Slot-Based Transmission Protocol for Real-Time NoCs – SBT-NoC

Borislav Nikolić
Institute of Computer and Network Engineering, TU Braunschweig, Germany
nikolic@ida.ing.tu-bs.de

Robin Hofmann
Institute of Computer and Network Engineering, TU Braunschweig, Germany
hofmann@ida.ing.tu-bs.de

Rolf Ernst
Institute of Computer and Network Engineering, TU Braunschweig, Germany
ernst@ida.ing.tu-bs.de

Abstract

Network on Chip (NoC) interconnects are some of the most challenging-to-analyse components of multiprocessor platforms. This is primarily due to the following two reasons: (i) NoCs contain numerous shared resources (e.g. routers, links), and (ii) the network traffic often concurrently traverses multiple of those resources. Consequently, complex contention scenarios among traffic flows might occur, some of the important implications being significant performance limitations, and difficulties when performing the real-time analysis.

In this work, we propose a slot-based transmission protocol for NoCs (called SBT-NoC), and an accompanying analysis method for deriving worst-case traffic latencies. The cornerstone of SBT-NoC is a contention-less slot-based transmission, arbitrated via a protocol running on a dedicated network medium. The main advantage of SBT-NoC is that, while not requiring any sophisticated hardware support (e.g. virtual channels, a flit-level arbitration), it makes NoCs amenable to real-time analysis and guarantees bounded low latencies of high-priority time-critical flows, which is a *sine qua non* for the inclusion of NoCs, and multiprocessors in general, in the real-time domain. The experimental evaluation, including both synthetic workloads and a use-case of an autonomous driving vehicle application, reveals that SBT-NoC offers a plethora of configuration opportunities, which makes it applicable to a wide range of diverse traffic workloads.

2012 ACM Subject Classification Computer systems organization → Real-time systems; Computer systems organization → Embedded systems

Keywords and phrases Real-Time Systems, Embedded Systems, Network-on-Chip, Protocols

Digital Object Identifier 10.4230/LIPIcs.ECRTS.2019.26

1 Introduction

Even though multiprocessors are now ubiquitous in almost all computing areas, they are still often considered as a new frontier technology in the real-time domain. Traditionally, in the real-time analysis of multiprocessors, the emphasis is on a single type of shared resources – processing elements (cores). However, due to the core proliferation trend in the multiprocessor area, contentions for other shared resources, such as an interconnect medium, become more apparent. This implies that, in order to perform the timing analysis of real-time applications deployed on multiprocessors, it is no longer sufficient to only take their computation requirements into account, but communication and memory traffic need to be considered as well. Therefore, the real-time analysis of network interconnects became a crucial prerequisite for the integration of multiprocessors in the real-time domain.

The Network-on-Chip architecture [2] (NoC) is among the prevalent choices for interconnects in contemporary multiprocessors, mostly due to its good performance and scalability potential [14]. However, the real-time analysis of NoCs is a challenging topic. This is primarily due the following two reasons: (i) NoCs are composed of numerous shared resources (e.g. routers, links), and (ii) network traffic flows often concurrently traverse multiple of those resources. As a consequence, infamous contention scenarios among traffic flows might occur (e.g. *head of line blocking* [5] and *backpressure* [27]), causing significant performance degradations as well as making the worst-case timing analysis difficult to perform.

Interestingly, the worst-case timing analysis of NoCs with single-channel ports and a fixed-priority packet-level arbitration (referred to as *basic NoCs* hereafter) is still an open research topic. This can be largely attributed to the aspects from the previous paragraph, implying that basic NoCs "out-of-the-box" (i.e. without any enhancements) are not amenable to real-time analysis, and hence do not represent satisfactory solutions for the real-time domain. This is discussed in more detail in Section 6 (Experiment 1).

How to make NoCs viable interconnect choices for the real-time domain? Intuitively, mechanisms are needed to ensure that a low priority traffic interferes with a high priority one only slightly (ideally not at all), thus effectively providing low latencies for high-priority time-critical flows, and big latencies for low-priority ones – which may not even have any timing requirements.

One promising solution to these problems is to use sophisticated hardware support and advanced router functionalities (e.g. virtual channels [5,6] and a flit-level arbitration [23]), and in that way make NoCs both more performant and more amenable to the real-time analysis (e.g. [11,17,26]). Some other notable solutions revolve around (i) organising NoC accesses in a time-division-multiplexing manner (e.g. [18]), (ii) explicitly reserving entire paths before transmissions, called the *virtual circuit* method (e.g. [4]), and (iii) utilising single-cycle multihop transfers to bypass intermediate routers (e.g. [8]). All these approaches suffer from one or more of the following limitations: complex and/or pessimistic timing analysis, expensive hardware requirements, inefficient use of resources, limited applicability to certain workload types, and finally, inability to meet real-time requirements (e.g. bounded low latencies of time-critical flows). This is discussed in more detail in Section 2.

Contribution. In this work, we propose a novel approach for making NoCs more applicable to the real-time domain. Specifically, we propose a slot-based transmission protocol, called SBT-NoC, and an accompanying analysis method for deriving worst-case traffic latencies. SBT-NoC ensures a contention-less slot-based transmission, arbitrated via a protocol running over a dedicated communication medium. By explicitly separating arbitration and data transmission domains, infamous contention scenarios are prevented by design, which contributes to a less complex and less pessimistic worst-case analysis, and perhaps even more importantly, to a better utilisation of network resources. At the same time, a slot-based transmission represents an efficient means to protect high priority flows from the lower-priority interference, thus guaranteeing their bounded low latencies, which is an essential real-time requirement. Finally, it is worth mentioning that besides a dedicated arbitration medium, SBT-NoC does not require any sophisticated router functionalities, nor any additional hardware support (e.g. virtual channels, flit-level arbitration), which implies that SBT-NoC can be easily adopted by next-generation commercial multiprocessors.

2 Related Work

All approaches for the integration of NoCs in the real-time domain can be broadly classified into two categories: contention-less and contention-aware. The former category supports uninterrupted transmissions by implementing a temporal and/or spatial allocation of NoC

resources to individual traffic flows. One popular strategy is to allocate resources in a time-division multiplexing (TDM) manner (e.g. [9, 10, 13, 18, 20]). The three limitations of TDM-based approaches are: (i) it might be challenging to find an efficient slots assignment configuration, (ii) significant buffer space might be required to store flits in traversed routers, and (iii) providing bounded low latencies for time-critical flows might require either resource over-provisioning (large slots for those flows), or workload type restrictions (e.g. strictly periodic traffic sources). The other approach from this category is a virtual circuit method. It achieves contention-less transmissions by reserving an entire path before a transmission (e.g. [4]). One downside is that a path establishing stage might be time consuming, and hence assuring bounded low latencies for time-critical flows might be non-trivial.

On the other hand, contention-aware approaches allow contentions among traffic flows, which are resolved via in-router arbitration mechanisms. One such method called *round-robin* is particularly popular among hardware manufacturers (e.g. [1, 25]). Its popularity comes from the fact that it offers a fair treatment to all traffic flows, thus promoting good performance. Several worst-case analysis methods for NoCs with the round-robin mechanism have been proposed, including the *Network Calculus* framework (e.g. [19]), the *Compositional Performance Analysis* framework (e.g. [24]) and the *Recursive Calculus* framework (e.g. [15]). However, one limitation of this mechanism is that it does not have any means to establish a notion of priorities, and it may be difficult to achieve low latencies of time-critical flows.

Another class of contention-aware approaches uses a priority-based flit-level arbitration via dedicated virtual channels. Currently, there exist several worst-case analysis methods, e.g. [11, 17, 26]. Although this type of NoCs fulfils almost all requirements of the real-time domain, its biggest limitations are substantial hardware requirements in a form of dedicated virtual channels to each traffic flow within each router along its path. Another relevant point is that dedicated virtual channels and the accompanying logic elevate power requirements, thus rendering such NoCs inapplicable in scenarios where low power consumption is required (e.g. embedded domain). Moreover, these approaches require routing mechanisms that guarantee a single continuous contention domain between any two interfering flows (e.g. dimension-ordered routing), which may be a limiting factor in some scenarios.

Recently, a novel type of interconnect architecture called SMART NoC [8] was introduced. This approach aims to avoid complex in-network interference patterns by utilising a router bypass mechanism which allows single-cycle multihop transmissions. However, one limitation of this approach is that it does not have a means to enforce prioritisation among traffic flows, and hence it may be challenging to achieve low latencies of time-critical flows.

Finally, it is worth mentioning that the arbitration protocol of SBT-NoC is, to an extent, inspired by CAN [7], Byteflight [3] and FlexRay [16] technologies, which are used for bus-based communication, predominantly in automotive in-vehicle networks. The common aspect of these approaches is that, during the arbitration phase, all traffic sources indicate their transmission requests, and at the end of the arbitration cycle one of them is granted the permission to transmit. The arbitration in SBT-NoC is performed in a similar way, with the following two fundamental distinctions: (i) instead of being interleaved with transmission phases, the arbitration is performed continuously over a dedicated arbitration medium, and (ii) instead of concluding the arbitration phase with a single transmission permission, multiple traffic sources are able to gain the transmission permission and subsequently concurrently transmit. The second aspect is of particular importance, because it allows to exploit the full potential of an underlying NoC architecture. More details on the SBT-NoC arbitration protocol (and the approach in general) are available in Section 5.

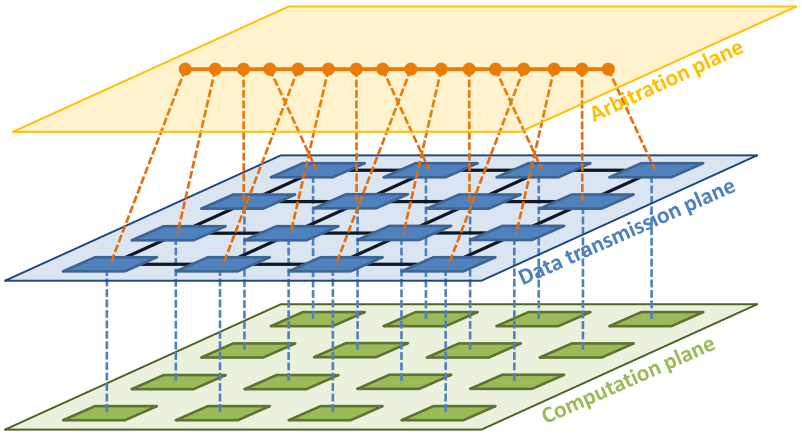


Figure 1 Illustrative example of assumed platform architecture.

3 System Model

3.1 Platform Architecture

The assumed platform architecture is a multiprocessor system $\mathcal{M}$, comprising: (i) a computation plane, (ii) a data transmission plane and (iii) an arbitration plane. One example of the assumed platform is illustrated in Figure 1.

A *computation plane* consists of $m \cdot n$ potentially heterogeneous processing elements (cores) $\{\mu_1, \mu_2, ..., \mu_{m \cdot n}\}$.

A *data transfer plane* consists of $m \cdot n$ interconnected mutually synchronised routers $\{\rho_1, \rho_2, ..., \rho_{m \cdot n}\}$, with identical physical characteristics. Routers are connected in a way to form a 2-D mesh topology, and each router ρ_i is, depending on its position, directly connected with 2, 3 or 4 neighbouring ones. Each two neighbouring routers ρ_i and ρ_j are connected via two unidirectional links of opposite directions, and it is assumed that all network links have the same capacity, where d_L stands for the transmission latency of one flit across one link. The connection between the computation and the data transfer plane is also organised in the form of two unidirectional links between each core μ_i and its corresponding router ρ_i. These links have the same physical characteristics as router-to-router links. As in the vast majority of NoCs, a data transfer is organised with the wormhole switching technique; prior to transmission, a packet is divided into small elements of fixed size called flits, which are successively injected into the NoC, where they travel to their destination in a pipelined manner. Moreover, routers have single-channel ports (no virtual channels necessary), where per-port buffers have the capacity to store at least two flits, so as to ensure a pipelined transmission. Buffer overflows are prevented with credit-based flow control. As a routing mechanism, any static technique can be applied (including source routing), with the only requirement that a flow should not put itself in a deadlock. The packet routing overhead is denoted by d_R, and only the first flit of the packet (header) experiences this delay. Finally, all routers have the same constant frequency ψ. Note that the link transfer latency d_L and the routing latency d_R are typically expressed as a number of cycles (e.g. in Intel's SCCC [12] $d_L = 1$ and $d_R = 3$).

An *arbitration plane*[1] consists of a bus system, to which all routers of the data transmission plane are connected. The bus frequency is assumed to be the same as the frequency of the routers. The term d_B denotes the worst-case latency of one router writing one bit on the bus, and all connected routers reading it. Similar to d_R and d_L, we also assume that d_B can be expressed as a number of cycles.

3.2 Workload

In this work, we take a communication-centric approach, and model the workload as a sporadic traffic flow-set Φ, which is a collection of z flows $\Phi = \{\phi_1, ..., \phi_z\}$. Each flow ϕ_i is characterised by: (i) a source core/router μ_i^{src}/ρ_i^{src}, (ii) a destination core/router μ_i^{dst}/ρ_i^{dst}, (iii) a path $\mathcal{L}_i$, expressed as a set of traversed network links (including those connecting μ_i^{src} and μ_i^{dst} to the NoC), (iv) a payload size σ_i, expressed as a number of bytes, (v) a minimum inter-arrival time T_i, (vi) a constrained deadline[2] $D_i \leq T_i$, and (vii) a unique priority P_i.

During each inter-arrival time, a flow may release at most one packet (consisting of a header flit, payload flit(s) and a tail flit). If it can be analytically proven that each packet of ϕ_i can complete its transfer before its deadline, even in the worst-case conditions, then ϕ_i is considered to be *schedulable*. If all flows of Φ are schedulable, then Φ itself is considered to be *schedulable*.

4 Problem Formulation

Given a platform $\mathcal{M}$ and workload Φ, propose a transmission protocol, and an accompanying worst-case timing analysis method, such that the schedulabtility of Φ on $\mathcal{M}$ can be evaluated. Additional requirements are as follows:

- The transmission protocol should exploit the full potential of the underlying platform by accommodating concurrent transmissions of multiple flows, whenever possible.
- The transmission protocol should ensure low worst-case traversal times (WCTT) of high priority time-critical flows, possibly at the expense of increased WCTTs of low priority ones.
- The timing analysis method should provide safe and tight upper-bounds on WCTTs, so as to avoid resource over-provisioning.

5 SBT-NoC

In this section, we introduce a slot-based transmission protocol for real-time NoCs, called *SBT-NoC*. As already mentioned, SBT-NoC provides contention-less packet transmissions. Before explaining how the contentions are prevented, let us first discuss inter-flow relations.

5.1 Inter-flow Relations

NoC routers and links are shared resources, and it often happens that two packets, belonging to different flows, simultaneously request to access the same resource. In such cases, a higher priority packet should be transmitted as soon as possible, while the lower priority one should

[1] An arbitration plane can be implemented in many different ways. In this work, we focus on one possible implementation strategy – a bus system. Investigating other options is a potential future work.
[2] Extending our approach to include arbitrary deadlines is a potential future work.

be deferred until the shared resource is available. When reasoning about the interference that packets of one flow may suffer, it is important to consider all potentially interfering flows, i.e. all flows that share at least one link[3] with the analysed one.

Let $\phi_i \in \Phi$ be the analysed flow. We classify all its interfering flows into two disjoint sets, namely $\mathcal{F}_i^H$ and $\mathcal{F}_i^L$. The former set is formally introduced with Definition 1.

▶ **Definition 1** (Set of directly interfering flows – $\mathcal{F}^H$)**.** *Consider $f_i \in \Phi$. Set $\mathcal{F}_i^H$ is a set of directly interfering flows of f_i,* **iff** *(if and only if) $\mathcal{F}_i^H$ contains all flows from Φ that have higher priorities than f_i, and share at least one link with it.*

Set $\mathcal{F}_i^H$ can be formally described as follows:

$$\forall f_i, f_j \in \Phi \mid P_j > P_i \wedge \mathcal{L}_j \cap \mathcal{L}_i \neq \emptyset \iff f_j \in \mathcal{F}_i^H$$

Analogously, the latter set of flows $\mathcal{F}_i^L$ is formally introduced with Definition 2.

▶ **Definition 2** (Set of directly interfered flows – $\mathcal{F}^L$)**.** *Consider $f_i \in \Phi$. Set $\mathcal{F}_i^L$ is a set of directly interfered flows of f_i,* **iff** *$\mathcal{F}_i^L$ contains all flows from Φ that have lower priorities than f_i, and share at least one link with it.*

Set $\mathcal{F}_i^L$ can be formally described as follows:

$$\forall f_i, f_j \in \Phi \mid P_j < P_i \wedge \mathcal{L}_j \cap \mathcal{L}_i \neq \emptyset \iff f_j \in \mathcal{F}_i^L$$

In order to achieve low latencies of packets of ϕ_i, it is essential to ensure that: (i) a transmission of any packet of ϕ_i can be delayed *only* by flows from $\mathcal{F}_i^H$, and (ii) a transmission of any packet of ϕ_i can delay transmissions of all flows from $\mathcal{F}_i^L$. These two aspects are the cornerstone of SBT-NoC.

5.2 Basic SBT-NoC

After defining flow relations, let us introduce a basic SBT-NoC variant. The advanced SBT-NoC variants have additional configuration possibilities, and they are described in Section 5.3.

5.2.1 Arbitration Mechanism (Basic SBT-NoC)

The arbitration process for SBT-NoC is a continuous activity comprising a potentially infinite sequence of *arbitration slots*. During one arbitration slot, all flows indicate their intentions to transmit packets. An arbitration slot concludes with transmission permissions granted to highest priority flows with pending requests, whose transmissions can be concurrently accommodated without causing any mutual in-network contentions. Upon the completion of one arbitration slot, and before the beginning of the next one, optionally, there might exist a short *pause* termed Δ of duration d_Δ. During the pause, all entities participating in the arbitration should make sure that the decisions derived during the previous arbitration slot have been implemented (e.g. flows that are granted a transmission permission should inject their packets), and that everything is ready for the next arbitration slot. An illustrative example of the basic SBT-NoC arbitration is illustrated in Figure 2.

[3] Due to the crossbar switching fabric inside routers, *router sharing* is only a necessary condition for interference between two flows, because two packets from different input ports can be transferred to different output ports simultaneously. Conversely, *link sharing* (and hence *port sharing*) is both a necessary and a sufficient condition for interference.

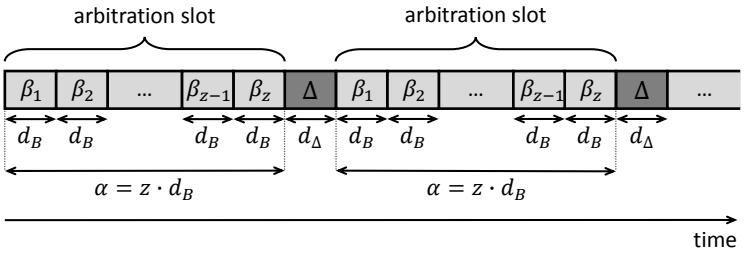


Figure 2 Illustrative example of basic SBT-NoC arbitration.

Each arbitration slot has a fixed duration termed α, and it contains a sequence of z dedicated *arbitration intervals* $\{\beta_1, \beta_2, ..., \beta_z\}$, one for each flow (recall that z denotes the number of flows in the flow-sets). All arbitration intervals have an equal duration d_B, and they are assigned to flows in order, with respect to their priorities, non-decreasingly, i.e. β_1 to the highest priority flow, and β_z to the lowest priority one. For the ease of exposition, let us assume that flow indexes are also assigned in the same manner, i.e. ϕ_1 and ϕ_z are the highest and the lowest priority flows, respectively.

During the arbitration interval β_i, a transmission of packets of ϕ_i is assessed. Similar to the CAN protocol, we assume a dominant "0" and a recessive "1" on the arbitration bus. At the beginning of β_i, the bus is in the recessive state. If ϕ_i has packets ready for transmission, its source router ρ_i^{src} will not attempt to change the state of the bus. The recessive "1" indicates a transmission request. In the opposite case (no packets of ϕ_i ready for transmission), ρ_i^{src} will, on behalf of ϕ_i, change the state of the bus to "0", and in that way indicate that it does not have packets ready for transmission.

However, ϕ_i (via ρ_i^{src}) is not the only flow that can manipulate the bus state during β_i. In fact, all flows from $\mathcal{F}_i^H$ can do that. The fact that these flows have higher priorities than ϕ_i implies that their respective arbitration intervals have already concluded, and that these flows are already either granted or denied transmission permissions. If flow $\phi_h \in \mathcal{F}_i^H$ was granted a transmission permission (manifested by a recessive "1" on the bus during β_h), then it will enforce, via ρ_h^{src}, a dominant "0" on the bus during β_i, and in that way deny transmission requests of ϕ_i. Conversely, if ϕ_h does not have packets ready for transmission, it will not manipulate the bus state during β_i. Additionally, if ϕ_i and ϕ_h originate from the same core, and ϕ_h already has a transmission permission, then during β_i the precedence will be given to ϕ_h to set the dominant state "0" on the bus via the common router, regardless of transmission intentions of ϕ_i.

In summary, from the perspective of flow ϕ_i, during arbitration interval β_i, only ϕ_i and flows from $\mathcal{F}_i^H$ are able to manipulate the bus state. Additionally, if the resulting bus state at the end of β_i is a recessive "1" (ϕ_i received a permission to transmit), ϕ_i will also manipulate the bus state during arbitration intervals dedicated to flows from $\mathcal{F}_i^L$ by setting a dominant state "0" (transmission denied).

5.2.2 Transmission Mechanism (Basic SBT-NoC)

All transmissions granted during one arbitration slot should start during a subsequent pause Δ, and should occur concurrently with the next arbitration slot. Granted transmissions must complete before the next pause. Large packets, which cannot complete an entire transfer during one arbitration slot, are transmitted in several stages. During the first transmission stage, the maximum number of flits that could complete the transfer before the next pause are selected, and those flits are transmitted as one *sub-packet*. A transmission of remaining

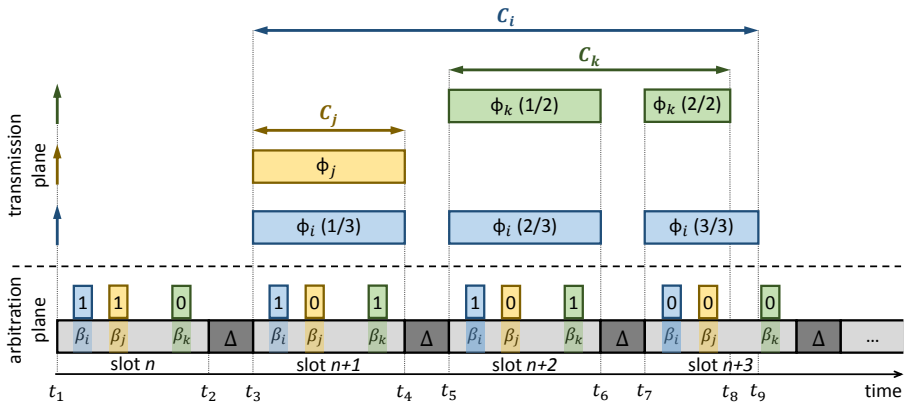


Figure 3 Illustrative example of basic SBT-NoC transmission.

flits is requested during the current arbitration slot. If the permission is granted, during the second transmission stage the maximum number of remaining flits that could complete the transfer before the next pause are selected, and those flit are transmitted as one sub-packet. This process is repeated until eventually the entire packet is transferred.

An illustrative example of a transmission process in SBT-NoC is presented in Figure 3. Three flows ϕ_i, ϕ_j and ϕ_k have packets ready for transmission at time instant t_1 (illustrated with three upward arrows). A packet of ϕ_i is the largest and it requires 3 transmission slots, a packet of ϕ_k requires 2 slots, while a packet of ϕ_j requires only one slot. Flow ϕ_i does not interfere with the remaining two flows, while $\phi_j \in \mathcal{F}_k^H$.

During arbitration slot n, all three flows try to indicate their transmission requests by setting a recessive "1" during their intervals β_i, β_j and β_k. In cases of ϕ_i and ϕ_j, the recessive value remains, i.e. $\beta_i = \beta_j = 1$, while in the case of ϕ_k, due to a potential contention, ϕ_j overrides the value with a dominant "0", i.e. $\beta_k = 0$. Consequently, during slot $n+1$, the first sub-packet of ϕ_i and a packet of ϕ_j are transmitted.

During slot $n+1$, flow ϕ_j does not participate because it does not have packets ready for transmission, and just sets a dominant "0" during β_j. Flows ϕ_i and ϕ_k have sub-packets ready for transmission, so they set recessive "1" during β_i and β_k. The bus remains in the recessive state during those intervals. Consequently, during slot $n+2$, the second sub-packet of ϕ_i and the first one of ϕ_k are being transmitted.

A similar scenario occurs during slot $n+2$, and therefore, during slot $n+3$, the third sub-packet of ϕ_i and the second sub-packet of ϕ_k are being transmitted.

Finally, during slot $n+3$, all three flows yield transmission opportunities to other flows by setting a dominant "0" during their respective intervals, because none of them have packets ready for transmission during slot $n+4$.

In Figure 3, we also illustrated the transmission latencies of the analysed flows, denoted by C_i, C_j and C_k. These latencies are formally introduced in the next section, and they represent the basic components for deriving the worst-case timing analysis method for SBT-NoC.

5.2.3 Packet Splitting and Transmission Latencies (Basic SBT-NoC)

In the previous section, it was mentioned that transmissions of large packets are performed in several stages, each including a transfer of one sub-packet. Before we discuss the process of packet splitting, let us introduce *NoC transmission latency.*

▶ **Definition 3** (NoC transmission latency). *Consider one packet of flow ϕ_i. NoC transmission latency of ϕ_i, termed C_i, is the time interval between the injection of a header flit from μ_i^{src} into the NoC, and the arrival of a tail flit at μ_i^{dst}, where a packet traversed its path without interference.*

NoC transmission latency is often also called the *isolation latency*. It can be computed by solving Equation 1.

$$C_i = \overbrace{(|\mathcal{L}_i| - 1) \cdot d_R}^{\text{header routing}} + \overbrace{|\mathcal{L}_i| \cdot d_L}^{\text{header traversal}} + \overbrace{\left(\left\lceil \frac{\sigma_i}{\sigma_F} \right\rceil + 1\right) \cdot d_L}^{\text{payload and tail traversal}} \tag{1}$$

Term $|\mathcal{L}_i|$ denotes the number of elements of $\mathcal{L}_i$, also called the *number of hops*, σ_F represents a size of a flit, in bytes, while d_L and d_R and σ_i were introduced in Section 3. NoC transmission latency is equal to the latency of a header flit reaching a destination (the first two terms of Equation 1), augmented by a traversal of payload flits and a tail flit across the last link, due to the pipelined transmission (the last term of Equation 1).

In SBT-NoC, each transmission needs to start during the pause, and complete before the end of the subsequent arbitration slot, which imposes a limit on the amount of payload that can be transferred within a single (sub-)packet. That limit, denoted by $\widehat{\sigma_i}$, can be computed by solving Equation 2.

$$\widehat{\sigma_i} = \left(\frac{\alpha - (|\mathcal{L}_i| - 1) \cdot d_R}{d_L} - |\mathcal{L}_i| - 1 \right) \cdot \sigma_F \tag{2}$$

Equation 2 was derived from Equation 1 by substituting σ_i with $\widehat{\sigma_i}$ and C_i with α. Recall that α denotes a slot duration.

Now we can obtain the minimum number of sub-packets ω_i, which are needed to transfer one packet of ϕ_i (Equation 3).

$$\omega_i = \left\lceil \frac{\sigma_i}{\widehat{\sigma_i}} \right\rceil \tag{3}$$

In SBT-NoC, large packets are transmitted with the minimum number of sub-packets in the following way: the first $\omega_i - 1$ sub-packets have the maximum payload size $\widehat{\sigma_i}$, and the last sub-packet has the payload size $\sigma_i - (\omega_i - 1) \cdot \widehat{\sigma_i}$, also denoted by $\sigma_i^{\dashv}$. Incidentally, these packet splitting and transmission rules also apply to flows ϕ_i, ϕ_j and ϕ_k from Figure 3, where a packet of ϕ_i is transmitted via 3 sub-packets, a packet of ϕ_j via a single sub-packet, and a packet of ϕ_k via 2 sub-packets.

In order to reason about transmission latencies in SBT-NoC, we need to slightly revise Definition 3, so as to account for large packets transmitted in several stages (Definition 4).

▶ **Definition 4** (SBT-NoC transmission latency). *Consider one packet of flow ϕ_i. SBT-NoC transmission latency of ϕ_i is the time interval between the injection of the header flit of the first sub-packet from μ_i^{src} into the NoC, and the arrival of the tail flit of the last sub-packet at μ_i^{dst}, where all transmission requests of ϕ_i during that interval were granted.*

Now we can express transmission latencies of flows in SBT-NoC. If an entire packet of a flow can be transmitted during α, then its transmission latency can be computed by solving Equation 1, i.e., a packet is transmitted in the same way as if it was a regular NoC. This is the case for flow ϕ_j from Figure 3. Conversely, if a packet of a flow is large and its transmission cannot finish during α, then its transmission latency can be computed by solving Equation 4. This is the case for flows ϕ_i and ϕ_k from Figure 3.

$$C_i = \overbrace{(\omega_i - 1) \cdot (\alpha + d_\Delta)}^{\text{transmission of first } \omega_i - 1 \text{ sub-packets}} + \overbrace{(|\mathcal{L}_i| - 1) \cdot d_R + |\mathcal{L}_i| \cdot d_L + \left(\left\lceil \frac{\sigma_i^{\dashv}}{\sigma_F} \right\rceil + 1 \right) \cdot d_L}^{\text{transmission of the last sub-packet}} \quad (4)$$

5.2.4 Worst-case Analysis (Basic SBT-NoC)

In this section, we provide a method to obtain upper bounds on WCTTs of flows in SBT-NoC. Several factors can contribute to the WCTT of the analysed flow, and in order to derive a safe upper bound, we need to cover all of them.

First, a packet release of ϕ_i may occur after its arbitration interval, in which case its router ρ_i^{src} must wait for the next arbitration slot to indicate the transmission request. In the worst case, a packet may arrive just after its arbitration interval and will have to wait for the remaining part of the slot, augmented by the pause, before it will be able to participate in the arbitration process. This delay is denoted by O_i.

$$O_i = \alpha - i \cdot d_B + d_\Delta \quad (5)$$

Moreover, before a packet can start traversing the NoC, a recessive "1" must be indicated during β_i in one of the subsequent arbitration slots. Once ϕ_i gains a transmission permission, during the next pause its packet is injected into the network. Assuming no higher priority interference, the worst-case delay of acquiring a transmission permission and preparing a packet for transmission is denoted by A_i (Equation 6).

$$A_i = \alpha + d_\Delta \quad (6)$$

Note, regardless of a packet size, only the delay of obtaining the first transmission permission needs to be considered. For large packets, remaining permissions are acquired concurrently with transmissions of preceding sub-packets (e.g. ϕ_i and ϕ_k in Figure 3).

Finally, a transmission of a packet of ϕ_i can be delayed by higher priority flows. This happens when $\phi_h \in \mathcal{F}_i^H$ also has a packet ready for transmission and participates in the same arbitration slot as ϕ_i. Consequently, ϕ_h prevents a transmission of ϕ_i by setting a dominant "0" during the β_i arbitration interval, and ϕ_i has to wait for the next arbitration slot to again attempt to gain a transmission permission.

The delay that higher priority flows inflict on ϕ_i is equal to the cumulative duration of full arbitration slots (augmented with respective pauses), in which a transmission request of ϕ_i was denied with a dominant "0" during β_i. Incidentally, each of these arbitration slots corresponds to one subsequent (sub-)packet transmission of higher priority flows. Therefore, the delay that one packet of a higher priority flow ϕ_h can cause to ϕ_i can be computed by multiplying the number of sub-packets of ϕ_h, with the full number of slots (augmented with respective pauses), i.e. $\omega_h \cdot (\alpha + d_\Delta)$.

After computing the interference that one packet of ϕ_h can cause to ϕ_i, now we need to compute the maximum number of packets of ϕ_h that can interfere with one packet of ϕ_i. An assumption that each two consecutive packets of ϕ_h interfering with ϕ_i must be at least T_h apart may be unsafe. This is due to the *indirectly interfering flows* (Definition 5).

▶ **Definition 5** (Indirectly interfering flow). *Consider three flows ϕ_g, ϕ_h and ϕ_i, where $\phi_g \in \mathcal{F}_h^H$ and $\phi_h \in \mathcal{F}_i^H$, but $\phi_g \notin \mathcal{F}_i^H$. Flow ϕ_g is an indirectly interfering flow of ϕ_i.*

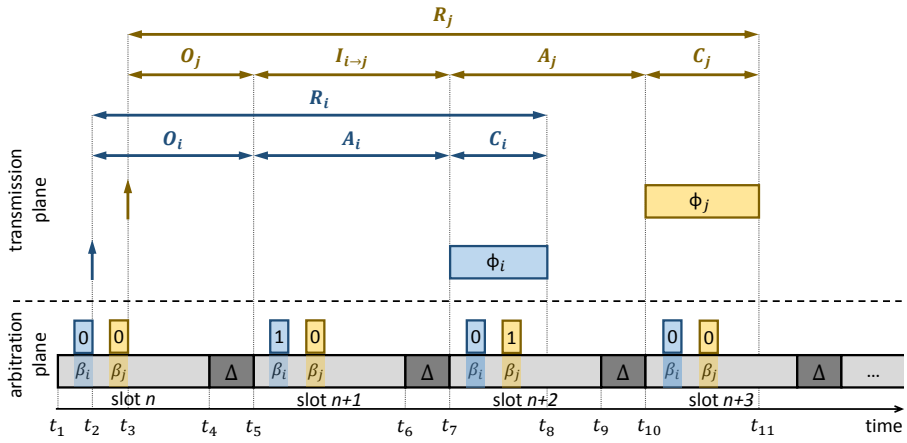


Figure 4 Illustrative example of transmissions of two flows.

Even though ϕ_g cannot cause direct interference to ϕ_i (no common parts of the path), ϕ_g can still cause indirect interference to ϕ_i in the following way: by interfering with ϕ_h, it may cause two consecutive packets of ϕ_h to interfere with ϕ_i within a time interval which is shorter than T_h.

In order to take indirect interference effects into account, from the perspective of ϕ_i, the first occurrence of ϕ_h should be assumed as late as possible, while remaining occurrences should be assumed as early as possible. Under the assumption that ϕ_h is schedulable, the effects of indirect interference on ϕ_i can be modelled with jitter $J_{h\to i}$ (Equation 7), which corresponds to the difference between the latest and the earliest time instants when a packet of ϕ_h can interfere with ϕ_i.

$$J_{h\to i} = \begin{cases} \overbrace{R_h - C_h}^{\text{latest occurrence}} - \overbrace{A_h - d_\Delta}^{\text{earliest occurrence}}, & \text{if } \exists \phi_g \mid \phi_g \in \mathcal{F}_h^H \wedge \phi_g \notin \mathcal{F}_i^H \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

Now, the worst-case interference that ϕ_h causes to one packet of ϕ_i, termed $I_{h\to i}$, can be obtained from Equation 8.

$$I_{h\to i} = \overbrace{\left\lceil \frac{R_i + J_{h\to i}}{T_h} \right\rceil}^{\text{maximum number of packets}} \cdot \overbrace{\omega_h \cdot (\alpha + d_\Delta)}^{\text{per-packet interference}} \tag{8}$$

In Equation 7 and Equation 8, the terms R_h and R_i denote the WCTTs of ϕ_i and ϕ_h, respectively.

Finally, the WCTT of ϕ_i can be obtained from Equation 9, which should be solved iteratively, until reaching a fixed converging point (if it exists).

$$R_i = O_i + A_i + C_i + \sum_{\forall \phi_h \in \mathcal{F}_i^H} I_{h\to i} \tag{9}$$

An illustrative example of transmissions of two flows ϕ_i and ϕ_j, and their WCTT components are shown in Figure 4.

5.3 Advanced SBT-NoC Variants

From the previous discussion it is noticeable that R_i is to a large extent affected by the duration of the arbitration slot α. This is because O_i and A_i directly depend on α, while the interference from higher priority flows is inflicted in multiples of $\alpha + d_\Delta$ intervals. Shorter α would lead to shorter O_i and A_i, but also to increased transmission times due to sequential transmissions of numerous sub-packets. On the other hand, longer α would increase O_i and A_i, but would lead to fewer sub-packets and more efficient transmissions.

Basic SBT-NoC operates under the assumption that an arbitration slot size α is fixed, and consists of z arbitration intervals, each dedicated to a single flow, i.e. $\alpha = z \cdot d_B$. However, based on the workload characteristics, in some scenarios it may be beneficial to either increase or decrease the duration of the arbitration slot. In this section, we present and discuss two advanced SBT-NoC variants, which allow to modify the size of the arbitration slot.

5.3.1 Advanced SBT-NoC with Slot Extension

One viable strategy to extend the duration of the arbitration slot is to introduce empty (non-used) arbitration intervals. These intervals should be added after the used, per-flow dedicated intervals. Therefore, assuming that γ intervals are added, the extended arbitration slot α_E consists of $z + \gamma$ intervals, of which only the first z are being used. The duration of the extended arbitration slot is $\alpha_E = (z + \gamma) \cdot d_B$.

Regardless of the number of added arbitration intervals, the worst-case analysis can be performed in the same way as for the basic SBT-NoC, with the only difference that instead of the basic arbitration slot α, the extended arbitration slot α_E should be used. This SBT-NoC variant is evaluated in Section 6 (Experiment 3).

Note that, an alternative approach for increasing the duration of the arbitration slot to a desired value α_E is to reduce the bus frequency and in that way increase the bus read/write latency to $d_B^* > d_B$. Such an approach can in fact be perceived as the basic SBT-NoC, because in this case the equality $\alpha_E = z \cdot d_B^*$ would remain valid.

5.3.2 Advanced SBT-NoC with Slot Reduction

One viable strategy to reduce the duration of the arbitration slot is to allow *arbitration interval sharing* among different lower-priority flows. This allows to achieve $\alpha_R < z \cdot d_B$, where α_R denotes the desired arbitration slot length. This SBT-NoC variant is explained in detail in the reminder of this section and evaluated in Section 6 (Experiment 2).

5.3.2.1 Arbitration Mechanism (Advanced SBT-NoC with Slot Reduction)

In this variant, each flow ϕ_i has two additional parameters. The first is ν_i, which defines how frequently ϕ_i participates in the arbitration. For example, $\nu_i = 1$ means that ϕ_i participates in every arbitration slot, $\nu_i = \frac{1}{2}$ every second, $\nu_i = \frac{1}{4}$ every fourth, etc. The closer ν_i is to 1, the more frequently ϕ_i is able to participate in the arbitration, and consequently, the smaller its WCTT is (analysed in Section 5.3.2.3). Therefore, we assume that ν values are assigned to flows according to their priorities (and indexes) non-increasingly. Note that the basic variant is a special case of this advanced variant where $\nu_i = 1, \forall \phi_i \in \Phi$.

The second parameter is $\theta_i \in [0, 1, ... \frac{1}{\nu_i})$, and together with ν_i, it defines exactly in which arbitration slots ϕ_i participates. The arbitration eligibility condition for ϕ_i is expressed with Equation 10, where n denotes the number of an arbitration slot.

$$n - \lfloor n \cdot \nu_i \rfloor \cdot \frac{1}{\nu_i} = \theta_i \tag{10}$$

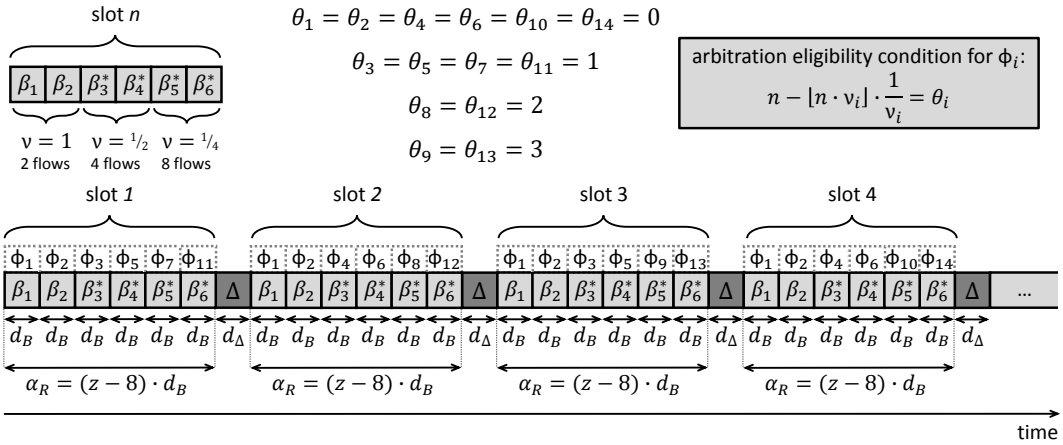


Figure 5 Illustrative arbitration example of advanced SBT-NoC with slot reduction.

In Figure 5 is illustrated an example of 14 flows $\phi_1 - \phi_{14}$. Flows ϕ_1 and ϕ_2 have $\nu_1 = \nu_2 = 1$. The next four flows $\phi_3 - \phi_6$ have $\nu_3 = \nu_4 = \nu_5 = \nu_6 = \frac{1}{2}$. Finally, the remaining eight flows $\phi_7 - \phi_{14}$ have $\nu_7 = ... = \nu_{14} = \frac{1}{4}$. Moreover, θ values are also illustrated in Figure 5. Flows participate only in slots for which their arbitration eligibility condition is fulfilled. For example, ϕ_1 and ϕ_2 participate in all slots, ϕ_3 and ϕ_5 in odd ones, ϕ_4 and ϕ_6 in even ones, ϕ_7 and ϕ_{11} in slots {1, 5, 9, 13, ...}, ϕ_8 and ϕ_{12} in slots {2, 6, 10, 14, ...}, etc.

5.3.2.2 Packet Splitting, Transmission Mechanism and Transmission Latencies (Advanced SBT-NoC with Slot Reduction)

The packet splitting in this variant is similar to that of the basic one. For flow ϕ_i, the maximum sub-packet payload size $\widehat{\sigma_i}$ and the number of sub-packets ω_i can be obtained from Equations 2-3 (Section 5.2.3), where α is replaced by α_R.

The transmission mechanism in this variant is also very similar to the one from the basic variant. After a flow receives a transmission permission during one arbitration slot, a transfer of one of its (sub-)packets is accommodated in the subsequent slot. One important difference from the basic variant is that, for flows with $\nu < 1$, two successive arbitration slots are separated from each other by $\frac{1}{\nu_i} - 1$ slots, and hence the corresponding transmission slots will be separated from each other as well. This implies that transmission latencies between the basic and this variant may differ, and assuming the latter case, the transmission latency of ϕ_i can be obtained from Equation 11. For flows with $\nu = 1$, Equation 11 becomes Equation 4.

$$C_i = \overbrace{(\omega_i - 1) \cdot (\alpha_R + d_\Delta) \cdot \frac{1}{\nu_i}}^{\text{transmission of first } \omega_i - 1 \text{ sub-packets}} \overbrace{+ (|\mathcal{L}_i| - 1) \cdot d_R + |\mathcal{L}_i| \cdot d_L + \left(\left\lceil \frac{\sigma_i^{\dashv}}{\sigma_F} \right\rceil + 1 \right) \cdot d_L}^{\text{transmission of the last sub-packet}} \quad (11)$$

5.3.2.3 Worst-case Analysis (Advanced SBT-NoC with Slot Reduction)

In this section, we provide a method to obtain upper-bounds on WCTTs of flows in advanced SBT-NoC with slot reduction. Several components constitute the WCTT, and in order to derive a safe upper bound, we need to cover all of them.

Recall from Section 5.2.4 that O_i corresponds to the time interval between a packet release and a beginning of the next slot when ϕ_i can participate in the arbitration. In the worst-case, a packet may arrive just after its arbitration interval, and has to wait until the

next slot in which it can participate in the arbitration. This is covered with Equation 12, where i stands for the index of the arbitration interval of ϕ_i in α_R. For flows with $\nu = 1$, Equation 12 becomes Equation 5.

$$O_i = \overbrace{\alpha_R - i \cdot d_B + d_\Delta}^{\text{until beginning of next slot}} + \overbrace{\left(\frac{1}{\nu_i} - 1\right) \cdot (\alpha_R + d_\Delta)}^{\text{until beginning of next } \phi_i\text{-eligible slot}} \tag{12}$$

The worst-case delay of acquiring a transmission permission and preparing a packet for transmission, denoted by A_i, requires a single slot (augmented by the pause). This term is the same as in basic SBT-NoC (Equation 6), where α is replaced by α_R.

The last component contributing to the WCTT of ϕ_i is the higher priority interference. Let us first discuss the effects of indirect interferences. Similar to the basic variant, the effects of the indirect interference from indirectly interfering flows to ϕ_i via ϕ_h can be modelled with jitter $J_{h\to i}$. The term $J_{h\to i}$ can be computed as before (Equation 7), because, relative to a packet release of ϕ_h, terms $R_h - C_h$ and $A_h - d_\Delta$ cover the latest and the earliest time instants, respectively, when a packet of ϕ_h may interfere with ϕ_i.

Now, let us obtain the interference that one packet of higher-priority flow ϕ_h can cause to ϕ_i. We have to analyse several cases:

Case 1 ($\nu_h = \nu_i = 1$): In this scenario, ϕ_i can suffer interference only from flows with $\nu = 1$. From the perspective of ϕ_i, the system behaves in the same way as the basic SBT-NoC, and the maximum interference caused by ϕ_h to a packet of ϕ_i can be obtained from Equation 8, where α is replaced by α_R.

Case 2 ($\nu_h = 1 \ \wedge \ \nu_i < 1 \ \wedge \ \nexists \phi_g \mid \phi_g \in \mathcal{F}_h^H \wedge \phi_g \notin \mathcal{F}_i^H$): In this scenario, ϕ_i does not suffer indirect interference via ϕ_h. Therefore, apart from flows in $\mathcal{F}_i^H$, there exist no other flows which can disrupt successive transmission requests of sub-packets of ϕ_h. This allows to compute the maximum interference from ϕ_h to a packet of ϕ_i by considering that sub-packets of ϕ_h traverse in consecutive slots (Equation 13).

$$I_{h\to i} = \overbrace{\left\lceil \frac{R_i}{T_h} \right\rceil}^{\text{maximum number of packets}} \cdot \overbrace{\lceil \omega_h \cdot \nu_i \rceil \cdot \frac{1}{\nu_i} \cdot (\alpha_R + d_\Delta)}^{\text{per-packet interference}} \tag{13}$$

In Equation 13, a multiplication by ν_i, a ceiling operator, and a division by ν_i are needed, because ϕ_i participates in the arbitration in every $\frac{1}{\nu_i}$th slot.

Case 3 ($\nu_h = \nu_i \wedge \theta_h \neq \theta_i$): In this scenario, ϕ_i and ϕ_h participate in different arbitration slots and hence ϕ_h cannot cause interference to ϕ_i (Equation 14).

$$I_{h\to i} = 0 \tag{14}$$

Case 4 (All other scenarios): In these scenarios, it is not safe to assume that sub-packets of ϕ_h are transmitted in consecutive slots, either due to the existence of indirectly interfering flows, or due to $\nu_h < 1$. Separated transmissions of successive sub-packets of ϕ_h may cause more interference to ϕ_i, than what would otherwise be caused by their transmissions in consecutive slots. A safe assumption is that, as long as ϕ_h has sub-packets ready for transmission, it will participate in the same arbitration slots with ϕ_i, and after each of them, transmit a single sub-packet. By following this reasoning, an upper-bound on the interference from ϕ_h to a packet of ϕ_i can be obtained by solving Equation 15.

$$I^{\circ}_{h\to i} = \overbrace{\left\lceil \frac{R_i + J_{h\to i}}{T_h} \right\rceil}^{\text{maximum number of packets}} \cdot \overbrace{\omega_h \cdot \frac{1}{\nu_i} \cdot (\alpha_R + d_\Delta)}^{\text{per-packet interference}} \tag{15}$$

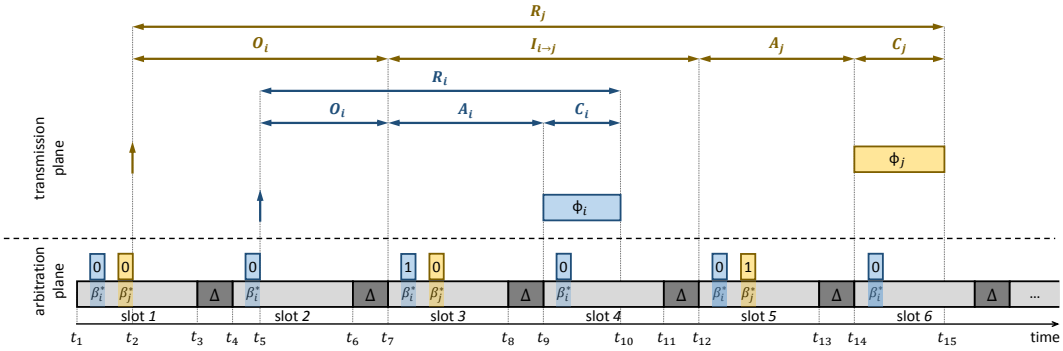


Figure 6 Illustrative example of transmissions of two flows.

Additionally, assuming that ϕ_h is schedulable, the distance between the first and the last sub-packets of its one packet is limited by R_h. Thus, by considering that sub-packets of ϕ_h may interfere with ϕ_i during the entire interval R_h, yet another upper bound on the interference can be derived (Equation 16).

$$I^{\bullet}_{h\to i} = \overbrace{\left\lceil \frac{R_i + J_{h\to i}}{T_h} \right\rceil}^{\text{maximum number of packets}} \cdot \overbrace{\left\lceil \left\lceil \frac{R_h}{\alpha_R + d_\Delta} \right\rceil \cdot \nu_i \right\rceil \cdot \frac{1}{\nu_i}(\alpha_R + d_\Delta)}^{\text{per-packet interference}} \tag{16}$$

Note that Equation 16 is derived using similar reasoning to that of Equation 13. The difference is that instead of ω_h slots, it is conservatively assumed that $\left\lceil \frac{R_h}{\alpha_R + d_\Delta} \right\rceil$ slots are needed to transfer one packet of ϕ_h.

Since both these bounds are safe, the minimum of them can be used (Equation 17).

$$I_{h\to i} = \min\{I^{\circ}_{h\to i}, I^{\bullet}_{h\to i}\} \tag{17}$$

Finally, the WCTT of ϕ_i can be computed by summing all components, as in the basic variant (Equation 9).

In Figure 6 are shown transmissions of two flows ϕ_i and ϕ_j for the advanced SBT-NoC with slot reduction, where $\phi_i \in \mathcal{F}^H_j$, $\nu_i = 1$, $\nu_j = \frac{1}{2}$ and $\theta_j = 1$. Moreover, the components contributing to the WCTTs of ϕ_i and ϕ_j are also illustrated.

6 Experimental Evaluation

In this section, we present the results of the experimental evaluation of SBT-NoC. The relevant NoC parameters which are common to all experiments are summarised in Table 1. The experiment-specific parameters are separately introduced in the context of each experiment. An asterisk sign denotes a randomly generated value, assuming a uniform distribution. Flow source and destination cores/routers are assigned randomly, with a restriction that they have to be different entities, i.e. $\forall \phi_i \in \Phi : \mu^{src}_i \neq \mu^{dst}_i \wedge \rho^{src}_i \neq \rho^{dst}_i$.

6.1 Experiment 1: SBT-NoC Run-time Performance Evaluation

In order to evaluate the performance of SBT-NoC, we have implemented a simulator of a multiprocessor platform. The simulator supports two different types of NoCs: (i) a regular NoC architecture with single-channel ports and 2-flit buffers, utilising the fixed-priority

Table 1 Analysis and simulation parameters.

NoC topology	2-D mesh
Routing mechanism	X-Y
Router frequency (ψ)	100MHz
Router latency (d_R) + link latency (d_L)	3 + 1 cycles
Bus writing/reading latency (d_B)	1 cycle
Pause between arbitration slots (d_Δ)	0 cycles
Link width = flit size (σ_F)	4B
Flow source core/router (μ_i^{src} / ρ_i^{src})	Random
Flow destination core/router (μ_i^{dst} / ρ_i^{dst})	Random
Flow deadline (D_i) = flow period (T_i)	[10ms - 50ms]*
Flow priority assignment policy	Rate monotonic

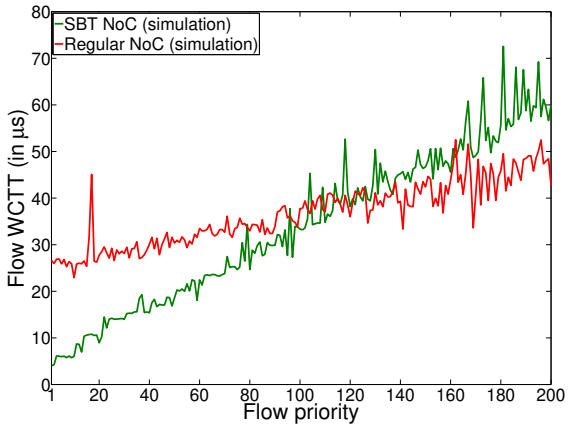


Figure 7 WCTTs of regular NoC (simulation) and basic SBT-NoC (simulation).

packet-level arbitration mechanism (in Section 1 referred to as the *basic NoC*), and (ii) the SBT-NoC with the basic arbitration variant (introduced in Section 5.2). In both cases, NoC parameters are identical to those from Table 1. The assumed NoC size is 4×4.

The workload characteristics are as follows. There exist 200 flows with unique priorities assigned with the rate-monotonic policy. Smaller numbers represent higher priorities. The flow periods and deadlines are as specified in Table 1. Regarding flow sizes, the following trend applies: higher priority flows have smaller sizes. The highest priority flow has the smallest payload size of 500B, the lowest priority flow has the biggest payload size of 10kB, and the sizes of intermediate flows are assigned equidistantly.

We run the simulations of the two aforementioned approaches, each for 100 seconds of simulated time. For each approach, we recorded the observed WCTTs of all 200 flows.

The results are illustrated in Figure 7. It is evident that basic NoCs do not have efficient mechanisms to leverage high priorities to achieve low WCTTs, which is one of the basic real-time requirements. In fact, when a lower priority flow starts its transmission through a shared router, it can block any later arriving higher priority flow for the duration of its entire traversal through that router. During a single transmission, a higher priority flow can experience blocking from multiple lower priority flows across different routers. Consequently, there exists an almost negligible difference in WCTTs of flows with highest and lowest priorities, despite the fact that higher priority ones have significantly smaller

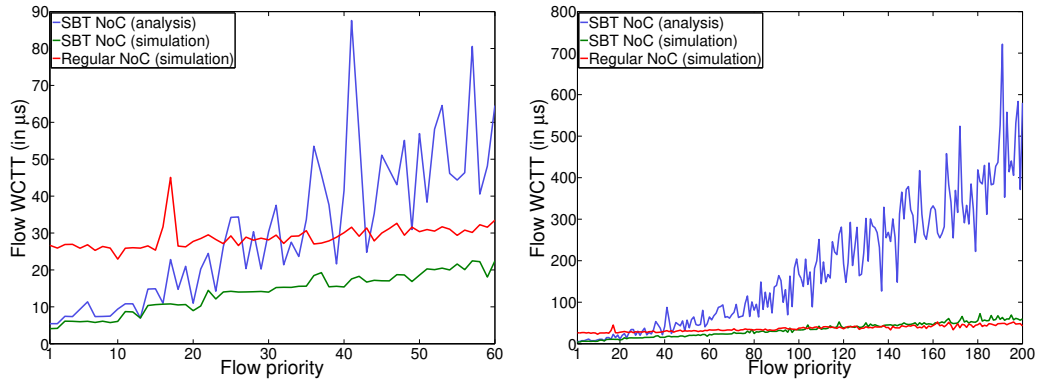


Figure 8 WCTTs of regular NoC (simulation) and basic SBT-NoC (simulation & analysis).

sizes. This result coincides with the statement from Section 1 that basic NoCs without any enhancements do not represent satisfactory solutions for the real-time domain, and further motivates research activities in the area of real-time oriented NoCs.

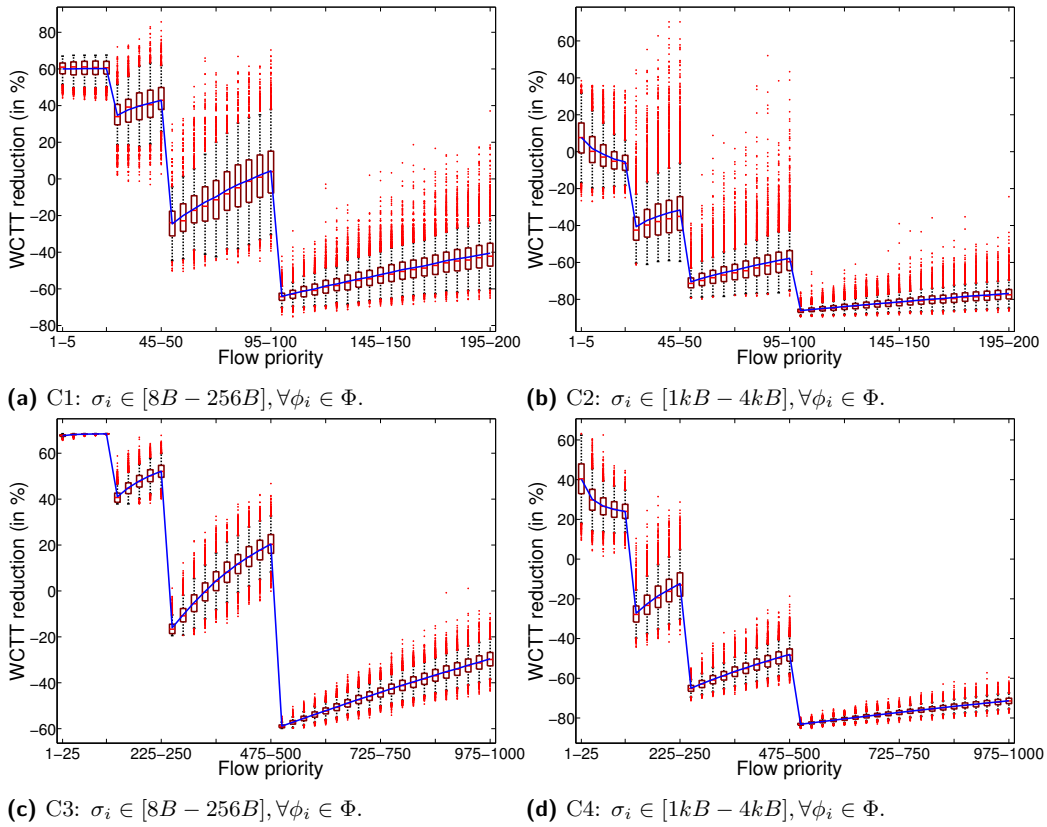
SBT-NoC demonstrates a significantly different behaviour. For high priority flows, SBT-NoC provides substantially smaller WCTTs than a regular NoC, and for the highest priority ones WCTTs are even several times smaller. These WCTT reductions are achieved at the expense of the lower priority traffic, and for low priority flows SBT-NoC provides bigger WCTTs than a regular NoC. The observed WCTTs suggest that SBT-NoC does not have negative effects on the run-time performance (no unusual and unexpected spikes in WCTTs), thus we conclude that SBT-NoC fulfils the first two objectives from Section 4.

Additionally, we derived the analytical WCTT upper-bounds by applying the method proposed in Section 5.2.4. A comparison of analytical and simulation results is illustrated in Figure 8a, where the focus is on 30% of flows with high priorities, (priorities 1 to 60). Interestingly, for the highest priority flows, even the SBT-NoC analytical method provides smaller WCTTs than the regular NoC simulations. The trends remain until priority 30, and imply that SBT-NoC, in conjunction with an analysis method, can produce low **bounded**[4] WCTTs of high priority flows. This means that the third objective from Section 4 is fulfilled.

For completeness, we have extended the observation interval to include all flows (Figure 8b). Unsurprisingly, as priorities decrease (i.e. bigger numbers on the X-axis), a difference between analytical and simulation results grows. This can be attributed to the fact that simulations are performed for only a limited time, which makes it unlikely that all worst-case scenarios were indeed captured. Moreover, the analysis method might contain a certain degree of pessimism, and reducing it is a potential future work activity.

Summary. SBT-NoC provides efficient means to achieve low WCTTs of high-priority flows at the expense of increased WCTTs of low priority ones. SBT-NoC does not have a negative effect on the run-time performance, which makes it a promising solution for soft real-time systems, where good performance might also be one of the requirements. Perhaps even more importantly, SBT-NoC provides low bounded WCTTs of high priority flows, which makes it a viable choice for hard real-time systems as well.

[4] Recall that for a regular NoC there exists no worst-case analysis method.



(a) C1: $\sigma_i \in [8B - 256B], \forall \phi_i \in \Phi$.

(b) C2: $\sigma_i \in [1kB - 4kB], \forall \phi_i \in \Phi$.

(c) C3: $\sigma_i \in [8B - 256B], \forall \phi_i \in \Phi$.

(d) C4: $\sigma_i \in [1kB - 4kB], \forall \phi_i \in \Phi$.

Figure 9 Relative ruction in analytically derived WCTTs of the advanced SBT-NoC with slot reduction – "A" against the basic SBT-NoC – "B".

6.2 Experiment 2: SBT-NoC Analytical Evaluation (Synthetic Workload)

In this experiment, we compare analytical results of the basic SBT-NoC variant – "B", and the advanced SBT-NoC variant with slot reduction – "A", for different workload configurations. "A" is configured as follows: 12.5% flows with the highest priorities have $\nu = 1$, the next 12.5% have $\nu = \frac{1}{2}$, the next 25% have $\nu = \frac{1}{4}$ and the remaining 50% have $\nu = \frac{1}{8}$. Phases (θ values) are derived from flow priorities in the following way: $\theta_i = P_i - \lfloor P_i \cdot \nu_i \rfloor \cdot \frac{1}{\nu_i}$.

The NoC size is extended to 8x8. Flow deadlines, periods, priorities, source and destination cores are assigned in the same way as in Experiment 1 (Table 1). Moreover, the evaluation includes 2 different setups for the workload size: (i) $z = 200$ flows and (ii) $z = 1000$ flows, as well as 2 different setups for the payload size: (i) $\sigma_i \in [8B - 256B]$ and (ii) $\sigma_i \in [1kB - 4kB]$. Assuming a certain payload size range, flow payloads are randomly generated values (a uniform distribution). The combinations of payload and workload size setups produce 4 distinctive evaluation configurations (C1-C4). For each of them, we generate 1000 flow-sets and analytically obtain WCTT upper-bounds with both evaluated SBT-NoC variants. We compare derived bounds by calculating the relative reduction in WCTTs achieved by "A" against "B". In cases where "B" outperforms "A", the WCTT reduction has negative values.

The results are illustrated in Figure 9. It is visible that a selection of the parameter ν has a significant impact on WCTTs. Therefore, let us analyse different sub-domains independently. For $\nu = 1$ (highest priority flows), in all configurations except C2, "A" derives

tighter bounds. This is expected, because "A" utilises shorter arbitration slots, and as discussed in Section 5.3.2.3, this reduces several WCTT components. As a number of flows increases, so do the improvements, because the difference in the durations of arbitration slots of "A" and "B" also grows. On the other hand, the increase in payload sizes leads to reduced improvements. This is because shorter slots yield more sub-packets, which limits the effects of a pipelined flit traversal and increases transmission overheads (more header and tail flits). These factors cause longer transmission latencies of analysed flows, and also inflate the interference they suffer from higher priority flows.

For flows with $\nu = \frac{1}{2}$, the observations regarding the effects of flow numbers and payload sizes on WCTT improvements are similar to those for flows with $\nu = 1$. In configurations C1 and C3, "A" is beneficial for flows with $\nu = \frac{1}{2}$. This implies that in scenarios with numerous flows and relatively small payload sizes, a strategy of assigning $\nu = \frac{1}{2}$ to flows with intermediate priorities may still lead to more favourable conditions for them, than what they could experience in "B". Another interesting observation from all evaluated configurations is that, within an observed sub-domain ($\nu = \frac{1}{2}$), improvements of "A" over "B" increase with decreasing priorities. This is because in "B" a flow can suffer interference from all higher priority flows sharing a part of the path with it, while in "A" that is not the case. In fact, if two flows in "A" have the same parameter ν, they can interfere only if they have the same θ. For $\nu = \frac{1}{4}$, "A" outperforms "B" only in C3, and in rare cases in C1.

Finally, for flows with $\nu = \frac{1}{8}$, in none of scenarios "A" produces smaller WCTTs than "B". This is expected, because the improvements of "A" over "B" for high priority flows were in fact achieved at the expense of increased latencies of the low priority traffic.

Summary. Variant "A" allows to even further reduce WCTTs of highest priority flows by decreasing a duration of an arbitration slot via an arbitration interval sharing among low priority flows. The improvements against variant "B" are the most significant for the highest priority flows ($\nu = 1$), while depending on the nature of the workload, significant WCTT reductions can also be achieved for flows with $\nu < 1$. Of course, WCTT reductions for high priority flows are achieved at the expense of increased WCTTs of the low priority traffic.

6.3 Experiment 3: SBT-NoC Analytical Evaluation (Use Case of Autonomous Driving Vehicle Application)

In this experiment, we perform the analytical evaluation of SBT-NoC. The workload is modeled after the use-case of the autonomous driving vehicle application [22]. The use-case consists of 33 functionalities producing 38 traffic flows in total. For a more detailed description of the use-case, a reader is advised to consult the work of Shi et al. [22].

The evaluation is performed in the following way. First, assuming the basic SBT-NoC variant – "B", the WCTT upper bounds of all flows are analytically obtained. Then, the same is performed for the two configurations of the advanced SBT-NoC with slot extension. The first one, referred to as "E10", has the slot length α_{E10} which is 10 times bigger than the slot length α_B of the basic variant, i.e. $\alpha_{E10} = 10 \cdot \alpha_B$. The second one, referred to as "E100", has the slot length α_{E100}, where $\alpha_{E100} = 10 \cdot \alpha_{E10} = 100 \cdot \alpha_B$. The incentive to evaluate "E10" and "E100" comes from the fact that there exist only 38 flows in this use-case, and the approach "B" is likely to lead to inefficient sequential transmissions of lots of small sub-packets, causing large WCTTs. Finally, in order to compare the performance of SBT-NoC with some other available approaches for real-time NoCs, we included priority preemptive NoCs with flit-level arbitration and per flow dedicated virtual channels in this evaluation [21], hereafter referred to as "PP". The WCTTs of all flows are obtained using the latest available analysis for such NoCs [17].

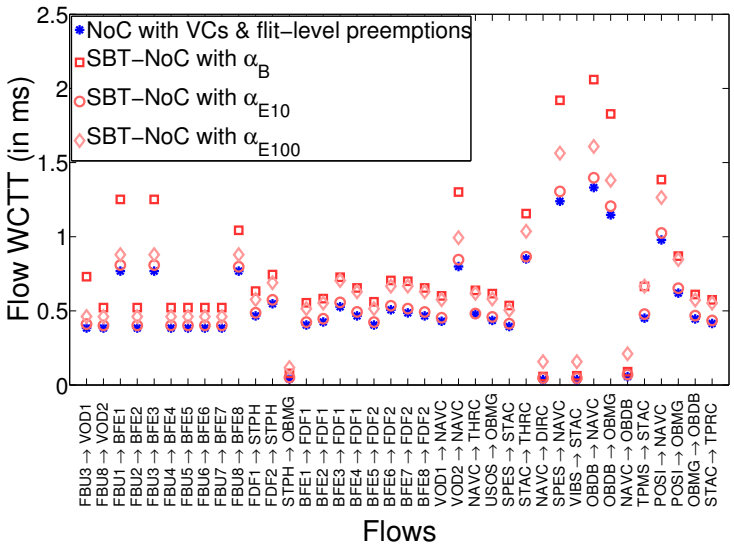


Figure 10 Analytically derived WCTTs of flows of the autonomous driving vehicle application [22].

The evaluation results are illustrated in Figure 10. As expected "B" displays the worst performance. Even though it offers short arbitration slots and short out-of-interval-arrival penalties (the first two terms in Equation 9), the transmissions are performed via numerous short sequentially transmitted sub-packets, thus causing large transmission latencies of entire packets (the third term in Equation 9). On the other hand, "E10" utilises 10 times longer arbitration slots, which may lead to 10 times longer out-of-interval-arrival penalties. However, "E10" performs transmissions with fewer sub-packets, which causes significantly shorter transmission latencies of entire packets. Consequently, "E10" produces $17.47 - 43.75\%$ smaller WCTTs than "B". The average WCTT reduction is 26.25%.

In the case of "E100", arbitration slots and out-of-interval-arrival penalties are 10 times larger than in "E10". On the other hand, "E100" performs more efficient (shorter) packet transmissions via fewer larger sub-packets, however, the achieved gains cannot compensate for the penalties arising from the increased slot size. Therefore, "E100" produces $8.72 - 233.28\%$ larger WCTTs than "E10". The average WCTT increase is 39.37%.

Finally, compared to the priority-preemptive approach "PP", the best performing SBT-NoC scheme "E10" produces $1.17 - 31.21\%$ larger WCTTs. The average WCTT increase is 7.33%. However, it is fair to point out that "PP" has more substantial hardware requirements than SBT-NoC schemes. Specifically, it operates under the assumption that there exist per-flow dedicated virtual channels in each of the traversed router ports, and that the arbitration is performed on a flit level. Implementing these features requires a sophisticated in-router logic and buffer space, which are typically not available in commercial NoCs. On the other hand, SBT-NoC requires a dedicated bus-based arbitration mechanism, which we believe is less costly and less demanding to implement.

Please note that "E10" may not be the most efficient SBT-NoC configuration for this use-case. It is only the best performing of the three SBT-NoC variants covered in this preliminary evaluation. In order to uncover the full potential of SBT-NoC, more detailed evaluations are necessary, and these activities are a potential future work.

Summary. A decision regarding arbitration slot sizes should be thoughtfully derived, because it significantly impacts the efficiency of SBT-NoC. The important aspects are platform architecture properties and a workload structure. Too short slots may lead to packet fragmentation into numerous sequentially transmitted sub-packets, which may cause longer WCTTs. On the other hand, too large slots may lead to significantly longer arbitration procedures, and longer out-of-interval-arrival penalties, both contributing to longer WCTTs. When compared with the priority-preemptive NoC scheme, the best of the three evaluated SBT-NoC approaches shows comparable results, and given the substantially higher hardware requirements associated with the former scheme, we believe that SBT-NoC is an attractive alternative, and a promising communication solution for real-time NoCs.

7 Conclusions and Future Work

In this work, we presented SBT-NoC – a slot-based transmission protocol for NoCs, and the accompanying worst-case analysis. SBT-NoC features contention-less slot-based transmissions, arbitrated via a protocol running on a dedicated network medium. SBT-NoC provides bounded low latencies of high-priority time-critical flows, at the expense of low priority ones. In this work, an SBT-NoC implementation via a dedicated bus medium was presented.

Moreover, this work includes a preliminary experimental evaluation of SBT-NoC. The initial results suggest that the proposed approach fulfils several important requirements of the real-time domain, and that it presents a viable choice for interconnect mediums in next-generation real-time-oriented multiprocessors. SBT-NoC offers a plethora of configuration options, of which only few have been evaluated in this work. Our future work plans include investigations of alternative technologies for an arbitration medium (e.g. a NoC interconnect), a practical implementation, and a design space exploration for deriving the most efficient SBT-NoC configurations for given platform and workload characteristics. This includes finding answers to the following questions: how to assign priorities, configure slot durations and assign parameters ν and θ?

References

1 Adapteva. *Epiphany Architecture*. URL: `www.adapteva.com/docs/epiphany_arch_ref.pdf`.

2 L. Benini and G. De Micheli. Networks on chips: a new SoC paradigm. *The Comp. J.*, 35(1):70–78, January 2002.

3 Josef Berwanger, Martin Peller, and Robert Griessbach. byteflight - A new protocol for safety-critical applications. In *FISITA World Automotive Congress*, 2000.

4 T. Bjerregaard and J. Sparso. Implementation of guaranteed services in the MANGO clockless network-on-chip. *IEE Proc. - Computers & Digital Techniques*, 153(4):217–229, July 2006.

5 W.J. Dally. Virtual-channel flow control. *Trans. Parall. & Distr. Syst.*, 3(2):194–205, March 1992.

6 W.J. Dally and C.L. Seitz. Deadlock-Free Message Routing in Multiprocessor Interconnection Networks. *Trans. Computers*, 1987.

7 Robert I. Davis, Alan Burns, Reinder J. Bril, and Johan J. Lukkien. Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised. *Real-Time Syst. J.*, 2007.

8 K. Duraisamy and P. P. Pande. Enabling High-Performance SMART NoC Architectures Using On-Chip Wireless Links. *Trans. Very Large Scale Integration Syst.*, 2017.

9 K. Goossens, J. Dielissen, and A. Radulescu. AEthereal network on chip: concepts, architectures, and implementations. *IEEE Design & Test Computers*, 2005.

10 Tim Harde, Matthias Freier, Georg von der Brüggen, and Jian-Jia Chen. Configurations and Optimizations of TDMA Schedules for Periodic Packet Communication on Networks on Chip. In *26th RTNS*, 2018.

11 Leandro Soares Indrusiak, Alan Burns, and Borislav Nikolić. Buffer-aware bounds to multi-point progressive blocking in priority-preemptive NoCs. In *21st DATE*, 2018.

12 Intel. *Single-Chip-Cloud Computer*, 2010. URL: `www.intel.com/content/dam/www/public/us/en/documents/technology-briefs/intel-labs-single-chip-cloud-article.pdf`.

13 E. Kasapaki, M. Schoeberl, R. B. Sørensen, C. Müller, K. Goossens, and J. Sparsø. Argo: A Real-Time Network-on-Chip Architecture With an Efficient GALS Implementation. *Trans. Very Large Scale Integration Syst.*, 2016.

14 N. K. Kavaldjiev and G. J. M. Smit. A Survey of Efficient On-Chip Communications for SoC. In *4th Symp. Emb. Syst.*, 2003.

15 M. Liu, M. Becker, M. Behnam, and T. Nolte. A tighter recursive calculus to compute the worst case traversal time of real-time traffic over NoCs. In *22nd ASPDAC*, 2017.

16 R. Makowitz and C. Temple. Flexray - A communication network for automotive control systems. In *Int. WS Factory Comm. Syst.*, 2006.

17 Borislav Nikolić, Sebastian Tobuschat, Leandro Soares Indrusiak, Rolf Ernst, and Alan Burns. Real-time analysis of priority-preemptive NoCs with arbitrary buffer sizes and router delays. *Real-Time Syst. J.*, 2018.

18 C. Paukovits and H. Kopetz. Concepts of Switching in the Time-Triggered Network-on-Chip. In *14th RTCSA*, pages 120–129, 2008.

19 Yue Qian, Zhonghai Lu, and Wenhua Dou. Analysis of worst-case delay bounds for best-effort communication in wormhole networks on chip. In *NOCS*, 2009.

20 Martin Schoeberl, Sahar Abbaspour, Benny Akesson, Neil Audsley, Raffaele Capasso, Jamie Garside, Kees Goossens, Sven Goossens, Scott Hansen, Reinhold Heckmann, Stefan Hepp, Benedikt Huber, Alexander Jordan, Evangelia Kasapaki, Jens Knoop, Yonghui Li, Daniel Prokesch, Wolfgang Puffitsch, Peter Puschner, Andrć Rocha, Cláudio Silva, Jens Sparsø, and Alessandro Tocchi. T-CREST: Time-predictable multi-core architecture for embedded systems. *J. Syst. Arch.*, 2015.

21 Zheng Shi and A. Burns. Real-Time Communication Analysis for On-Chip Networks with Wormhole Switching. In *NOCS*, 2008.

22 Zheng Shi, Alan Burns, and Leandro Soares Indrusiak. Schedulability Analysis for Real Time On-Chip Communication with Wormhole Switching. *Int. J. Emb. & Real-Time Comm. Syst.*, 2010.

23 Hyojeong Song, Boseob Kwon, and Hyunsoo Yoon. Throttle and preempt: a new flow control for real-time communications in wormhole networks. In *1997 Int. Conf. Parall. Processing*, August 1997.

24 S. Tobuschat and R. Ernst. Real-time communication analysis for Networks-on-Chip with backpressure. In *20th DATE*, 2017.

25 D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C. Miao, J. F. Brown III, and A. Agarwal. On-Chip Interconnection Architecture of the Tile Processor. *MICRO*, 2007.

26 Q. Xiong, F. Wu, Z. Lu, and C. Xie. Extending Real-Time Analysis for Wormhole NoCs. *Trans. Computers*, 66(9), 2017.

27 Qin Xiong, Zhonghai Lu, Fei Wu, and Changsheng Xie. Real-Time Analysis for Wormhole NoC: Revisited and Revised. In *26th ACM Great Lakes Symp. VLSI*, 2016.