},E)}=7$
$B_{(r_2,I)}=1$	$B_{(r_5,E)}=7$	$B_{(r_7,E)}=7$	$B_{(r_{10},E)}=68$	$B_{(r_{13},I)}=204$	$B_{(r_{15},I)}=16$
$B_{(r_2,E)}=1$	$B_{(r_5,S)}=262$	$B_{(r_7,S)}=1$	$B_{(r_{10},S)}=84$	$B_{(r_{13},N)}=1$	$B_{(r_{15},N)}=1$
$B_{(r_3,I)}=1$	$B_{(r_6,I)}=1$	$B_{(r_8,I)}=1$	$B_{(r_{11},I)}=17$	$B_{(r_{13},E)}=68$	$B_{(r_{15},E)}=7$
$B_{(r_3,W)}=1$	$B_{(r_6,E)}=1$	$B_{(r_8,W)}=1$	$B_{(r_{11},N)}=77$	$B_{(r_{14},I)}=2$	$B_{(r_{16},I)}=1$
$B_{(r_3,E)}=1$	$B_{(r_6,S)}=17$	$B_{(r_9,I)}=68$	$B_{(r_{11},E)}=1$	$B_{(r_{14},W)}=84$	$B_{(r_{16},N)}=1$
$B_{(r_4,I)}=1$	$B_{(r_7,I)}=1$	$B_{(r_9,N)}=16$	$B_{(r_{11},S)}=16$		

Sub-index (r, L) in Table VI refers to input channel L of router r, where r is the number of the router and L is a letter assigned to the input port which is defined as I: *Injection channel*, W: *Western input channel*, N: *Northern input channel*, E: *Eastern input channel*, and S: *Southern input channel*. For example, $B_{(r_3,W)}$ indicates the buffer size threshold in western input channel of router 3.

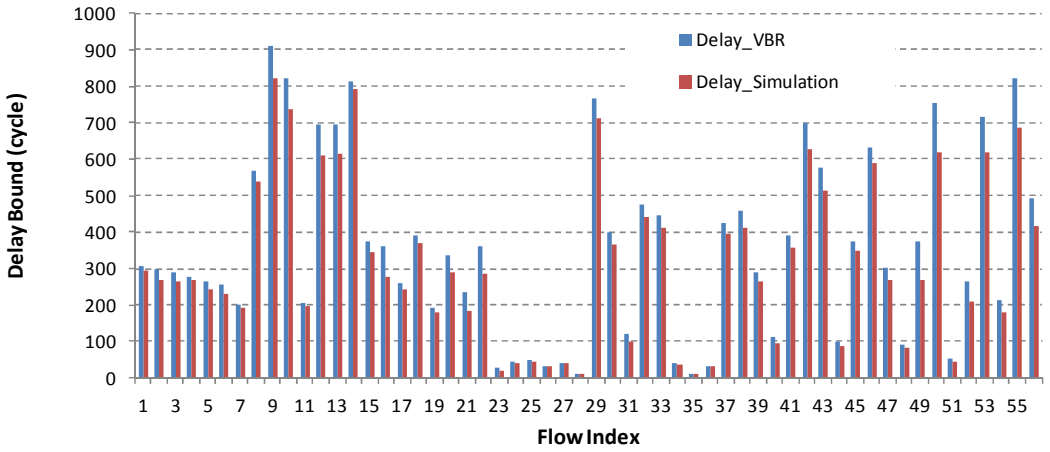

Fig. 23. Comparison of delay bounds from the proposed model and simulator under the transpose traffic pattern

7.4. Transpose Traffic Pattern

To investigate a larger network, we experiment a 8×8 mesh network under the transpose traffic pattern with 56 communication flows characterized by TSPEC. In this traffic pattern, the node with binary value $a_{n-1}, a_{n-2}, ..., a_1, a_0$ communicates with the node $\bar{a}_{n/2-1}, ..., \bar{a}_0, \bar{a}_{n-1}, ..., \bar{a}_{n/2}$. For all traffic flows, we assume the same values for L_i and p_i which are $1\ flit$ and $1\ flit/cycle$, respectively. For different flows, ρ_i varies between 0.001 and $0.03\ flits/cycle$, and σ_i between 2 and $128\ flits$. Table VII presents the source and destination of flows along with the index assigned to them.

Similar to previous case studies, delay bounds from the proposed analytical model, $\bar{D}_{f_i,VBR}$, and BookSim simulator, $\bar{D}_{f_i,Sim}$ are derived for all flows and presented as Figure 23. As can be seen from this figure, all delays observed in simulations are below the LUDB but not too far, suggesting that the analytical bound is fairly tight since the simulation typically does not exercise the worst case.

To consider the accuracy of the analytical model, the relative errors with respect to simulation results are computed. The calculations show that the maximum and average relative errors are about 33.3% and 13%, respectively.

We also calculate per-flow delay bounds from our proposed method, $\bar{D}_{f_i,VBR}$, and CBR analytical model, $\bar{D}_{f_i,CBR}$, as depicted in Figure 24 and compare the results by computation of improvement percentages per flow, η_{f_i}. As shown in Figure 25, our

Table VII. The list of flows

$f_1 : 0 \longrightarrow 63$	$f_{11} : 20 \longrightarrow 29$	$f_{21} : 25 \longrightarrow 52$	$f_{30} : 55 \longrightarrow 1$	$f_{39} : 29 \longrightarrow 20$	$f_{48} : 44 \longrightarrow 26$
$f_2 : 1 \longrightarrow 55$	$f_{12} : 10 \longrightarrow 46$	$f_{22} : 24 \longrightarrow 60$	$f_{31} : 47 \longrightarrow 2$	$f_{40} : 46 \longrightarrow 10$	$f_{49} : 52 \longrightarrow 25$
$f_3 : 2 \longrightarrow 47$	$f_{13} : 9 \longrightarrow 54$	$f_{23} : 34 \longrightarrow 43$	$f_{32} : 39 \longrightarrow 3$	$f_{41} : 54 \longrightarrow 9$	$f_{50} : 60 \longrightarrow 24$
$f_4 : 3 \longrightarrow 39$	$f_{14} : 8 \longrightarrow 62$	$f_{24} : 33 \longrightarrow 51$	$f_{33} : 31 \longrightarrow 4$	$f_{42} : 62 \longrightarrow 8$	$f_{51} : 43 \longrightarrow 34$
$f_5 : 4 \longrightarrow 31$	$f_{15} : 19 \longrightarrow 37$	$f_{25} : 32 \longrightarrow 59$	$f_{34} : 23 \longrightarrow 5$	$f_{43} : 37 \longrightarrow 19$	$f_{52} : 51 \longrightarrow 33$
$f_6 : 5 \longrightarrow 23$	$f_{16} : 18 \longrightarrow 45$	$f_{26} : 41 \longrightarrow 50$	$f_{35} : 15 \longrightarrow 6$	$f_{44} : 45 \longrightarrow 18$	$f_{53} : 59 \longrightarrow 32$
$f_7 : 6 \longrightarrow 15$	$f_{17} : 17 \longrightarrow 53$	$f_{27} : 40 \longrightarrow 58$	$f_{36} : 22 \longrightarrow 13$	$f_{45} : 53 \longrightarrow 17$	$f_{54} : 50 \longrightarrow 41$
$f_8 : 13 \longrightarrow 22$	$f_{18} : 16 \longrightarrow 61$	$f_{28} : 48 \longrightarrow 57$	$f_{37} : 30 \longrightarrow 12$	$f_{46} : 61 \longrightarrow 16$	$f_{55} : 58 \longrightarrow 40$
$f_9 : 12 \longrightarrow 30$	$f_{19} : 27 \longrightarrow 36$	$f_{29} : 63 \longrightarrow 0$	$f_{38} : 38 \longrightarrow 11$	$f_{47} : 36 \longrightarrow 27$	$f_{56} : 57 \longrightarrow 48$
$f_{10} : 11 \longrightarrow 38$	$f_{20} : 26 \longrightarrow 44$				

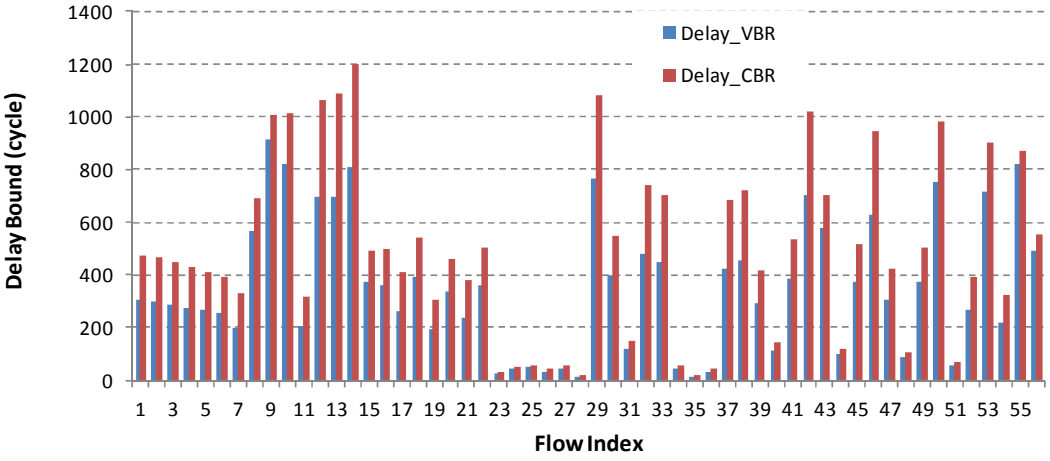

Fig. 24. Comparing $\bar{D}_{VBR}$ and $\bar{D}_{CBR}$ under the transpose traffic pattern

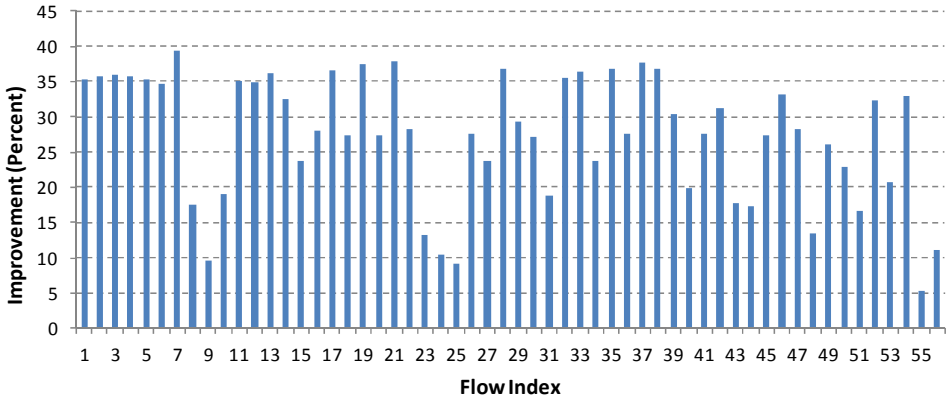

Fig. 25. Improvement percentage of $\bar{D}_{VBR}$ than $\bar{D}_{CBR}$ under the transpose traffic pattern

proposed analytical model is up to 39.3% more accurate than CBR analytical model and more than 31% on average over all flows.

The run-time of the proposed method in C++ is typically in the order of a few seconds. It is about 0.58 *sec* and 1.02 *sec* for the VOPD application and transpose traffic pattern, respectively.

7.5. Discussion About Other Metrics

Although the paper targets an analytical model for latency bound, we briefly consider evaluating other metrics including throughput, communication load, energy consumption, and area requirements.

The network throughput is the sum of the data rates that are delivered to all ejection channels in a network and communication load is estimated by utilized bandwidth and calculated as the sum of the data rates injected to the network. As the paper models the network which is not saturated, the throughput and communication load have

the same values. This value is equal to $0.296\ flits/cycle$ for the synthetic example in Section 7.2 and $0.73\ flits/cycle$ for VOPD application in Section 7.3.

Network calculus does not directly evaluate energy consumption and area requirements. However, we can present a comparative discussion between VBR and CBR analyses, which is the main contribution of this work. Since we study the classic input-queuing virtual-channel router, there is nothing new or changed in the structure and design details of routers. In terms of area, what brings difference is in the calculated backlog, which determines the buffer size thresholds. In network calculus, the upper bound on backlog along the network is computed by the sum of the individual bounds on every element [Le Boudec et al. 2004]. Thus, the total required buffer for flow i is bounded by:

$$\bar{B}_i = \sum_{j \in L_{f_i}} \bar{B}_{ij} \tag{26}$$

where $\bar{B}_{ij}$ is the upper bound on the buffer size for flow i in each channel $j \in L_{f_i}$ and L_{f_i} is the set of channels along the path of flow i. $\bar{B}_{ij}$ for VBR traffic flows, $\bar{B}_{ij}^{VBR}$, and CBR traffic flows, $\bar{B}_{ij}^{CBR}$, are given by Eq. (27) and Eq. (28), respectively.

$$\bar{B}_{ij}^{VBR} = \sigma_i + \rho_i T_j + ((\sigma_i - L_j)/(p_i - \rho_i) - T_j)^+ [(p_i - R_j)^+ - p_i + \rho_i] \tag{27}$$

$$\bar{B}_i^{CBR} = \sigma_i + \rho_i T_j \tag{28}$$

In Eq. (27), term $[(p_i - R_j)^+ - p_i + \rho_i]$ is negative because $R_j \geq \rho_i$ and $p_i \geq R_j$ due to channel capacity constraint and the assumption stated in Section 4, respectively. Further, term $((\sigma_i - L_j)/(p_i - \rho_i) - T_j)^+$ is always positive because $a^+ = a$, if $a \geq 0$; $a+ = 0$, otherwise. Therefore, $((\sigma_i - L_j)/(p_i - \rho_i) - T_j)^+ [(p_i - R_j)^+ - p_i + \rho_i] < 0$ which means that $\bar{B}_{ij}^{VBR} \leq \bar{B}_{ij}^{CBR}$. In Section 7.2.3, we have calculated the required buffer size (buffer size threshold) in each input port of routers for a synthetic example. The sum of these values is the total required buffer size, $\bar{B}^{VBR}$, which is equal to 42 ***flits***. If we calculate the total required buffer size for CBR analysis, $\bar{B}^{CBR}$, by Eq. (26) and (28), it would be equal to 51 ***flits***, which is about 21.4% larger than $\bar{B}^{VBR}$. Similarly, $\bar{B}^{VBR}$ is calculated for VOPD application as a realistic traffic pattern by summing buffer size bounds derived in Section 7.3. The calculations show that $\bar{B}^{VBR} = 1673$ ***flits*** and $\bar{B}^{CBR} = 2827$ ***flits***. Therefore VBR analysis leads to about 40.8% reduction of the total required buffers. We have also derived $\bar{B}^{CBR}$ and $\bar{B}^{VBR}$ for the case study represented in Section 7.4 which is a 8×8 mesh network under the transpose traffic pattern. Due to calculations, $\bar{B}^{CBR} = 18256$ ***flits*** and $\bar{B}^{VBR} = 12556$ ***flits*** which shows that the total required buffers is reduced about 31.2% by VBR analysis. As a result, under the same network and application, VBR analysis gives tighter backlog bound than CBR analysis and can thus give more accurate bounds on the buffer requirements. From the design perspective, the tighter backlog bounds lead to the area saving in the router buffers.

Regarding power consumption, the network power comprises router power (buffer, switch, control circuit) and link power which are traffic dependent. It is notable that although VBR analysis derives tighter delay bounds, it does not change the packet transfer behavior, because it is only deriving more accurate analytical delay bounds without any change in design features of the router like switching, control, and link traversal. Therefore, the design decision of the router which our analysis brings impact on is the buffer dimensioning. Assuming the same system model, VBR analysis can indeed derive tighter bounds than CBR analysis on buffer requirements, leading to power consumption saving. Following a power model for the buffers using, e.g. Orion [Shi et al. 2002], we can safely assume that the power consumption for buffers will decrease proportionally to the buffer size.

8. CONCLUSIONS

In this work, we have derived the analysis procedure to investigate per-flow delay bound. To this end, we have given theorems to calculate end-to-end ESC and internal output arrival curves in a FIFO multiplexing network. Based on the proposed analysis technique, we have conducted case studies of worst-case performance analysis, considered the accuracy of the proposed model through simulation, and compared it with a method without considering the traffic peak behavior. We have developed algorithms to automate analysis steps. The algorithms run very fast and can be applied for larger networks with more flows. In the future, we plan to develop network calculus models to investigate different scheduling policies and then compare them. We also plan to extend the proposed analytical method in case of back-pressure in the network. There are some network calculus-based analytical models [Qian et al. 2009; Zhao et al. 2013] which analyze worst-case delay bounds for CBR flows due to back-pressure in the network. It would be interesting to derive possibly tighter delay bound for VBR flows. In this respect, we have to extend the analytical models under a given fixed buffer size rather than to-be-determined bounded buffer size.

REFERENCES

BAKHOUYA, M., SUBOH, S., GABER, J.,EL-GHAZAWI, T., AND NIAR, S. 2011. Performance evaluation and design tradeoffs of on-chip interconnect architectures. *Simulation Modelling Practice and Theory, Elsevier. 19,* 6, 1496-1505.

BAUER, H., SCHARBARG, J. L., AND FRABOUL, C. 2010. Improving the Worst-Case Delay Analysis of an AFDX Network Using an Optimized Trajectory Approach. *IEEE Transactions on Industrial Informatics. 6,* 4, 521–533.

BEN-ITZHAK, Y., CIDON, I., AND KOLODNY, A. 2011. Delay Analysis of Wormhole Based Heterogeneous NoC. In *Proceedings of the International Networks-On-Chip Symposium (NOCS)*. 161–168.

BENNETT, J. C. R., BENSON, K.,CHARNY, A.,COURTNEY, W. F., AND LE BOUDEC, J. -Y. 2002. Delay jitter bounds and packet scale rate guarantee for expedited forwarding. *IEEE/ACM Transactions on Networking. 10,* 4, 529–540.

BERTOZZI, D., JALABERT, A., MURALI, S., TAMHANKAR, R., STERGIOU, S., BENINI, L., AND DE MICHELI, G. 2005. NoC synthesis flow for customized domain specific multiprocessor systems-on-chip. *IEEE Transactions on Parallel and Distributed Systems. 16,* 2, 113–129.

BISTI, L., LENZINI, L., MINGOZZI, E., AND STEA, G. 2010. DEBORAH: A Tool for Worst-case Analysis of FIFO Tandems. In *Proceedings of ISoLA 2010, LNCS 6415*. 152–168.

BLAKE, S., BLACK, D., CARLSON, M., DAVIES, E., WANG, Z., AND WEISS, W. 1998. *An architecture for differentiated services*. IETF RFC 2475.

BOGDAN, P. AND MARCULESCU, R. 2007. Quantum-like effects in network-on-chip buffers behavior. In *Proceedings of the 44th Design Automation Conference (DAC)*. 266–267.

BOGDAN, P., KAS, M., MARCULESCU, R., AND MUTLU, O. 2010. QuaLe: A Quantum-Leap Inspired Model for Non-stationary Analysis of NoC Traffic in Chip Multi-processors. In *Proceedings of the International Networks-On-Chip Symposium (NOCS)*. 241–248.

BOUILLARD, A., JOUHET, L., AND THIERRY, E. 2010. Tight performance bounds in the worst-case analysis of feed-forward networks. In *Proceedings of Infocom*. 1316–1324

BOUILLARD, A. AND JUNIER, D. 2011. Worst-case delay bounds with fixed priorities using network calculus. In *Proceedings of Valuetools*. 381–390

BOYER, M. 2010. Half-modelling of shaping in FIFO net with network calculus. *RTNS 2010*.

CHANG, C. 2000. *Performance Guarantees in Communication Networks*. Springer-Verlag.

CHARNY, A. AND LE BOUDEC, J. -Y. 2000. Delay Bounds in a Network with Aggregate Scheduling. In *Proceedings of QofIS*. 1–13.

CIUCU, F., AND SCHMITT, J. 2012. Perspectives on Network Calculus - No Free Lunch but Still Good Value. *ACM Sigcomm. 42,* 4, 311–322.

GEBALI, F. AND ELMILIGI, H., EDITORS 2009. *Networks on Chip: Theory and Practice*. Taylor and Francis Group LLC - CRC Press.

JAFARI, F., LU, Z., JANTSCH, A., AND YAGHMAEE, M. H. 2010. Buffer Optimization in Network-on-Chip Through Flow Regulation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD). 29,* 12, 1973–1986.

JAFARI, F., JANTSCH, A., AND LU, Z. 2011. Output Process of Variable Bit-Rate Flows in On-Chip Networks Based on Aggregate Scheduling. In *Proceedings of the International Conference on Computer Design (ICCD'11)*. Amherst, USA, 445–446.

JAFARI, F., JANTSCH, A., AND LU, Z. 2012. Worst-Case Delay Analysis of Variable Bit-Rate Flows in Network-on-Chip with Aggregate Scheduling. In *Proceedings of the Design, Automation and Test in Europe Conference (DATE'12)*. Dresden, Germany, 538–541.

JIANG, Y. 2002. Delay bounds for a network of guaranteed rate servers with FIFO aggregation. *Computer Networks. 40,* 6, 683–694.

JIANG, N., BECKER, D. U., MICHELOGIANNAKIS, G., BALFOUR, J., TOWLES, B., KIM, J., AND DALLY, W. J. 2013. A Detailed and Flexible Cycle-Accurate Network-on-Chip Simulator. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 86–96.

KIASARI, A. E., JANTSCH, A., AND LU, Z. 2013. Mathematical formalisms for performance evaluation of networks-on-chip. *ACM Computing Surveys. 45,* 3.

KIEFER, A., GOLLAN, N., AND SCHMITT, J.B. 2010. Searching for Tight Performance Bounds in Feed-Forward Networks. In *Proceedings of MMB/DFT*. 227–241.

HANSSON, A., WIGGERS, M.,MOONEN, A., GOOSSENS, K., AND BEKOOIJ, M. 2008. Applying dataflow analysis to dimension buffers for guaranteed performance in networks on chip. In *Proceedings of NOCS*. 211–212.

LE BOUDEC, J. Y. AND THIRAN, P. 2004. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. (LNCS, vol. 2050). Berlin, Germany: Springer-Verlag.

LEE, S. Real-Time Wormhole Channels. *Parallel Distributed Computer. 63,* 299–311.

LENZINI, L., MARTORINI, L.,MINGOZZI, E., AND STEA, G. 2006. Tight end-to-end per-flow delay bounds in fifo multiplexing sink-tree networks. *Performance Evaluation. 63,* 9, 956–987.

LENZINI, L., MINGOZZI, E., AND STEA, G. 2008. A Methodology for Computing End-to-end Delay Bounds in FIFO-multiplexing Tandems. *Elsevier Performance Evaluation. 65,* 11-12, 922–943.

MARTIN, S., MINET, P., AND GEORGE L. 2003. Deterministic End-to-End Guarantees for Real-Time Applications in a DiffServ-MPLS Domain. In *Proceedings of SERA 2003, LNCS 3026*. 51–73.

MARTIN, S. AND MINET, P. 2006. Schedulability analysis of flows scheduled with FIFO: Application to the Expedited Forwarding class. In *Proceedings of IPDPS*. Rhodes Island, 25–29.

MOADELI, M., SHAHRABI, A., VANDERBAUWHEDE, W., AND OULD-KHAOUA M. 2007. An analytical performance model for the spidergon noc. In *Proceedings of 21st AINA*. 1014–1021.

OGRAS, U. Y., HU, J., AND MARCULESCU R. 2005. Key research problems in noc design: A holistic perspective. In *Proceedings of CODES+ISSS 2005*. 69–74.

QIAN, Y., LU, Z., AND DOU, W. 2009. Analysis of Worst-case Delay Bounds for Best-effort Communication in Wormhole Networks on Chip. In *Proceedings of the 3rd ACM/IEEE International Symposium on Networks-on-Chip (NOCS'09)*. ACM/IEEE, San Diego, CA, 44–53.

QIAN, Y., LU, Z., AND DOU, Q. 2010. QoS Scheduling for NoCs: Strict Priority Queueing versus Weighted Round Robin. In *Proceedings of the 28th International Conference on Computer Design (ICCD'10)*. Amerstedam, the Netherlands, 52–59.

RAHMATI, D., MURALI, S., BENINI, L., ANGIOLINI, F., DE MICHELI, G., AND SARBAZI-AZAD, H. A method for calculating hard QoS guarantees for Networks-on-Chip. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD'09)*. 579–586.

RAHMATI, D., MURALI, S., BENINI, L., ANGIOLINI, F., DE MICHELI, G., AND SARBAZI-AZAD, H. Computing Accurate Performance Bounds for Best Effort Networks-on-Chip. *IEEE Transactions on Computers (IEEE-TC). 62,* 3, 452–467.

RIZK, A. AND FIDLER, M. 2012. Non-asymptotic End-to-end Performance Bounds for Networks with Long Range Dependent fBm Cross Traffic. *Computer Networks. 56,* 1, 127–141.

SCHMITT, J. B., ZDARSKY, F. A., AND FIDLER, M. 2008. Delay bounds under arbitrary multiplexing: When network calculus leaves you in the lurch .... In *Proceedings of INFOCOM*. 1669–1677.

SHI, Z., AND BURNS A. 2008. Real-time communication analysis for on-chip networks with wormhole switching. In *Proceedings of the 2nd ACM/IEEE International Symposium on Networks-on-Chip (NOCS 2008). IEEE Computer Society*. 161–170.

VAN DER TOL, E.B. AND JASPERS, E.G. T. 2002. Mapping of MPEG4 Decoding on a Flexible Architecture Platform. *SPIE. 4674,* , 1–13.

STILIADIS, D. AND VARMA, A. 1998. Latency-rate servers: A general model for analysis of traffic scheduling algorithms. *IEEE/ACM Transactions on Networking. 6,* 5, 611–624.

WROCLAWSKI, J. 1997. *The Use of RSVP with IETF Integrated Services*. IETF RFC 2210.

WANG, H. S., ZHU, X.,PEH, L. S., AND MALIK S. 2002. Orion: A Power-Performance Simulator for Interconnection Networks. In *Proceedings of the 35th annual ACM/IEEE international symposium on Microarchitecture (MICRO)*.

ZHAO, X. and LU, Z. 2013. Per-flow Delay Bound Analysis Based on a Formalized Micro-architectural Model. In *Proceedings of the 7th ACM/IEEE International Symposium on Networks-on-Chip (NOCS'2013)*, Tempe Arizona, USA, April 2013.

Electronic Appendix