Computing Accurate Performance Bounds for Best Effort Networks-on-Chip

Dara Rahmati*, Srinivasan Murali¥§, Luca Benini‡, Federico Angiolini¥§, Giovanni De Micheli§, Hamid Sarbazi-Azad**

*HPCAN, Sharif University of Technology, Tehran, Iran, †Institute for Research in Fundamental Sciences (IPM), Tehran, Iran, (d_rahmati@ce.sharif.edu, azad@sharif.edu, azad@ipm.ir)
§NoCs Sarl, (murali, angiolini)@inocs.com
£LSI, EPFL, Lausanne, Switzerland, giovanni.demicheli@epfl.ch
‡DEIS, University of Bologna, Bologna, Italy, luca.benini@unibo.it

Abstract—Real-time (RT) communication support is a critical requirement for many complex embedded applications which are currently targeted to Network-on-chip (NoC) platforms. In this paper, we present novel methods to efficiently calculate worst-case bandwidth and latency bounds for RT traffic streams on wormhole-switched NoCs with arbitrary topology. The proposed methods apply to best-effort NoC architectures, with no extra hardware dedicated to RT traffic support. By applying our methods to several realistic NoC designs, we show substantial improvements (more than 30% in bandwidth and 50% in latency, on average) in bound tightness with respect to existing approaches.

Index Terms—SoC, NoC, Performance, QoS, Best-effort analysis, Real-time, Analytical model, Wormhole switching.

1. Introduction

The Network-on-Chip [1, 2] paradigm has emerged in recent years to overcome the power and performance scalability limitations of point-to-point signal wires, shared buses, and segmented buses [1-6]. While the scalability and efficiency advantages of NoCs have been demonstrated in many occasions, their timing predictability and suitability to transport real-time communication are still a source of technical concern.

Many applications have strict requirements on latency and bandwidth of on-chip communication, which are often expressed as real-time constraints on traffic flows. On a NoC fabric, this translates to guaranteed QoS (quality of service) requirements for packet delivery. Different approaches have been used to support guaranteed QoS for NoCs: priority-based switching schemes [7], time-triggered communication [8], time-division multiple access [9], and many variations thereof. All these approaches require the use of special hardware mechanisms and often come with strict service disciplines that limit NoC flexibility and penalize the average performance to provide worst-case guarantees. In fact, NoC prototypes are often classified as being either best-effort or guaranteed-service, depending on the availability of hardware support form RT traffic.

Our work takes a new viewpoint. We consider best-effort NoC architectures without special hardware support for QoS traffic. We only assume that the traffic injected by the network’s end-nodes is characterized in terms of worst-case behavior. We then formulate algorithms to find latency and bandwidth bounds on end-to-end traffic flows transported by a best-effort wormhole NoC fabric with no special hardware support for RT traffic. For applications with traffic streams that have RT latency and/or bandwidth constraints, it is critical to be able to bound the maximum delay and minimum injectable bandwidth for packets of such streams. This helps in choosing topologies that meet the RT constraints with minimum area, power overhead and optimum utilization of resources. Our approach is inspired by the work by Lee et al. [10] for traditional multiprocessor networks, and extends it in several directions. We propose two different methods for characterizing worst-case performance. The first method, RTB-HB (Real-Time Bound for High-Bandwidth traffic), is conceived for NoCs supporting workloads where injected flows have high demands of average bandwidth and require a guaranteed worst-traffic minimum bandwidth (mBW) and maximum upper bound NoC traversal latency (UB). In this case, we do not assume any a-priori regulation on the traffic injection rate; a core can send packets at any time, as long as the network has buffer capacity to accept them.

The second method considers applications with latency-critical flows that require low and guaranteed UB values, but have moderate bandwidth requirements, and thus can send packets at intervals no shorter than a minimum permitted interval - which obviously implies a maximum bandwidth (MBW) limitation. This method, called RTB-LL (Real-Time Bound for Low-Latency traffic) requires a very simple traffic regulation at network injection points. RTB-LL is a significant improvement to the WCFC bound proposed in [10], while RTB-LL is completely new. Table 1 compares typical values for upper bound delay and bandwidth of RTB-HB, RTB-LL and WCFC methods. In [11] we presented these methods in their basic modes of operation. In this paper, we extend the methods to be more comprehensive, considering more generic NoC models, supporting various modes of operation and more experiments. In particular, we have several new and important contributions from our earlier work in [11].

The remainder of the paper is organized as follows. Section 2 summarizes related work. Section 3 gives definitions and basic concepts. Section 4 describes RTB-HB and RTB-LL methods. Section 5 focuses on experimental results and quantitative comparisons. Section 6 describes the time complexity of the proposed methods. Finally, Section 7 concludes the paper.
Table 1: Typical upper bound delay and bandwidth in different methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>RTB-LL (LL), RTB-HB (HB), WCFC (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upper Bound Delay</td>
<td>$UB_{LL} \leq UB_{HB} \leq UB_{W}$</td>
</tr>
<tr>
<td>Minimum Guaranteed Bandwidth Comparison</td>
<td>$MBW_{LL} \geq MBW_{HB} \geq MBW_{W}$</td>
</tr>
</tbody>
</table>

2. RELATED WORK

The body of knowledge on macro-scale RT networks is extensive and an overview of the state-of-the-art is beyond the scope of this work. The interested reader is referred to [12 - 21]. Here, we focus on RT-NoCs, which have often been called guaranteed-service or QoS-enabled NoCs.

QoS is an important issue in many application domains such as multimedia, aerospace, healthcare and military. Many of these applications have one or more traffic flows that have real-time requirements and need hard QoS guarantees. Two major parameters that account for QoS guarantees in NoC are worst-case delay and worst-case bandwidth. They are sometimes referred as upper-bound delay and lower-bound bandwidth of flows. Historically, designers have focused on extracting the average delay and average bandwidth, and a large body of work to extract such parameters exists [22-33]. Simulation and mathematical modeling are two different approaches to do so. While simulation is widely used in many situations, it is time-consuming and it gives limited insight on sensitivity to traffic parameters and worst-case conditions. In contrast, devising accurate mathematical models of a system is complicated, but if such models can be extracted, they are usually computationally efficient and insightful. Therefore, they can be used within design tools, for example to iterate in the NoC synthesis process to tailor NoC architectures for specific applications. Frequently used mathematical frameworks are queuing theory and statistical timing analysis [25, 26, 28, 31]. A node is modeled as a queuing system, which can be $M/G/1, M/G/1/t, G/G/1$, etc. The NoC then is modeled by interconnecting a number of queues and the parameters are then extracted using standard solutions from queuing theory. In these models, the applicability and accuracy are the main concerns.

In order to provide QoS, some NoC architectures use special hardware mechanisms. They are known as Guaranteed Service (GS) NoCs, as opposed to Best Effort (BE) NoCs. To distinguish briefly, GS NoCs commit to a performance level for one or more flows (typically latency or bandwidth). Hard or soft QoS can be provided, depending on whether the NoC actually strictly enforces the desired performance level or merely strives to achieve it. GS NoCs can leverage resource reservation or priority-based scheduling mechanisms. The former technique usually achieves hard QoS, but the resource utilization may be poor because reserved resources are under-utilized. The latter may achieve better resource utilization as resources are used on demand, like a best effort fashion with priority, but generally only ensures soft QoS and problems like starvation of low priority flows may occur. GS NoCs require extra hardware complexity with respect to BE NoCs to support redundant resources or priority mechanisms. On the other hand, the performance of flows in a GS NoC can be more easily characterized. In a pure BE NoC, analyzing the temporal behavior of the flows is very complex, due to the large number of contentions that may block a packet of a flow several times along its journey to the destination. Also simulation is very complicated and time-consuming, as identifying the worst-case scenario, and enforcing the network to operate in such worst-case situation, is extremely difficult, if not impossible. Thus, the modeling approach may be an option; however, due to extreme complexity, until recently there have been no applicable approaches to model the performance parameters in worst-case situations. Thus, most of the efforts to provide QoS have been in the context of GS or combinations of GS and BE NoCs.

In [9], Goossens et al. present the Ethereal NoC which combines GS with BE. The MARS [34], aSoc [35] and Nostrum [36] architectures use TDMA (time division multiplexing) mechanisms to provide real-time guarantees on packet-switched networks. The aelute NoC [37] provides a GS and scalable TMDA based architecture, using mesochronous or asynchronous links. In Shi et al. [7], a priority-based wormhole switching for scheduling RT flows is presented. In [8], Paukovits et al. propose the concept of a predictable Time-Triggered NoC (TTNoC) that realizes QoS based communication services. Diemer et al. in [38] introduce Back Suction, a flow control scheme to implement service guarantees using a prioritized approach between BE and GS services. In [39], Hansson et al. provide the latency and throughput guarantees based on the approach of data flow analysis technique, determining required buffer size at network interfaces, which is applicable to the Ethereal NoC. Many other works have been published with variations over these basic ideas [40-49].

However, most NoC architectures are of best-effort [50] type and do not have special hardware mechanisms to guarantee QoS. Today, to the best of our knowledge, there are only few works that calculate worst-case bandwidth and delay values for a BE NoC. In [51], the lumped link model was proposed where the links a packet traverses are lumped into a single link. This model does not distinguish direct contention (due to arbitration losses) from indirect contention (due to full buffers ahead along the path), thus the estimated bounds are pessimistic. In [52], Qian et al. provide a method based on network calculus [53, 54] to calculate real-time bounds for NoCs; the method uses service curves and arrival curves that characterize the service characteristics of switches and injected traffic. Extracting arrival curves is not a straightforward task for many applications. Thus, for an arbitrary injected traffic load, traffic regulators may be needed to make sure that the amount of injected traffic in a specified period does not exceed a specified level. In [55] a buffer optimization problem is solved under worst-case performance constraints based on network calculus. In [56], Bakhouya et al. also present a model based on network calculus to estimate the maximum end-to-end and buffer size for mesh networks; the delay bounds calculated for the flows are not hard bounds and real values may be larger.

In [11], we proposed some methods for worst-case analysis that do not need traffic regulators. The bounds
behind by those methods are tighter than those reported by previous studies. In this paper, we have extended these methods in several directions: we provide a more detailed switch model to differentiate stage delay and buffer depth; this results in tighter bounds in calculations of RTB-LL method. Our analysis also considers networks with virtual channels and variable buffer lengths; the analysis for small buffers is also extended to account for message lengths, which results in tighter bounds with respect to our previous results.

3. THE NETWORK MODEL

A router model is essential to characterize network latency and bandwidth. We consider the very general reference architecture shown in Fig. 1 where a crossbar handles the connections among input and output channels inside the router. For more generality, we consider optional buffering at input and output ports. We assume round-robin arbitration in the switches, a commonly used arbitration scheme in many NoCs. Each port is equipped with some virtual channels sharing the bandwidth of the physical channel associated with it. Links, which can be pipelined to maximize the operating frequency, connect the output ports to the input ports of adjacent routers. Note that due to backpressure signaling, packet loss and packet dropping do not happen in switches. Table 2 summarizes the parameters used to describe the model. For the sake of simplicity, we use a single parameter $Freq$ for the operating frequency of all cores and $FlitWidth$ as the data width of all NoC links.

The buffer depth ($B_d$) parameter is used in the paper frequently. As seen in Fig. 1, $B_d$ is the summation of a number of registers and/or number of the slots of one or two FIFO(s)\(^1\), from the arbitration point (at the entry of the crossbar) of switch $j$ to the arbitration point of switch $j+1$. The input buffer depth is denoted by $b_1$ (we assume at least one register), $b'_1$ is the minimum delay of the header flit of a packet in the input buffer of a switch, $b_2$ is the number of cycles to traverse a switch crossbar (if pipelined), $b_3$ is the depth of the output buffer (if any), $b'_3$ is the minimum delay of the header flit of a packet in the output buffer of a switch and $a$ is the number of registers (if any) along a link to compensate the propagation delay of the wires. Note that $b_2$ and $a$ represent latencies that packets face even in the absence of congestion, while $b_3$ and $b'_3$ become most relevant in case of blocking, when buffers fill up; in the absence of congestion, input and output buffers can be traversed in a single cycle instead ($b'_1$ and $b'_3$). Blocking always happens because of arbitration conflicts, either directly in front of a switch crossbar, or indirectly due to full buffers ahead. For simplicity, throughout the paper, we consider the buffering between two adjacent switches to be lumped, so we mention ‘output buffer of switch $j$’ and ‘input buffer of switch $j+1$’ equivalently, referring to the same number of intermediate registers or FIFOs between the arbitration points of switches $j$ and $j+1$, i.e. to $B_d$. The stage delay ($S_d$) parameter instead describes the minimum delay the header flit of a packet will face in the absence of any contention with other packets, between arbitration points of two adjacent switches. Please also note that the switches along a path are indexed $j=1..m$, but $j=0$ can conveniently be seen as a virtual switch inside the source node, which acts like a physical switch to model source conflicts (i.e. sending more than one flow from a source node). The parameters $t_{S_1}$ and $t_{S_2}$ model the setup time at NoC sources and consumption time at NoC destinations to inject and eject packets. Of course, to be able to use finite parameters, we assume that the receiving nodes are able to accept incoming data at the required rates.

### Table 2: Network parameters, symbols, and used notation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Freq$</td>
<td>Clock frequency of the system</td>
</tr>
<tr>
<td>$a$</td>
<td>Number of pipeline stages (registers) used to segment NoC links to compensate the propagation delay of the wires</td>
</tr>
<tr>
<td>$b_1$</td>
<td>Depth of switch input buffer, as a FIFO</td>
</tr>
<tr>
<td>$b'_1$</td>
<td>Delay of packet header in switch input buffer without contention</td>
</tr>
<tr>
<td>$b_2$</td>
<td>Number of pipeline stages of the crossbar (0 if combinational)</td>
</tr>
<tr>
<td>$b_3$</td>
<td>Depth of switch output buffer, as a FIFO (0 if none)</td>
</tr>
<tr>
<td>$b'_3$</td>
<td>Delay of switch output buffer without contention</td>
</tr>
<tr>
<td>$b_{d}$</td>
<td>Buffer depth parameter $b_{d} = a + b_1 + b_2 + b_3$</td>
</tr>
<tr>
<td>$S_{d}$</td>
<td>Stage delay $S_{d} = a + b'_1 + b_2 + b'_3 = a + b$</td>
</tr>
<tr>
<td>$t_{S_1}$</td>
<td>Latency overhead for injecting a packet into the network</td>
</tr>
<tr>
<td>$t_{S_2}$</td>
<td>Latency overhead for ejecting a packet at the destination</td>
</tr>
</tbody>
</table>

### Table 3: Traffic model parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_i$</td>
<td>$i$-th traffic flow in the network</td>
</tr>
<tr>
<td>$L$</td>
<td>General terminology for length of packets, in flits</td>
</tr>
<tr>
<td>$L_i$</td>
<td>Length of the packets of $F_i$, in flits</td>
</tr>
<tr>
<td>$S_i$</td>
<td>Source node of $F_i$</td>
</tr>
<tr>
<td>$D_i$</td>
<td>Destination node of $F_i$</td>
</tr>
<tr>
<td>$P_i$</td>
<td>Packet of $F_i$</td>
</tr>
<tr>
<td>$h_i$</td>
<td>Number of switches (hops) along the path of $F_i$</td>
</tr>
</tbody>
</table>

\(^1\)Throughout this paper we assume registers to be clocked pipeline registers, in which to traverse a number of n cascaded registers, n clock cycles are always required. A FIFO is the traditional First-In, First-Output queue. In case a flit traverses an empty FIFO of length n, in the absence of traffic contention, it will face one cycle and in case of a filled FIFO, will face n cycles of delay. We use the general term buffer throughout the paper to address both types of registers and FIFOs or a combination of them.
4. NETWORK TRAVERSAL DELAY ANALYSIS

Table 3 lists the parameters we use to describe traffic flows across the network, while Table 4 summarizes the parameters that we use to model the performance of such flows. Most notably, $UB_f$ represents the upper-bound delay for a packet of flow $F_f$ traversing the network, and is a key factor for the interconnect designer. We try to use a notation as close as possible to that used in [10] for ease of comparison. We first present a method called RTB-HB (Real Time Bound for High-Bandwidth traffic) which calculates $UB_f$ in a completely worst-case traffic situation. Crucially, this includes the possibility for other system cores to inject unregulated bandwidth, i.e. any amount of traffic at irregular intervals. This is a key property for real-world interconnects analysis, as most available IP cores operate on an unregulated-injection basis. In order to calculate $UB_f$ in such a case, we consider all intermediate buffers along the route to be full, and we assume arbitration loss at all switches where other flows are contending for the same output port. As it will be seen, the calculations always provide solutions for worst-case situation on different flows that are unique due to the employed deterministic calculation procedure. The calculated values are tight in most network scenarios as the worst-case situation can really happen. The bounds may be slightly pessimistic in rare network scenarios where some of the containing flows connect two switches on different routes that can prevent providing enough contending packets to create worst situations.

Deadlock and livelock do not occur as we assume the routing path along the switches for all the flows are deterministic and predefined (like the networks used in [57]). As we are modeling the worst-case behavior, we consider that the flows send the packets at maximum possible rate that the network permits; thus at this level of design, knowing the flow behavior is not important. Since switches are assumed to feature round-robin arbitration, even though we assume the current flow to be serviced last, the maximum delay is bounded, i.e. starvation cannot occur. Therefore, the packets sent by the source $S_i$ are eventually delivered. RTB-HB calculates the Maximum Interval $M_I$, i.e. the number of cycles after which the output buffer of $S_i$ is guaranteed to be free again for further injection. From this value, the worst-traffic minimum injectable bandwidth ($mBW_i$) can also be easily derived. This analysis can be applied to most NoC architectures, without any specific QoS hardware or software provisioning. We then move on to the description of another method, called RTB-LL (Real Time Bound for Low-Latency traffic). In this scenario, we assume that traffic injection can be regulated, as in some application scenarios. Therefore, we also calculate a minimum permitted interval ($mI$) between two consecutive packets from the same source, which can be translated into a maximum permitted bandwidth ($mBW_i$). This approach is similar to a previously reported method [10] (Real-time wormhole channel feasibility checking or WCFC, which will be briefly described later) but provides much better results in terms of bound tightness. For a proper operation, the system must then respect $mBW_i$ bounds at runtime.

A. The Proposed Delay Model RTB-HB

The goal here is to calculate the parameters $UB_f$ (worst-case latency to traverse the network) and $M_I$ (maximum worst-case interval). Let us first consider the case $b_d = L$.

1) The case $b_d = L$

Note that $b_d = L$ means that a packet fills exactly the buffering resources between the arbitration points of two adjacent switches. Considering the case where the network is completely loaded (an unrealistic scenario just for visualization purposes) and $b_d = L$, the network operates by shuffling packets around in lockstep: all switches simultaneously re-arbitrate every L cycles and packets trail each other, filling up the buffers as soon as they become free.

Table 4: Performance model parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$UB_f$</td>
<td>Upper bound delay for a packet $P_i$ of $F_f$ to traverse the NoC</td>
</tr>
<tr>
<td>$M_I$</td>
<td>Maximum interval until the ability to inject a new packet of $F_f$, used in method RTB-HB</td>
</tr>
<tr>
<td>$mI$</td>
<td>Minimum permitted interval between two consecutive packets of $F_f$, used in methods WCFC and RTB-LL</td>
</tr>
<tr>
<td>$mBW_i$</td>
<td>Worst-traffic minimum injectable bandwidth for $F_i$, used in method RTB-HB and is equal to: $mBW_i = \frac{\text{FlitWidth}}{M_I} = \frac{\text{FlitWidth}}{mI}$</td>
</tr>
<tr>
<td>$MBW_i$</td>
<td>Maximum permitted bandwidth for $F_i$, used in methods WCFC and RTB-LL and is equal to: $MBW_i = \frac{\text{FlitWidth}}{mI}$</td>
</tr>
<tr>
<td>$u_i'$</td>
<td>The time needed for $P_i$ to go from the input buffer of $SW_j(i &gt; 0)$ or from generating process in $S_i(j = 0)$ to the input buffer of $SW_j(i \leq j \leq h - 1)$ or $D_i(j = h)$</td>
</tr>
<tr>
<td>$u_i^D$</td>
<td>The time needed for $P_i$ to go from the output buffer of $SW_j(i &gt; 0)$ or from the output buffer of $S_i(j = 0)$ to the output buffer of $SW_j+1$. This parameter is used for convenience and, mathematically it can be written based on $u_i'$ and $u_i^D = u_i' + u_i^D$, $j \geq 0$</td>
</tr>
<tr>
<td>$z_i(0, 0)$</td>
<td>Number of flows contending with $F_i$ at $S_i$</td>
</tr>
<tr>
<td>$z_i(i, j)$</td>
<td>Number of flows contending with $F_i$ at $SW_j(i)$ at output channel $c$</td>
</tr>
<tr>
<td>$l(x)$</td>
<td>Index of the x-th flow contending with $F_i$ at $S_i$ or $SW_j(i)$</td>
</tr>
</tbody>
</table>

More formally, when $P_i$ is generated in $S_i$, we consider all intermediate buffers along its route being full of packets from different flows. In the worst case, for $P_i$ to reach its destination, all these packets must leave their buffers. Focusing on hop $j$, $P_i$ may have arbitration conflicts with a number $z_c(i, j)$ of other flows contending for the output channel $c$, e.g. flows $F_a$ and $F_r$. Since round-robin arbitration is assumed, it is enough to consider all contending flows to send a packet before $P_i$ to guarantee a worst-case analysis. The order in which contending flows obtain the arbitration is not important for the latency calculation of $P_i$. So, $P_i$ should make a one-hop forward progress. While $P_a$ frees the buffers at hop $j$, flit by flit, the flits from $P_i$ will smoothly replace the free buffer spaces. Eventually, $P_i$ also goes through. Section 4.C presents a simple example to visualize this. The parameter $u_i'$ represents the time needed for $P_i$ to be ejected from $S_i$ and be placed in the output buffer of $S_i$ (or input buffer of the first switch of $F_f$). $u_i^D$ then represents the time needed for $P_i$ to go from the input buffer of $SW_j$ to the input buffer of $SW_j+1$, except for the last switch. At the last switch, $P_i$ is
ejected, so it is instead the time needed to get into the input buffer of destination $D_i$. To calculate $UB_i$, as shown in Eq. 1, all these contributions must be aggregated, plus the fixed overhead for the packet creation and ejection:

$$UB_i = ts_1 + ts_2 + \sum_j u'_j, j = 0..h_i$$  (1)

The time needed for $S_i$ to inject the next packet is the time to create such a packet, plus the time needed for this packet to move on to the input buffer of the first switch. Thus,

$$MI_i = ts_1 + u'_0$$  (2)

To be consistent with the notations from [10], we introduce the upper case $U'_i$ symbol, which models the hop delay from output buffer to output buffer (instead of from input buffer to input buffer).

Let us consider a packet of flow $F_i$ initiated at the source $S_i$. For this packet to reach the input buffer of the first switch, all existing packets at that buffer have to leave. Such an existing packet could be a packet from the same flow $F_i$ or any of the contending flows at the output channel of the source. Thus, the worst-case time taken for any existing packet to leave the buffer is given by $MAX_x(U'_i, U'_0(x))$, where $I(x)$ is the index of contending flows at the output channel, with $x = 1...z_o(i,0)$. Also, all other contending flows of $F_i$ may have to send a packet before this flow. Thus, the total delay for a packet from $F_i$ to reach the input buffer of the first switch is given by:

$$u'_i = MAX_x(U'_i, U'_0(x)) + \sum_x U'_0(x)$$

$$x = 1...z_o(i,0)$$  (3)

Similarly, for the subsequent hops, $u'_j$ can be calculated as:

$$u'_j = MAX_x(U'_j, U'_0(x)) + \sum_x U'_0(x),$$

$$x = 1...z_o(i,j), 1 \leq j \leq h_i$$  (4)

Please note that, if there is no contention for the flow, the above equation reduces to $u'_j = U'_j$. This is again akin to a packet moving in a pipeline fashion in the network.

In order to calculate $U'_j$ values, let us consider the packet from flow $F_j$ moving from output buffer of the source to the output buffer of the first switch. For the packet to move, any existing packet from the output buffer of the first switch should move to the output buffer of the second switch. Similar to the above calculations, the maximum delay is given to be $MAX_x(U'_j+1, U'_0(x))$. Please note the small difference from the $u'_j$ calculations that, in this case, the values of $U'_j$ at a switch $(j)$ depends on the values at the next switch $(j + 1)$ on the path. The $U'_j$ values can be obtained as:

$$U'_j = MAX_x(U'_j+1, U'_0(x)) + \sum_x U'_0(x),$$

$$x = 1...z_o(i,j+1), 0 \leq j \leq h_i - 1$$  (5)

For the case of the last switch, from the output port, the packet can be ejected in $L_i$ cycles (one flit per cycle). Thus,

$$U'_h = L_i$$

Based on Eq. 3 and 4, now the problem of finding $UB_i$ and $MI_i$ (Eq. 1 and 2) is mapped onto a summation of $U'_i$ values, which can be solved by Eq. 5. Please note that we assume that the destination has enough buffers to eject the packets at the rate at which the network delivers them. By applying the above formulae recursively, we can obtain the worst-case delay ($UB$) and injection rate ($MI$) for the different flows.

To describe the details of different aspects of analytical method RTB-HB for the calculation of upper bound delay and interval, we apply them step-by-step to an example NoC (shown in Fig. 2). The NoC contains four switches and there are 4 message flows from $S_1$ to $D_1$, $S_2$ to $D_3$, $S_3$ to $D_2$, and $S_4$ to $D_4$ ($S_2$ and $D_4$ are source and destination with two flows originated from or finished at them). We consider $B_d = L = 4$, in this example.

**Figure 2. A simple example network**

As an example, we study the time needed for a packet $P^0$ of flow $F_1$ to cross the network. In general, from Eq. 1, we can write:

$$ts_2 + \sum_j u'_j, j = 0...3$$

To start, let us model the time $u'_0$ needed to move from $S_1$ to the input buffer of switch $SW_1$. We start from the most congested network possible, so there exists another packet $P^1$ of the same flow ahead, and this packet needs $u'_0$ to move from the output buffer of the source (remember that source nodes are tagged with superscript 0) to the output buffer of $SW_1$. So, $u'_0 = U'_0$, $U'_0$ has to be calculated recursively based on the delays of the contending packets and delays of the packets ahead along the same route.

We observe that two factors mainly contribute when calculating the delay: first, the possibility of losing arbitrations at $SW_1$; second, the fact that there may be no available buffer space at the output of $SW_1$ (due to arbitration losses ahead), which also effectively stalls packets at the input of $SW_1$. For what concerns the arbitration loss, it can be seen that flow $F_1$ contends with flow $F_2$ at the output of $SW_1$. Thus, a packet $P^2$ of $F_2$ currently in the input buffer of $SW_1$ could be arbitrated before $P^1$. For what concerns the output buffer full condition, in the worst case, there will be a single ($B_d = L$) packet $P^3$ in the output buffer of $SW_1$. $P^3$ could belong to either $F_1$ or $F_2$, where respectively $U'_1$ or $U'_2$ models the time for such a packet to move ahead, from the output buffer of $SW_1$ to the output buffer of $SW_2$. $MAX(U'_2, U'_1)$ models the worst-case delay affecting the flow under study. During the time $MAX(U'_2, U'_1)$, the packet $P^3$ moves on to the output buffer of $SW_2$, leaving the output buffer at $SW_1$ empty. However, in the worst case, an arbitration loss occurs to $P^1$, so it is packet $P^2$ which will smoothly replace $P^3$ (Fig. 3(a)). Before $P^1$ can move on by one hop, we must also consider the time for packet $P^2$ to go from the output buffer of $SW_1$ to the output buffer of $SW_2$ (Fig. 3(b)), which is $U'_2$. Thus, we can write:
\[ u_i^0 = U_i^0 = \text{MAX}(U_i^1, U_i^2) + U_i^2 \]

which traces back to Eq. 3. As mentioned above, this is the delay for \( P^1 \) to move one hop on, but equivalently is also the delay for \( P^0 \) to replace it in the previous location (Fig. 3(c)).

Now, similarly, \( P^0 \) needs to move another hop on, from the input buffer of \( SW_1 \) to the input buffer of \( SW_2 \), with a delay which is defined as \( u_i^1 \). It is possible to use the relation \( u_i^1 = U_i^0 \) based on the equations described in previous section, but for clarity, we always describe \( u_i^1 \) based on \( U_i^1 \).

As shown in Fig. 3(d, e, f), in the worst case, packet \( P^0 \) should wait for a packet \( P^4 \) of \( F_2 \). A packet \( P^1 \), again either from \( F_1 \) or \( F_2 \), should be considered at the output buffer of \( SW_1 \). So again, during the time \( \text{MAX}(U_i^1, U_i^2) \), while \( P^1 \) moves on to the output buffer of \( SW_2 \), \( P^4 \) will replace it. \( P^4 \) itself then takes \( U_i^1 \) to move on, allowing \( P^0 \) to eventually get to the input buffer of \( SW_2 \). Thus,

\[ u_i^1 = \text{MAX}(U_i^1, U_i^2) + U_i^2 \]

In a similar manner \( u_i^2 \) can be calculated. Once \( P^0 \) is in the input buffer of \( SW_3 \), it is only one hop away from its destination and as there is no contending flow at the destination, the ejection time for the messages equals \( L_1 \). For the sake of uniformity of presentation, we can write

\[ u_i^2 = U_i^1 = L_1 \]

Now, the target metric \( U_{B1} \) can be calculated recursively, as a function of \( U_i^1 \) variables for the whole network starting from the last hop of each flow. The calculation of all intermediate values, e.g. \( U_i^1, U_i^2, U_i^3, U_i^4, U_i^5 \) and of the relevant metrics \( U_B \) and \( M_{L_1} \), is shown in Fig. 4. Please note that to speed up the recursion steps, intermediate values can be calculated only once and then stored and used later.

When considering the source \( S_{2,3} \) it can be noticed that two flows \( F_2 \) and \( F_3 \) can originate from it; therefore, source conflicts may happen. As Fig. 4 shows, for example, when analyzing flow \( F_2 \), \( u_i^3 \) (the time to transfer a packet of \( F_2 \) into the input buffer of the first switch) should include a delay \( \text{MAX}(U_i^3, U_i^4) \), which accounts for a packet of either \( F_2 \) or \( F_1 \) to move away from the input of \( SW_2 \) towards the input of \( SW_2 \) (during which time we must assume, in the worst case, that it is a packet of \( F_3 \) which replaces it), and then again the time \( U_i^0 \) for this latter packet to also move on, and finally letting a packet from \( F_2 \) in. The calculation for \( u_i^0 \) is done similarly.

\[
\begin{align*}
 u_i^1 &= U_i^0 + u_i^1 \\
 u_i^2 &= \text{MAX}(U_i^1, U_i^2) + U_i^2 \\
 u_i^3 &= \text{MAX}(U_i^3, U_i^4) + U_i^4 \\
 u_i^4 &= \text{MAX}(U_i^5, U_i^6) + U_i^6 \\
 u_i^5 &= \text{MAX}(U_i^7, U_i^8) + U_i^8 \\
\end{align*}
\]

Figure 4: The complete calculation of \( U_{B1} \) and \( M_{L_1} \) for the example NoC of Fig. 2 in method RTB-HB

2) The case \( B_d < L \)

The same values calculated for the case \( B_d = L \) can be used for the case \( B_d < L \). Assume network \( A \) with \( B_d < L \) and a completely equivalent network \( B \) but with \( B_d = L \). Let us consider \( U_{B1}(A) \) and \( U_{B1}(B) \) to be the upper bound delays for these networks. It is possible to use value of \( U_{B1}(B) \) instead of \( U_{B1}(A) \) for network \( A \). The reason is simply that the number of contending flows along the path of a flow is the same in both the networks \( A \) and \( B \), but the number of in-flight packets in the worst-case situation is larger for network \( B \).

In order to tighten the performance bounds, we also present equations to calculate \( U_{B1}(A) \) directly, achieving lower figures than those calculated above for network \( B \). For this purpose, we introduce a new parameter \( \delta_i \) and extend the definitions of \( u_i^i \) and \( U_i^i \) to support the case \( B_d < L \) for network \( A \); the new definitions comply with those used for the case \( B_d = L \).

Definition 1: We denote by \( \delta_i \) the worst-case delay (in cycles) elapsing from the moment when the header flit of \( P_i \) enters the arbitration point of switch \( j + 1 \) to the moment when the tail flit leaves the arbitration point of switch \( j \).

In particular, when \( B_d = L \) (as in network \( B \)), this time is zero, since as soon as the header flit enters the arbitration point of switch \( j + 1 \), the tail flit has left the arbitration point of switch \( j \).

In order to calculate the parameters for network \( A \), it is needed to calculate the values of \( \delta_i \) for different \( i \) and \( j \) values. Here, for convenience, we consider \( L \) being divisible by \( B_d \); thus, when there is a sufficiently long cascade of switches, a whole packet can fit between the arbitration points of the first and last switches. For this purpose, the number of required switches is \( L/B_d + 1 \) (2 when \( B_d = L \)).
To calculate $\delta^j_i$, since the header flit is at the arbitration point of switch $j + 1$, $L - B_d$ flits of the packet have not passed the arbitration point of switch $j$, and therefore the header flit needs to traverse $S$ more switches ($S = (L - B_d)/B_d = L/B_d - 1$), to ensure that the tail flit passes the arbitration point of switch $j$. If the number of remaining switches in the path of flow $i$ is smaller than $S$, the packet simply exits the network at the destination node flit by flit, until the tail flit has left the arbitration point of switch $j$; thus,

$$
\delta^j_i = \begin{cases} 
\sum_{k=1}^{S} u^{j+k}_i (A) & \text{when } j + s \leq h_i \\
\sum_{k=1}^{h_i - j} u^{j+k}_i (A) + (j + s - h_i) \times B_d & \text{when } j + s > h_i, j = 0, 1, \ldots, h_i 
\end{cases}
$$

**Definition 2:** We denote by $U^j_i (A)$ the worst-case delay elapsing from the moment when the header flit of $P_i$ enters the arbitration point of switch $j + 1$ to the moment when the tail flit leaves this point. When $B_d = L$ (as in network $B$), this definition matches the previous definition for $U^1_i (B)$ in Eq. 5.

$$
U^j_i (A) = \max_x \left( U^{j+1}_i (A) - \delta^j_i, U^{j+1}_i (A) - \delta^j_i + \sum_x u^{j+1}_i (A) + \delta^j_i \right), x = 1, \ldots, z_c (i, j + 1), 0 \leq j \leq h_i - 1
$$

and

$$
U^{h_i}_i (A) = L
$$

In case $B_d < L$, a packet occupies more buffering space than that available between arbitration points of two adjacent switches. Consider now a situation where the header flit of a packet $p$ of flow $i$ is at the arbitration point of switch $j + 1$, while an interfering packet $q$ of flow $t$ is also traversing switch $j + 1$, but with its header already at the arbitration point of switch $j + 2 + S$ ($S = L/B_d - 1$). Also, a number $z_c (i, j + 1)$ of packets at different input ports of switch $j + 1$ are contending with $p$ for the same output port. Before the tail flit of packet $p$ can leave the arbitration point at switch $j + 1$, the following must happen:

1) Packet $q$ should proceed for $B_d$ steps (registers), so that its tail flit leaves the arbitration point of switch $j + 2$. After this step the header flit of one of the $z_c (i, j + 1)$ packets should be at the arbitration point of switch $j + 2$. This time is calculated as $u^{j+2}_i (A)$; it is easy to see that $u^{j+2}_i (A) = U^{j+2}_i (A) - \delta^j_i$. As $t$ can be selected from any of $i$ or $z_c (i, j + 1)$ flows, the term to calculate the required time for this step is:

$$
\max_x \left( U^{j+2}_i (A) - \delta^j_i, U^{j+2}_i (A) - \delta^j_i + \sum_x u^{j+2}_i (A) + \delta^j_i \right), x = 1, \ldots, z_c (i, j + 1).
$$

2) All the $z_c (i, j + 1)$ packets, to account for the worst case, shall be arbitrated before $p$; so, after this step the header of $p$ is at the arbitration point of switch $j + 2$. The required time for this step is

$$
\sum_x U^{j+3}_i (A) + x + 1 \cdot z_c (i, j + 1).
$$

3) Packet $p$, whose header is at the arbitration point of switch $j + 2$, must proceed for a number of steps until its tail flit leaves the arbitration point of switch $j + 1$. This time is calculated as $\delta^j_i + 1$.

The summation of the time for these steps results in Eq. 7. As can be seen in Eq. 8, in the last switch for a flow, $L$ cycles are required for the packet to be ejected to the destination.

**Definition 3:** We denote by $u^j_i (A)$ the worst-case delay elapsing from the moment when the header flit of $P_i$ enters the arbitration point of switch $j$ to the moment when the tail flit of the packet is at the arbitration point of the first switch. Also, for $j = h_i$, it is the worst-case delay elapsing from the moment the header flit of $P_i$ enters the arbitration point of the last switch $h_i$ to when a number of $B_d$ flits of the packet have exited the network at $D_1$.

$$
u^j_i (A) = \max_x \left( u^j_i (A) - \delta^j_i, u^j_i (A) - \delta^j_i + \sum_x u^j_i (A) + \delta^j_i \right), x = 1, \ldots, z_c (i, j), 0 \leq j \leq h_i - 1
$$

and

$$
u^{h_i}_i (A) = B_d + \sum_x u^{h_i}_i (A), x = 1, \ldots, z_c (i, h_i)
$$

It is obvious that calculation of $u^j_i (A)$ is like $u^{j-1}_i (A)$ except for step 3 which is not required and term $\delta^j_i$ is omitted; thus, we can write:

$$
u^{j-1}_i (A) = u^j_i (A) + \delta^j_i, j \geq 1
$$

Using the above definitions the equations to calculate $UB_i (A)$ and $MI_i (A)$ can be summarized as:

$$
UB_i (A) = ts_1 + ts_2 + \sum_j u^i_j (A) + (L - B_d)
$$

and

$$
MI_i (A) = ts_1 + u^i_0 (A) + \delta^0
$$

As seen in Eq. 9, in order to calculate $u^j_i (A)$, different values of $\delta^j_i$ should be calculated, which in turn requires $u^k_i (A)$ with $k > j$; so the order of calculation of $u^j_i (A)$ values should be from the larger switch indices towards the smaller ones, i.e. from destinations towards sources.
In order to show the implementation for the case $B_d < L$, Fig. 5 shows the case of $L = 4$ and $B_d = 2$ for the example scenario in Fig. 2. In order to calculate $UB_1$, the values $u_1^I$, $u_2^I$, $u_1^D$, and $u_2^D$ should be calculated. $u_1^I$ as described in Eq. 10 is equal to $B_d$ as there is no flow contention on the path connected to $D_1$ in $SW_3$. Eq. 9 is used to calculate $u_1^D$, which is equal to the time needed for the header flit of a packet $P_i$ of $F_1$ to move from the arbitration point of $SW_2$ to the arbitration point of $SW_3$. As there is no contention from flows from different input ports of the term $\sum_j u_{1(j)}^I(A)$ in Eq. 9 is zero; thus, the value of $\text{MAX}(u_1^D - \delta_j^D, u_1^D - \delta_1^D)$ is calculated (the reason for using the $\text{MAX}$ operator is that the type of the packet just ahead of $P_i$ may be either from $F_1$ or $F_2$). It is obvious that flow contention happens between $F_1$ and $F_2$ for the output port of $SW_4$ which is connected to $SW_2$; thus, to calculate $u_2^D$ this contention should be considered and based on Eq. 9, the value is $\text{MAX}(u_2^D - \delta_j^D, u_2^D - \delta_j^D) + u_2^D$. To calculate $u_2^D$, as there is no source conflict, the term $\sum_j u_{1(j)}^I(A)$ is zero and the term $\text{MAX}(u_1^I(A) - \delta_j^I, u_{1(j)}^I(A) - \delta_1^I)$ will result in $u_1^I - \delta_1^I$. The calculation for other flows is done in the same manner.

$$\begin{align*}
 u_1^I &= B_d \\
 u_1^D &= \text{MAX}(0 - \delta_j^D, u_1^D - \delta_1^D) = 2L - B_d \\
 u_2^I &= \text{MAX}(0 - \delta_j^D, u_2^D - \delta_j^D) + u_2^D = 2L - B_d \\
 u_2^D &= \text{MAX}(0 - \delta_j^D, u_2^D - \delta_j^D) + u_2^D = 2L - B_d \\
\end{align*}$$

In other words, the minimum guaranteed bandwidth in RTB-HB does not depend on the network's buffering space when buffer depth is larger than message length. Indeed, in this scenario, we assume that a buffer of size $m \times L$ exists just before the arbitration point of the inputs of the switches and for the calculation purposes we assume $m - 1$ dummy switches are placed to split the buffer to $m$ stages; as the delay for all the stages are equal, we can compare these stages in the network with a pipeline in which increasing or decreasing the number of stages does not affect the data injection rate into the pipeline although the delay will increase when $m$ increases. This is true for the buffers just at the outputs of the source cores and thus the injection rate to the network is not affected when $m$ changes. This is better shown in an example in the next section. Also as in section 4.4.1.2 we have proposed the solution for parameterized packet Length of $L$ and $B_d = 1$, it is possible to combine this solution with dummy switches solution to support arbitrary $B_d$ on different switches (i.e. $B_{d-j}$ on switch $j$). It is possible to consider spreading $B_{d-j} - 1$ dummy switches among the elements of a buffer of depth $B_{d-j}$, then re-enumerate all the real and dummy switches in the network and solve the equation for $B_d = 1$ (in the whole network) as suggested in section 4.4.2.

The message length $L_i$ can vary on a per-flow basis. To tackle this challenge, let us call the shortest message length $L_{\text{min}}$. Let us first consider $B_d \leq L_{\text{min}}$. In this case, it is intuitively possible to use the same equation as in Section 4.4.1. The only difference is that for every individual flow, $L_i$ must be used as a parameter instead of $L$. Also, for $B_d > L_{\text{min}}$ it is possible to use exactly the same approach used in Section 4.4.2. To further describe this, when $B_d = m \times L_{\text{min}} + k$, with $0 < k < m$, it is possible to consider $m$ dummy switches (or alternatively $B_d - 1$ dummy switches as described above) between each two consecutive switches and do the calculations as described in Section 4.4.2, using $L_i$ values for different flows. Thus, based on the approach of variable message length described here, throughout the paper, we have used $L_i$ instead of $L$.

4) Virtual channels

The extension of the proposed method to support virtual channels is straightforward, provided that the index of the
virtual channel used for a flow in different switches is a predefined parameter. For this purpose, all virtual channels in intermediate switches can be considered as physical channels in the proposed model. Arbitration conflicts would be among all the incoming virtual channels, and each output virtual channel would account for a separate output port (so, e.g., the value for $U_i^{h_1}$ would change from $L_i$ to $m \times L_i$, assuming $m$ virtual channels per physical channel).

In the following equations, the modifications needed to support virtual channels are shown. The parameter $u_i^{[v_{ij}]}$ denotes the maximum delay to move from the arbitration point of switch $j$ to the arbitration point of switch $j+1$ when using the predefined input virtual channel $v_{ij}$ used for flow $i$ at the input of switch $j$. Similarly, $U_i^{[v_{ij}+1]}$ is the maximum delay to move from the output buffer of switch $j$ to the output buffer of switch $j+1$ using input virtual channel $v_{ij+1}$ at the input of switch $j+1$. Also $v_{ij+1}$ is defined as the virtual channel index used for flow $i$ at the input port of the destination ($D_i$) IC core. The parameter $u_0^{[1]}$ is used to calculate $M_i$ as shown below; in this case, the selection of virtual channel number 1 is mandatory as only one flow enters the virtual switch inside an IP core.

$$u_i^{[v_{ij}]} = \max(J_i^{[v_{ij}]} U_i^{[v_{ij+1}]} [v_{ij+1}], U_i^{[v_{ij+1}]} [v_{ij+1}]) + \sum \delta_i^{[v_{ij+1}]} [v_{ij+1}], x = 1 - \delta_i^{[v_{ij+1}]} [v_{ij+1}], 0 \leq j \leq h_i$$

$$U_i^{[v_{ij}+1]} = \max(J_i^{[v_{ij}+1]} U_i^{[v_{ij+1}]} [v_{ij+1}], U_i^{[v_{ij+1}]} [v_{ij+1}]) + \sum \delta_i^{[v_{ij+1}]} [v_{ij+1}], x = 1 - \delta_i^{[v_{ij+1}]} [v_{ij+1}], 0 \leq j \leq h_i - 1$$

$$U_i^{h_1} [v_{ij+1}] = m \times L_i$$

$$U_i = t_s + ts_2 + \sum_j u_i^{[v_{ij}]} j = 0 \ldots h_i$$

$$M_i = t_s, u_0^{[1]}$$

(17)

Table 5: Virtual channel selection for the example network in Fig. 2

<table>
<thead>
<tr>
<th>Source IP</th>
<th>Input of $SW_1$</th>
<th>Input of $SW_2$</th>
<th>Input of $SW_3$</th>
<th>Input of $SW_4$</th>
<th>Destination IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow 1</td>
<td>Vch-1</td>
<td>Vch-1</td>
<td>Vch-1</td>
<td>--</td>
<td>Vch-1</td>
</tr>
<tr>
<td>Flow 2</td>
<td>Vch-1</td>
<td>Vch-1</td>
<td>Vch-1</td>
<td>Vch-1</td>
<td>--</td>
</tr>
<tr>
<td>Flow 3</td>
<td>Vch-1</td>
<td>Vch-2</td>
<td>--</td>
<td>Vch-1</td>
<td>--</td>
</tr>
<tr>
<td>Flow 4</td>
<td>Vch-1</td>
<td>--</td>
<td>--</td>
<td>Vch-1</td>
<td>Vch-2</td>
</tr>
</tbody>
</table>

Figure 7: The calculation of $UB_i$ and $M_i$ for the example NoC in Fig. 2 using RTB-HB, considering 2 virtual channels per physical channel.

To illustrate the support of virtual channels, let us consider the example shown in Fig. 2 with 2 virtual channels per physical channel. In this case, Table 5 shows the mapping between flows and virtual channels at each switch input port. Fig. 7 shows the calculation for this example. By comparing the results with the case where no virtual channels are used, it can be observed that virtual channels can reduce the number of flow contentions in different switches. At the same time, the bandwidth of a physical channel is shared among its virtual channels, thus depending on the application and on the strategy that assigns virtual channels to different flows, the worst-case latency and bandwidth values can be better or worse compared to the case without virtual channels, regardless of extra hardware cost and complexity of virtual channels.

B. The Proposed Delay Model RTB-LL

We present a substantial improvement to the previously published method WCFC [10]. WCFC also calculates the upper bound propagation delays and permitted injection intervals for the flows in a wormhole network. It considers the arbitration contention packets face and the delay incurred by other packets sharing some part of their route due to such blockings. With a notation similar to that used above, WCFC employs [10] the following equation to calculate $UB_i$ and $M_i$:

$$UB_i = t_s + ts_2 + l_i + a + \sum_j u_i^{[v_{ij}]} j = 0 \ldots h_i$$

$$M_i = t_s, U_i^{[v_{ij}]}$$

(18)

In the WCFC method, the calculations are based on the assumption that each flow injects packets with a minimum permitted interval. For the applications that can support such an assumption, we present a method that provides significant improvement in bound tightness over the WCFC method, which we call RTB-LL. As RTB-LL is less pessimistic than WCFC in evaluating worst-case performance, it enables the design of more hardware-efficient NoCs. To improve upon WCFC, a new concept, called overlapping flows, is introduced. If two or more different flows contend for the same output port at a switch, and they also share the same input port, we call such flows overlapping at the switch. This notion allows us to significantly optimize the bound tightness.

When $F_i$ contends with multiple overlapping flows at a switch, it is possible to locally coalesce all such overlapping flows into a single one. This is because the arbitration cannot be lost to multiple of those flows, as they cannot physically produce a contending packet simultaneously given that they enter the switch through the same input port. If there exist e.g. two overlapping contending flows at hop $j$ with delay parameters $U_i$ and $U_i^{[v]}$, it is possible to consider $MAX(U_i, U_i^{[v]})$ as their representative delay, instead of $U_i^{[v]}$, for calculating the parameters of $F_i$. Moreover, whenever $F_i$ overlaps with other flows, those other contending flows should be ignored. By applying these optimizations, we have noticed a significant improvement in bound tightness for RTB-LL, shown in the next section and as summarized in Table 1.

Fig. 8 and Fig. 9 show the calculated $UB_i$ and $M_i$ values for both WCFC and RTB-LL methods for the same example in Fig. 2. Since flows $F_1$ and $F_2$ are overlapping at $SW_2$, our proposed RTB-LL improves the bound tightness compared to WCFC. Consider, for the sake of exemplification, a NoC variant shown in Fig. 10, with another contending flow at $SW_2$. In RTB-LL, the delay $U_2$ of $F_3$ at $SW_2$ can be modeled as $S_d + MAX(U_2, U_2^{[v]})$ instead of the overly pessimistic value $S_d + S_2$ calculated by WCFC.
applications, we assume that NoC topologies are predefined based on application communication requirements, but without any feedback from the proposed algorithms to customize the network structure for better upper bound delay and interval time results (considering such a feedback is a possible extension for future work). In particular, for many applications, it is possible to identify a small subset of flows as critical, and then to optimize the NoC based on feedback loops from RTB-HB and RTB-LL to improve the performance of such critical flows. It is possible to do this without dedicated hardware support or any priority scheme.

\[ S = A + 4S_d + 2L_2 + L_3 + L_4 \]

Figure 8: The complete calculation of \( UB_1 \) and \( ml_1 \) for the example NoC of Fig. 2 with method WCFC

\[ U_w^2 = 0 \]
\[ U_w^1 = S_1 + U_1^1 \]
\[ U_w^0 = S_2 \]
\[ U_1^2 = U_1^2 + U_1^3 \]
\[ U_1^1 = U_1^2 + U_1^2 + L_1 + L_3 \]
\[ UB_1 = a + 3S_d + L_1 + L_2 + L_4 \]
\[ ml_1 = L_1 + L_2 + L_3 + L_4 \]

Figure 9: The complete calculation of \( UB_1 \) and \( ml_1 \) for the example NoC of Fig. 2 with method RTB-LL

\[ U_w^2 = 0 \]
\[ U_w^1 = S_1 + U_1^1 \]
\[ U_w^0 = S_2 \]
\[ U_2^1 = U_2^1 + U_2^1 \]
\[ U_2^0 = U_2^1 + U_2^2 + L_2 + L_4 \]
\[ UB_1 = a + 4S_d + L_1 + L_2 + L_3 + L_4 \]
\[ ml_1 = L_1 + L_2 + L_3 + L_4 \]

Figure 10: NoC variant where flow \( F_3 \) contends with two overlapping flows \( F_1 \) and \( F_2 \) at \( SW_2 \)

To show how virtual channels are supported in RTB-LL, the same example for method RTB-HB is considered. Fig. 11 shows the results. Like in RTB-HB, using virtual channels results in better latency and bandwidth bounds, since flow contents are resolved.

\[ U_B = a + 5S_d + L_3 \]
\[ ml_1 = 2L_4 \]

Figure 11: Calculation of \( UB \) and \( ml_1 \) for the example in Fig. 2 using RTB-L and WCFC, with 2 virtual channels per physical channel

\[ U_B = a + 4S_d + 2L_2 + L_2 + L_4 \]
\[ ml_2 = 2L_4 \]

5. STUDIES ON APPLICATIONS

The proposed methods RTB-HB and RTB-LL can be used to analyze the scheduling of traffic flows in real-world applications. In this section, we present studies on a multimedia application (four other multimedia and RT applications are considered in the appendix). We compare methods RTB-HB and RTB-LL to the baseline method WCFC, using the parameters listed in Table 6. In these

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline
\textbf{L} & \textbf{a} & \textbf{b}_1 & \textbf{b}_2 & \textbf{b}_3 & \textbf{b}_4 & \textbf{S}_d & \textbf{t}_c & \textbf{F} & \textbf{Freq} \\
\hline
4 & 1 & 1 & 1 & 1 & 2 & 0 & 0 & 4 & 0 & 4 & 400 \\
\hline
\end{tabular}
\caption{Network parameters for the study}
\end{table}

A. Case Study: A Multimedia Application

In this section, we compare the results of applying RTB-HB, RTB-LL and WCFC to D26-media (Fig. 12), a real-time multimedia application with 67 communication flows, some of which critical. The application is mapped onto different NoC topologies, each with different switch counts and switch radices. Fig. 14 shows the average flow latency for different analysis methods and topologies. In particular, we have shown the implementation for switch counts up to seven that are typical values and a topology with 20 switches that is a reasonable point with many longer hop flows. Topologies with one or few switches (e.g. 1-3 switches in this example) need them to be “fat” (high-radix), while other cases need “medium” or “thin” (low-radix) switches. It is important to note that the limitations in physical implementation (like power consumption, area, frequency, etc.) may limit the use of fat switches in practice; we still present the results for these cases for the sake of latency comparison, and assume that proper constraints can be implemented at a higher level in the NoC synthesis flow. Two detailed implementations with 5 and 20 switches are shown in Fig. 13. Fig. 15 presents the results in terms of latency, intervals and bandwidth for the whole set of flows for 1-, 5- and 20-switch networks. Fig. 15(a, b, c) compares \( UB_2 \). The RTB-LL model always provides the tightest bounds. Compared to WCFC, the largely improved tightness (more than 50% on average) is due to the analysis of overlapping flows, a novelty of this paper. Please note that the improved tightness comes without any impact on the accuracy of the bounds, which are still under worst-case assumptions. For the topology with only one switch and without overlapping flows in Fig. 15(a), the results for RTB-LL and WCFC are identical but increasing switch counts triggers different performance profiles. RTB-HB is intrinsically expected to return higher worst-case latencies, due to the assumption that no hardware traffic injection regulation facilities are available. Still, due to the more accurate calculation approach, the bounds are on average 30% lower than those given by WCFC, despite the less
restrictive assumptions. There are, however, a few flows for which WCFC predicts lower delays than RTB-HB, due to the regulated injection assumption. In a zero-load scenario (with no contention at all), the minimum theoretical latency to traverse the 5-switch NoC for flows spanning a single hop is 8 cycles ($\delta_d + L$), while RTB-LL gives a minimum upper-bound of 17 cycles in worst-case contention. The delays calculated for the 20-switch topology, in this example, are higher, as a result of longer paths (more hops) per flow, higher probability of contention, and especially for RTB-HB, more in-flight packets. This suggests, as intuitively expected, that NoCs with fewer hops guarantee, on average, lower delay bounds. As described earlier, physical implementation limitations may prevent using fat switches in practice. On the other hand, increasing the number of switches not only increases the system cost but requires a careful consideration in the design process to reduce flow contentions to acquire tighter upper bound delays; so, a trade off may be considered for the number of switches. For RTB-LL, the number of contending in-flight packets is unrelated with the NoC topology, so the delay will not increase as a result of more hops.

Fig. 15(d, e, f) shows the maximum and minimum injection intervals ($Ml_i$ and $ml_i$). Intuitively, if traversal delays are lower, new packets can be injected sooner, so $Ml_i$ ($ml_i$) plots resemble UB$_i$ trends: flows with lower latencies can be injected more frequently. Thus, the $ml_i$ intervals are always shorter in RTB-LL and the $Ml_i$ intervals often shorter in RTB-HB when compared to $ml_i$ in WCFC (except when using a few fat switches). These intervals can be directly translated into $mBW_i$ and $MBW_i$ using the equations in Table 3. Results are shown in Fig. 15(g, h, i). The maximum injectable bandwidths ($MBW_i$) are, on average 35% higher according to RTB-LL when compared to WCFC, and 25% higher according to the minimum bandwidth ($mBW_i$) in RTB-HB. The maximum theoretical injectable bandwidth is 1600 MB/s ($Freq \times FlitWidth$); according to RTB-LL, even under worst-case assumptions, some flows on the 5-switch NoC are guaranteed injection rates of as much as 533 MB/s. In the 20-switch network, the higher contention likelihood affects injectable bandwidth negatively, but the use of more resources has a positive effect on many-hop flows, resulting in comparable injectable bandwidths. In summary, NoCs with few hops exhibit clearly better flow average upper-bound traversal delays, but in terms of injectable bandwidths, the mapping of the flows (i.e. the contention patterns) and the amount of used resources play a decisive performance role.

B. The Effect of Virtual Channels

The results of employing different number of virtual channels in the network in Fig. 12 are reported in Fig. 16 and Fig. 17. Here we consider D26-media application with RTB-HB and RTB-LL methods for one, two and four virtual channels per physical channel. The strategy that assigns virtual channels to the flows is to share the load of input ports among the virtual channels and thus minimizing the contentions on switch output ports. The figures show that for both methods increasing the number of virtual channels results in better average RT metrics for different flows.

C. Study with Variable Buffer Depths

Fig. 18 and Fig. 19 show the comparison of worst-case delay and maximum interval for D26-media application for the case $B_d < L$. The figure suggests that shallower buffers, when the message length is fixed, will result in smaller worst-case delay and maximum interval bounds. Fig. 20 illustrates the effects of increasing the buffer depth $B_d$, in all switches of the NoC, as an integer multiple of message length $L$. The test is run for the D26-media application using method RTB-HB. A linear relationship can be observed between $B_d$ and the upper bound delay UB$_i$. But because of the pipeline effect, as described in section 4.2.1, there is no such a relation between $B_d$ and $mBW_i$.

A contradiction may be perceived since deeper buffers are generally expected to improve performance, while Fig. 20(a) reveals worse latency with deeper buffers. In fact, the average latency is probably improved with a larger $B_d$, but worst-case latency is not, as shown in Fig. 20(a).
To understand why the worst-case latency deteriorates, consider the following explanation. When the basic case of $B_d = L$ is considered, RTB-HB calculates the maximum delay for a packet $P_i$ from the time the packet is supposed to be injected into the network until it is ejected at the destination. Since the output buffer of the source core has a depth of $B_d = L$, it is not possible to have more than one packet in this buffer awaiting to be serviced before $P_i$. In the worst situation, the traffic generator can inject a packet every $W_l$ cycles, at most; if it injects more, the traffic generator may be stalled by NoC backpressure until this interval elapses. In the more complex case $B_d > L$, some packets may be queued in the source core buffer ahead of $P_i$, and since RTB-HB imposes no restriction on injection rates, they are expected to be there by a worst-case analysis. Thus, the calculated worst-case delay for $P_i$ includes the extra time needed to service them. Exactly the same reasoning applies to $B_d > L$ in other intermediate buffers in the network. As a consequence, deeper buffers increase worst-case latencies, but this is not incompatible with the fact that increasing buffer size will decrease the average delay. Indeed if we apply the same traffic pattern to two identical networks, but with different buffer sizes, the total average delay in the network with larger buffers will typically be lower than that of the other network. Our method RTB-LL does not consider the buffering space $B_d$ to calculate the worst-case delay; instead it uses stage delay $S_j$. Therefore changing the buffer size will not affect the calculated worst-case delay.

D. Suitability to Critical Flows

Fig. 21 shows the average $UB_i$ traversal delay and the average $mBW_i$ injectable bandwidth for the flows traversing $x$ hops of the 5-switch NoC, considering the D26-media application and using RTB-HB. It is seen 1-hop flows exhibit reasonably low latencies and high bandwidths, suitable for critical traffic loads. Thus, the proposed methodology has a clear applicability to industrial RT applications.

6. COMPLEXITY OF THE METHODS

To estimate the time complexity of the proposed methods, we calculate the maximum number of required operations. As Eq. 1, Eq. 3 and Eq. 5 show, the only operations are additions and comparisons (for the MAX operator); we consider one cycle to execute each of such operations. We call $h$ the maximum number of switches traversed by a flow, and $k$ the number of flows. We also pessimistically assume the maximum number of contending flows at a switch output to be $k$. For calculating one $U_i^j$ parameter, we need (according to Eq. 5) at most $k$ comparisons and $k$ additions, thus a total of $2k$ operations. The number of $U_i^j$ parameters to be calculated is $hk$; so, the maximum number of operations is $2hk^2$. Each $u_i^j$ parameter (except for $j = 0$) can be derived from the equality $u_i^j = u_i^{j+1} + 1$; thus, we only need to calculate the case of $u_i^0$ for all flows. In this case, one $u_i^0$ (Eq. 3) needs $2k$ operations; for all $u_i^0$ parameters we need $2k^2$ operations.

In RTB-HB, the outcome is $k \times UB_i$ and $k \times M_l$ values. For calculating one $UB_i$ value (according to Eq. 1), we need $h + 1$ additions; so, for all $k \times UB_i$ values, we need $(h + 1)k$ operations, while $k$ operations are needed in the case of $M_l$. The total number of operations is the summation of the all above, i.e. $2hk^2 + 2k^2 + (h + 1)k + k$. Therefore, the complexity of the algorithm is $O(hk^2)$.

For RTB-LL, using the same approach, we can show that the complexity of the algorithm for calculating $UB_i$ and $m_i$ is again $O(hk^2)$. Thus, both algorithms have quadratic time complexity. Also the timing complexity of WCFC algorithm is similar to RTB-LL as it exhibits a similar recursive behavior [10]. In practice, the execution time for all our test applications is very small (few seconds on a standard PC) and the modeling of delay and bandwidth parameters does not pose significant runtime issues.
7. CONCLUSION AND FUTURE WORK

We have proposed two different methods to characterize bandwidth and latency for NoC-based real-time SoCs, aiming at guaranteed QoS provisions. The choice of the most suitable method depends on the performance demands of the system and on whether dedicated hardware facilities can be provided by the NoC. One method is aimed at applications demanding the minimum latencies and requires injection regulation, while the other is suitable for applications where packet injection must be flexible to accommodate for higher average injected bandwidths and no hardware regulation is possible. We have proved that the proposed methods return the worst-case metrics in a much tighter way than existing approaches, rendering them quite applicable for real-world SoC applications. The next step is to use the results of this work as an input to NoC synthesize and optimization tools whereby the QoS demands of critical traffic flows are met.

ACKNOWLEDGMENT

This work is partly supported by EU ARTIST-DESIGN and ARTEMIS project SMECY.
Dara Rahmati received his B.Sc. in computer engineering from University of Tehran, Tehran, Iran, in 1998, his M.Sc. in computer engineering from University of Tehran in 2001. He is currently Ph.D. student of computer engineering at Sharif University of Technology, Tehran, Iran. His research interests are in design and performance evaluation of optimized Networks-on-Chip architectures including topology selection, routing and switching algorithms, performance boundary evaluation and QoS support for Real Time Networks-on-Chip.

Srinivasan Murali received the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Palo Alto, CA, in 2007. He is a Co-Founder and the Chief Technical Officer with NoCs, Lausanne, Switzerland. He is also currently a Research Scientist with Ecole Polytechnique Federale de Lausanne, Lausanne. He has authored a book and presented over 40 publications in leading conferences and journals. His current research interests include interconnect design for systems-on-chips, thermal modeling, and reliability of multi-core systems. Dr. Murali is a recipient of the European Design and Automation Association Outstanding Dissertation Award in 2007 for his work on interconnect architecture design. He received the Best Paper Award at the Design Automation and Test in Europe Conference in 2005. One of his papers has also been selected as one of “The Most Influential Papers of 10 Years DATE.”

Luca Benini is Full Professor at the Department of Electrical Engineering and Computer Science (DEIS) of the University of Bologna. He also holds a visiting faculty position at the Ecole Polytechnique Federale de Lausanne (EPFL) and he is currently serving as Chief Architect for the Platform 2012 project in STMicroelectronics, Grenoble. He received a Ph.D. degree in electrical engineering from Stanford University in 1997. Dr. Benini’s research interests are in energy-efficient system design and Multi-Core SoC design. He is also active in the area of energy-efficient smart sensors and sensor networks for biomedical and ambient intelligence applications. He has published more than 500 papers in peer-reviewed international journals and conferences, four books and several book chapters. He has been general chair and program chair of the Design Automation and Test in Europe Conference. He has been Associate Editor of several international journals, including the IEEE Transactions on Computer Aided Design of Circuits and Systems and the ACM Transactions on Embedded Computing Systems. He is a Fellow of the IEEE, a member of the Academia Europaea, and a member of the steering board of the ARTEMISIA European Association on Advanced Research & Technology for Embedded Intelligence and Systems.

Federico Angiolini received the M.S. degree (summa cum laude) in electrical engineering from the University of Bologna, Bologna, Italy, in 2003, and the Ph.D. degree from the Department of Electronics and Computer Science, University of Bologna, in 2008. He is currently the Vice President of Engineering with the InNoCs, Lausanne VD, Switzerland. His current research interests include memory hierarchies, multiprocessor-embedded systems, and networks-on-chip.

Giovanni De Micheli (S’79–M’79–SM’80–F’94) is currently a Professor and the Director of the Institute of Electrical Engineering and the Integrated Systems Centre, Ecole Polytechnique Federale de Lausanne, Switzerland. He is also a Program Leader of the Nano-Tera.ch program. He was a Professor of Electrical Engineering at Stanford University. His research interests include several aspects of design technologies for integrated circuits and systems, such as synthesis for emerging technologies, networks on chips 3-D integration, heterogeneous platform design including electrical components and biosensors, as well as in data processing of biomedical information. Prof. De Micheli is the recipient of the 2003 IEEE Emanuel Piore Award. He is a Fellow of the Association for Computing Machinery. He was also the recipient of the Golden Jubilee Medal for outstanding contributions to the IEEE CAS Society in 2000 and the 1987 D. Pederson Award for the best paper on the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS (TCAD/ICAS). He was the Division-1 Director (2008–2009), Co-Founder and the President Elect of the IEEE Council on EDA (2005–2007), the President of the IEEE CAS Society 2003, the Editor in Chief of the IEEE TCAD/ICAS (1987–2001). He has been the Chair of several conferences, including DATE (2010), pHealth (2006), VLSI SOC (2006), DAC (2000), and ICCD (1989).

Hamid Sarbazi-Azad received the BSc degree in electrical and computer engineering from Shahid-Beheshti University, Tehran, Iran, in 1992, the MSc degree in computer engineering from Sharif University of Technology, Tehran, in 1994, and the PhD degree in computing science from the University of Glasgow, Glasgow, United Kingdom, in 2002. He is currently professor of computer engineering at Sharif University of Technology, and heads the School of Computer Science of the Institute for Research in Fundamental Sciences (IPM), Tehran, Iran. His research interests include high performance computer architectures, NoCs and SoCs, parallel and distributed systems, performance modeling/evaluation, graph theory and combinatorics, and wireless/mobile networks, on which he has published more than 200 refereed conference and journal papers. He received Khwarizmi International Award in 2006 and TWAS Young Scientist Award in engineering sciences in 2007. He is a member of the managing board of Computer Society of Iran (CSI), and has been serving as the editor in chief for the CSI Journal on Computer Science and Engineering since 2005. He is an associate editor of IEEE Transactions on Computers and has guest edited several special issues on high-performance computing architectures and networks (HPCANS) in related journals. He is a member of the ACM and IEEE.