A Method for Calculating Hard QoS Guarantees for Networks-on-Chip

Dara Rahmati*, Srinivasan Murali[¥§], Luca Benini[‡], Federico Angiolini[¥§],
Giovanni De Micheli[§], Hamid Sarbazi-Azad*
*HPCAN, Sharif University of Technology, Tehran, Iran, {d_rahmati@ce.sharif.edu, azad@sharif.edu}
[¥]iNoCs Sarl, {murali, angiolini}@inocs.com
[§]LSI, EPFL, Lausanne, Switzerland, giovanni.demicheli@epfl.ch
[‡]DEIS, University of Bologna, Bologna, Italy, luca.benini@unibo.it

ABSTRACT
Many Networks-on-Chip (NoC) applications exhibit one or more critical traffic flows that require hard Quality of Service (QoS). Guaranteeing bandwidth and latency for such real time flows is crucial. In this paper, we present novel methods to efficiently calculate worst-case bandwidth and latency bounds and thereby provide hard QoS guarantees. Importantly, the proposed methods apply even to *best-effort NoC architectures, with no extra hardware dedicated to QoS support*. By applying our methods to several realistic NoC designs, we show substantial improvements (on average, more than 30% in bandwidth and 50% in latency) in bound tightness with respect to existing approaches.[1]

Categories and Subject Descriptors
B.4.3 [**Hardware**]: Interconnections (Subsystems)
C.1.2 [**Processor Architectures**]: Multiple Data Streaming Architecture (MultiProcessor) – Interconnection Architectures

General Terms
Algorithms, Performance and Design.

Keywords
SoC, NoC, QoS, Wormhole switching, Real-time guarantees, Performance analytical model.

1. INTRODUCTION
The Networks-on-Chip [1, 2] paradigm has emerged in recent years to overcome the scalability limitations of point-to-point signal wires, shared buses or segmented buses [1, 3], which do not scale well in power, performance and design complexity [4, 5, 6]. While the scalability and efficiency advantages of NoCs have been demonstrated in many occasions, their timing predictability and suitability to transport real-time (RT) 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 inter-core traffic flows. On a NoC fabric this translates to guaranteed QoS 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 of these ideas. All these approaches imply hardware overhead and often come with strict service disciplines that limit NoC flexibility and penalize average performance to provide worst-case guarantees. In fact, NoCs prototypes are often classified as being either best-effort or QoS, depending on the availability of hardware support for RT traffic.

Our work takes a new viewpoint. We consider best-effort NoC architectures without special HW support for QoS traffic. We only assume that the traffic injected by network end-nodes is known and characterized in terms of its worst-case behavior. We then formulate algorithms to *find conservative 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*. Our approach is inspired by the work by Lee et al. [10] for traditional multiprocessor networks, which we extend in several directions.

We propose two different methods for characterizing worst-case packet injection. The first method, RTB-HB (Real-Time Bound for High-Bandwidth traffic), is used for NoCs supporting application workloads where the injected flows have high demands of average bandwidth and require a guaranteed worst-traffic minimum bandwidth (*mBW*) and maximum upper bound latency (*UB*). In this case we do not assume any restriction on traffic injection rate: a flow can send packets whenever the network has buffer capacity to accept them. The second method considers applications with latency-critical flows which require low and guaranteed UB values, but have moderate bandwidth requirements and do not send packets at intervals 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 RBT-HB is completely new. Table 1 summarizes a cross-comparison of RTB-HB, RTB-LL and WCFC methods.

ICCAD'09, November 2–5, 2009, San Jose, CA, USA.

[1]This work has been financially supported partly by EU ARTIST-DESIGN and Predator projects and also partly by Iran Telecommunication Research Center (ITRC), an affiliation of the Ministry of Communications and Information Technology.

Table 1: Upper bound delay and bandwidth comparison in different methods

Methods	WCFC (W)	RTB-LL (LL)
RTB-HB (HB)	$UB_{HB} \leq UB_W$ $mBW_{HB} \geq MBW_W$	$UB_{HB} \geq UB_{LL}$ $mBW_{HB} \leq MBW_{LL}$
RTB-LL (LL)	$UB_{LL} \leq UB_W$ $MBW_{LL} \geq MBW_W$	

The remainder of the paper is organized as follows. Section 2 summarizes related work. Section 3 gives definitions and basic concepts. Section 4 describes methods RTB-HB and RTB-LL. Section 5 focuses on experimental results and quantitative comparisons. Section 6 describes the time complexity of the methods. Finally, section 7 concludes the paper.

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 [11 - 20]. Here we focus on RT-NoCs, which have often been called QoS-NoCs.

QoS is an important issue for many application domains, such as multimedia, aerospace and military applications. Many of these applications have one or more traffic flows that have real time requirements and need hard QoS guarantees. Some NoC architectures provide hard QoS support by using special hardware mechanisms. In [9], Goossens et. al present the Æthereal NoC, which combines guaranteed services with best effort services to guarantee QoS in NoCs. The MARS [21] architecture uses TDMA (Time division multiplexing) mechanism to provide real-time guarantees on packet switched networks. In Shi et. al [7], a priority-based wormhole switching method for scheduling of RT flows is presented. In [8], Paukovits et. al propose the concept of a predictable Time-Triggered Network-on-Chip (TTNoC) that realizes QoS based communication services. Many other works have been published with variations and improvements over these basic ideas [22-33]. However, most NoC architectures are best-effort [34] and do not have special hardware mechanisms to guarantee QoS. Today, there is no available method that can calculate worst-case bandwidth and delay values of a general best-effort NoC design. Our work addresses this open issue.

3. 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. For maximum generality, we consider buffering at input and (optionally) output ports. We assume round-robin arbitration in the switches, a commonly used arbitration scheme in many NoCs. Virtual channels can be supported by the proposed methods. Links, which can be pipelined to maximize the operating frequency, connect the output ports to the input ports of adjacent switches. Table 2 summarizes the parameters that we will use in the following to describe the network. For the sake of simplicity, we use a single parameter *Freq* for the operating frequency of all cores and a single *FlitWidth* as the data width of the NoC links.

Table 2: Network model parameters and symbols

Parameter	Description
Freq	Clock frequency of the system
a	Number of pipeline stages (registers) used to segment NoC links
b_1	Depth of switch input buffers
b_2	Number of pipeline stages of the switch crossbar (0 if combinational)
b_3	Depth of switch output buffers (0 if none)
b	$:= b_1+b_2+b_3$
B_d	Buffer depth parameter, $:= a+b_1+b_2+b_3 = a+b$
ts_1	Latency overhead for injecting a packet into the network
ts_2	Latency overhead for ejecting a packet at the destination
FlitWidth	Width of NoC links, in bytes
SW_j	j-th switch of the network

The buffer depth (B_d) parameter will be used frequently in the paper. As can be seen in Fig. 1, B_d is the summation of the number of registers or buffers from the arbitration point (at the entry of the crossbar) of switch j to the arbitration point of switch $j+1$. b_1 is the depth of the input buffer (we assume at least one register), b_2 is the number of registers in crossbar switch (if any), b_3 is the depth of the output buffer (if any), and a is the number of registers (if any) along a link to compensate the propagation delay of the wires. It is important to note that b_2 and a represent latencies that packets will face even in the absence of congestion, while b_1 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. Blocking always happens because of arbitration conflicts, either directly, when 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 between the arbitration points of switches j and $j+1$, i.e. to B_d. 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 to model source conflicts (i.e. sending more than one flow from a source node).

The parameters ts_1 and ts_2 model the setup time at NoC sources and 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.

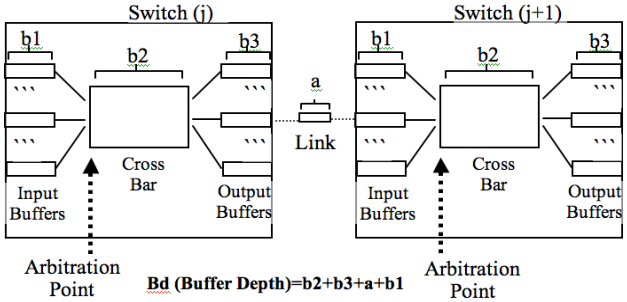

Figure 1: Switch model and parameters

4. NOC TRAVERSAL DELAY ANALYSIS

Table 3 lists the parameters we define to describe traffic flows across the network, while Table 4 summarizes the parameters that we identify to model the performance of such flows. Most

notably, UB_i represents the upper-bound delay for network traversal by a packet of flow F_i, and is a key figure 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_i 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 interconnect analysis methods, as most available IP cores operate on an unregulated-injection basis. In order to calculate UB_i in such a case, we consider all the intermediate buffers along the route full, and we assume arbitration losses at all switches against all other contending flows. 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 flow source S_i are eventually delivered. RTB-HB calculates the interval MI_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. Please note that 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 possible in some application scenarios. Therefore, we also calculate a minimum permitted interval (mI_i) between two consecutive packets from the same source, which can be translated into a maximum permitted bandwidth (MBW_i). This approach is similar to previously published method [10] (Real-time wormhole channel feasibility checking or WCFC, which will be briefly described later), but delivers substantially better results in terms of bound tightness. For proper operation, the system must then respect MBW_i bounds at runtime. Compared to method RTB-HB, in the calculations for method RTB-LL, it is possible to consider parameters *b1* and *b3* to represent the buffers delay (*b1*=1 and *b3*=1 or 0, when there is no output buffer), instead of the exact buffer size.

4.1 The Proposed Delay Model RTB-HB

The goal here is the calculation of the parameters UB_i (worst-case latency to traverse the network) and MI_i (maximum worst-case interval). In this paper, due to lack of space, we only show the case where B_d <= L. For a B_d > L, we can introduce dummy switches for each port, each with B_d <= L and use the same analysis below. Let us first consider the case $B_d = L$. To grasp intuitively the analysis, please observe that $B_d = L$ means, in concrete terms, that a packet fills exactly the buffering resources between the arbitration points of two adjacent switches. Considering as an example a network completely full of traffic (an unrealistic scenario just for visualization purposes), and $B_d = L$ globally, the network operates by shuffling packets around in lockstep: all switches re-arbitrate simultaneously every L cycles, and packets trail each other, filling up buffers as soon as they free up.

Table 3: Traffic model parameters

Parameter	Description
F_i	*i*-th traffic flow in the network
L_i	Length of the packets of F_i, in flits
S_i	Source node of F_i
D_i	Destination node of F_i
P_i	Packet of F_i
h_i	Number of switches (hops) along the path of F_i

Table 4: Performance model parameters

Parameter	Description
UB_i	Upper bound delay for a packet P_i of F_i to traverse the NoC
MI_i	Maximum interval until the ability to inject a new packet of F_i, used in method RTB-HB
mI_i	Minimum permitted interval between two consecutive packets of F_i, used in methods WCFC and RTB-LL
mBW_i	Worst-traffic minimum injectable bandwidth for F_i, used in method RTB-HB and is equal to: $mBW_i := L*FlitWidth/MI_i*Freq$
MBW_i	Maximum permitted bandwidth for F_i , used in methods WCFC and RTB-LL and is equal to: $MBW_i := L*FlitWidth/mI_i*Freq$
u_i^j	The time needed for P_i to go from the input buffer of SW_j (j>0) or from generating process in S_i (j=0) to the input buffer of SW_{j+1} ($0 \leq j \leq h_i$-1) or Dj ($j=h_i$)
U_i^j	The time needed for P_i to go from the output buffer of SW_j (j>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^j, as $U_i^j = u_i^{j+1}$, $j \geq 0$
$z_0(i,0)$	Number of flows contending with F_i at S_i
$z_c(i,j)$	Number of flows contending with F_i at SW_j at output channel c
$I(x)$	Index of the x-th flow contending with F_i at S_i *(or* SW_j*)*

More formally, when P_i is generated in S_i, we consider all intermediate buffers along its route as 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 e.g. 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, for example flows F_q 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 that the analysis is worst-case. The order in which contending flows obtain the arbitration is not important for the calculation of the delay of P_i. So P_q should make a one-hop forward progress. While P_q frees the buffers at hop j flit by flit, the flits from P_r will smoothly replace the free buffer spaces. Eventually P_i also goes through. Section 4.3 presents a simple example to visualize this. The parameter u_i^0 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_i). u_i^j 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 added up, plus the fixed overhead for packet creation and ejection.

$$UB_i = ts_1 + ts_2 + \sum_j u_i^j \quad \text{with } j = 0 \ldots h_i \qquad (1)$$

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:

$$MI_i = ts_1 + u_i^0 \qquad (2)$$

To be consistent with the notations from [10] and for convenience, we introduce the uppercase U_i^j 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, any already existing packet at that buffer has to leave, freeing the buffer. This existing packet could be a previous 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 $MAX_x(U_i^0, U^0_{I(x)})$, where $I(x)$ is the index of contending flows at the output channel, with $x = 1 \ldots z_0(i, 0)$. Also, all the 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:

$$u_i^0 = \text{MAX}\,(U_i^0, U^0_{I(x)}) + \sum_x U^0_{I(x),} \text{ with } x = 1 \ldots z_0(i, 0) \quad (3)$$

For the subsequent hops, u_i^j can be calculated similarly:

$$u_i^j = \text{MAX}\,(U_i^j, U^j_{I(x)}) + \sum_x U^j_{I(x)} \text{ with } x = 1 \ldots z_c(i,j),\ 1 \le j \le h_i \quad (4)$$

Please note that, if there is no contention for the flow, then the above equation reduces to $u_i^j = U_i^j$. This is again akin to a packet moving in a pipeline fashion in the network.

In order to calculate the U_i^j values, let us consider the packet from flow *Fi* 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 for this is given by $MAX_x\,(U_i^{j+1}, U^{j+1}_{I(x)})$. Please note the small difference from the u_i^j calculations that, in this case, the values of U_i^j at a switch (j) depends on the values at the next switch (j+1) on the path. The U_i^j values can be obtained using the following equations:

$$U_i^j = \text{MAX}_x\,(U_i^{j+1}, U^{j+1}_{I(x)}) + \sum_x U^{j+1}_{I(x)} \text{ with } x = 1 \ldots z_c(i,j+1),\ 0 \le j \le h_i - 1 \quad (5)$$

For the case of the last switch, from the output port, the packet can be ejected in L_i cycles, with one flit of the packet ejected each cycle. Thus,

$$U^{hi}_i = L_i,\ U^{h_{I(x)}}_{I(x)} = L_{I(x)}$$

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^j values, which can be solved by Eq. 5. Please note that we assume the destination have 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. The same set of formulae also apply to the case where $B_d < L$. It is intuitive that the queuing effects are similar for this case as that of $B_d = L$. The methods can be easily extended to support virtual channels and different message lengths, so in the examples throughout the paper, we have used L_i instead of L. But due to lack of space, we omit these extensions in this paper.

4.2 The Proposed Delay Model RTB-LL

We present a substantial improvement to the previously published method WCFC [10]. WCFC also calculates upper bound propagation delays and permitted injection intervals for flows in wormhole networks. It considers the arbitration contention that packets will face and, recursively, the delay incurred by other packets sharing some part of their route due to these blockings. With the same notation as above, WCFC proposes [10] the following formulae for calculating UB_i and mI_i:

$$UB_i = ts_1 + ts_2 + L_i + (h_i + 1)a + \sum_j u_i^j \quad \text{with } j = 0 \ldots h_i$$
$$mI_i = ts_1 + L_i + \sum_j u_i^j - h_i\, b \quad \text{with } j = 0 \ldots h_i$$

where

$$u_i^j = b + \sum_x U^j_{Ii(x)} \quad \text{with } x = 1 \ldots z_c(i, j)$$

In the WCFC method, the calculations are based on the assumption that each flow injects packets spaced by at least a minimum permitted interval. For applications that can support such an assumption, we present a method that gives significant improvement over the WCFC method, which we call RTB-LL. To improve upon WCFC, a new concept of *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 having delay parameters U^j_{i1} and U^j_{i2}, then it is possible to consider $\text{MAX}(U^j_{i1}, U^j_{i2})$ as their representative delay, instead of $U^j_{i1} + U^j_{i2}$, for calculating the parameters for F_i, and so the main contending flows (i_1 and i_2) are ignored in the calculations. When F_i overlaps with other flows, in the calculations for F_i the other contending flows should be ignored. By applying these optimizations, we have noticed a significant improvement in bound tightness in RTB-LL, as will be seen in next section and as Table 1 summarizes.

4.3 Delay Calculation Examples

To describe in detail different aspects of the analytical methods RTB-HB, RTB-LL and WCFC for upper bound delay and interval calculation, we apply them step-by-step to an example NoC (shown in Fig. 3) and then compare them. There are 4 message flows from S_1 to D_1, $S_{2,3}$ to D_3, $S_{2,3}$ to $D_{2,4}$ and S_4 to $D_{2,4}$. The NoC contains four switches. For the sake of simplicity, we consider $B_d = L$ in this example.

4.3.1 Method RTB-HB

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

$$UB_1 = ts_1 + ts_2 + \sum_j u_1^j \quad \text{with } j = 0 \ldots 3$$

To start, let us model the time u^0_1 needed to move from S_1 to 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_1 to go from the output buffer of the source (remember that source nodes are tagged with superscript 0) to the output buffer of SW_1, so $u^0_1 = U^0_1$. U^0_1 has to be calculated recursively based on the delays of the contending packets and of the packets ahead along the same route.

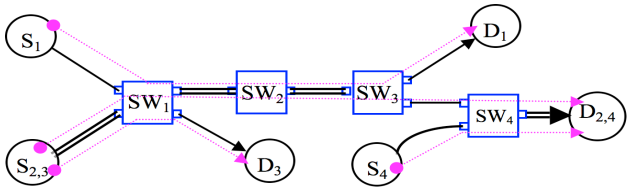

Figure 3. A simple example network

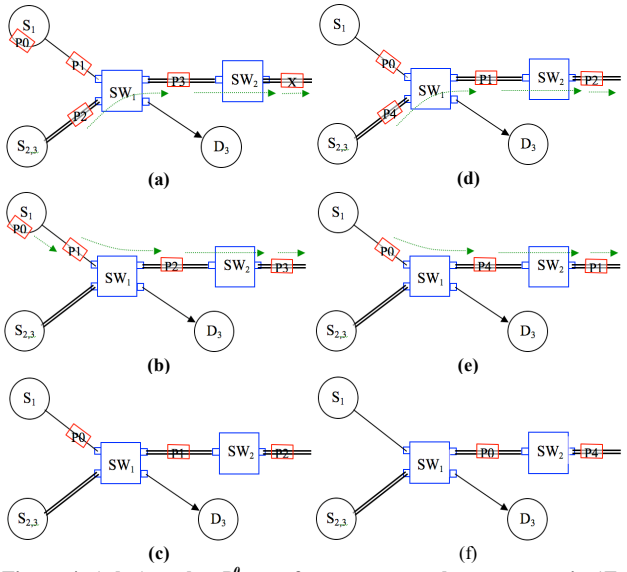

Figure 4: (a,b,c) packet P^0 goes from a process that generates it (F_1 in S_1) to the input buffer of SW_1. (d, e, f) packet P^0 goes from input buffer of SW_1 to input buffer of SW_2

We observe that two factors contribute to its calculation: 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 packet P^3 in the output buffer of SW_1. P^3 could belong to either F_1 or F_2, in which cases, respectively, either U^1_1 or U^1_2 models the time for such packet to move ahead, from the output buffer of SW_1 to the output buffer of SW_2. MAX(U^1_2, U^1_1) models the worst-case delay affecting our flow under study. During the time MAX(U^1_2, U^1_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. 4(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. 4(b)), which is U^1_2. In summary: $u^0_1 = U^0_1 = \text{MAX}(U^1_2, U^1_1) + U^1_2$

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. 4(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^1_1.

It is possible to use the equation $u^1_1 = U^0_1$ based on the equality described in previous section, but for clarity we always describe u^j_i based on U^j_i. As shown in Fig. 4(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 MAX(U^1_2, U^1_1), while P^1 moves on to the output buffer of SW_2, P^4 will replace it. P^4 itself then takes U^1_2 to move on, allowing P^0 to eventually get to the input buffer of SW_2. Thus: $u^1_1 = \text{MAX}(U^1_2, U^1_1) + U^1_2$

In a similar manner u^2_1 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 time that is needed to be ejected is equal to L_1. For uniformity of presentation we can write: $u^3_1 = U^3_1 = L_1$

Now, the target metric UB_1 can be calculated recursively, as a function of a set of U^j_i variables for the whole network starting from the last hop of each flow, where they assume a known value. The calculation of all intermediate values, e.g. U^1_1, U^1_2, U^2_1, U^2_2, U^3_1, and of the relevant metrics UB_i and MI_i, is shown in Fig. 5. Please note that intermediate values can be calculated once only and then stored, to speed up the recursion.

$u^0_1 = U^0_1$ $u^1_1 = \text{MAX}(U^1_2, U^1_1) + U^1_2$ $u^2_1 = \text{MAX}(U^2_2, U^2_1)$ $u^3_1 = U^3_1$ $U^2_1 = U^3_1 = L_1$ $U^2_2 = U^3_2 = U^4_4 + U^4_2 = L_4 + L_2$ $U^1_1 = \text{MAX}(U^2_1, U^2_2) = L_2 + L_4$ $U^1_2 = \text{MAX}(U^2_1, U^2_2) = L_2 + L_4$ $U^0_1 = \text{MAX}(U^1_2, U^1_1) + U^1_2 = 2L_2 + 2L_4$ $UB_1 = L_1 + 5L_2 + 5L_4, MI_1 = 2L_2 + 2L_4$	$u^0_2 = \text{MAX}(U^0_2, U^0_3) + U^0_3$ $u^1_2 = \text{MAX}(U^1_2, U^1_1) + U^1_1$ $u^2_2 = \text{MAX}(U^2_2, U^2_1)$ $u^3_2 = U^3_2$ $u^4_2 = U^4_4 + U^4_2$ $U^0_2 = \text{MAX}(U^1_2, U^1_1) + U^1_1 = 2L_2 + 2L_4$ $U^0_3 = U^1_3 = L_3$ $UB_2 = 6L_2 + L_3 + 6L_4$ $MI_2 = 2L_2 + L_3 + 2L_4$
$u^0_3 = \text{MAX}(U^0_2, U^0_3) + U^0_2$ $u^1_3 = U^1_3 = L_3$ $UB_3 = 4L_2 + L_3 + 4L_4$ $MI_3 = 4L_2 + 4L_4$	$u^0_4 = U^0_4$ $u^4_4 = U^4_2 + U^4_4$ $U^0_4 = U^4_2 + U^4_4 = u^4_4$ $UB_4 = 2L_2 + 2L_4, MI_4 = L_2 + L_4$

Figure 5: The complete calculation of UB_i and MI_i for the example NoC of Fig. 3 in method RTB-HB

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. 5 shows, analyzing e.g. the flow F_2, in this case u^0_2 (the time to transfer of a packet of F_2 into the input buffer of the first switch) should include a delay MAX(U^0_2, U^0_3), which accounts for a packet of either F_2 or F_3 to move away from the input of SW_1 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^0_3 for this latter packet to also move on, and finally letting a packet from F_2 in. The calculation for u^0_3 is the same.

4.3.2 *Methods RTB-LL and WCFC*

Fig. 6 and Fig. 7 show the calculated UB_i and mI_i values for both WCFC and RTB-LL methods for the same example of Fig. 3. Since flows F_1 and F_2 are overlapping at SW_2, our proposed RTB-LL improves the bound tightness compared to WCFC. Considering for the sake of exemplification a NoC variant as in Fig. 8, with another contending flow at SW_2, then in RTB-LL the delay u^2_3 of F_3 at SW_2 can be modeled as $b + \text{MAX}(U^2_1, U^2_2)$ instead of the overly pessimistic value $b + U^2_1 + U^2_2$ calculated by WCFC.

Table 5: Network parameters for the study

Parameter	L (flits)	a (regs.)	b_1 (regs.)	b_2 (regs.)	b_3 (regs.)	ts_1 (cycles)	ts_2 (cycles)	$FlitWidth$ (bytes)	$Freq$ (MHz)
Value	4	1	1	2	0	0	0	4	400

$u^0_1 = 0$ $u^1_1 = b + U^1_2$ $u^2_1 = b + U^2_2$ $u^3_1 = b$ $U^1_2 = U^2_2 + U^2_1$ $U^2_2 = U^3_2 = U^4_2 + U^4_4 = L_2 + L_3$ $U^2_1 = U^3_1 = L_1$ $UB_1 = 4a+3b + L_1+2L_2+2L_4$ $mI_1 = 2L_1+2L_2+2L_4$	$u^0_2 = U^0_3$ $u^1_2 = b + U^1_1$ $u^2_2 = b + U^2_1$ $u^3_2 = b$ $u^4_2 = b + U^4_4$ $U^0_3 = U^1_3 = L_3$ $U^1_1 = U^2_1 + U^2_2$ $UB_2 = 5a+4b+2L_1+2L_2+L_3+2L_4$ $mI_2 = 2L_1+2L_2+ L_3+2L_4$
$u^0_3 = U^0_2$, $u^1_3 = b$ $U^0_2 = U^1_2 + U^1_1$ $UB_3 = 2a+b+2L_1+2L_2+L_3+2L_4$ $mI_3 = 2L_1+2L_2+L_3+2L_4$	$u^0_4 = 0$ $u^4_4 = b + U^4_2$ $UB_4 = 2a+b+L_2+L_4$ $mI_4 = L_2+L_4$

Figure 6: The complete calculation of UB_i and mI_i for the example NoC of Fig. 3 in method WCFC

$u^0_1 = 0$ $u^1_1 = b + U^1_2$ $u^2_1 = b$ $u^3_1 = b$ $U^1_2 = U^2_2 = U^3_2 = U^4_2 + U^4_4 = L_2 + L_4$ $U^2_1 = U^3_1 = L_1$ $UB_1 = 4a+3b + L_1 + L_2 + L_4$ $mI_1 = L_1+L_2+L_4$	$u^0_2 = U^0_3$ $u^1_2 = b + U^1_1$ $u^2_2 = b$ $u^3_2 = b$ $u^4_2 = b + U^4_4$ $U^0_3 = U^1_3 = L_3$ $U^1_1 = U^2_1$ $UB_2 = 5a+4b+L_1+L_2+L_3+L_4$ $mI_2 = L_1+L_2+L_3+L_4$
$u^0_3 = U^0_2$, $u^1_3 = b$ $U^0_2 = U^1_2 + U^1_1$ $UB_3 = 2a+b+L_1+L_2+L_3+L_4$ $mI_3 = L_1+L_2+L_3+L_4$	$u^0_4 = 0$ $u^4_4 = b + U^4_2$ $UB_4 = 2a+b+L_2+L_4$ $mI_4 = L_2+L_4$

Figure 7: The complete calculation of UB_i and mI_i for the example NoC of Fig. 3 in method RTB-LL

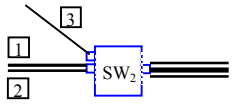

Figure 8: NoC variant where flow F_3 contends with two overlapping flows F_1 and F_2 at SW_2

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 five multimedia and RT applications, comparing RTB-HB and RTB-LL to the WCFC baseline, under the parameters of Table 5. In these applications, we assume that NoC topologies are predefined based on application communication needs, but without any feedback from the proposed algorithms to customize the network structure for better upper bound delay and interval time results. This is a possible extension for future work. In particular, for many applications it is possible to identify a small subset of few 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.

5.1 Case Study: A Multimedia Application

In this section, we compare the results of applying RTB-HB, RTB-LL and WCFC to D26-media, a real-time multimedia application with 67 communications flows, some of which critical, shown in Fig. 9. The application is mapped onto two NoC topologies, one with 5 "fat" (high-radix) switches (shown in Fig. 10) and the other one with 20 "thin" (low-radix) switches. Fig. 11 presents the results of the study in terms of latency, interval and bandwidth for the whole set of flows. Fig. 11(a, b) compare the worst-case NoC traversal latency UB_i. 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, but without any impact on the accuracy of the bounds, which are still under worst-case assumptions. RTB-HB naturally returns higher worst-case latency, due to the assumption that no hardware traffic injection regulation facilities are available. In fact, due to the different calculation approach, the bounds are on average still 30% lower than in 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 (no contention at all), the minimum theoretical latency to traverse the 5-switch NoC for flows spanning a single hop is 8 cycles ($a + b_1 + b_2 + b_3 + L$), while RTB-LL gives a minimum upper-bound of 17 cycles in worst-case contention. In general, for all methods, the delays calculated for the 20-switch topology 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 lower delay bounds.

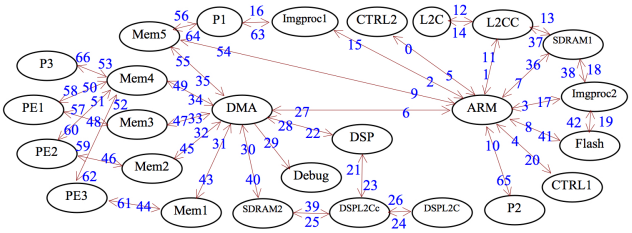

Figure 9: Communication graph for D26-media

Fig. 11(c, d) shows maximum and minimum injection intervals (MI_i and mI_i). Intuitively, if traversal delays are lower, new packets can be injected sooner, so MI_i (mI_i) plots resemble UB_i trends: flows with lower traversal latencies can be injected more frequently. Thus, the mI_i intervals are always shorter in RTB-LL and the MI_i intervals often shorter in RTB-HB when compared to mI_i in WCFC. These intervals can be directly translated into minimum and maximum injectable bandwidths (mBW_i, MBW_i) using the formulae in Table 3; results are shown in Fig. 11(e, f). Maximum injectable bandwidths (MBW_i) are on average 35% higher according to RTB-LL 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 ($Freq$* $FlitWidth$); 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 overall in comparable injectable bandwidths. In

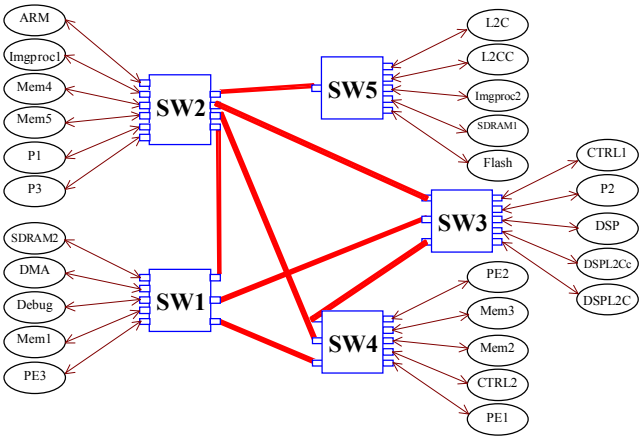

Figure 10 : D26-media application mapped on a 5-switch NoC

summary, NoCs with few hops exhibit clearly better 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 role in NoC performance.

5.2 Suitability to Critical Flows

Fig. 12 shows the average UB_i traversal delay and the average mBW_i injectable bandwidth for flows traversing x hops of the 5-switch NoC, considering the D26-media application and using RTB-HB. It can be seen that 1-hop flows exhibit reasonably low latencies and high bandwidths, suitable for critical traffic. Therefore, the proposed methodology has a clear applicability to industrial RT applications.

Table 6: Studied applications

	Cores	NoC switches	Clock frequency (MHz)	Traffic flows
D26-media	26	5	400	67
Pipeline	65	6	300	378
Bottleneck	35	6	300	128
36core-4	36	6	400	144
36core-6	36	7	400	216

5.3 Comparison for Different Applications

Fig. 13 shows the implementation results of the 3 different methods to the five applications listed in Table 6. 36core-4 and 36core-6 are different mainly because in the former application each core handles 4 communication flows to as many other cores, while in the latter each core handles 6 such flows. Fig. 13 proves that, for all the applications, RTB-HB and RTB-LL provide tighter average bounds than the reference WCFC method. RTB-LL is strictly tighter than WCFC, while RTB-HB, although it provides much tighter bounds on average, can return higher bounds for specific flows since it does not rely on regulated traffic assumptions of any kind.

6. COMPLEXITY OF THE METHODS

To estimate the time complexity of the proposed algorithms, implemented for the proposed methods, we calculate the maximum number of operations that are required. As Eq. 1, Eq. 3 and Eq. 5 show, the only operations are additions and comparisons (for the MAX operator); so we may consider one cycle to execute each of these operations.

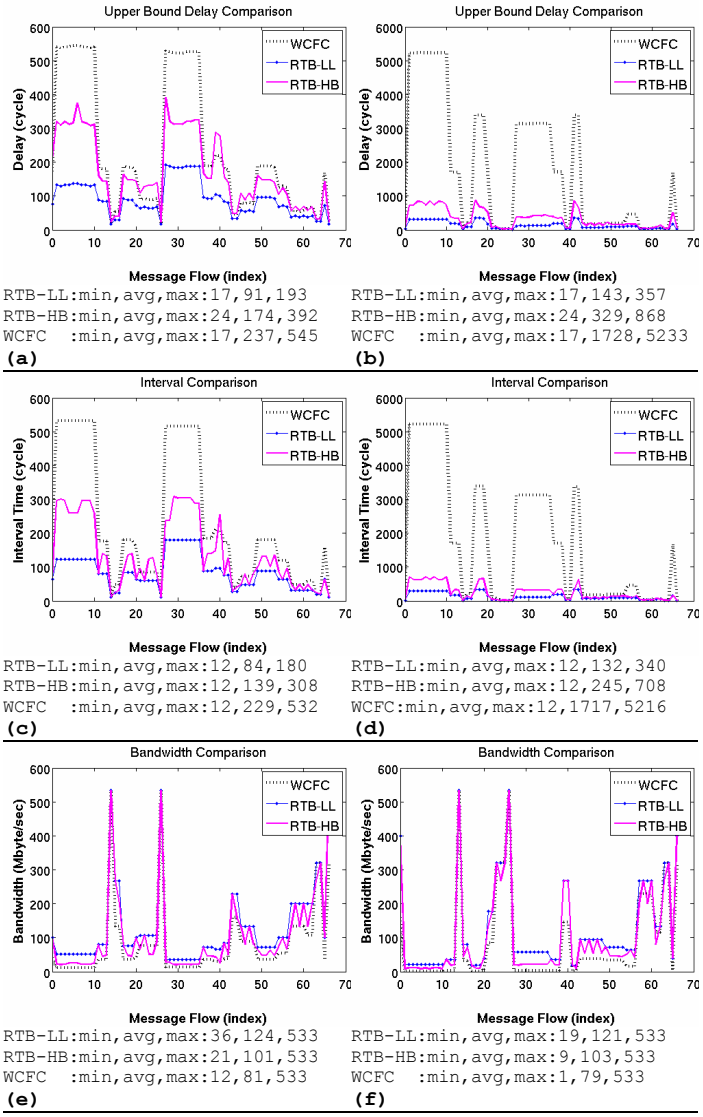

Figure 11: (a, c, e) UB_i, MI_i and mBW_i characterized with WCFC, RTB-LL and RTB-HB for D26-media mapped onto a 5-switch NoC. (b, d, f) The same metrics mapping on 20-switch NoC. Horizontal axes enumerate the communication flows.

We call h the maximum number of switches in 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^j_i parameter, we need (Eq. 5) at most k comparisons and k additions, thus $2k$ operations. The number of U^j_i parameters to be calculated is hk, so the maximum number of operations is $2hk^2$. u^j_i parameters (except for $j=0$) can be derived from the equality $U^j_i = u^{j+1}_i$, so we only need to calculate the case of u^0_i for all flows. In this case, one u^0_i (Eq. 3) needs $2k$ operations; for all u^0_i parameters we need $2k^2$ operations.

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

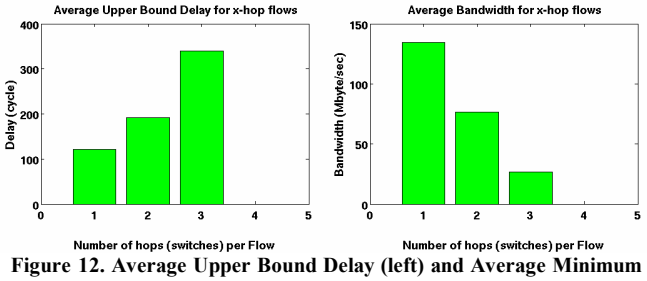

Figure 12. Average Upper Bound Delay (left) and Average Minimum Bandwidth (right) for x-hop flows in D26-media for RTB-HB

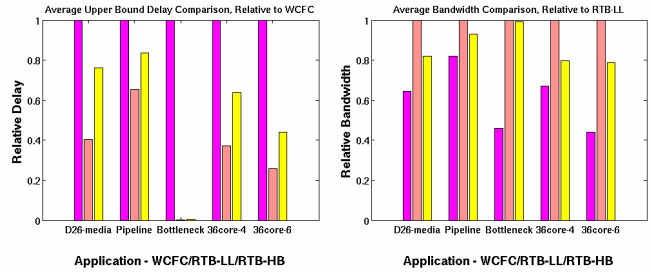

Figure 13. Comparisons: (left) average UB_i (WCFC as reference), (right) average MBW_i or mBW_i (RTB-LL as reference)

For RTB-LL, using the same approach, we can show that the complexity of the algorithm for calculating UB_i and mI_i is again $O(hk^2)$. Thus, both algorithms have quadratic time complexity. 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 WORKS

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 supplied in the NoC. One method is aimed at applications demanding 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 available. We 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 major next step is to use the results of this work as an input to an iterative procedure to synthesize optimized NoCs whereby the QoS demands of critical traffic flows are met.

8. REFERENCES

[1] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks, " Proceedings of the 38th Design Automation Conference, 2001.

[2] L. Benini and G. D. Micheli, "Networks on Chips: A New SoC Paradigm, " Computer, 35(1):70–78, 2002.

[3] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet-switched interconnections," in Proceedings of Design, Automation and Test in Europe Conference ands Exhibition(DATE '00), pp. 250–256, Paris, France, March 2000.

[4] Z. Guz, I. Walter, E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "Network Delays and Link Capacities in application-Specific Wormhole NoCs, " VLSI Design, Volume 2007, Article ID 90941

[5] J. Henkel, W. Wolf, and S. Chakradhar, "On-chip networks: a scalable, communication-centric embedded system design paradigm," in Proceedings of the 17th International Conference on VLSI Design (VLSID '04), vol. 17, pp. 845–851, Mumbai, India, January 2004.

[6] S. Furber and J. Bainbridge. "Future trends in SoC interconnect, " In VLSI Design, Automation and Test, pages 183– 186, 2005.

[7] Z. Shi and A. Burns , "Real-Time Communication Analysis for On-Chip Networks with Wormhole Switching, " Second ACM/IEEE International Symposium on Networks-on-Chip, 2008

[8] C. Paukovits, H. Kopetz, "Concepts of Switching in the Time-Triggered Network-on-Chip," Proceedings of the 2008 14th IEEE International Conference on Embedded and Real-Time Computing Systems, pp.120-12.

[9] K. Goossens, J. Dielissen, and A. Radulescu, "The Æthereal network on chip: Concepts, architectures, and implementations," IEEE Design and Test of Computers, 22(5):414–421,2005.

[10] S. Lee, "Real-time wormhole channels," J. Parallel Distrib. Comput. 63 (2003) 299–311

[11] D. Kandlur, K. Shin, D. Ferrari , "Real-Time Communication in Multihop Networks," IEEE Trans. on Para. and Distributed Systems, vol. 5, no. 10, Oct. 1994.

[12] M. Zhang, J. Shi, T. Zhang, Y. Hu, "Hard Real-time Communication over Multi-hop Switched Ethernet, " Int. Conf. on Networking, Architecture, and Storage, 2008

[13] S. Gopalakrishnan, S. Lui, and M. Caccamo, "Hard Real-Time Communication in Bus-Based Networks, " In Proc. 25th IEEE Int. Real-Time Systems Symp., 2004.

[14] A. Yiming, and T. Eisaka, "A Switched Ethernet Protocol for Hard Real-Time Embedded System Applications," In 19th Conf. on Advanced Information Networking & Applications, March 2005, pp. 41-44.

[15] K. Watson and J. Jasperneite, "Determining end-to-end delays using network calculus," In Proc. 5th IFAC Int. Conf. on Fieldbus Systems and Their Applications (IFAC-FET2003), July 7-8, 2003, pp. 255-260.

[16] J. Chen, Z. Wang, and Y. Sun, "Real-time capability analysis for switch industrial Ethernet traffic priority-based," In Proc. of Int. Conf. on Control Applications, Glasgow, UK, Sep. 2002, pp. 525-529.

[17] J. Jaspernite, P. Neumann, M. Theis, and K. Watson, "Deterministic Real-Time Communication with Switched Ethernet," In Proc. of WFCS'02, Vasteras, Sweden.

[18] S. Lee; K. C. Lee, and H. H. Kim, "Maximum communication delay of a real-time industrial switched Ethernet with multiple switching hubs," In 30th Conf. of IEEE Industrial Electronics Society, 2004.

[19] J. Loeser and H. Haertig, "Low-latency hard real-time communication over switched Ethernet," In Proc. of ECRTS 2004.

[20] J. Kiszka, B. Wagner, Y. Zhang, J. Broenink," RTnet – A Flexible Hard Real-Time Networking Framework," In: 10th IEEE International Conference on Emerging Technologies and Factory Automation, 2005.

[21] H. Kopetz, A. Damm, C. Koza, et al., "Distributed fault-tolerant real-time systems: the Mars approach," IEEE Micro, 1989, 9(1): 25-40.

[22] E. Bolotin et al., "QNoC: QoS architecture and design process for network on chip," Journal of Systems Architecture, vol. 50, no. 2–3, pp. 105–128, Feb. 2004.

[23] T. Bjerregaardand, J. Sparsoe, "Arouter architecturefor connection-oriented service guarantees in the MANGO clockless network-on-chip," in DATE, vol. 2, Mar. 2005, pp. 1226–1231.

[24] A. Bouhraoua and M. E. Elrabaa, "A high-throughput network-on-chip architecture for systems-on-chip interconnect," in Intl. Symposium on Soc, Nov. 2006.

[25] F. Felicijan and S. Furber, "An asynchronous on-chip network router with quality-of-service (QoS) support," in SOCC, Sep. 2004, pp. 274–277.

[26] N. Kavaldjiev et al., "A virtual channel network-on-chip for GT and BE traffic," in ISVLSI, vol. 00, Mar. 2006.

[27] A. Leroy et al., "Spatial division multiplexing: a novel approach for guaranteed throughput on NoCs," in CODES+ISSS, 2005, pp. 81–86.

[28] A. Mello, L. Tedesco, N. Calazans, and F. Moraes, "Evaluation of current QoS mechanisms in network on chip," in Intl. Symposium on Soc, 2006, pp. 115–118.

[29] M. Millberg, R. T. E. Nilsson, and A. Jantsch, "Guaranteed bandwidth using looped containers in temporally disjoint networks within the Nostrum network on chip," in DATE, 2004, pp. 890–895.

[30] F. Mondinelli, M. Borgatti, and Z. Vajna, "A 0.13 um 1Gb/s/channel store-and-forward network on-chip," in SOCC, Sep. 2004, pp. 141–142.

[31] R. Mullins, A. West, and S. Moore, "The design and implementation of a low-latency on-chip network," in ASP-DAC, 2006.

[32] A. Radulescu et al., "An efficient on-chip NI offering guaranteed services, shared-memory abstraction, and flexible network configuration," IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol. 24, no. 1, pp. 4–17, Jan. 2005.

[33] E. Rijpkema et al., "Trade offs in the design of a router with both guaranteed and best-effort services for network on chip," IEE Proc. Computers and Digital Techniques, vol. 150, no. 5, pp. 294–302, 2003.

[34] E. Salminen, A. Kulmala, T. Hamalainen, "Survey of Network-on-Chip Proposals," www.ocpip.org, March 2008.