Weighted Round Robin Configuration for Worst-Case Delay Optimization in Network-on-Chip

Fahimeh Jafari*, Axel Jantsch†, and Zhonghai Lu‡
*Liverpool Hope University, UK
†TU Wien, Vienna, Austria
‡KTH Royal Institute of Technology, Sweden

***Abstract*—We propose an approach for computing the end-to-end delay bound of individual variable bit-rate flows in a FIFO multiplexer with aggregate scheduling under Weighted Round Robin (WRR) policy. To this end, we use network calculus to derive per-flow end-to-end equivalent service curves employed for computing Least Upper Delay Bounds (LUDBs) of individual flows. Since real time applications are going to meet guaranteed services with lower delay bounds, we optimize weights in WRR policy to minimize LUDBs while satisfying performance constraints. We formulate two constrained delay optimization problems, namely, *Minimize-Delay* and *Multi-objective* optimization. *Multi-objective* optimization has both total delay bounds and their variance as minimization objectives. The proposed optimizations are solved using a genetic algorithm. A Video Object Plane Decoder (VOPD) case study exhibits 15.4% reduction of total worst-case delays and 40.3% reduction on the variance of delays when compared with round robin policy. The optimization algorithm has low run-time complexity, enabling quick exploration of large design spaces. We conclude that an appropriate weight allocation can be a valuable instrument for delay optimization in on-chip network designs.**

***Index Terms*—Network-on-chip, performance evaluation, network calculus, worst-case delay optimization, weight configuration.**

I. Introduction

Many multi-core Systems on Chip (SoC) require different levels of service for different applications. Real-time applications have stringent performance requirements; the correctness relies not only on the communication result but also the end-to-end delay bound. A data packet received too late could be useless. In other words, the Least Upper Delay Bound (LUDB) for each packet must not exceed its deadline. In such systems, it is desirable to minimize the end-to-end delay bound of the traffic streams satisfying their QoS requirements. Therefore, the first important consideration is to derive the LUDB for a given communication flow. To this end, based on the Network Calculus theory [1], we have presented and proved required theorems in [2] [3]. We then presented a methodology [4] to consider resource sharing scenarios and also derive end-to-end Equivalent Service Curve (ESC) and LUDB by applying the proposed theorems. We assume that all traffic can be well characterized as flows and scheduled as aggregates 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. Our proposed models [4] have been defined and validated under Round Robin (RR) policy. RR policy uses the same service level for each connection while multiple service levels allow to better adapt to the application requirements by providing different bandwidth and latency guarantees. A Weighted Round Robin (WRR) scheduling policy assigns weights to concurrent communications to define multiple service levels. Higher service levels have greater weights and do not preempt lower ones. It is important for designers to find appropriate weights in WRR policy such that the corresponding service levels can support QoS requirements for each communication connection. It is desirable to also optimize delay and throughput in the network.

In this paper, we extend our earlier proposed methodology for RR [4] to WRR policy. We then address an optimization problem of minimizing the total delay bounds subject to the performance constraints of the applications running on the SoC. To avoid unfair services, we consider another objective for minimizing the variance of delay bounds in different flows. Minimizing the variance of the delay bounds avoids an intolerable delay of some flows, caused by processing and transmission of other flows. As both mentioned objective functions are important for the real-time applications, we formulate them as a multi-objective problem under QoS constraints. Finally, we show the benefits of the proposed method and quantify performance improvement.

Random variables appear in the formulation of the optimization problem which causes random objective functions. Such optimization problems are usually solved by metaheuristic methods which do not guarantee an optimal solution. However, they usually find high-quality solutions in reasonable time [5]. There is a wide variety of metaheuristics like simulated annealing, tabu search, iterated local search, evolutionary computation, and genetic algorithms. We compare the performance of several metaheuristics (*pure random search*, *markov monotonous approach*, *adaptive search*, and *genetic algorithm*) and conclude that a genetic algorithm based method is most suitable.

The rest of this paper is organized as follows. Section II discusses related works. Section III introduces the basics of Network Calculus. Section IV is devoted to the underlying system model and notations in our analysis. Section V introduces the major features of our formal method for analyzing the contention scenarios and computation of LUDB along with an example. The proposed optimization problems and corresponding solutions are represented in Section VI and VII. Section VIII implements the algorithms for solving the proposed optimization problems. Experimental results are reported in Section IX. Finally, Section X concludes the paper.

II. Related Work

A. *Performance Evaluation of Real-time Services*

There are different mathematical formalisms for performance evaluation in on-chip networks. We have surveyed four popular techniques along with their applications and also reviewed their strengths and weaknesses [6]. Most of the current works use queuing theory-based approaches. For example, Ben-Itzhak et al. [7] propose an analytical model for deriving the average end-to-end latency and link utilization of wormhole NoCs with heterogeneous link capacities and heterogeneous number of virtual channels per unidirectional link. Queuing approaches often use probability distributions like Poisson to model traffic in the network while Poisson distribution is not appropriate for characterizing all traffic patterns in NoC applications. Qian et al. [8]–[10] suggest a methodology for addressing the limitation of assuming traffic arrivals as Poisson processes. They present analysis for a general arrival process based on G/G/1 queue model [9]. In SVR-NoC [8] [10], the proposed analytical model is embedded into the learning process to form the feature vectors and in turn consider a more generalized traffic model. Although queuing approaches have been widely used for performance analysis, they cannot properly model some significant features, such as nonstationary, self-similarity, higher order statistics, for NoC-based multicore platform designs. To overcome this limitation, Bogdan et al. [11] suggest an approach which analyzes the traffic dynamics and captures the non-stationary effects of the NoC workload. They propose QuaLe model [12] based on statistical physics to analyze the information flow and buffer usage in NoCs, and also investigate the impact of packet injection rate and data packet sizes on the multi-fractal spectrum of traffic. In following up to this work, Bogdan et al. [13] propose mathematical frameworks for exascale computing in data-centers-on-a-chip architectures that exploit the NoC paradigm for interconnecting large number of heterogeneous processing elements. The frameworks can account and exploit the non-stationary and multi-fractal characteristics of computation and communication workloads. To address the problem of distant and large volume data exchange, Xue and Bogdan [14] present a user-cooperation network coding strategy for NoCs.

Network calculus is another mathematical approach specialized for deriving worst-case performance analysis. Network calculus is able to model all traffic patterns with bounds defined by arrival curves. It provides the facility of capturing dynamic features of the network based on the traffic flows' shapes. Network calculus has been applied in a number of papers for performance analysis of real-time services in networks with aggregate scheduling. For example, Charny and Boudec [15] derive a closed-form delay bound in a generic network assuming a fluid model. An extended model is proposed [16] to look into packetization effects. The main limitation of these models is that they work well only for small utilization factors in a generic network configuration. Lenzini et al. [17] describe a methodology for obtaining per-flow worst-case delay bound in tandem networks of rate-latency nodes traversed by leaky-bucket shaped flows. This method yields better bounds than those previously proposed. Qian et al. [19] 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.

All previous works based on network calculus investigate computing delay bounds only for average behavior of flows and they do not consider peak behavior, which results in less accurate bounds. Since a considerable number of real time applications are transmitted by Variable Bit-Rate (VBR) traffics, we have proposed a methodology [4] to consider performance analysis for VBR traffic characterized by (L, p, σ, ρ) in on-chip networks employing aggregate resource management. This method achieves more accurate delay bounds. In this paper, we extend this method to WRR and then regulate weights in each router to minimize delay bounds while satisfying performance constraints.

B. *Performance Improvement*

There exist some previous related works focused on flow-control, adaptive routing, and also domain isolation based on non-interfering router microarchitecture. Concer et al. [20] propose an end-to-end flow control protocol, called Connection-Then-Credit (CTC), to handle message-dependent deadlocks in on-chip networks. Although this protocol adds a latency overhead to the transfer of the message, under some conditions, the performance of the system can be improved. As CTC is an extension to the normal credit-based flow control protocol, Sallam et al. [21] implement both protocols and investigate implementation trade-offs along with performance analysis. Joshi and Mutyam [22] consider a prevention flow control mechanism which satisfies cost/performance constraints in torus networks while preventing deadlocks by combining priority arbitration with prevention slot cycling. However, the mechanism can lead to deadlocks with variable-sized packets.

There are a wide range of adaptive algorithms varying from partially to fully adaptive and have the potential for improving system performance [23]–[26]. For instance, Lin and Tang [23] develop a bufferless routing algorithm and have shown improvements of up to 10% for average latency when compared to older designs. Najib et al. [24] present a look-ahead partial adaptive routing for their prior proposed low-latency NOC router [27]. They show that their algorithm improves the performance of a baseline design of the router under imbalanced traffic. Sheng et al. [25] [26] propose a flow control technique, called Whole Packet Forwarding, which is a VC re-allocation scheme for fully adaptive routing in wormhole on-chip networks. They show that fully adaptive routing can offer higher performance than several partial adaptive routing algorithms.

Wassel et al. [28] propose SurfNoC which is a low-latency time-division-multiplexed packet-switched k-ary n-cube network. Channels are scheduled in each dimension of a mesh network in a pipelined fashion using the dimension-ordered routing algorithm. Psarras et al. [29] present a non-interfering VC-based architecture for on-chip networks, called PhaseNoC, which adopts TDM at the VC level. Applications are mapped to disjoint sets of VCs to isolate them both inside the router's pipeline and at the network level. They also design appropriate scheduling of flows to reduce area/delay cost in the network.

All above mentioned works consider reducing average latency in different system models while, in this paper, we calculate worst-case delay bounds under WRR scheduling policy and deterministic routing and minimize worst case delay and variance.

III. Network Calculus Background

Network calculus is a mathematical approach to compute worst case bounds for guaranteed services in communication networks [1]. It uses min-plus algebra to convert non-linear queueing systems into linear systems. The algebra structure of min-plus is $(\Re \cup \{+\infty\}, \wedge, +)$ in which $\wedge$ represents the minimum operation, $f \wedge g = min(f, g)$. The min-plus convolution, denoted by $\otimes$, is defined as $(f \otimes g)(t) = inf_{0 \leq s \leq t} \{f(t-s) + g(s)\}$; where two functions f and g are wide-sense increasing functions.

An arrival curve defines an upper bound on the cumulative arrival process to characterize a traffic flow and a service curve defines a lower bound on the cumulative service process. In this paper, we use Traffic SPECification (TSPEC) [35] for characterizing traffic to look into both the average and peak behaviors of a flow. With TSPEC, the arrival curve of flow f_j is defined as $\alpha_j(t) = min(L_j + p_j t, \sigma_j + \rho_j t)$ 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 $f_j \propto (L_j, p_j, \sigma_j, \rho_j)$. A well-formulated service model to reflect the service capability of a node is the rate-latency function defined as $\beta_{R,T} = R(t-T)^+$, where R is the minimum service rate and T the maximum processing latency of the node. We use x^+ to denote the function $x^+ = x$ if $x > 0; x^+ = 0$, otherwise. $\vee$ represents the maximum operation, $f \vee g = max(f, g)$. *Burst delay* function $\delta_T(t) = +\infty$, if $t > T$; $\delta_T(t) = 0$, otherwise. *Affine function* $\gamma_{b,r}(t) = b + rt$, if $t > 0$; $\gamma_{b,r}(t) = 0$, otherwise. Therefore, $\delta_T \otimes \gamma_{b,r}(t) = b + r(t-T)$. $\oslash$ represents the min-plus deconvolution as $(f \oslash g)(t) = sup_{s \geq 0} \{f(t+s) - g(s)\}$.

IV. System Model

We consider a Network-on-Chip (NoC) architecture in which every node contains a core equipped with a Network Interface (NI) and a router with input and output channels. Assumptions are given as follows:

- The NoC architecture can have arbitrary topologies.
- A flow consists of packets and each packet is broken into flits. We consider the *arbitration granularity* of one word with a fixed word length of L_w for all flows. L_w is assumed to be 1 *flit*.
- Packets have fixed length and traverse the network in a best-effort fashion with virtual-cut-through switching technique using deadlock-free deterministic routing.
- Routers have multiple flit input buffers but no output buffers.
- The router can have multiple Virtual Channels (VCs) per in-port. VC allocation is deterministic and each VC receives an aggregate service.

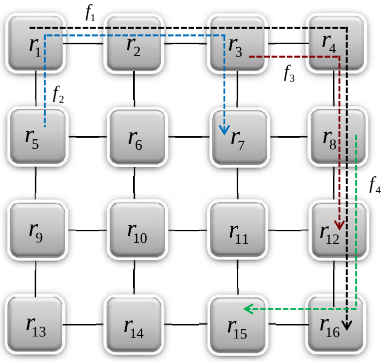

Fig. 1. An example of a NoC with 16 nodes and 4 flows.

- Buffers are bounded by Eq. (7) and the network is lossless.
- All traffic is modeled as TSPEC flows $f = TSPEC(L, p, \sigma, \rho)$ at the entry into the network.
- To characterize flows based on TSPEC, we assume unbuffered leaky bucket controllers (regulators) which do not buffer the packets, but stall the traffic producers or IPs [18].
- We assume weighted round robin arbitration and model it by a rate-latency service curve as $\beta = \delta_T \otimes \gamma_{0,R}$, it is assumed that $\rho \leq R$ and $p \geq R$, where ρ is the average rate, p the peak rate, and R the minimum service rate.
- Flows are classified into a pre-specified number of aggregates and traffic of each aggregate is buffered and transmitted in FIFO order, denoted as FIFO multiplexing.
- Different aggregates are buffered separately and each aggregate is guaranteed a rate-latency service curve.
- The hardware limits the peak rate to 1 *flit/cycle*.

Figure 1 depicts an example with 16 nodes and 4 flows. Multiple flows which share the same buffer and channel in the same router, for example f_1 and f_2 in router 2, are scheduled as an aggregate flow denoted as $f_{\{1,2\}}$. The *tagged flow* is a flow for which the delay bound is derived and the other flows which compete with the tagged flow for the same resource are called *contention flows*. In the example, if f_1 is the tagged flow, f_2, f_3, and f_4 would be contention flows. Table I presents notations in this work.

Sub-indices "(f_i, r_j)" indicate that they are related to flow f_i in router r_j. For instance, $\alpha_{(f_1, r_2)}$ indicates the arrival curve of f_1 in router r_2. Using f_{s_i} instead of f_i in the sub-index means that the notation is related to the f_{s_i} which can be one flow or an aggregate flow. For example, $\beta_{(\{1,2\}, r_2)}$ refers to aggregate flow $f_{\{1,2\}}$ in router r_2.

V. LUDB Derivation for WRR Policy

To derive a delay bound per flow passing a series of nodes, one simple way is to sum up the delay bounds at each node, which results in a loose delay bound. A theorem called *Pay Bursts Only Once* is known to give a tighter upper estimate on delay bounds, when an end-to-end service curve is obtained prior to delay computations. This accounts for bursts of the tagged flow only once instead of at each link independently. This principle also holds in aggregate scheduling networks.

TABLE I
THE LIST OF NOTATIONS

f_i	Flow i
$F^{RPV}_{(j,i,k)}$	The set of flows passing through VC k in physical channel i of router j
$F_{(j,l,s,k)}$	The set of flows passing from VC s of input channel l to output channel k in router j
Input PC#	The number assigned to an input physical channel
Output PC#	The number assigned to an output physical channel
VC#	The number assigned to an input virtual channel
$InPC$	The set of input physical channels in each router
$OutPC$	The set of output physical channels in each router
$InVC$	The set of input virtual channels in each input physical channel
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 the router and equals to the sum of T^l and T^{HoL}
D_{router}	Time spent for packet routing decision *(cycles)*
L_w	The word length in the flow *(flits)*
C	The channel capacity *(flits/cycle)*
CF_t	The set of contention flows of tagged flow f_t
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
$\|s_i\|$	The cardinality of set s_i, which is a measure of the ”number of elements of the set”
$F^B_{(s_i,r_j)}$	The set of flows which share the same buffer in router r_j with flow f_{s_i}
$w_{(j,l,s,k)}$	The weight assigned to node r_j, input Physical Channel (PC) l, input VC s, and output PC k
L_{WR}	The length of a round in WRR policy

To this end, we propose the two following steps to derive the end-to-end service curve for a tagged flow:

- **Step 1:** *Intra-router ESC*: This step derives intra-router ESCs for each router through which the tagged flow is passing. Different resource sharing scenarios in each router are distinguished and *intra-router analysis models* are built.
- **Step 2:** *Inter-router ESC*: In this step, according to the intra-router analysis models, we present a set-theoretic approach which recognizes and investigates different contention scenarios that a flow may experience along its routing path and in turn derive an end-to-end ESC for the tagged flow.

For extending our proposed analytical method to weighted round robin policy, we expand the first step while the second step is unchanged. Similarly, to support some other arbitration policies, only the first step must be modified.

A. Intra-router ESC

In this step, we consider three types of resource sharing, including channel&buffer sharing, channel sharing, and buffer sharing.

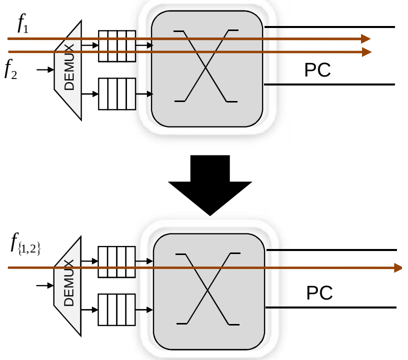

Fig. 2. An example of channel&buffer sharing

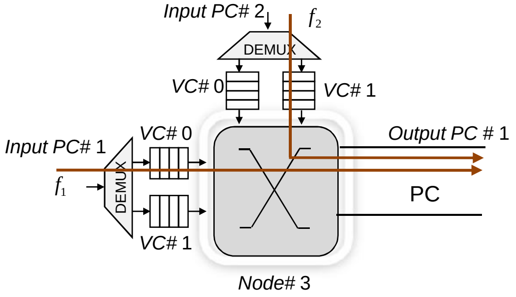

Fig. 3. An example of channel sharing

1) Channel&Buffer Sharing: As shown in Figure 2, multiple flows share both the same buffer and channel in the router, and are scheduled as a flow called aggregate flow. An aggregate flow including the tagged flow is named as tagged aggregate flow. In this case, intra-ESC is derived for the tagged aggregate flow instead of the tagged flow. In Section V-B, due to contention scenarios, we will remove contention flows from the ESC of a tagged aggregate flow in order to extract the ESC of the tagged flow.

2) Channel Sharing: Figure 3 depicts an example of a channel shared between two flows f_1 and f_2. The WRR arbiter associates a weight $w_{(j,l,s,k)}$ in cycles on each aggregate/single flow f_{s_i} passing from input VC s of input Physical Channel (PC) l in router r_j to output PC k. The value of the weight assigned to a channel depends on flows passing through that channel. Then, the router will try to give the flow a period of $w_{(j,l,s,k)}$ cycles before moving to the next node. In each round, for a non-empty VC buffer encountered, the router serves up to corresponding configured weight in cycles. The maximum length of a round consequently equals to $\sum_{l,s} w_{(j,l,s,k)}$ cycles, denoted as L_{WR}. The least service offered to one flow in a VC is completely dependent on the weight of that VC and the sum of all other weights. With the WRR scheduling, the worst case appears for a flow when it just misses its slot in the current round. Consequently it will have to wait for its slot assigned at the next round. In the worst case, each flow f_{s_i} passing from input VC s of input PC l in router r_j to output PC k will have to wait up to $\left(\sum_{p,q} w_{(j,p,q,k)} - w_{(j,l,s,k)}\right) \times \left(\frac{L_w}{C} + D_{router}\right)$ cycles before to be served, and get at least a $\frac{w_{(j,l,s,k)}}{\sum_{p,q} w_{(j,p,q,k)}} \times C$ of the channel bandwidth, where C is the channel capacity,

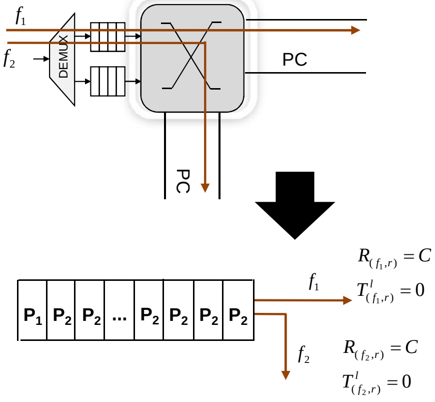

Fig. 4. An example of buffer sharing

L_w the word length, and D_{router} the delay for packet routing decision in a router. A flow may get more service rate if other flows use less, but we now know a worst-case lower bound on the bandwidth. Based on network calculus theory, we can use the abstraction of service curve to model a weighted round robin arbiter in router r_j for flow f_{s_i} as a rate-latency server $\beta_{(s_i,r_j)} = R_{(s_i,r_j)}(t - T^l_{(s_i,r_j)})^+$, 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:

$$R_{(s_i,r_j)} = \frac{w_{(j,l,s,k)}}{\sum_{p,q} w_{(j,p,q,k)}} \times C \qquad (1)$$

$$T^l_{(s_i,r_j)} = \left(\sum_{p,q} w_{(j,p,q,k)} - w_{(j,l,s,k)}\right) \times \left(\frac{L_w}{C} + D_{router}\right) \qquad (2)$$

In the example of Figure 3:

$$R_{(f_1,r_3)} = \frac{w_{(3,1,0,1)}}{w_{(3,1,0,1)}+w_{(3,2,1,1)}} \times C$$

$$T^l_{(f_1,r_3)} = w_{(3,2,1,1)} \times \left(\frac{L_w}{C} + D_{router}\right)$$

As aforementioned, the sum of weights of flows sharing a channel is equal to the round time. Thus, as the value of round time (the sum of weights) is increased or decreased, individual flows proportionally get more or less time slots respectively, which means the weights proportionally increased or decreased. Therefore, the model is able to adjust the weights based on the round time.

3) Buffer Sharing: Figure 4 depicts a buffer shared between two flows f_1 and f_2. In this type of sharing, we introduce two kinds of delay for a tagged flow including:

- *Head-of-Line delay (HoL)* is the maximum waiting time of the packet in the FIFO queue, which is denoted by T^{HoL}.
- *Processing delay* is the maximum processing latency of the router's arbiter for the flow, which is denoted by T^l.

Therefore, total delay for tagged flow f_i in router r_j is calculated as $T^{Total}_{(f_i,r_j)} = T^{HoL}_{(f_i,r_j)} + T^l_{(f_i,r_j)}$.

$T^l_{(f_i,r_j)}$ and $R_{(f_i,r_j)}$ can be calculated according to Equation (2) and (1), respectively. To show how $T^{HoL}_{(f_i,r_j)}$ is calculated, we consider the example in Figure 4 and assume that f_1 is the tagged flow. As depicted in the figure, $T^{HoL}_{(f_1,r)}$ is equal to the maximum delay for passing packets of flow f_2 in the buffer. According to [1], the maximum delay for flow f_j is bounded by Equation (3).

$$\bar{D}_{(f_j,r)} = T^l_{(f_j,r)} + \frac{L_j + \theta_j(p_j - R_{(f_j,r)})^+}{R(f_j,r)} \qquad (3)$$

where $\theta = (\sigma - L)/(p - \rho)$.

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

$$T^{HoL}_{(f_1,r)} = T^l_{(f_2,r)} - \theta_2 + \frac{L_2 + \theta_2 p_2}{R_{(f_2,r)}} \qquad (4)$$

In the case of more than one flow sharing the same buffer with the tagged flow, HoL delay for tagged flow f_{s_i} in router r_j is calculated as below:

$$T^{HoL}_{(s_i,r_j)} = \sum_{\forall f_c \in F^B_{(s_i,r_j)}} T^{HoL(f_c)}_{(s_i,r_j)} \qquad (5)$$

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}. Also $T^{HoL(f_c)}_{(s_i,r_j)}$ is given by

$$T^{HoL(f_c)}_{(s_i,r_j)} = T^l_{(f_c,r)} - \theta_c + \frac{L_c + \theta_c p_c}{R_{(f_c,r)}} \qquad (6)$$

Therefore router r_j can give flow f_{s_i} service bounded by curve $\beta_{(s_i,r_j)} = \delta_{T^{Total}_{(s_i,r_j)}} \otimes \gamma_{0,R_{(s_i,r_j)}}$, where $T^{Total}_{(s_i,r_j)}$ is equal to $T^{HoL}_{(s_i,r_j)} + T^l_{(s_i,r_j)}$ and $R_{(s_i,r_j)}$ is calculated by Equation (1).

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 k in physical channel i of router j is given as below:

$$B_{(j,i,k)} = \sum_{\forall f_c \in F^{RPV}_{(j,i,k)}} \Big(\sigma_c + \rho_c T^p_{(f_c,r_j)} + \left(\theta - T^p_{(f_c,r_j)}\right)^+ \times \left[\left(p_c - R_{(f_c,r_j)}\right)^+ - p_c + \rho_c\right]\Big) \qquad (7)$$

where $F^{RPV}_{(j,i,k)}$ is the set of flows passing through VC k in physical channel i of router j.

B. Inter-router ESC

In this step, we aim to extract ESC of the tagged flow by removing the contention flows from the ESC of the tagged aggregate flows. We have described this stage in elaborate detail through our previous paper [4]. For the sake of completeness, it is explained in the appendix. Algorithm 1 presents the main steps of deriving the end-to-end ESC for a given tagged flow. The only difference between this algorithm and the one presented for RR [4] results from the different methods proposed for calculating intra-router ESCs (Line 9). The algorithm with all stages, including details of *inter-router ESC* step, is presented in Appendix.

Now, we can obtain LUDB from end-to-end ESC according to the proposed theorem for calculating delay bounds [3]. We have automated our proposed analytical approach as a tool for worst-case performance analysis. The weighted RR gives flow isolation. Each flow is served at its own weight in the worst case. It is notable that the proposed approach is independent of the routing algorithm, but the routing algorithm must be predefined (deterministic).

Algorithm 1 End-to-End ESC Algorithm

```
Find the set of contention flows of tagged flow $f_t$, denoted by $CF_t$
for $\forall j \in CF_t$ do
    if $Src(j) \notin Path(t)$ then
        Find $joiningnode = JoiningPoint(f_j)$
        Calculate $X = ESC(f_j, Src(j), joiningnode)$
        $\alpha_j = \alpha_j \oslash X$
    end if
end for
Calculate intra-router ESC for WRR based on Section V-A.
Calculate inter-router ESC (See Appendix).
return end-to-end ESC for tagged flow $f_t$
```

In our proposed model, σ and ρ represent the congestion level. The effect of these parameters on delay bounds can be analyzed by following theorems and formula used in the proposed approach.

VI. Optimization Problem Formulation

Latency is one of the most critical challenges for on-chip interconnection network architectures [30]. However, there exists a huge search space to explore for minimizing latency. Thus, to design a low latency on-chip network, designers need to investigate optimization problems and make appropriate decisions. The general problem is defined as follows:

> **General Problem Definition**
> **Given** Architecture specifications, application parameters, and traffic characteristics (e.g. TSPEC in this paper);
> **Find** A set of decision variables;
> **Such that** network delay is minimized and performance constraints are satisfied.

Decision variables capture application mapping to processing cores, traffic regulation parameters (e.g. peak rate, burstiness, and packet injection rates to the network), switch architecture, a resource allocation strategy (e.g., bandwidth of channels, etc.), weight configuration in WRR policy, and a routing algorithm.

In networks with WRR policy, the weight configuration for flows can be in conflict because of contention for shared resources. This makes weight parameters non-trivial and thus, given single or multiple objectives, a parameter selection becomes necessary. In this paper, we find a weight configuration in WRR policy to minimize total worst-case delay in the network. Weight allocation is actually a resource allocation strategy in which a flow with larger weight gets more bandwidth or a higher service level. The weight of each non-empty VC is selected based on traffic specifications of flows passing through that VC, and to minimize interference between them. In Section IX, we describe how weights affect the delays of flows. For example, results for a real-time application show that an optimized weight allocation leads to about 48.8% reduction in total worst-case delay compared to a random configuration. Optimized WRR weight assignment leads to a 81.1% decrease of delay over a poor weight configuration and 15.4% decrease over a RR based allocation.

On the other hand, faster transmission is not necessarily better in a shared communication channel since faster delivery requires higher link bandwidth reservation and may incur a larger delay for another contention flow in a shared channel, leading to an intolerable delay. To avoid throttling some communications, we investigate another objective function which is minimizing the variance of delay bounds in different flows. As both mentioned goals are worthwhile for the real-time applications, we formulate them as a multi-objective problem in Section VI-B.

A. Delay Optimization

As stated before, our objective is to choose appropriate weights in a weighted round robin policy, assigned to channels on the path of flows, so as to minimize the sum of LUDBs while satisfying acceptable performance in the network. Note that $w_{(j,l,s,k)} = 0$ when no flow is passing from virtual channel s of input channel l to output channel k in router j. Thus, the delay bound minimization problem, *Minimize-Delay*, can be formulated as follows.

Given a set of flows $F = \{f_i \propto (L_i, p_i, \sigma_i, \rho_i)\}$, routing matrix R, the number of weight cycles L_{WR}, ***find*** the weights in weighted round robin policy as $w_{(j,l,s,k)}$ for $\forall i \in N$, $\forall j \in InPC$, $\forall s \in InVC$, and $\forall k \in OutPC$, such that

$$\min_{w_{(j,l,s,k)}} \sum_{\forall f_i \in F} D_i \tag{8}$$

subject to:

$$\textstyle\sum_{l,s} w_{(j,l,s,k)} = L_{WR} \qquad \forall j \in N; \forall k \in OutPC \tag{9}$$

$$\frac{L_{WR} \times \sum_{m \in F_{(j,l,s,k)}} \rho_m}{C} \leq w_{(j,l,s,k)} \leq L_{WR} \tag{10}$$

$$\forall j \in N, \forall l \in InPC, \forall s \in InVC, \forall k \in OutPC$$

where $w_{(j,l,s,k)}$ for $\forall j \in N$, $\forall l \in InPC$, $\forall s \in InVC$, and $\forall k \in OutPC$ are optimization variables.

Eq. (8) is the objective function of this optimization problem which minimizes total LUDBs. Constraint (9) says that the sum of weights assigned to flows which pass through the same output channel k in router j, the same weighted round robin scheduler, is equal to L_{WR}. Although we have assumed the same value of L_{WR} for all arbiters, the optimization problem can be easily adapted to different values of the sum of weights. To reach acceptable performance in the network, the share of $w_{(j,l,s,k)}$ from L_{WR} should be proportionate to $\frac{\sum_{m \in F_{(j,l,s,k)}} \rho_m}{C}$, where $F_{(j,l,s,k)}$ is the set of flows which pass through virtual channel s of input channel l to output channel k in router j. Therefore, we can consider $\frac{\sum_{m \in F_{(j,l,s,k)}} \rho_m}{C}$ as a criterion of minimum guaranteed performance for flows in $F_{(j,l,s,k)}$. In this respect, we have $\frac{\sum_{m \in F_{(j,l,s,k)}} \rho_m}{C} \leq \frac{w_{(j,l,s,k)}}{L_{WR}}$ which means $\frac{L_{WR} \times \sum_{m \in F_{(j,l,s,k)}} \rho_m}{C} \leq w_{(j,l,s,k)}$ as stated in Constraint (10). It is also clear that the value of each weight should be less than the number of weight cycles which means $w_{(j,l,s,k)} \leq L_{WR}$.

By following the equations described in Section V, the effect of optimization variables on the objective function of the defined problem is obvious.

In the literature, problem (8) is called a stochastic and nonlinear optimization problem [31]. We solve it using genetic algorithms because of their well-known robustness and ability to solve large and complex discrete optimization problems.

B. Multi-objective Optimization Problem

In order to avoid an intolerable delay of some flows due to processing and transmission of other flows, we would like to find appropriate weights in weighted round robin policy so that variance of delay bounds in the network is minimized. Using a general variance formula, we can calculate the variance of the delay bounds as $\frac{1}{|F|} \times \sum_{\forall f_i \in F} (E(D) - D_i)^2$. Hence, another optimization problem can be formulated to minimize both the total delay bounds and their variance while satisfying the constraints (9) and (10), as follows.

Given a set of flows $F = \{f_i \propto (L_i, p_i, \sigma_i, \rho_i)\}$, routing matrix R, the number of weight cycles L_{WR}, *find* the weights in weighted round robin policy as $w_{(j,l,s,k)}$ for $\forall j \in N$, $\forall l \in InPC$, $\forall s \in InVC$, and $\forall k \in OutPC$, such that

$$\min_{w_{(j,l,s,k)}} \sum_{\forall f_i \in F} D_i \tag{11}$$

$$\min_{w_{(j,l,s,k)}} \frac{1}{|F|} \times \sum_{\forall f_i \in F} (E(D) - D_i)^2 \tag{12}$$

subject to:

$$\textstyle\sum_{l,s} w_{(j,l,s,k)} = L_{WR} \qquad \forall j \in N; \forall k \in OutPC \tag{13}$$

$$\frac{L_{WR} \times \sum_{m \in F_{(j,l,s,k)}} \rho_m}{C} \leq w_{(j,l,s,k)} \leq L_{WR} \tag{14}$$

$$\forall j \in N, \forall l \in InPC, \forall s \in InVC, \forall k \in OutPC$$

Although the solution of multi-objective optimization problems consists of a set of solutions, the user needs only one solution. The decision about which solution is best depends on the *decision maker* and there is no universally accepted definition of *optimum* as in single-objective optimizations [36]. A multi-objective problem is often solved by composing the objective function as the weighted sum of the objectives which is in general known as the *weighted-sum* or *scalarization* method. In this approach, a relative preference factor of the objectives should be known in advance. In more detail, the weighted-sum method minimizes a positively weighted sum of the objectives, that is,

$$min(\gamma_1 f_1 + \gamma_2 f_2) \tag{15}$$

where γ_1 and γ_2 are the weighting coefficients representing the relative importance of the objectives.

The simplicity and efficiency of this method makes it an appropriate option for solving multi-objective optimizations with complex and nonsmooth objective functions. Therefore, we convert our proposed multi-objective problem into a scalar optimization problem with equal weighting coefficients. Since the problem is still a nonsmooth and stochastic optimization, we use the genetic algorithm to solve it.

VII. Solution Method

The proposed optimization problems have complex and highly nonlinear objective functions. Moreover, due to Eq. (1) and (19), minimization functions of decision variables appear in the formulation of per-flow LUDBs and consequently in the objective formulation which cause random objective functions.

Such optimization problems are usually solved by metaheuristic methods which make few assumptions about the problem being solved and do not guarantee an optimal solution. However, they can usually find a good solution [5].

Algorithm 2 A General Scheme of GA in Pseudo-code

```
1: $P1 \leftarrow$ Generate random population of $n$ chromosomes
2: Evaluate the fitness $f(x)$ for each $x \in P1$
3: repeat                                        ▷ Create a new population
4:     Selection: Select two parents from a population.
5:     Crossover: With a crossover probability cross over the
         parents to form a new offspring (children).
6:     Mutation: With a mutation probability mutate new
         offspring at each locus (position in chromosome).
7:     Accepting: Place new offspring in a new population
8: until the new population is not complete
9: Use new generated population for a further run.
10: if the end condition is satisfied then
11:     return The best solution in current population
12: else
13:     Go to step 2
14: end if
```

Among different types of metaheuristics, we choose genetic algorithms to solve the proposed optimization problems because they are most appropriate for large and complex nonlinear models specially where the objective function is discontinuous, stochastic, very rugged and complex, noisy, or has many local optima [37] [38]. Moreover, they have been proven to be effective in avoiding local optima and discovering the global optimum in even a problem with very complex objective functions [37]. GAs tend to be computationally expensive for the solutions of optimization problems with nonlinear equality and inequality constraints [38], which do not occur in our proposed problems. Although a GA does not always find a global optimum to a problem, it almost always finds high-quality solutions [37].

GA generates solutions to optimization problems mimicking the process of natural evolution such as inheritance, mutation, selection, and crossover. Algorithm 2 presents a general scheme of GA in pseudo-code. The algorithm is started with an initial population of solutions represented by *chromosomes*. A chromosome contains the solution as a set of parameters in form of *genes*. A gene is a position or set of positions in a chromosome, represented as a simple string or other data structures. The algorithm selects solutions, called *parents*, from the population and produces a new solution, called *offspring*, to form a new population. Although parents can be selected in many different ways, the main idea is that better parents according to their *fitness* hopefully will produce better offspring. *Crossover* and *mutation* are two basic operators of GA which produce a new offspring. This process is repeated until some condition, such as the number of populations or improvement of the best solution, is satisfied.

A method for encoding potential solutions of the problem is needed. There are different approaches to encode solutions like binary encoding, value encoding, permutation encoding, and tree encoding.

VIII. Implementation

In Section VII, we have discussed why GA is selected for solving the optimization problems and presented a general description of GA. This section shows how we have mapped our proposed optimization problems into the problems which can be solved by GA. We present Algorithm 3 to detail

Algorithm 3 Genetic Algorithm based Weight Optimization

```
1: $Pop1 \leftarrow Initilization\_FirstPopulation()$
2: $Encoded\_Pop1 \leftarrow Encoding(Pop1)$
3: $Temp\_Pop \leftarrow Encoded\_Pop1$
4: for i=1 to Iteration# do
5:     $New\_Pop[0] \leftarrow Elitism(Lb, Ub)$
6:     for j=1 to Pop_Size do
7:         $Cross\_Rate \leftarrow MersenneTwister()$
8:         if $(Cross\_Rate \leq Cross\_Prob)$ then
9:             $Chromosome1 \leftarrow Selection(Lb, Ub)$
10:            $Chromosome2 \leftarrow Selection(Lb, Ub)$
11:            $Offspring \leftarrow Crossover(Chromosome1, Chromosome2)$
12:        else
13:            $Offspring \leftarrow Selection(Lb, Ub)$
14:        end if
15:        $Mut\_Rate \leftarrow MersenneTwister()$
16:        if $(Mut\_Rate \leq Mut\_Prob)$ then
17:            $Offspring \leftarrow Mutation(Offspring)$
18:        end if
19:        $New\_Pop[j] \leftarrow Offspring$
20:    end for
21:    $Temp\_Pop \leftarrow New\_Pop$
22: end for
23: $Decoded\_Pop \leftarrow Encoding(Temp\_Pop)$
24: $Optimal\_Weight \leftarrow Minimum(Decoded\_Pop)$
25: return $Optimal\_Weight$
```

the procedure of deriving optimal weights for the proposed optimization problems. Objective and constraint functions in GA are the same as what we have defined for the proposed optimization problems. Objective functions are implemented as *Fitness* function which is called whenever the population is created or a selection is made from the population. Weights, which are decision variables, are considered as a vector and uniquely mapped onto a *chromosome*. As the proposed optimizations have only boundary constraints, these constraints in GA can be reflected as intervals of chromosomes' domain. *Parent*, which is a chromosome, presents the current solution for this round and offspring is a new vector generated from the parent which may be the next solution.

The algorithm uses a binary representation of *chromosomes* as fixed-length strings over the alphabet $\{0, 1\}$, such that they are well suited to handle the optimization problems. It uses function $Encoding()$ to map solutions $\vec{w} \in W$ to a binary string $\{0, 1\}^l$ and defines function $Decoding()$ to do the reverse. To this end, real-valued vector $\vec{w} \in \Re^n$ is presented by a chromosome in form of a binary string $\vec{x} \in \{0, 1\}^l$. The chromosome is logically divided into n segments (*gene*) of equal length S_{gene} as $(w_1 ... w_n)$, where S_{gene} is gene size and $l = n \times S_{gene}$. Each gene w_i is decoded to yield the corresponding integer value, and the integer value is in turn linearly mapped to its interval of real values, denoted as $[Lb_i, Ub_i] \subset \Re$, where Lb_i and Ub_i indicate lower and upper bound constraints on w_i, respectively. In this work, we use a *gray code* interpretation of the binary string. The main advantage of gray codes is that they are different by only one bit.

Figure 5 shows an example of the decoding process for string segments of length $S_{gene} = 8$ which allows the representations of integers $\{0, 1, ..., 255\}$. As shown in the figure, function $Decoding()$ first converts a given gray code to an integer value $p_i \in \{0, ..., 2^{S_{gene}} - 1\}$ and then maps p_i linearly to its corresponding interval $[Lb_i, Ub_i]$ as $Lb_i + \frac{Ub_i - Lb_i}{2^{S_{gene}} - 1} \times p_i$.

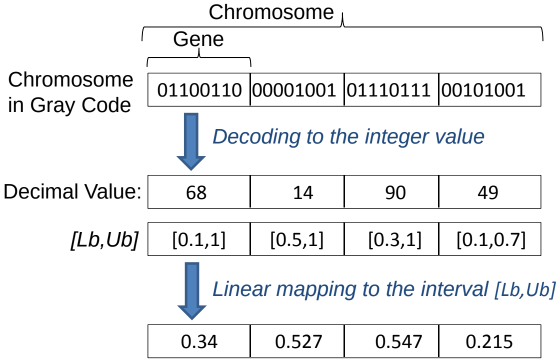

Fig. 5. An example of decoding and linear mapping

After encoding, the algorithm starts producing a new population in Line 5-20. Function $Elitism()$ in Line 5 copies the best chromosome of the current population to the new population, so the best chromosome found can survive. Elitism can very rapidly increase performance of GA, because it prevents losing the best found solution. To create other new offsprings, three basic operators including *selection*, *crossover*, and *mutation* are applied as follows.

Selection in GA means how to select parents for crossover or mutation. The main idea is to select the better parents in hope that the better parents will produce better offspring. Thus, function $Selection()$ in the algorithm selects randomly two chromosomes from the current population, evaluates their fitness values, and finally returns the one which has the smaller fitness value as one of parents. Another parent is selected in the same way.

$Cross\_Prob$ in Line 8 is the crossover probability which states how often a crossover is performed. If there is a crossover, two parents' chromosomes are selected and offspring is made from their crossover. If there is no crossover, offspring is the exact copy of a chromosome from the old population. Due to $Cross\_Prob$, the new generation is a mix of offsprings made by crossovers and chromosomes from the old population. Although crossovers have the tendency to improve chromosomes, it has been shown to be beneficial to keep part of the old population.

Crossover selects genes from parents' chromosomes and creates a new offspring. There are different ways to make a crossover. This algorithm chooses randomly two crossover points and everything before the first point and after the second point is copied from the first parent and the section between the two crossover points is copied from the second parent. Figure 6 shows an example of crossover applied in this algorithm (| denotes the crossover point).

After crossover, *mutation* is performed. Mut_Prob in Line 16 is the mutation probability which states how often a chromosome is mutated. If mutation is performed, parts of chromosome are changed. If there is no mutation, the offspring is copied after crossover without any change. Mutation is made to prevent an entire population being trapped in a local

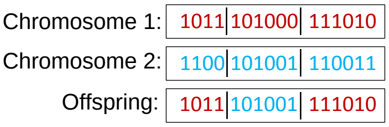

Fig. 6. An example of crossover

optimum. Mutation in Algorithm 3 changes the new offspring by randomly switching a few bits. It is worth mentioning that the mutation should not occur very often, because then GA will convert into a random search. Figure 7 shows an example of mutation used in the algorithm.

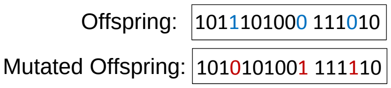

Fig. 7. An example of mutation

This process repeats for a specified number of iterations.

As shown in Figure 8, we have developed a tool in C++, divided into two main sub-tools including *End-to-End Delay Program* and *Optimization Program*. The former derives per-flow worst-case bounds by applying the proposed formal approach in Section V. The bounds are represented as functions of weights in WRR policy. The latter optimizes weights in WRR policy based on the optimization problem formulated in Section VI. Input for the first sub-tool includes an application communication graph, specification of flows, topology graph, routing matrix, and characteristics of routers. The outputs from the first sub-tool along with the set of system constraints will be inputs for the second part. If flows or traffic pattern are changed, per-flow end-to-end delay bounds and optimization problem need to be resolved. Since we aim for a design phase tool, it is executed once for static flows.

IX. Experimental Results

To evaluate the potential of our method, we applied it to two real applications and a synthetic traffic pattern on a larger network.

A. VOPD Application

We have applied our model to a real-time multimedia application with a random mapping to the tiles of a 4×4 on-chip network. Figure 9 shows the task graph and flow mapping of a Video Object Plane Decoder (VOPD) [39] 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 [40]. There are 21 communication flows characterized by TSPEC.

Hence, each flow i is characterized by $(L_i, p_i, \sigma_i, \rho_i)$. The maximum transfer size and peak rate refer to the real traffic flow over the flit channel between routers. They are constrained by the flit channel capacity. Packets may have different burst sizes. They are sent flit by flit over the flit channel. This means the maximum transfer size of $1\ flit$ and peak rate $1\ flit/cycle$. Therefore, 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 9 and σ_i varies between 8 and 128 $flits$ for different flows. The length of a round in WRR scheduling, L_{WR}, is assumed to be 10 *cycles*.

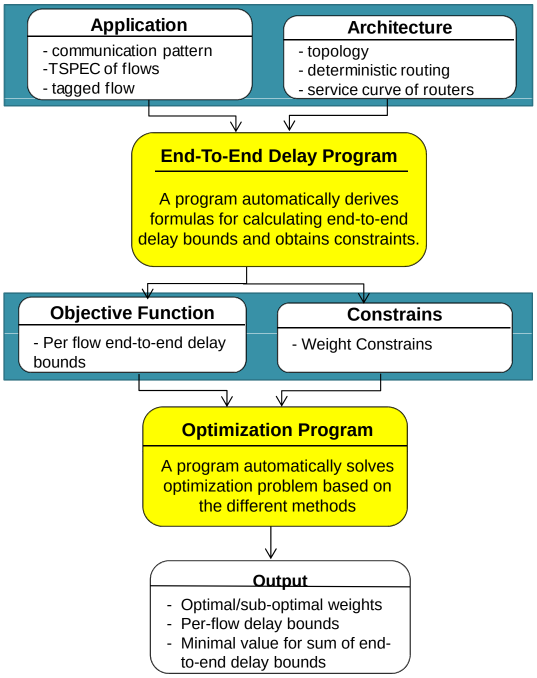
Application
- communication pattern
-TSPEC of flows
- tagged flow
Architecture
- topology
- deterministic routing
- service curve of routers
End-To-End Delay Program
A program automatically derives formulas for calculating end-to-end delay bounds and obtains constraints.
Objective Function
- Per flow end-to-end delay bounds
Constrains
- Weight Constrains
Optimization Program
A program automatically solves optimization problem based on the different methods
Output
- Optimal/sub-optimal weights
- Per-flow delay bounds
- Minimal value for sum of end-to-end delay bounds

Fig. 8. The flow chart of the developed tool

1) Delay Optimization: As mentioned before, decision variables in the proposed optimization problems are the weights on shared channels. Due to shared channels in VOPD application, 20 weights are formulated in the optimizations as a weight vector W defined as follows:

$$\begin{aligned} W = \big(& w_{(6,3,0,4)}, w_{(10,2,0,0)}, w_{(14,0,0,2)}, w_{(13,3,0,2)}, w_{(12,0,0,2)}, \\ & w_{(9,4,0,0)}, w_{(4,3,0,4)}, w_{(4,0,0,2)}, w_{(8,2,0,0)}, w_{(8,4,0,2)}, \\ & w_{(6,2,0,4)}, w_{(10,4,0,0)}, w_{(14,3,0,2)}, w_{(13,1,0,2)}, w_{(12,3,0,2)}, \\ & w_{(9,3,0,0)}, w_{(4,0,0,4)}, w_{(4,4,0,2)}, w_{(8,4,0,0)}, w_{(8,0,0,2)} \big) \end{aligned} \tag{16}$$

The "End-to-End Delay Program" calculates per-flow worst-case bounds as functions of weights for each flow in VOPD application and derives corresponding constraints. The "Optimization Program" formulates *Minimize-Delay* problem and derives weights for VOPD application.

To show how these weights affect the communication delay, we consider four different schemes:

- Random Scheme: The weights are selected randomly.
- *Round Robin Scheme*: The weights have the same values to represent round robin policy.
- *Optimized Scheme*: The weights are optimized based on the optimization problem (8).
- *Unoptimized Scheme*: The weights are not optimized and there are many unoptimized configurations. In this scheme, we allocate weights so as to maximize the optimization problem (8) instead of minimization.

Then, the total maximum delay are calculated for different schemes and depicted in Table II. From this table, we can

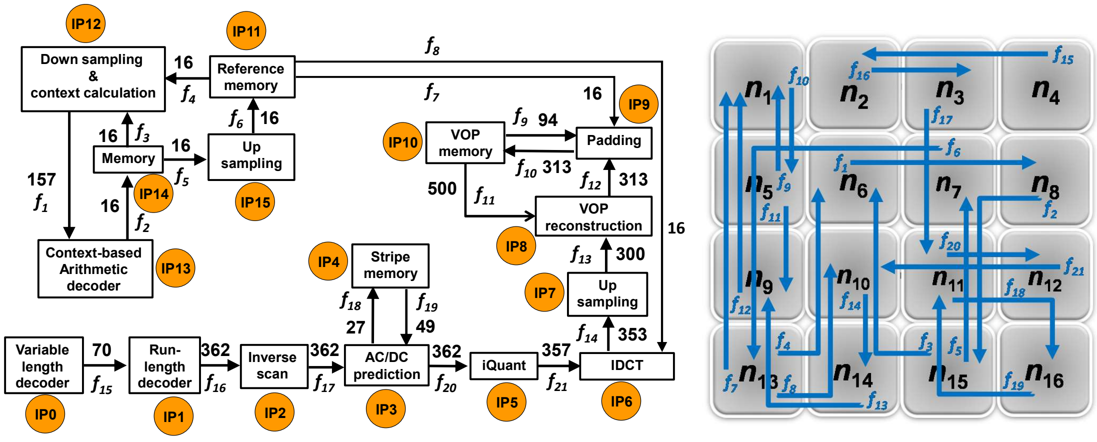

Fig. 9. VOPD Application

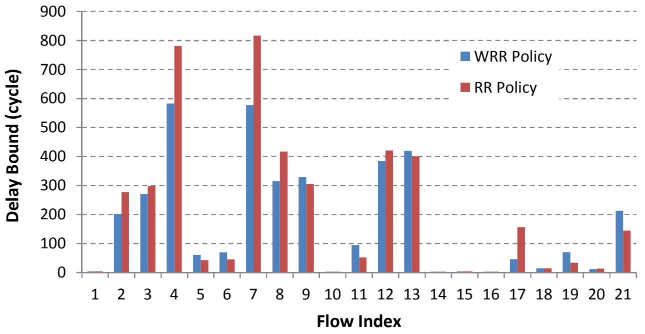

Fig. 10. Maximum worst-case delay for every flow in VOPD application

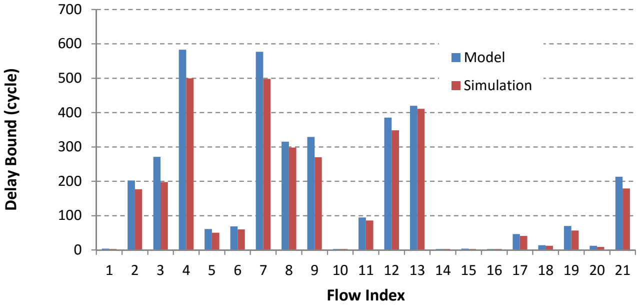

Fig. 11. Comparison of delay bounds from the proposed model and simulator for VOPD application

see that the optimized scheme leads to about 15.4%, 48.8%, and 81.1% reduction in total maximum delay when compared with *Round Robin*, *Random*, and *Unoptimized* schemes, respectively. The results show that although WRR is able to make better performance in terms of latency than RR scheduling, if the weights are not allocated properly, it may be worse. Therefore, an appropriate weight configuration makes WRR able to reduce total and average maximum delay by balancing the allocation of shared network bandwidth to different traffic flows with respect to their specifications and contentions for shared resources.

To better understand the effects of the weights, per-flow delay bounds for RR and WRR with the *Optimized* scheme are shown in Fig 10. This figure illustrates that flows in WRR can experience longer or shorter delays than the RR scheme which depends on the amount of network bandwidth allocated to each flow (due to the assigned weights). However, from Table II, we can see that the total and average worst-case delay are decreased in WRR with the *Optimized* scheme because the weights are assigned in a way to minimize total delay, satisfy performance constraints, and reduce contentions for shared resources leaving room for other contention interfering flows. Therefore, WRR can be used to control the per-flow delay bound by controlling its assigned weight.

It is worth mentioning that RR is a special case of WRR (all weights equal) and will most likely be found by the optimization algorithm when it is preferable according to the defined optimization objectives.

We have investigated the accuracy of the proposed analytical model with the BookSim simulator in our previous work [4]. However, as we have extended the model to WRR policy, we compare per-flow delay bounds obtained from the analytical model and BookSim simulator [32] for the *Optimized* scheme. The simulation uses the same assumptions as explained in [4]. As shown in Fig. 11, 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.

We have also computed the relative errors with respect to

TABLE II
HOW GOOD ARE OPTIMIZED WEIGHTS?

Scheme Type	Weight Vector	Total Worst-case Delay (*cycles*)	Average Worst-case Delay (*cycles*)
Optimized	$(2, 8, 8, 2, 6, 6, 4, 2, 3, 6, 8, 2, 2, 8, 4, 4, 6, 8, 7, 4)$	3671	174
Round Robin	$(5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5)$	4237	202
Random	$(1, 4, 2, 7, 2, 3, 9, 5, 8, 5, 9, 6, 8, 3, 8, 7, 1, 5, 2, 5)$	7177	343
Unoptimized	$(1, 1, 1, 9, 1, 1, 9, 9, 9, 5, 9, 9, 9, 1, 9, 9, 1, 1, 1, 5)$	19432	926

simulation results to consider the accuracy of the analytical model. The calculations show that the maximum and average relative errors are about 33.33% and 16.3%, respectively.

Monitoring the delay of packets shows that worst-case delay is much larger than average-case delay, which is reasonable because worst-case bounds cover corner cases. We have also noticed that traffic burstiness has the most influence on the delay distribution. The larger the burstiness, the larger the delay variance.

2) Multi-objective Optimization: In the multi-objective optimization minimizing delay and variance, we have calculated two parameters: *Total Worst-case Delay* and *Variance* listed in Table III. As can be observed from Table III, Minimize-Delay problem guarantees that weight allocation is carried out in favor of minimizing total worst-case delay while there is no such guarantee for the variance over various flows. In contrast, the Multi-objective optimization provides a trade-off between such parameters.

Under the Multi-objective optimization the standard deviation is less than 0.89 of the average delay because variance is an explicit target for minimization. It is also fairly small under the Minimize-Delay objective (standard deviation < 1.1 of the average delay) because greater imbalances of flows (bigger variance) tend to lead to worse contention between flows and thus to higher average delays. Hence, the Minimize-Delay algorithm implicitly tends to reduce variance as well.

Although we have assumed the same importance for total delay and variance in the multi-objective problem by considering the same weighting coefficients in Equation (15), it is possible for designers to change the value of the weighting coefficients γ_1 and γ_2 to specify another relative importance of objective functions.

3) Comparing with Other Solution Methods: As a comparative study, we implement three other metaheuristics, namely *Pure Random Search (PRS)* [33], *Markov Monotonous Approach (MMA)* [34], and *Adaptive Search (AS)* [34] to compare them with the genetic algorithm in terms of run-time and efficiency. These algorithms belong to a category of metaheuristics called trajectory-based methods. A trajectory-based algorithm works on single solutions at any time, namely, it starts from an initial state (initial solution) and follows a trajectory to reach a successor solution which may or may not belong to the neighborhood of the current solution. Population-based metaheuristics, on the contrary, deal with a set (a population) of solutions in each iteration and in turn provide an intrinsic method for exploring the search space. The way of manipulating the population has a significant impact on the performance of these methods. Genetic algorithms belong to this category. We also extend *PRS*, *MMA*, and *AS* to support a population of solutions instead of a single solution. Hereby, they produce m solutions in every iteration and select n solutions for the next iteration. The extended versions of *PRS*, *MMA*, and *AS* are called *PRS* $(m + n)$, *MMA* $(m + n)$, and *AS* $(m + n)$. Table IV presents the iteration number and run time required for solving the optimization problem (Eq. 8).

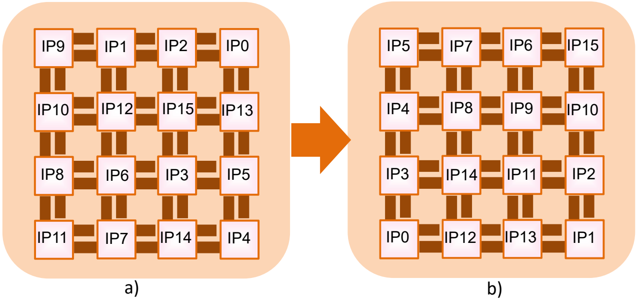

Fig. 12. Two different mappings for VOPD application

The results show that all metaheuristics presented in this table obtain the same solution for the problem. Therefore, we can say with some confidence that the solution is of high quality.

The table shows that the genetic algorithm has a shorter execution time with fewer iterations. GA is no exhaustive optimization method. However, as it is well known that GAs provide an efficient and robust method for solving problems in which the objective function is discontinuous, nondifferentiable, or highly nonlinear and due to the results from table IV, we believe that GA is a well suited solution method for our problem.

4) Comparing with An Optimized Mapping: By this point, we have considered a random mapping for VOPD application as shown in Fig. 12a). To show how a good mapping affect the results from our approach, we take the optimized mapping shown in Fig. 12b) from PERMAP algorithm [42]. Table V presents *Total Worst-case Delay* parameter derived from our approach for different scenarios on these two mappings. As it can be seen, applying our technique along with a good mapping can give much better results in terms of delay minimization.

TABLE III
HOW GOOD IS MULTIOBJECTIVE OPTIMIZATION?

	Weight Vector	Total Worst-case Delay *(cycles)*	Variance
Round Robin	$(5, 5, 5, 5, 5, 5, 5, 5, 5, 5,$ $5, 5, 5, 5, 5, 5, 5, 5, 5, 5)$	4237	59324.49
Minimize-Delay	$(2, 8, 8, 2, 6, 6, 4, 2, 3, 6,$ $8, 2, 2, 8, 4, 4, 6, 8, 7, 4)$	3671	35416.67
Multi-objective	$(6, 9, 9, 1, 6, 7, 2, 1, 1, 6$ $4, 1, 1, 9, 4, 3, 8, 1, 1, 6)$	4045	29320.71

TABLE IV
COMPARISON OF THE RUN TIME FOR DIFFERENT METHODS

Optimal point obtained by the methods		
Optimal Weight Vector $(2, 8, 8, 2, 6, 6, 4, 2, 3, 6, 8, 2, 2, 8, 4, 4, 6, 8, 7, 4)$		Total Delay 3671 *cycles*
Performance in different methods		
	Iteration#	Time *(sec)*
PRS	$100,000$	2.71
MMA	$100,000$	2.8
AS	$100,000$	2.85
PRS $(10 + 10)$	$5,000$	13
MMA $(10 + 10)$	$5,000$	13.37
AS $(10 + 10)$	$5,000$	12.83
GA	250	1.05

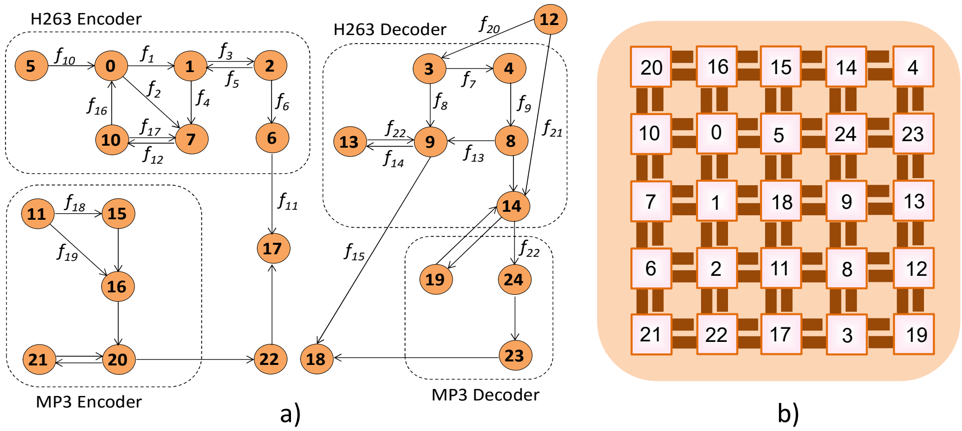

Fig. 13. MMS Application

B. Mutlimedia Application

We have evaluated our method for a generic MultiMedia System (MMS). MMS is an integrated video/audio system which includes an H263 video encoder, an H263 video decoder, an MP3 audio encoder, and an MP3 audio decoder. This application can be partitioned into 40 concurrent tasks and then these tasks are assigned onto 25 selected IPs [43]. These IPs range from DSPs, generic processors, and embedded DRAMs to customized application specific integrated circuits (ASICs). Real video and audio clips are used as inputs to derive the communication patterns among these IPs. Fig. 13a) reveals the communication task graph of this system [43]. We have applied the PERMAP algorithm [42] to get an optimized mapping for this system as shown in Fig. 13b).

Due to shared channels in the MMS application, weight vector W consisting of 37 weights is defined as below:

$$
\begin{aligned}
W = \Big(& w_{(16,1,0,0)}, w_{(15,3,0,2)}, w_{(15,0,0,2)}, w_{(5,3,0,0)}, w_{(14,4,0,2)}, \\
& w_{(19,1,0,2)}, w_{(18,4,0,2)}, w_{(18,3,0,2)}, w_{(13,2,0,4)}, w_{(23,3,0,0)}, \\
& w_{(14,1,0,4)}, w_{(10,2,0,4)}, w_{(1,0,0,4)}, w_{(1,3,0,4)}, w_{(11,4,0,0)}, \\
& w_{(6,0,0,3)}, w_{(13,1,0,0)}, w_{(8,2,0,0)}, w_{(2,4,0,0)}, w_{(21,3,0,0)}, \\
& w_{(16,4,0,0)}, w_{(15,4,0,2)}, w_{(5,4,0,0)}, w_{(14,0,0,2)}, w_{(19,4,0,2)}, \\
& w_{(18,0,0,2)}, w_{(13,3,0,4)}, w_{(23,2,0,0)}, w_{(14,2,0,4)}, w_{(10,0,0,4)}, \\
& w_{(1,1,0,4)}, w_{(11,2,0,0)}, w_{(6,1,0,3)}, w_{(13,4,0,0)}, w_{(8,4,0,0)}, \\
& w_{(2,1,0,0)}, w_{(21,2,0,0)} \Big)
\end{aligned}
\tag{17}
$$

Fig. 14 depicts delay bounds for each flow under the RR policy and the optimized WRR scheme with *Minimize-Delay*. Although flows in WRR can experience longer or shorter delays than under the RR scheme, the optimized WRR scheme guarantees an appropriate weight allocation in terms of minimizing total worst-case delay.

For more detail, we have presented two parameters *Total Worst-case Delay* and *Variance* for different defined scenarios in Table VI. As explained before, *Minimize-Delay* problem allocates weights such that guarantees total worst-case delay minimization and Multi-objective problem provide a trade-off between these two parameters.

TABLE V
COMPARISON OF TOTAL WORST-CASE DELAY WITH THE RANDOM AND OPTIMIZED MAPPINGS

	Random Mapping	PERMAP Mapping
Round Robin	4237	2385
Minimize-Delay	3671	2159
Multi-objective	4045	2544

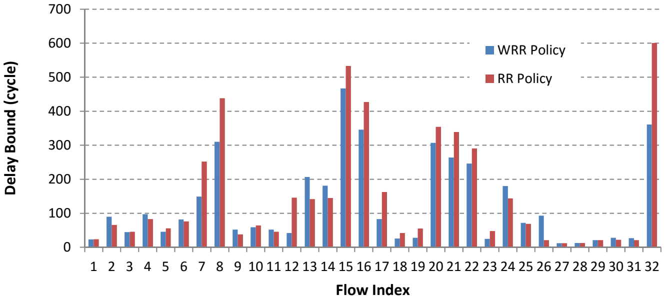

Fig. 14. Maximum worst-case delay for every flow in MMS application

TABLE VI
COMPARISON AMONG DIFFERENT SCENARIOS FOR MMS APPLICATION

	Weight Vector	Total Worst-case Delay (*cycles*)	Variance
Random Scheme	(5, 4, 4, 7, 9, 3, 1, 1, 9, 9, 7, 4, 5, 2, 4, 2, 9, 9, 2, 5, 5, 2, 3, 1, 7 8, 1, 1, 3, 6, 3, 6, 8, 1, 1, 8, 5)	13509	671014
Round Robin	(5, 3, 3, 5, 5, 5, 3, 3, 5, 5, 5, 5, 3, 3, 5, 5, 5, 5, 5, 5, 5, 4, 5, 5, 5 4, 5, 5, 5, 5, 4, 5, 5, 5, 5, 5, 5)	4783	26324.9
Minimize-Delay	(5, 3, 4, 7, 2, 8, 1, 4, 5, 4, 7, 5, 5, 2, 4, 2, 1, 4, 2, 5, 5, 3, 3, 8, 2 5, 5, 6, 3, 5, 3, 6, 8, 9, 6, 8, 5)	4034	14832.5
Multi-objective	(9, 1, 1, 8, 2, 8, 1, 4, 4, 4, 2, 9, 1, 1, 1, 5, 1, 1, 6, 1, 1, 8, 2, 8, 2 5, 6, 6, 8, 1, 8, 9, 5, 9, 9, 4, 9)	4855	10628.4

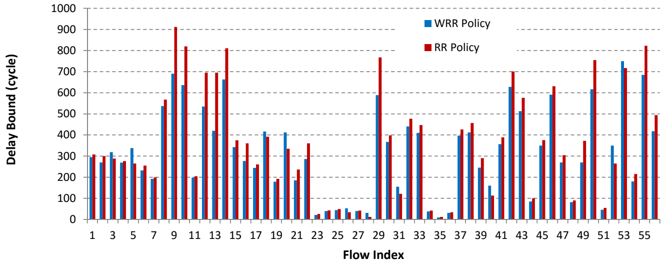

Fig. 15. Maximum worst-case delay for every flow under transpose traffic pattern

C. Transpose Traffic Pattern

To investigate a larger network, we experiment a 8×8 mesh network under the transpose traffic pattern with 56 communication flows. The values of L and p are assumed 1 $flit$ and 1 $flit/cycle$, respectively. For different flows, ρ varies between 0.001 and 0.03 $flits/cycle$, and σ 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, we depict per-flow worst-case delay bounds for RR policy and the optimized WRR scheme in Fig. 15. Regarding this figure, it is apparent that some flows in the optimized WRR policy may suffer longer delays than RR scheme. However, total delay bound in the optimized WRR scheme is equal to 17610 *cycles* while it is 19761 *cycles* in RR scheme. These values confirm that an appropriate weight allocation can guarantee total delay bound minimization in the network.

X. Conclusions

We have extended our earlier proposed methodology [4] for deriving per-flow delay bounds under RR policy to WRR scheduling and then compared them. We have developed algorithms to automate analysis steps. It is notable that the proposed methodologies for both RR and WRR do not deal with the back-pressure, but we have calculated the buffer size thresholds to make sure the back-pressure does not occur in the network. Based on these analytical models, we have presented two optimization problems for weight allocation in WRR scheduling, one for minimizing the total worst-case delays and one for minimizing both total worst-case delays and their variance under performance requirements to control per-flow delay bounds. We have also demonstrated that the proposed model exerts significant impact on communication performance. The algorithm for solving the proposed minimization problems runs very fast. For the case study, the optimized solution is found within about one second. The contribution of this paper is providing a performance evaluation tool for designers to make a good decision at the design phase. The algorithm can be performed at run time as it is quite fast but it needs a control infrastructure to get feedback from the network and distribute the results. On the other hand modifying weights at run time is not easy. The way of applying the algorithm at run time can be considered as a future work. In the future, we intend to investigate other scheduling policies. We also plan to extend the proposed analytical method in case of back-pressure in the network. Zhao and Lu [41] propose analytical models to derive worst-case bounds for constant bit rate flows due to back-pressure in the network. The current work does not consider speculative VC-allocation/switch allocation techniques. Extending the model to adjust these techniques can be considered as another future work. In this paper, we have assumed virtual-cut-through switching as the model is suitable for NoCs with small packets only. Small packet NoCs are a relevant and important, even in practice. Extension of the model to support wormhole routing is under our investigation. There are currently some papers in our group on wormhole routing but they consider only average behavior of flows [19] [44].

References

[1] J. Y. L. Boudec and P. Thiran, *”Network Calculus: A Theory of Deterministic Queuing Systems for the Internet”*, Number 2050 in LNCS, 2004.

[2] F. Jafari, A. Jantsch, and Z. Lu, ”Output Process of Variable Bit-Rate Flows in On-Chip Networks Based on Aggregate Scheduling”, *in Proc. of the International Conference on Computer Design (ICCD)*, pp. 445-446, 2011.

[3] F. Jafari, A. Jantsch, Z. Lu, ”Worst-Case Delay Analysis of Variable Bit-Rate Flows in Network-on-Chip with Aggregate Scheduling”, *in Proc. of Design, Automation and Test in Europe Conference (DATE)*, pp. 538-541, 2012.

[4] F. Jafari, Z. Lu, and A. Jantsch, ”Least Upper Delay Bound for VBR Flows in Networks-on-Chip with Virtual Channels”, ACM Transactions on Design Automation of Electronic Systems (TODAES), Vol. 20, No. 3, Article No. 35, June 2015.

[5] C. Blum and A. Roli, ”Metaheuristics in combinatorial optimization: Overview and conceptual comparison”, *ACM Comput. Surv.*, Vol. 35, No. 3, pp. 268-308, 2003.

[6] A. E. Kiasari, A. E., Jantsch, A., and Z. Lu, Mathematical formalisms for performance evaluation of networks-on-chip, *ACM Computing Surveys*. Vol. 45, No. 3, Article No. 38, 2013.

[7] Y. Ben-Itzhak, I. Cidon, A. Kolodny, Average latency and link utilization analysis of heterogeneous wormhole NoCs, *Integration, the VLSI Journal*, Vol. 51, Issue C, pp. 92-106, 2015

[8] Qian et al. ”A Support Vector Regression (SVR) based Latency Model for Network-on-Chip (NoC) Architectures”, *IEEE Transactions on CAD*, Vol. PP, No. 99, pp. 1

[9] Qian et al. ”A comprehensive and accurate latency model for Network-on-Chip performance analysis,” *in Proc. of Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 323-328, 2014.

[10] Qian et al. ”SVR-NoC: a performance analysis tool for network-on-chips using learning-based support vector regression model”. *In Proc. of Design, Automation and Test in Europe (DATE)*, pp. 354-357, 2013.

[11] P. Bogdan, M. Kas, R. Marculescu, O. Mutlu, ”QuaLe: A Quantum-Leap Inspired Model for Non-stationary Analysis of NoC Traffic in Chip Multi-processors”, *in Proc. of the International Symposium on Networks-on-Chip (NOCS)*, pp. 241-248, 2010

TABLE VII
The list of flows under transpose traffic pattern

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

[12] P. Bogdan, R. Marculescu, ”Workload Characterization and Its impact on Multicore Platform Design”, *in Proc. of Hardware/software codesign and system synthesis (CODES+ISSS)*, pp. 231-240, 2010.

[13] P. Bogdan, ”Mathematical Modeling and Control of Multifractal Workloads for Data-Center-on-a-Chip Optimization”, in Proc. of Symposium on Networks-on-Chip (NOCS), Article No. 21, 2015.

[14] Y. Xue and P. Bogdan, ”User Cooperation Network Coding Approach for NoC Performance Improvement”, *in Proc. of Symposium on Networks-on-Chip (NOCS)* Article No. 17, 2015.

[15] A. Charny and J.L. Boudec, ”Delay Bounds in a Network with Aggregate Scheduling”, *in Proc. of QofIS*, pp.1-13, 2000,

[16] Y. Jiang, ”Delay bounds for a network of guaranteed rate servers with FIFO aggregation”, *Computer Networks*, Vol. 40, No. 6, pp. 683-694, 2002.

[17] L. Lenzini, L. Martorini, E. Mingozzi, and G. Stea, ”Tight end-to-end per-flow delay bounds in fifo multiplexing sink-tree networks”, *Performance Evaluation*, Vol. 63, No. 9, pp. 956-987, 2006.

[18] F. Jafari, Z. Lu, A. Jantsch, and M. H. Yaghmaee, ”Buffer Optimization in network-on-Chip through Flow Regulation”, IEEE Transactions on CAD, Vol. 29, No. 12, pp. 1973-1986, 2010.

[19] Y. Qian, Z. Lu, and Q. Dou, ”QoS Scheduling for NoCs: Strict Priority Queueing versus Weighted Round Robin”, *in Proc. of the 28th International Conference on Computer Design (ICCD)*, pp. 52-59, 2010.

[20] N. Concer, L. Bononi, M. Soulie, R. Locatelli and L. Carloni, The Connection-Then-Credit Flow Control Protocol for Heterogeneous Multicore Systems-on-Chip. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Sys, Vol. 29, No. 6, pp. 869-882, 2010.

[21] M. Sallam, M. W. El-Kharashi, M. Dessouky, “The Connection-Then-Credit Flow Control Protocol for Networks-On-Chips: Implementation Trade-offs”, *in Proc. of International Workshop on Network on Chip Architectures (NoCArc)*, pp. 25-30, 2014.

[22] A. Joshi, M. Mutyam, “Prevention flow-control for low latency torus networks-on-chip”, in Proc. of Symposium on Networks on Chip (NoCS), pp. 41-48, 2011.

[23] J. Lin, X. Lin, L. Tang, “Making-a-stop: A new bufferless routing algorithm for on-chip network,” Journal of Parallel and Distributed Computing, vol. 72, no. 4, pp. 515-524, Jan. 2012.

[24] N. Najib, A. Monemi, and M.N. Marsono, “Partially adaptive look-ahead routing for low latency Network-on-Chip”, in Proc. of Research and Development (SCOReD), pp. 1-5, 2014.

[25] S. Ma, N. E. Jerger, Z. Wang, “Whole packet forwarding: Efficient design of fully adaptive routing algorithms for networks-on-chip”, *in Proc. of High-Performance Computer Architecture (HPCA)*, pp. 1-12, 2012.

[26] S. Ma, Z. Wang, N. E. Jerger, L. Shen, N. Xiao, ”Novel Flow Control for Fully Adaptive Routing in Cache-coherent NoCs”, IEEE Transactions on Parallel & Distributed Systems, No. 1, pp. 1, doi:10.1109/TPDS.2013.166

[27] A. Monemi, C. Y. Ooi, M. N. Marsono, ”Low Latency Network-on-Chip Router Microarchitecture Using Request Masking Technique”, International Journal of Reconfigurable Computing, Vol. 2015, Article No. 2, 13 pages.

[28] H. M. G. Wassel, Ying Gao, J. K. Oberg, T. Huffmire, R. Kastner, F. T. Chong, T. Sherwood, ”SurfNoC: a low latency and provably non-interfering approach to secure networks-on-chip”, *in Proc. of the International Symposium on Computer Architecture (ISCA)*, pp. 583-594, 2013.

[29] A. Psarras, I. Seitanidis, C. Nicopoulos, G. Dimitrakopoulos, “PhaseNoC: TDM scheduling at the virtual-channel level for efficient network traffic isolation”, *in Proc. of the Design, Automation & Test in Europe Conference (DATE)*, pp. 1090-1095, 2015.

[30] J.D. Owens et al., ”Research Challenges for On-Chip Interconnection Networks”, IEEE Micro, Vol. 27, No. 5, pp. 96-108, 2007.

[31] D. P. Bertsekas, ”Stochastic optimization problems with nondifferentiable cost functionals”, *Journal of Optimization Theory and Applications*, Vol. 12, No. 2, pp. 218-231, 1973.

[32] N. Jiang, D. U. Becker, G. Michelogiannakis, J. Balfour, B. Towles, J. Kim, and W. J. Dally, ”A Detailed and Flexible Cycle-Accurate Network-on-Chip Simulator”, *in Proc. of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 86–96, 2013,

[33] S. H. Brooks, ”A discussion of random methods for seeking maxima”, *The computer journal*, Vol. 6, No. 2, 1958.

[34] R. White, ”A survey of random methods for parameter optimization”, *SIMULATION*, Vol. 17, pp. 197-205, 1971.

[35] J. Wroclawski, The Use of RSVP with IETF Integrated Services, September 1997. RFC 2210, IETF.

[36] CA. Coello Coello, ”A comprehensive survey of evolutionary based multiobjective optimization techniques”, *Knowledge and Information systems*, Vol. 1, No. 3, pp. 269-308, 1999.

[37] P. Bajpai and M. Kumar, ”Genetic Algorithm – an Approach to Solve Global Optimization Problems”, Indian Journal of Computer Science and Engineering, Vol. 1 No. 3, pp. 199-206.

[38] J. Guan, M. M. Aral, ”Progressive genetic algorithm for solution of optimization problems with nonlinear equality and inequality constraints”, Applied Mathematical Modelling, Vol.23, No. 4, pp. 329–343, 1999.

[39] D. Bertozzi, A. Jalabert, S. Murali, R. Tamhankar, S. Stergiou, L. Benini, G. De Micheli, ”NoC synthesis flow for customized domain specific multiprocessor systems-on-chip”, *IEEE Transaction on Parallel and Distributed Systems*, Vol. 16, No. 2, pp. 113-129, 2005.

[40] E.B. van der Tol and E.G. Jaspers, “Mapping of MPEG4 Decoding on a Flexible Architecture Platform”, SPIE, Vol. 4674, pp. 1-13, 2002.

[41] X. Zhao and Z. Lu. ”Per-flow Delay Bound Analysis Based on a Formalized Micro-architectural Model”, *in Proc. of ACM/IEEE International Symposium on Networks-on-Chip (NoCS)*, 2013.

[42] A. E. Kiasari, S. Hessabi, and H. Sarbazi–Azad, “PERMAP: A Performance–Aware Mapping for Application–Specific SoCs”, *in Proc. of Application–specific Systems, Architectures and Processors (ASAP)*, pp. 73–78, 2008.

[43] J. Hu, and R. Marculescu, ”Energy-aware mapping for tile-based NoC architectures under performance constraints”, *in Proc. of Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 233-239, 2003.

[44] Y. Qian, Z. Lu, W. Dou, ”Analysis of Worst-case Delay Bounds for Best-effort Communication in Wormhole Networks on Chip”, *in Proc. of Symposium on Networks-on-Chip (NOCS)*, pp. 44-53, 2009.

Fahimeh Jafari received her B.Sc. and M.Sc. degrees in Computer Engineering from Ferdowsi University of Mashhad, Iran, in 2002 and 2005, respectively, and Ph.D. degree in Electronic and Computer Systems from the KTH Royal Institute of Technology, Stockholm, Sweden, in 2015. Then, she joined, as Post-Doctoral Teaching Fellow, the Department of Mathematics and Computer Science, Liverpool Hope University, UK, and she is currently Lecturer (Assistant Professor) at Liverpool Hope University. Her research interests include design methodologies, interconnection networks, optimization theory, and performance evaluation.

Axel Jantsch (M’97) received a Dipl.Ing. (1988) and a Dr. Tech. (1992) degree from TU Wien, Vienna, Austria. Between 1993 and 1995 he received the Alfred Schrödinger scholarship from the Austrian Science Foundation for a postdoc at the Royal Institute of Technology (KTH), Stockholm, Sweden. From 1995 through 1997 he was with Siemens Austria in Vienna as a system validation engineer. Between 1997 and 2002 he was Associate Professor at KTH, and from 2002 to 2014 he was full Professor in Electronic Systems Design at KTH. Since September 2014, he is Professor of Systems on Chips in the Institute of Computer Technology at the TU Wien. A. Jantsch has published over 300 articles and one book in the areas of VLSI design and synthesis, system level specification, modeling and validation, HW/SW codesign and cosynthesis, reconfigurable computing, and networks on chip, and has more recently turned his attention to self-awareness in cyber-physical systems.

Zhonghai Lu Zhonghai Lu (M’05) received the B.Sc. degree in radio and electronics from Beijing Normal University, Beijing, China, in 1989, and the M.Sc. degree in system-on-chip design and the Ph.D. degree in electronic and computer system design, both from the KTH Royal Institute of Technology, Stockholm, Sweden, in 2002 and 2007, respectively.

He was an Engineer in the field of electronic and embedded systems from 1989 to 2000. He was a Post-Doctoral Researcher at the KTH Royal Institute of Technology, for two years, where he is currently an Associate Professor with the Department of Electronics and Embedded Systems, School for Information and Communication Technology. His current research interests include on-chip networks, many-core architectures, performance analysis, and design automation etc. He has published over 110 peer-reviewed papers in the above areas.

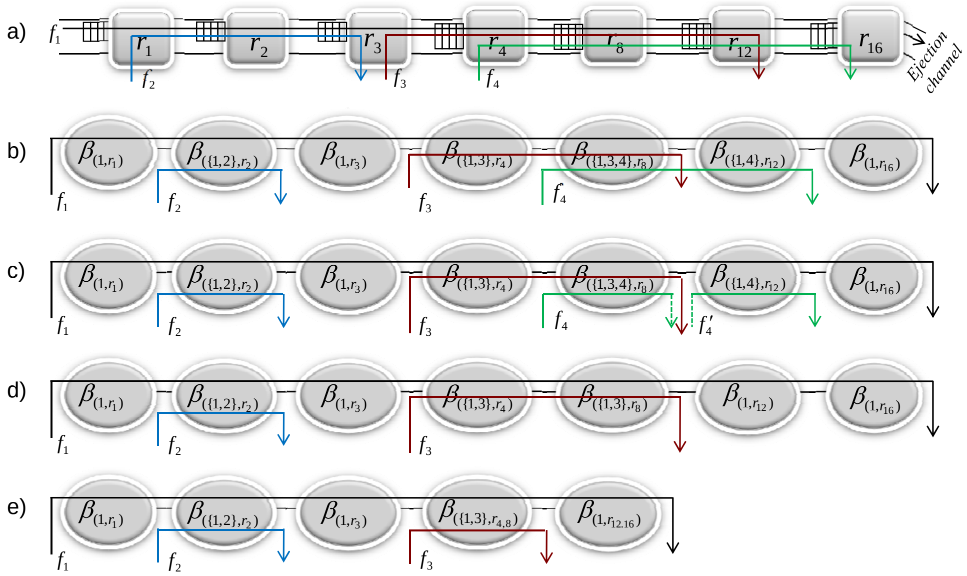

Fig. 16. An example of end-to-end ESC computation

Appendix

This appendix is included for completeness, but is already published in paper [4].

Here, we show the procedure of deriving end-to-end ESC for a tagged flow with the help of the example in Figure 1. Assuming flow f_1 is the tagged flow, its routing path is shown in a tandem of routers in Figure 16(a).

After analyzing per-router resource sharing scenarios and deriving intra-router ESCs, we can view an analysis model which keeps per-router ESCs of a tagged flow or tagged aggregate flow as shown in Figure 16(b). This model is called *aggregate analysis model*. 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, $\beta_{(\{1,2\},r_2)}$ is the service curve of aggregate flow $f_{\{1,2\}}$ in router r_2. A set of s_i's in a tandem of routers is denoted as $S = \{s_i\}$. For example, in Figure 16(b), $S = \{\{1\},\{1,2\},\{1\},\{1,3\},\{1,3,4\},\{1,4\},\{1\}\}$. It is notable that the elements should be placed in S with the same order as in the tandem of routers because the place of each element effects on the value of some parameters in the rest of the paper.

We use the theorem of *concatenation of network elements* [1] to model nodes sequentially connected and each is offering a rate-latency service curve to each of the aggregate flows $\beta_{(s_i,r_j)}, j = 1,2,...,n$ as a rate-latency server as follows:

$$\beta_{(s_i,r_{1,2,\ldots,n})} = \beta_{(s_i,r_1)} \otimes \beta_{(s_i,r_2)} \otimes \ldots \otimes \beta_{(s_i,r_n)} \tag{18}$$

where the minimum service rate and the maximum processing latency in an equivalent rate-latency server are defined as follows:

$$\begin{aligned} R_{(s_i,r_{1,2,\ldots,n})} &= \min\left(R_{(s_i,r_1)}, R_{(s_i,r_2)}, ..., R_{(s_i,r_n)}\right) \\ T^l_{(s_i,r_{1,2,\ldots,n})} &= T^l_{(s_i,r_1)} + T^l_{(s_i,r_2)} + ... + T^l_{(s_i,r_n)} \end{aligned} \tag{19}$$

In Figure 16(b), sequentially connected service curves for the same aggregate flows do not exist. Thus, we can directly go to the next step which considers contention scenarios.

As illustrated in Figures 16(a), contention flow f_2 is nested in flow f_1 and contention flow f_3 is crossed with f_4. To consider contentions in this model and obtain inter-router ESC, we decompose a complex contention scenario to basic contention patterns and then remove contention flows one by one. The contention scenarios can be classified into two basic patterns, namely, *nested* and *crossed*. We apply the algebra of sets to recognize contention scenarios. Convenient notations are defined through the example in order to facilitate our discussion. To recognize the contention scenarios, we first find $s^m = \left\{s_x \middle| |s_x| = max\left(|s_i|\right); \forall s_i \in S\right\}$, where $|s_x|$ is the cardinality (the number of elements) of set s_x. In other words, s^m is $s_x \in S$ with the maximum cardinality. The service curve, flow, and router related to s^m are denoted as f_{s^m}, β^m, and r^m, respectively. Thus, these notations in Figure 16(b) are given by $S = \{\{1\},\{1,2\},\{1\},\{1,3\},\{1,3,4\},\{1,4\},\{1\}\}$, $s^m = \{1,3,4\}$, $f_{s^m} = f_{\{1,3,4\}}$, $r^m = r_8$, and $\beta^m = \beta_{(\{1,3,4\},r_8)}$.

The set placed before s^m in S is called s^{Prev} and the set after that s^{Next}. In this respect, the related aggregate flow, service curve, and router to s^{Prev} are denoted as $f_{s^{Prev}}$, β^{Prev}, and r^{Prev}, respectively. $f_{s^{Next}}$, β^{Next}, and r^{Next} are related to s^{Next} as well. Therefore, due to $s^m =_{\{1,3,4\}}$ in Figure 16(b), $s^{Prev} = \{1,3\}$, $\beta^{Prev} = \beta_{(\{1,3\},r_4)}$, $f_{s^{Prev}} = f_{\{1,3\}}$, $r^{Prev} = r_4$, $s^{Next} = \{1,4\}$, $\beta^{Next} = \beta_{(\{1,4\},r_{12})}$, $f_{s^{Next}} = f_{\{1,4\}}$, and $r^{Next} = r_{12}$.

Now, we can recognize contention scenarios as below:

1) if $s^{Next} \subset s^{Prev}$ then the contention is *nested*;
   - Remove $f_{s^m-(s^m \cap s^{Next})}$ from β^m
2) else if $s^{Prev} \subset s^{Next}$ then the contention is *nested*;
   - Remove $f_{s^m-(s^m \cap s^{Prev})}$ from β^m.
3) else
   - if $s^{Next} \subset s^m$ then the contention is *nested*;
     - Remove $f_{s^m-(s^m \cap s^{Next})}$ from β^m

Algorithm 4 End-to-End ESC Algorithm

```
Find the set of contention flows of tagged flow $f_t$, denoted by $CF_t$
for $\forall j \in CF_t$ do
    if $Src(j) \notin Path(t)$ then
        Find $joiningnode = JoiningPoint(f_j)$
        Calculate $X = ESC(f_j, Src(j), joiningnode)$
        $\alpha_j = \alpha_j \oslash X$
    end if
end for
Calculate intra-router ESC for WRR based on Section V-A.
Calculate $\beta_{\left(s_{i_1},r_{j_1}\right)} \otimes \beta_{\left(s_{i_2},r_{j_2}\right)} \otimes ... \otimes \beta_{\left(s_{i_n},r_{j_n}\right)}$ if $i_1 = i_2 = ... = i_n$.
Find $s^m = \left\{s_x \middle| |s_x| = max\left(|s_i|\right); \forall s_i \in S\right\}$.
repeat
    if $s^{Prev} \subset s^{Next}$ then
        Remove $f_{s^m-(s^m \cap s^{Next})}$ from $\beta^m$
    else if $s^{Next} \subset s^{Prev}$ then
        Remove $f_{s^m-(s^m \cap s^{Prev})}$ from $\beta^m$.
    else
        if $s^{Prev} \subset s^m$ then
            Remove $f_{s^m-(s^m \cap s^{Prev})}$ from $\beta^m$
        else if $s^{Next} \subset s^m$ then
            Remove $f_{s^m-(s^m \cap s^{Next})}$ from $\beta^m$.
        else
            Find $joiningnode =$
              $JoiningPoint(f_{(s^m-s^{Prev})})$.
            Calculate $X =$
              $ESC(f_{(s^m-s^{Prev})}, joiningnode, r^{Next})$.
            $\acute{\alpha}_{(s^m-s^{Prev})} = \alpha_{(s^m-s^{Prev})} \oslash X$
            Remove $f_{(s^m-s^{Prev})}$ from $\beta^m$.
            Remove $\acute{f}_{(s^m-s^{Prev})}$ from $\beta^{Next}$.
        end if
    end if
    Calculate $\beta_{\left(s_{i_1},r_{j_1}\right)} \otimes \beta_{\left(s_{i_2},r_{j_2}\right)} \otimes ... \otimes \beta_{\left(s_{i_n},r_{j_n}\right)}$ if
        $i_1 = i_2 = ... = i_n$.
    Find $s^m$.
until $|s^m| \neq 1$
return end-to-end ESC for tagged flow $f_t$
```

- else if $s^{Prev} \subset s^m$ then the contention is *nested*;
  - Remove $f_{s^m-(s^m \cap s^{Prev})}$ from β^m.
- else, it is *crossed.*
  - The problem is strictly transformed to the combination of two nested flows

Regarding Figure 16(b), $s^m = \{1,3,4\}$, $s^{Prev} = \{1,3\}$, and $s^{Next} = \{1,4\}$. Since s^{Prev} is not a subset of s^{Next}, and vice versa, due to contention recognition procedure, this case is a crossed contention. There are two cross points, one between r_4 and r_8 and the other between r_8 and r_{12}. We cut f_4 at the second cross point, i.e., at the ingress of r_{12}, f_4 will be split into two flows, f_4 and $\acute{f}_4$, as shown in Figure 16(c). Then the problem is strictly transformed to the combination of nested flows such that f_4 is nested in flow f_3 and $\acute{f}_4$ in f_1. It is clear that the arrival curve $\alpha_{\left(\acute{f}_4,r_{12}\right)}$ equals to output arrival curve f_4 in router r_8, $\alpha^*_{(f_4,r_8)}$. To compute $\alpha^*_{(f_4,r_8)}$, we need to get the ESC of r_8 for f_4, $\beta_{(f_4,r_8)}$. Then, we calculate the output arrival curve of f_4 as $\alpha^*_{(f_4,r_8)} = \alpha_{(f_4,r_8)} \oslash \beta_{(f_8,r_8)}$ and remove nested flows f_4 and $\acute{f}_4$ from the tandem as shown in Figure 16(d). Deriving output arrival curve and removing the contention flows are done by applying our proposed Theorems in [2] [3].

After subtracting each contention flow from the ESC, we should apply the concatenation theorem again to find more equivalent servers and reduce the number of service curves. For instance, after removing contention flows f_4 and $\acute{f}_4$, the example looks like Figure 16(d). In this figure, the service curve of sub-tandem $\{r_4, r_8\}$ for aggregate flow $f_{\{1,3\}}$ is computed as $\beta_{(\{1,3\},r_{4,8})} = \beta_{(\{1,3\},r_4)} \otimes \beta_{(\{1,3\},r_8)}$ and also $\beta_{(1,r_{12,16})}$ is calculated as $\beta_{(1,r_{12})} \otimes \beta_{(1,r_{16})}$. The aggregate analysis model with new equivalent servers is shown in Figure 16(e).

We similarly repeat contention recognition and convolution steps until $|s^m| \neq 1$. When $|s^m| = 1$, it means, the end-to-end ESC of tagged flow is obtained.

Algorithm 4 presents all stages of deriving the end-to-end ESC for a given tagged flow as described through the example.