Time-Division Multiplexing vs Network Calculus: A Comparison

Wolfgang Puffitsch, Rasmus Bo Sørensen, Martin Schoeberl
Department of Applied Mathematics and Computer Science
Technical University of Denmark
{wopu, rboso, masca}@dtu.dk

ABSTRACT

Networks-on-chip are increasingly common in modern multicore architectures. However, general-purpose networks-on-chip are not always well suited for real-time applications that require bandwidth and latency guarantees. Two approaches to provide real-time guarantees have emerged: time-division multiplexing, where traffic is scheduled according to a precalculated static schedule, and network calculus, a mathematical framework to reason about dynamically scheduled networks. This paper compares the two approaches to provide insight into their relative advantages and disadvantages. The results show that time-division multiplexing leads to better worst-case latencies, while network calculus supports higher bandwidths. Furthermore, time-division multiplexing leads to a simpler hardware implementation, while dynamically scheduled networks-on-chip allow the integration of best-effort traffic in the on-chip network in a more natural way.

CCS Concepts

•Networks → Network on chip; Network performance modeling; •Computer systems organization → Real-time systems;

Keywords

Network-on-Chip; Time-Division Multiplexing; Network Calculus

1. INTRODUCTION

Packet-switched networks-on-chip (NoCs) are becoming more and more common in modern multicore architectures. They enable a modular, tile-based design methodology and provide better scalability than traditional bus-based interconnects. Consequently, chips with dozens or hundreds of cores are now available commercially.

However, congestion can lead to excessive worst-case latencies in NoCs. Congestion is problematic in hard real-time systems, where the applications require that all deadlines are met. There are two common approaches to avoid congestion in on-chip networks for real-time systems: (1) static arbitration according to a predetermined schedule and (2) dynamic arbitration with routers that buffer packets and network interfaces that shape traffic to avoid buffers overflows. Both approaches rely on knowledge about the application's communication requirements in terms of latency and bandwidth.

This paper aims at shedding some light onto the advantages and disadvantages of these two approaches. We evaluate two concrete instances of each approach with regard to the latencies and bandwidths they can guarantee. Furthermore, we discuss properties of the two approaches that cannot be captured by these two simple measures.

NoCs based on time-division multiplexing (TDM) [22, 9, 10, 14] are examples of the first approach. In a TDM NoC, traffic is scheduled statically such that congestion in the network is avoided. A packet is delayed at the source until its slot in the schedule; afterwards, it travels through the NoC without being delayed by flow control, buffering, or interference from other traffic.

Network calculus [5, 6, 19, 20] is a mathematical framework that is used to bound buffering requirements and latencies in the second approach. Traffic is modeled through arrival and departure curves, which permit the computation of the maximum backlog and delay in network routers. By limiting the rates of packets at the sources of traffic, backlog and delay can be reduced and bounded.

TDM NoCs fit a *bottom-up*, *constructive* approach to time predictability: The predictability of a system emerges from the predictability of its building blocks. Each layer can rely on the predictable behavior of lower layers. A bottom-up approach can provide tight guarantees by leveraging the predictable behavior of all layers, but requires a time-predictable hardware platform.

In contrast, network calculus can also be used in a *top-down*, *analytical* approach to time predictability: The system is analyzed from the top down to establish guarantees on its behavior. Such an approach is useful when lower layers cannot be changed, for example when using commercial-off-the-shelf hardware platforms.

The research in our group is centered around time-predictable computer architectures. In the T-CREST project, we followed a bottom-up approach in the design of a time-predictable platform [24] and used a TDM NoC for the on-chip communication. This paper tries to answer the question whether our approach provides the benefits we expect or whether a network calculus approach may be a more promising direction for future research.

More generally, this paper aims at providing a better insight into the advantages and disadvantages of a TDM-based approach and a network calculus approach. In particular, the paper provides a quantitative evaluation of a network calculus approach and a TDM NoC by comparing two concrete platforms. We compare the worst-case latencies and bandwidths that can be guaranteed by the two approaches for a generic all-to-all communication graph and for communication graphs used in network benchmarking. Furthermore, we discuss benefits and drawbacks that are not captured by latency

RTNS 2015, November 04-06, 2015, Lille, France

DOI: http://dx.doi.org/10.1145/2834848.2834868

and bandwidth numbers, but may influence the choice between the two approaches nonetheless, for example hardware costs.

The paper is organized as follows. Section 2 presents the context of this paper and highlights related work. Section 3 describes the system model and details the assumptions made throughout the paper. Section 4 compares latencies and bandwidths for a generic all-to-all communication graph and for several application-specific communication graphs. Section 5 discusses benefits and drawbacks of the two approaches that cannot be captured by mere latency and bandwidth numbers. Section 6 concludes the paper.

2. BACKGROUND AND RELATED WORK

This section presents the context of the work described in this paper and presents related work. As the literature in the area of TDM based NoCs and network calculus is too extensive to be described comprehensively, we focus on the approaches that are of immediate importance for our work.

2.1 TDM NoCs

The use of TDM in communication protocols for hard real-time systems has long been advocated by Kopetz [18, 17]. While the approach proposed by Kopetz had off-chip networks in mind, the principle of using time to avoid dynamic traffic scheduling is also realized in TDM NoCs.

In TDM NoCs, shared resources such as network links are allocated to communication channels according to a predefined schedule. Due to the predefined schedule, bandwidths and latencies of channels are guaranteed, which makes TDM NoCs attractive for hard real-time applications. While TDM NoCs may waste bandwidth, they can avoid buffering in the NoC and can thus be implemented efficiently [25].

TDM NoCs require a common notion of time throughout the whole system. In synchronous hardware designs, this common notion of time can be established when the system is reset by resetting the counters that keep track of the current TDM slot. However, signal distribution issues make it difficult to design large systems that are fully synchronous.

In mesochronous hardware designs, different parts of the system operate at the same frequency but with unknown phase differences. Such a timing organization can solve the signal distribution issues, but leads to skew in the local notions of time. As both the clock signal and the reset signal may be skewed, the local notions of time can deviate by several cycles. The Argo NoC [14, 15] solves this problem by combining mesochronous network interfaces with asynchronous routers. The asynchronous routers compensate the skew in the notions of time at the network interfaces, such that a common notion of time is established without requiring a fully synchronous hardware design.

Examples for TDM NoCs are Nostrum [22], Æthereal [9], and aelite [10]. For the scope of this paper, we use the Argo NoC [14, 15] as reference TDM NoC. As pointed out above, Argo combines TDM with an asynchronous hardware implementation of the routers. Furthermore, communication in Argo is based in DMA transfers rather than individual read/write transactions, which enables an efficient hardware implementation of the network interfaces.

In a TDM NoC, the worst-case latency depends on the time between the slots of a channel and the time to traverse the routers and links. Let T_r be the length of the TDM schedule. For the simple case of one slot per channel per TDM round, a packet has to wait at most $T_r - 1$ cycles until its slot arrives. If there is more than one slot per channel in a TDM round, the worst-case latency depends on the maximum distance between slots in the schedule. However, $T_r - 1$ is a conservative upper bound. The worst-case latency over n hops in a TDM NoC is given by

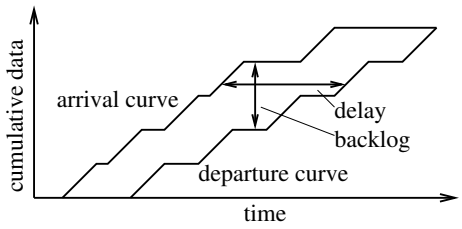

Figure 1: Arrival and Departure Curves

$$L_{\max}^{\mathrm{TDM}} = T_r - 1 + (n-1)p + nd + l_{\max} \quad (1)$$

In this equation, n denotes the number of hops of a flow, $l_{\max}$ denotes the maximum packet length including header data, and p and d denote the time to traverse a router or link, respectively. The first part of the equation, $T_r - 1$, models the time until a packet's slot arrives. The second part, $(n-1)p$, models the time to traverse the routers. As there are no collisions, the time to traverse a router is constant. The final part of the equation, $nd + l_{\max}$, models the link traversal times. As there are no collisions in a TDM NoC, packets can be forwarded immediately, and the packet size has to be counted only once.

2.2 Network Calculus

Network calculus is a set of mathematical rules that give insight into the behavior of packet-switched networks. In particular, network calculus is useful to reason about the delays that packets may experience and to compute the maximum backlog in routers. Network calculus was pioneered by Cruz [5, 6] and then later simplified and extended by Le Boudec [19, 20].

Network calculus models traffic through arrival and departure curves. An arrival curve models the traffic arriving at an incoming port of a router; a departure curve models the traffic departing at an outgoing port of a router. Network calculus provides results on how curves can be combined and which guarantees can be given for particular types of curves.

Figure 1 illustrates an arrival and a departure curve, with time on the x-axis and the cumulative amount of data on the y-axis. In this representation, the delay and backlog in a router can be found easily. The delay packets may experience in a router is given by the horizontal distance between the two curves and the backlog in a router is given by their vertical distance.

While network calculus was originally applied to off-chip networks, it has also been applied to on-chip networks [2, 23, 29]. These works investigated the applicability of network calculus to NoCs and assumed a rather generic NoC model. An apparent benefit of network calculus is that it can be applied to NoCs that were not designed with real-time communication in mind.

There are numerous variations of network calculus that are adapted and optimized for different types of networks. We base our comparison on the variant that applies to the NoC of the Kalray MPPA-256 processor [8, 7], which itself is based on the work by Zhang [31]. We use this approach for the comparison for two reasons. First, the approach models the NoC of a commercially available platform, such that we can be sure that all of its assumptions are realistic. Second, the NoC of the MPPA-256 was designed with network calculus in mind, such that network calculus is given a chance to present itself in a favorable light.

In the approach presented by Dupont de Dinechin et al. for the MPPA-256 [8, 7], data flows are characterized by two parameters,

ρ and σ. The parameter ρ represents the bandwidth of the flow and is defined as the number of words $N_{\max}$ that a flow may inject into the network over a sliding time window of length T_w: $\rho = N_{\max}/T_w$. The parameter ρ is therefore also referred to as *injection rate*. In the MPPA-256, a rate control unit ensures that flows cannot exceed their assigned injection rate. The parameter σ represents the "burstiness" of the flow and can, for the MPPA-256, be computed as $\sigma = \rho(1-\rho)T_w$.

The burstiness parameter models the fact that packets may be spread out evenly over the time window or arrive in a burst. For $\rho = 0$, the flow never sends data and cannot create bursts. For $\rho = 1$, the flow constantly sends data, such that the traffic is necessarily spread out evenly over the time window. For intermediate values, flows can exhibit bursty behavior. The worst case is $\rho = 0.5$, where packets could be spread out evenly by sending in every other slot, or arrive in a single burst that occupies half of the time window.

The approach by Dupont de Dinechin et al. uses network calculus to compute the maximum bandwidths the network can sustain. The input to the approach is a set of flows with predetermined routes. They then formulate a set of constraints on the injection rates ρ of all flows and subsequently maximize the bandwidth according to proportional fairness [16]. On the one hand, the constraints bound the utilization of links. On the other hand, the constraints apply constraints to limit the backlog and avoid buffer overflows in the routers. If all constraints are met, the bound for the end-to-end delay of packets is given by

$$L_{\max}^{\text{RC}} = \frac{\sigma}{\rho} + \frac{(n-1)l_{\max}}{\rho} + n(d + l_{\max}) \tag{2}$$

As in Equation 1, n denotes the number of hops of a flow, $l_{\max}$ denotes the maximum packet length including header data, and d denotes the link traversal time.

The first part of Equation 2, σ/ρ, models the delay introduced by the rate control. When substituting σ with its definition, σ/ρ becomes $(1-\rho)T_w$. A flow that is assigned the full bandwidth ($\rho = 1$) is never affected by the rate control, whereas a flow with minimal bandwidth ($\rho = 1/T_w$) may have to wait for $T_w - 1$ cycles before the rate controller forwards a packet.

The second part of Equation 2, $(n-1)l_{\max}/\rho$, models the delays caused by intermediate routers. In every router, packets of the flow in question have to traverse the router and may be delayed by packets from other flows. As the available bandwidth may have to be shared, flows with a lower bandwidth may experience more interference and hence a longer delay.

The final part of Equation 2, $n(d + l_{\max})$, models the link traversal times. In contrast to the formula for the TDM NoC, the packet length is counted for every hop.

2.3 Response Time Analysis

Network calculus is not the only approach to reason about latencies in dynamically scheduled NoCs. Response time analysis [13] is the classic schedulability analysis for periodic and sporadic real-time tasks. The work by Shi and Burns [26, 27] applies ideas from response time analysis to NoCs. Similar to network calculus, response time analysis evaluates the maximum interference of other flows (originally tasks) to the flow (task) under investigation.

Network traffic is shaped by allowing only periodic or sporadic insertion of packets into the NoC. An integral part of the approach by Shi and Burns is that flows are assigned priorities. Packets are scheduled on the links according to the priorities of the respective flows, and higher-priority packets may preempt lower-priority packets. In contrast, the scheduling of packets on links in the NoC of the MPPA-256 follows a non-preemptive round-robin policy.

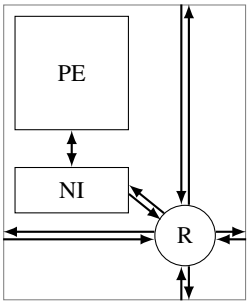

Figure 2: A tile, comprising a processing element PE, a network interface NI, and a router R.

Indrusiak [12] extends the approach by Shi and Burns and combines response time analysis for tasks with a response time analysis approach for NoC traffic. By doing so, he is able to provide end-to-end schedulability tests for chains of tasks that communicate over a NoC.

This paper limits itself to a comparison between TDM and network calculus, and does not consider approaches that are based on response time analysis. However, future work could extend the comparison presented in this paper to such approaches. Approaches based on response time analysis could potentially provide low latencies to high-priority flows, while still using the available bandwidth efficiently. However, further work is necessary for a sound comparison.

3. SYSTEM MODEL

Our system model comprises a set of *processing elements* that are connected through an on-chip network. Processing elements may contain one or more processors cores or hardware accelerators. The on-chip network comprises *network interfaces* and *routers*. The network interfaces translate between transactions issued by the processing elements and network packets. The routers steer the packets through the network according to the route encoded in the packet (i.e., we assume source routing). A processing element, a network interface, and a router are grouped into a *tile*, as shown in Figure 2. To distinguish between the NoCs we assume for the two approaches, we call the NoC for the TDM approach "TDM NoC", and the NoC for the network calculus approach "rate-controlled NoC".

With regard to the topology of the NoC, we consider a platform with 16 tiles, arranged as a 4×4 bitorus. Figure 3 illustrates the NoC topology. The figure shows the logical topology; a hardware layout would rearrange the tiles to shorten the maximum link length.

In the NoCs we assume, the minimum number of hops n in a flow is three. A packet has to first traverse the local link from its source processing element to the tile's router, then the link to the next router, and finally the local link from the destination tile's router into the processing element. The maximum number of hops in a 4×4 bitorus is six (the two local links plus four links between routers).

The routers have five input and output ports (north, south, east, west, and local). Packets can be routed in parallel as long as they are routed to different output ports. For example, a packet coming from east and routed to west does not collide with a packet coming from south and routed to north. The routers are non-blocking and therefore have no flow control. The packet flow is regulated at the source either by a static TDM schedule or by traffic shaping.

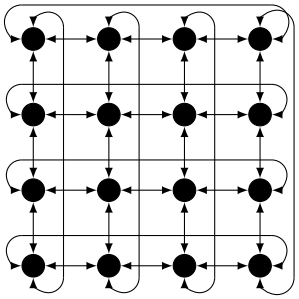

Figure 3: Topology of 4×4 bitorus NoC

We model the TDM NoC after the Argo NoC; we assume a router with a three-stage pipeline and a crossbar to forward packets from any input to any output. Within the router there is no buffering, flow control, or dynamic arbitration. When a packet arrives at the input port, it flows freely through the router in three cycles. The static (global) TDM schedule ensures that there is no collision on any output port.

For the rate-controlled NoC we assume a router with a three-stage pipeline and input buffers that can store whole packets. Each of the five output ports has four buffers associated to the input ports. The arbitration between those four packet sources for each output port is round robin. Except for this arbitration, there is no further flow control. The arrival rate of packets needs to be restricted so that no buffer overflow can occur. This router organization is similar to the routers in the NoC of the Kalray MPPA-256.

The traffic generated by the processing elements is characterized through *flows* (also called *channels*). Flows are unidirectional with one source and one destination. We model *push* communication, where the transmission of data is initiated by the source of the data. This model fits both the Argo NoC and the Kalray MPPA-256 NoC. *Pull* communication, where the sink of the data initiates the transmission of data, could be modeled by two separate flows, one for the request and one for the data. However, we limit the scope of this paper to push communication.

We assume that packets are routed along the shortest path. For the TDM NoC, we allow that different packets take different paths if there is more than one shortest path. For the rate-controlled NoC, we assume that all packets of a flow take the same path. In principle, it would be possible to split flows to allow packets to take alternative paths in the rate-controlled NoC. On the one hand, this could lead to higher bandwidths by using spare capacities along the alternative paths. On the other hand, the alternative paths would cause more interferences for other flows. Consequently, it is difficult to predict when flow splitting would have an overall positive effect. For simplicity, we follow the approach as described by Dupont de Dinechin et al. and assume that all packets of a flow take the same path.

We permit multiple packets of the same flow to be in flight at the same time. However, shortest-path routing ensures that packets always arrive in order. Neither the Argo NoC nor the NoC of the Kalray MPPA-256 are restricted to shortest path routing, but we make this assumption for simplicity. We do not assume flow control or acknowledgment that a packet has been received at the destination.

In accordance to the packet format of Argo, we assume that packets contain 1 header word and 2 payload words, i.e., $l_{\max} = 3$. In the scope of this paper, all packets have the same length and we always include the header word in bandwidth calculations. For example, a flow that has one slot in a TDM schedule has a bandwidth of $l_{\max}/T_r$.

For the router traversal time p we assume 2 cycles, and for the link traversal time d we assume 1 cycle. These values correspond to the Argo NoC and may be different in the NoC of the Kalray MPPA-256. To keep the comparison fair, we chose to assume the same values for both approaches.

For the network calculus approach, we also require parameters T_w and Q_{size}, which model the window size and the buffer sizes in the routers, respectively. In accordance with the MPPA-256, we assume that Q_{size} is 401 and that T_w may vary between 1 and 512. While T_w may in principle be different for each flow, we assume that T_w is a global value. The latter assumption is also made by the approach by Dupont de Dinechin et al. [8, 7].

4. BANDWIDTH AND LATENCY

Bandwidth and latency are important parameters of a NoC. While general-purpose NoCs typically emphasize bandwidth, real-time systems require guarantees on the worst-case latency. This section evaluates these two parameters for a TDM NoC and a network calculus approach for different communication graphs.

4.1 All-to-All Communication

As a starting point for the comparison between a TDM NoC and network calculus, we consider an all-to-all communication graph where all flows have equal bandwidth. What is the maximum end-to-end delay that packets may experience? What bandwidth can the two approaches guarantee?

There are $k \times (k-1)$ flows in an all-to-all communication graph with k nodes. For the 4×4 bitorus we consider for this comparison, this computes to 240 flows. We require that each of these flows sends one packet per TDM round T_r or time window T_w.

4.1.1 TDM Schedule

To generate TDM schedules, we use the meta-heuristic scheduler by Sørensen et al. [28]. The scheduler takes as input a communication graph with bandwidth requirements. While the mapping of the nodes of the communication graph to the target platform's processing elements is fixed, the scheduler decides the routes that flows take between their source and their destination. The scheduler always uses the shortest path for routing, but different packets of a flow may take different paths if there is more than one shortest path.

For an all-to-all communication graph in a 4×4 bitorus, the scheduler quickly finds a solution with a schedule length of $T_r = 19 \times l_{\max} = 57$ cycles. This is close to the optimal solution with a schedule length of $18 \times l_{\max} = 54$ cycles [4]. As each flow may send one packet per round in the generated schedule, the bandwidth of each flow corresponds to $\frac{1}{19}$ of the bandwidth of a single link.

In a 4×4 bitorus, the minimum number of hops is three and the maximum number of hops is six. For flows with the minimum number of hops, $n = 3$, Equation 1 yields a worst-case latency $L_{\max}^{\text{TDM}} = 57 - 1 + 2 \times 2 + 3 \times 1 + 3 = 66$ cycles. For flows with the maximum number of hops, $n = 6$, Equation 1 yields $L_{\max}^{\text{TDM}} = 57 - 1 + 5 \times 2 + 6 \times 1 + 3 = 75$ cycles. This indicates that the largest factor for the latency in a TDM NoC is the schedule length T_r, whereas the number of hops has a minor impact. It is also an indication that a TDM scheduled NoC may scale well to larger structures.

4.1.2 Network Calculus

The network calculus approach by Dupont de Dinechin et al. [8, 7] takes as input flows with a fixed route. For this comparison, we use the routes generated by the TDM scheduler. As in the

approach by Dupont de Dinechin et al., we formulate constraints on the rates of the flows and then maximize the rates according to a proportional fairness measure [16]. As weights for the flows we use the bandwidths in the TDM schedule.

We formulate the constraints for the ILOG OPL constraint solver [11]. As this constraint solver does not support floating-point decision variables, we use the number of packets that a flow may inject as decision variables. In addition to the rates, we let the constraint solver also choose a suitable window size T_w.

For the all-to-all graph in a 4×4 bitorus, the constraint solver finds a solution where all flows may inject one packet within a time window T_w of $15 \times l_{\max} = 45$ cycles. As each processing element has to send to 15 other processing elements, a bandwidth of $\frac{1}{15}$ per flow for all flows is an optimal solution. The bandwidth permitted by network calculus is 26% higher than the bandwidth of $\frac{1}{19}$ in the TDM schedule.

The minimum and maximum number of hops for a flow are the same for the TDM schedule and the network calculus. For flows with the minimum number of hops, $n = 3$, Equation 2 yields $L^{\mathrm{RC}}_{\max} = 42 + 2 \times 3 \times 15 + 3 \times 4 = 144$ cycles. For flows with the maximum number of hops, $n = 6$, Equation 2 yields $L^{\mathrm{RC}}_{\max} = 42 + 5 \times 3 \times 15 + 6 \times 4 = 291$ cycles. This indicates that the number of hops has a large impact on the latency as calculated by network calculus. Consequently, the usefulness of network calculus may be limited for large NoCs.

4.1.3 Discussion

We make the following observations for all-to-all communication:

- Network calculus achieves a bandwidth that is 26% higher than the bandwidth under TDM scheduling.
- TDM scheduling results in significantly lower latencies than network calculus. The advantage of TDM grows as the number of hops increases.

These observations indicate that network calculus can provide benefits if bandwidth is valued higher than worst-case latency. Furthermore, finding an application mapping that minimizes the number of hops is of high importance for network calculus, but of lesser importance in TDM NoCs.

4.2 Application-Specific Flows

As an all-to-all communication pattern is not necessarily realistic, we also use application-specific communication graphs for our comparison. In particular, we use communication graphs extracted from the MCSL (multi-constraint system-level) NoC traffic patterns [21]. These traffic patterns contain communication traces from eight different benchmark applications and exhibit a range of different communication patterns, such as one-to-many or many-to-many communication. From the communication traces, we extract communication graphs by calculating the maximum communication rate between each pair of processing elements. Table 1 shows a brief description of the benchmark applications along with the number of flows in the extracted communication graphs.

4.2.1 TDM Schedule

When converting the extracted communication graphs to the XML format understood by the TDM scheduler, the communication rates are normalized such that the flow with the smallest rate sends one packet per TDM period. In cases where the ratio between the largest and the smallest rate is high, this would lead to overly long TDM schedules. To overcome this issue, we compress the schedules by increasing the smallest communication rates according to the method described by Sørensen et al. [28]. To generate the results presented in this section, we compressed the schedules such that the actual rates of flows with the highest communication rate are degraded by no more than 5%. The schedules were then calculated by running the scheduler with the GRASP meta-heuristic for 200 seconds.

Benchmark	Description	Flows
FFT-1024_compl	Fast Fourier transformation	226
Fpppp	SPEC Fpppp	226
H264-1080_dec	H.264 decoder, high resolution	44
H264-720p_dec	H.264 decoder, low resolution	44
Robot	Robot control	53
RS-32_28_8_dec	Reed-Solomon decoder	85
RS-32_28_8_enc	Reed-Solomon encoder	28
Sparse	Sparse matrix solver	42

Table 1: Benchmark applications

		Worst-Case Latency			Bandwidth		
Benchmark	T_r	min	avg	max	min	avg	max
FFT-1024_compl	66	36	73.8	81	.0455	.0583	.2273
Fpppp	123	72	130.6	138	.0244	.0304	.1463
H264-1080_dec	93	69	98.7	108	.0323	.0543	.0645
H264-720p_dec	93	69	98.6	108	.0323	.0543	.0645
Robot	174	135	178.3	186	.0172	.0332	.0862
RS-32_28_8_dec	93	72	100.7	108	.0323	.0421	.0645
RS-32_28_8_enc	87	69	91.7	96	.0345	.0579	.0690
Sparse	30	27	36.7	45	.1000	.1333	.2000

Table 2: Results for TDM scheduling. Latencies are given in cycles, bandwidths are given as fractions of the capacity of a single link.

Table 2 presents some key measures for the generated schedules. The column labeled T_r shows the length of the TDM schedule in cycles. The columns under the label "Worst-Case Latency" characterize the worst-case latencies of the communication flows. The column "min" displays the worst-case latency of the flow with the minimum worst-case latency, while the column labeled "max" displays the worst-case latency of the flow with the maximum worst-case latency. The column "avg" shows the average worst-case latency over all flows. Similarly, the columns under the label "Bandwidth" characterize the bandwidths of the individual flows.

Worst-case latencies that are lower than T_r are due to several packets of a flow being scheduled within one round. Consequently, these flows do not have to wait the full $T_r - 1$ cycles until their slot arrives. In contrast, worst-case latencies that are longer than T_r are due to the traversal time within the NoC, which is computed as $(n-1)p + nd + l_{\max}$.

4.2.2 Network Calculus

We use the routes generated by the TDM scheduler as input to the network calculus approach. If packets of a flow may take different routes, we use the route of the last packet of the flow. In addition to the link capacity and buffer size constraints, we also use constraints for the minimum flow rates ρ, such that each flow has at least the bandwidth of the TDM schedule.

The TDM scheduler tries to minimize the schedule length and thus implicitly optimizes for latency. In contrast, the approach described by Dupont de Dinechin et al. [8, 7] aims at maximizing the bandwidths. For this comparison, we evaluate the network calculus approach with two different objective functions. On the one hand,

Benchmark	T_w	Worst-Case Latency			Bandwidth		
		min	avg	max	min	avg	max
FFT-1024_compl	57	74	226.8	357	.0526	.0708	.2632
Fpppp	120	151	273.1	735	.0250	.0707	.1500
H264-1080_dec	45	25	144.9	285	.0667	.2727	.9333
H264-720p_dec	45	25	145.6	285	.0667	.2621	.9333
Robot	138	67	182.5	561	.0217	.2112	.6522
RS-32_28_8_dec	45	30	176.0	285	.0667	.1325	.7333
RS-32_28_8_enc	42	37	129.7	222	.0714	.1582	.7143
Sparse	27	19	67.7	147	.1111	.2857	.8889

Table 3: Results for network calculus, optimizing average latency. Latencies are given in cycles, bandwidths are given as fractions of the capacity of a single link.

Benchmark	T_w	Worst-Case Latency			Bandwidth		
		min	avg	max	min	avg	max
FFT-1024_compl	228	192	381.2	519	.0526	.0708	.2895
Fpppp	120	135	293.8	735	.0250	.0705	.2000
H264-1080_dec	465	52	452.9	686	.0581	.2730	.9355
H264-720p_dec	372	46	389.7	635	.0484	.2652	.9354
Robot	486	193	464.1	975	.0185	.2073	.6420
RS-32_28_8_dec	93	42	218.9	338	.0645	.1332	.7419
RS-32_28_8_enc	126	65	201.7	300	.0714	.1624	.6905
Sparse	120	28	135.6	220	.1000	.2958	.9000

Table 4: Results for network calculus, optimizing bandwidths. Latencies are given in cycles, bandwidths are given as fractions of the capacity of a single link.

we let the constraint solver minimize the average latency. On the other hand, we let the constraint solver optimize the bandwidths according to proportional fairness.

As the constraint solver is not always able to find the optimal solution within a reasonable amount of time, we used an execution time limit of 100 hours for 10 parallel worker threads. While it is not guaranteed that the solutions are optimal, we are confident that they are close enough to the optimum to allow for a meaningful comparison.

Table 3 shows the results for network calculus when optimizing the average latency, while Table 4 shows the results when maximizing the bandwidths. The latencies calculated by the network calculus approach are not always integral; the minimum and maximum worst-case latencies in Tables 3 and 4 are rounded up to the next cycle.

The window lengths T_w are lower than the TDM schedule lengths T_r when optimizing for latency, but can be substantially longer when optimizing only bandwidths. The average and maximum worst-case latencies calculated by network calculus are higher than for the TDM schedules. However, the worst-case latencies for the flows with the smallest latencies can be lower than for the TDM schedules. As enforced by the constraints, the bandwidths calculated by the network calculus are no less than the bandwidth in the TDM schedules.

When comparing Tables 3 and 4, we observe that the effect of the objective function on the bandwidths if relatively minor. In contrast, latencies can be affected severely when optimizing bandwidth without taking latencies into account.

4.2.3 Discussion

The results clearly show that TDM scheduling generally leads to lower latencies. This is a result we expected because packets in a TDM NoC are only delayed to wait for their slot, whereas packets in a rate-controlled NoC may be delayed by the rate controller and suffer from additional interferences along their route.

When comparing the results for TDM scheduling and network calculus, an interesting phenomenon is that the network calculus approach results in substantially higher bandwidths. However, this is in part caused by the fact that the constraint solver is free to maximize bandwidths by using spare link capacities. There are fewer chances to find spare capacities for the benchmarks with a high number of flows (FFT-1024_compl and Fpppp). Therefore, the bandwidth advantage of network calculus is smaller for these two benchmarks.

Even when optimizing for latency, Equation 2 entails that larger bandwidths lead to smaller latencies for the network calculus approach. In contrast, the TDM scheduler only allocates the requested bandwidth and does not make use of spare link capacities. Consequently, the bandwidth figures in Table 2 are not directly comparable with the bandwidth figures in Tables 3 and 4. A future extension of the TDM scheduler could include such an optimization to make the results more comparable. However, we do not expect such an optimization to fully close the bandwidth gap between network calculus and TDM scheduling.

A second interesting phenomenon is that for some flows network calculus can guarantee lower latencies than the TDM scheduling. We believe that this is (at least partly) due to the fact that the TDM scheduler does not actively try to distribute the slots of flows with more than one slot over the schedule. For example, the latency of a flow with two slots could be almost halved by placing its slots appropriately in the schedule. A future extension of the TDM scheduler could make use of this observation to reduce latencies.

We are aware that the latency numbers computed by the network calculus approach we evaluated include some pessimism. We look forward to future research in this area to see in how far this pessimism can be removed. Future work could revisit the work presented in this section to follow the developments in the area of both TDM scheduling and network calculus.

5. DISCUSSION

Not all advantages and disadvantages of TDM NoCs and a network calculus approach can be captured by the latency and bandwidth numbers presented in the previous section. This section discusses the respective benefits and drawbacks to provide a more complete picture.

5.1 Hardware Costs

A thorough comparison of hardware costs between different NoCs would have to take into account that all components of the NoC (i.e., network interfaces, routers, and links) consume hardware resources. Unfortunately, published results are rarely sufficiently detailed to deduce the costs of all these components. Furthermore, different target technologies and optimization goals lead to results that are not comparable. However, we can estimate the hardware costs for the routers in a TDM NoC and a rate-controlled NoC.

The core functionality of routers in the two NoC approaches investigated in this paper is quite similar. Packets arrive at input ports and are multiplexed to output ports according to a route that is encoded in the packet. Therefore, we can expect this core functionality to require a similar amount of hardware resources.

However, routers in a rate-controlled NoC require buffers, while routers in a TDM NoC do not. While the available data in the literature is insufficient to do a precise comparison, we can still estimate the relative overhead inferred by the buffers. As buffers

are typically implemented with SRAM, we can relate the size of a TDM router design to the size of an SRAM bit.

Routers in the Argo NoC occupy a cell area of around 8000 μm^2 in 65 nm technology [14]. An SRAM bit in 65 nm technology uses around 0.5 µm [1, 30]. Consequently, we can estimate that around 16 kbit of buffers are equivalent to the area occupied by a router without buffers. In other words, we can expect a router that includes buffers for 500 32-bit words to occupy about twice the area of a router without any buffers. For comparison, a router in the NoC of the MPPA-256 contains 20 queues with 401 32-bit entries. In 65 nm technology, the SRAM cells to implement these queues would occupy about 16 times the area of a router in the Argo NoC.

This observation could be utilized in two ways. On the one hand, one could reduce hardware costs by replacing a rate-controlled NoC with a TDM NoC. On the other hand, one could increase the bandwidth of a TDM NoC by making it wider or replicating the NoC without exceeding the hardware costs of a rate-controlled NoC. However, future work is necessary to gain a better understanding of the trade-off between hardware costs and bandwidth when taking into account the costs for all components of the NoC.

Furthermore, we observe that the constraints presented by Dupont de Dinechin et al. [8, 7] relate the buffer sizes with the rate control time window. The larger the time window, the bigger the buffers have to be. Fine-grained rate control however requires a large time window. Consequently, the required rate control granularity impacts the required buffer sizes and the size of the hardware implementation.

5.2 Best-Effort and Mixed-Criticality Traffic

In a TDM NoC, all traffic is statically scheduled and flows through the network without buffering or flow control. It is therefore not straightforward to integrate best-effort traffic without substantial changes to the network hardware.

The Æthereal [9] NoC, which combines TDM scheduling with best effort traffic, has in fact separate TDM and best-effort routers placed in parallel. Support for best-effort traffic is implemented through buffers on the input ports of the routers together with link-level flow control. When a best-effort packet arrives at a router, it is stored in the input buffer. The best-effort packet is forwarded to the next router when no TDM packet contends for the same output port and there is room in the receiving buffer.

In a rate-controlled NoC without flow control, all flows need to be rate-constrained to guarantee the absence of buffer overflows. Therefore, best-effort flows cannot benefit from unused bandwidth in such a NoC. In addition to buffers, best-effort traffic requires flow control. However, we expect the hardware overhead for flow control to be relatively low compared to the buffers. Consequently, extending a rate-controlled NoC with flow control introduces little hardware overhead.

Mixing best-effort traffic with traffic that requires hard latency and bandwidth guarantees bears some similarity with mixed-criticality models that distinguish high-criticality traffic and low-criticality traffic. In such a model, high-criticality traffic takes priority over low-criticality traffic. As priorities can be modeled in network calculus [31], such a model of mixed criticality could be integrated into network calculus.

Network calculus can also be applicable to mixed-criticality models where the criticality of a flow depends on the tightness of its timing requirements. A variant of network calculus for such a model of mixed criticality is presented by Boyer et al. [3]. While this network calculus variant was developed for an off-chip network, similar ideas could be applied to on-chip networks.

5.3 Possible Optimizations

Similar to the approach by Dupont de Dinechin et al. [8, 7] we assume a single, global time window of length T_w for all flows. However, in contrast to a TDM NoC where the TDM period is a global property, there is no fundamental reason to have a single, global window for traffic shaping.

As pointed out in Section 5.1, the window size impacts the required buffer sizes. Smaller windows lead to smaller buffer requirements in the routers, but imply a coarser rate control granularity. One optimization point could be to have a different time window for each tile and to adapt that window according to the flows that originate in the tile. The hardware of the Kalray MPPA-256 platform would already support such an optimization.

A further possible optimization would be the use of different time windows for different flows from the same node. However, this would require several traffic shapers and a buffer to merge the individual flows after traffic shaping. Therefore, such an optimization would increase hardware costs.

The TDM scheduler we used in this paper tries to minimize the TDM period for given bandwidth requirements. The bandwidths are not actively changed during scheduling. If bandwidth is more important than latency, we can envision a TDM scheduler that trades longer TDM periods (and hence higher latencies) for higher bandwidths of some or all flows.

6. CONCLUSION

TDM and network calculus are two competing approaches to provide latency and bandwidth guarantees for real-time systems in NoCs. This paper compared these two approaches to gain a better understanding of their respective advantages and disadvantages.

We evaluated latencies and bandwidths for a generic all-to-all communication graph and several application-specific communication graphs. To compute concrete values, we assumed a TDM NoC and a rate-controlled NoC that are both modeled after actual NoC implementations. The results show that TDM provides better latency guarantees, while network calculus results in higher bandwidths. We identified optimization potential in the TDM scheduling approach we used and are looking forward to future research that may reduce pessimism in the network calculus approach.

This paper also discussed properties of a TDM approach and a network calculus approach that are not reflected in mere latency and bandwidth numbers. On the one hand, TDM NoCs can be implemented very efficiently in hardware. On the other hand, best-effort traffic can be accommodated more easily in a rate-controlled NoC. Future work will have to show whether the size advantage of a TDM NoC can make up for the greater flexibility of a rate-controlled NoC.

Acknowledgments. The work presented in this paper was funded by the Danish Council for Independent Research | Technology and Production Sciences under the project RTEMP,[1] contract no. 12-127600.

7. REFERENCES

[1] F. Arnaud, F. Boeuf, F. Salvetti, D. Lenoble, F. Wacquant, C. Regnier, P. Morin, N. Emonet, E. Denis, J. Oberlin, et al. A functional 0.69 μm^2 embedded 6T-SRAM bit cell for 65 nm CMOS platform. In *2003 Symposium on VLSI Technology. Digest of Technical Papers.*, pages 65–66. IEEE, 2003.

[1] http://rtemp.compute.dtu.dk

[2] M. Bakhouya, S. Suboh, J. Gaber, and T. El-Ghazawi. Analytical modeling and evaluation of on-chip interconnects using network calculus. In *3rd ACM/IEEE International Symposium on Networks-on-Chip (NoCS)*, pages 74–79, May 2009.

[3] M. Boyer, N. Navet, M. Fumey, J. Migge, L. Havet, and T. Avionics. Combining static priority and weighted round-robin like packet scheduling in afdx for incremental certification and mixed-criticality support. In *5th European Conference for Aeronautics and Space Sciences (EUCASS), Munich, Germany*, 2013.

[4] F. Brandner and M. Schoeberl. Static routing in symmetric real-time network-on-chips. In *20th International Conference on Real-Time and Network Systems (RTNS 2012)*, pages 61–70, Pont a Mousson, France, November 2012.

[5] R. L. Cruz. A calculus for network delay. I. Network elements in isolation. *IEEE Transactions on Information Theory*, 37(1):114–131, Jan 1991.

[6] R. L. Cruz. A calculus for network delay. II. Network analysis. *IEEE Transactions on Information Theory*, 37(1):132–141, Jan 1991.

[7] B. Dupont de Dinechin, Y. Durand, D. van Amstel, and A. Ghiti. Guaranteed services of the NoC of a manycore processor. In *International Workshop on Network on Chip Architectures (NoCArc)*, pages 11–16, New York, NY, USA, Dec. 2014. ACM.

[8] B. Dupont de Dinechin, D. van Amstel, M. Poulhiès, and G. Lager. Time-critical computing on a single-chip massively parallel processor. In *Conference on Design, Automation and Test in Europe*, DATE '14, pages 97:1–97:6, 3001 Leuven, Belgium, Belgium, 2014. European Design and Automation Association.

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

[10] A. Hansson, M. Subburaman, and K. Goossens. Aelite: a flit-synchronous network on chip with composable and predictable services. In *Conference on Design, Automation and Test in Europe*, DATE '09, pages 250–255, 2009.

[11] IBM ILOG. CPLEX Optimization Studio, 2014.

[12] L. S. Indrusiak. End-to-end schedulability tests for multiprocessor embedded systems based on networks-on-chip with priority-preemptive arbitration. *Journal of systems architecture*, 60(7):553–561, 2014.

[13] M. Joseph and P. K. Pandya. Finding response times in a real-time system. *Comput. J*, 29(5):390–395, 1986.

[14] E. Kasapaki. *An Asynchronous Time-Division-Multiplexed Network-on-Chip for Real-Time Systems*. PhD thesis, Technical University of Denmark (DTU), 2015.

[15] E. Kasapaki, M. Schoeberl, R. B. Sørensen, C. T. Müller, K. Goossens, and J. Sparsø. Argo: A real-time network-on-chip architecture with an efficient GALS implementation. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, PP, 2015.

[16] F. P. Kelly, A. K. Maulloo, and D. K. Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research society*, pages 237–252, 1998.

[17] H. Kopetz and G. Bauer. The time-triggered architecture. *Proceedings of the IEEE*, 91(1):112–126, 2003.

[18] H. Kopetz and G. Grünsteidl. TTP — a protocol for fault-tolerant real-time systems. *Computer*, 27(1):14–23, 1994.

[19] J.-Y. Le Boudec. Network calculus made easy. Technical Report EPFL-DI 96/218, EPFL, 1996.

[20] J.-Y. Le Boudec. Application of network calculus to guaranteed service networks. *IEEE Transactions on Information Theory*, 44(3):1087–1096, May 1998.

[21] W. Liu, J. Xu, X. Wu, Y. Ye, X. Wang, W. Zhang, M. Nikdast, and Z. Wang. A NoC traffic suite based on real applications. In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 66–71, July 2011.

[22] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch. Guaranteed bandwidth using looped containers in temporally disjoint networks within the Nostrum network on chip. In *Conference on Design, Automation and Test in Europe*, DATE '04, pages 20890–, Washington, DC, USA, 2004. IEEE Computer Society.

[23] Y. Qian, Z. Lu, and W. Dou. Analysis of worst-case delay bounds for best-effort communication in wormhole networks on chip. In *3rd ACM/IEEE International Symposium on Networks-on-Chip (NoCS)*, pages 44–53. IEEE, 2009.

[24] M. Schoeberl, S. Abbaspour, B. Akesson, N. Audsley, R. Capasso, J. Garside, K. Goossens, S. Goossens, S. Hansen, R. Heckmann, S. Hepp, B. Huber, A. Jordan, E. Kasapaki, J. Knoop, Y. Li, D. Prokesch, W. Puffitsch, P. Puschner, A. Rocha, C. Silva, J. Sparsø, and A. Tocchi. T-CREST: Time-predictable multi-core architecture for embedded systems. *Journal of Systems Architecture*, 0(0):–, 2015. published online, to appear in print.

[25] M. Schoeberl, F. Brandner, J. Sparsø, and E. Kasapaki. A statically scheduled time-division-multiplexed network-on-chip for real-time systems. In *6th International Symposium on Networks-on-Chip (NoCS)*, pages 152–160, Lyngby, Denmark, May 2012. IEEE.

[26] Z. Shi and A. Burns. Real-time communication analysis for on-chip networks with wormhole switching. In *2nd ACM/IEEE International Symposium on Networks-on-Chip (NoCS)*, pages 161–170, April 2008.

[27] Z. Shi and A. Burns. Schedulability analysis and task mapping for real-time on-chip communication. *Real-Time Systems*, 46(3):360–385, 2010.

[28] R. B. Sørensen, J. Sparsø, M. R. Pedersen, and J. Højgaard. A metaheuristic scheduler for time division multiplexed network-on-chip. In *Software Technologies for Future Embedded and Ubiquitous Systems (SEUS), 2014*. IEEE, 2014.

[29] S. Suboh, M. Bakhouya, J. Gaber, and T. El-Ghazawi. Analytical modeling and evaluation of network-on-chip architectures. In *International Conference on High Performance Computing and Simulation (HPCS)*, pages 615–622, June 2010.

[30] K. Utsumi, E. Morifuji, M. Kanda, S. Aota, T. Yoshida, K. Honda, Y. Matsubara, S. Yamada, and F. Matsuoka. A 65nm low power CMOS platform with 0.495 μm^2 SRAM for digital processing and mobile applications. In *2005 Symposium on VLSI Technology. Digest of Technical Papers.*, pages 216–217, June 2005.

[31] H. Zhang. Service disciplines for guaranteed performance service in packet-switching networks. *Proceedings of the IEEE*, 83(10):1374–1396, Oct 1995.