INSTITUT FÜR
DATENTECHNIK UND
KOMMUNIKATIONS-
NETZE

Technische
Universität
Braunschweig

Resource Manager and Control Layer

Basic Concepts, Technical Report Version 1.0

Adam Kostrzewa, Sebastian Tobuschat, Rolf Ernst

February 23, 2018

Contents

Chapter 1: Introduction

1.1 Real-Time Applications

It is a common intuition that many automotive functions, especially when controlled with ECUs, have temporal constrains, e.g., breaking systems, steering or engine control. The upcoming ADAS functionality extends this spectrum even further on gathering of sensor data (e.g. timely video frames and / or radar signals), its processing (e.g. denoising, filtering, compression), and finally decision making (e.g. big data processing and machine learning with convolutional neural networks). However, real time requirements appear also in other electronic domains, e.g., traffic control, multimedia, mobile communication, medical applications, industrial robotics or avionic. In this section, we provide a brief overview and summary of the most important properties of such setups with a special focus on their requirements towards the communication architecture.

1.1.1 Real-Time Traffic Requirements

In systems with real-time (temporal) requirements, the correctness of the system design and working depends equally on temporal and functional aspects. Real-time applications/systems must guarantee not only that their results are logically correct, but also that they are delivered on time. The physical time by which a specific result must be produced is called deadline. Deadlines are frequently dictated by the environment and physical nature of the controlled process.

Fundamentally, real-time applications can be classified by a metric, which uses the consequences that a deadline miss can cause. The theory of real-time system design [63] distinguishes three application categories w.r.t the aforementioned criterion: hard real-time (HRT), soft real-time (SRT) and best-effort (BE). These categories will be briefly discussed in the following, as properties of the applications directly influence the properties of the initiated NoC traffic, i.e., a hard real-time application has hard real-time communication requirements w.r.t to bandwidth or latency. However, the presented classification is orthogonal to the different traffic metrics described in the next sections. For instance, it is possible to have a hard-real time application requiring guaranteed worst-case latencies or a soft-real time application requiring guaranteed throughput.

HRT applications have firm deadlines, i.e., the utility of the produced result is zero when the deadline is crossed. Consequently, any delay in their execution, including high latencies of initiated transmissions, may have severe consequences for the whole system, i.e., fatal faults and prohibitive degradation of service. Note, that this frequently includes situations when the user's safety may be in danger, e.g., activation of anti-lock breaking system (ABS), high latency of video frames from cameras in an ADAS system leading to malfunction.

Indeed, as presented in Figure 1.1, in practice many safety critical systems have hard deadlines - therefore systems are both safety and time critical. Therefore, transmissions conducted by HRT

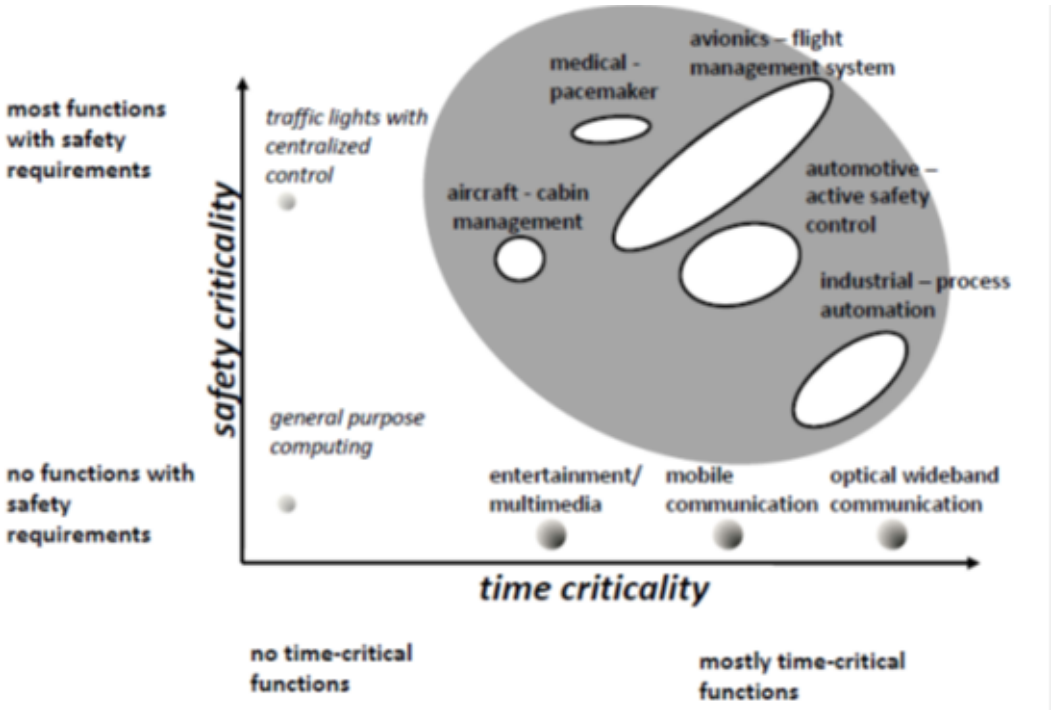


Figure 1.1: Dependency between safety and time-critical systems.

senders, i.e. hard real-time transmissions (HRTTs) are usually not allowed to miss any deadline, i.e., its worst-case latency must stay within assumed upper/lower limit. Similarly, even if the functionality does not directly affects user safety it may have hard real-time requirements from the producers point of view, as a failure of the service could cause client loss or substantial financial penalty. Accommodation of the traffic from HRT applications requires worst-case dimensioning during the design process and, in case of safety-critical systems, also a verification proving adherence to the standards. Due to these requirements the characteristics of traffic originated from hard real-time senders is usually well specified and tested, e.g., periodic DMA transfers.

Similarly to HRTTs, transmissions initiated by SRT senders, i.e. soft real-time transimssions (SRTTs), must also comply to the overall real-time performance objectives, e.g., guaranteed latency or throughput. The main difference, when compared to hard real-time senders, is that these applications are rarely required to rigorously meet all their deadlines, i.e., the produced results have some utility after the deadline see Figure 1.2. For instance, video streaming done as a part of infotainment functions in a car or a plane does not influence vehicle safety, but video frames must still arrive with a certain latency to prevent quality drops and glitches. Another example are control algorithms based on a feedback loop. Frequently, the algorithm may tolerate a limited number of cases when instead of new sampling data old values are used e.g. [113, 103, 88]. However, also in case of SRT senders timing can be a critical factor depending on other non-functional requirements. For a producer of an infotainment system the quality of user experience may play a critical role in the market success of a product. Consequently, it may accept sporadic drop of the video quality but may loose clients whenever it happens to often.

The last category of best-effort (BE) senders is populated by the majority of existing applications.

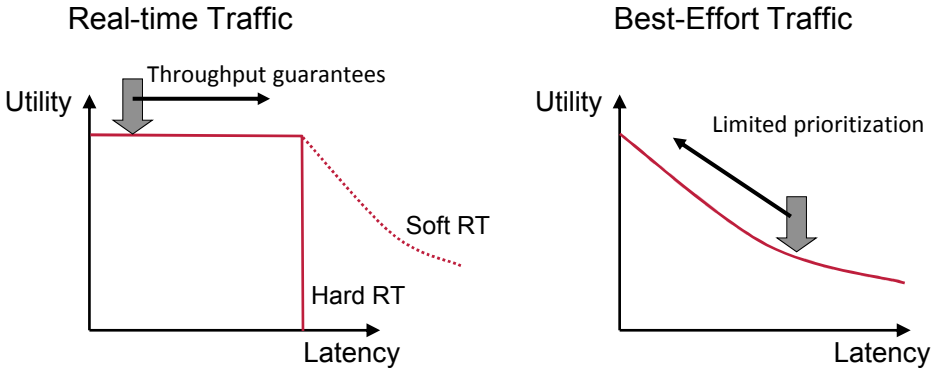


Figure 1.2: NoC requirements of different applications w.r.t the temporal isolation, from [92]

On the contrary to both aforementioned real-time classes, BE transmissions have no strict temporal requirements, i.e, a deadline missed by them does not endanger the system. Consequently, the behavior of best-effort applications is rarely tested w.r.t temporal properties, e.g., the number, duration and frequency of memory accesses. BE senders are usually designed targeting average performance metric, e.g., average latency, average throughput and compiled with standard toolchains without considering the temporal properties nor certification requirements, e.g., GNU compilers and applications running on processors with caches. Consequently, in case of initiated accesses to the on-chip interconnect they may exhibit high burstiness, i.e., an unpredictable and highly variable resource usage (number of and length of transmissions). BE senders suffer from lack of temporal models and their integration with other applications having real-time requirements is difficult.

Figure 1.2 presents a short summary of temporal properties of NoC traffic. The X-axis depicts worst-case communication latency and the Y-axis the utility of data, i.e., the usefulness for the receiver. A missed deadline is denoted by the utility value equal to zero. Punctual data arrival (before or on deadline) is denoted by the utility value equal to one. All other values of the utility function (between these bounds) denote a performance degradation. For instance, HRTTs have utility only if they arrive before the deadline. For SRTTs utility function (usefulness of the data) decreases after the missed deadline.

1.1.2 Integration of Traffic

As discussed in Section 1, Networks-on-Chip are foreseen as a communication backbone for large SoCs integrating different ECUs. As a result of such integration, it can happen that diverse traffic classes, i.e., HRTTs, SRTTs and BE, must share the SoC resources. This causes co-dependencies between applications running on different cores, what may endanger safety. Unpredictable and bursty accesses from BE senders may lead to contention in network buffers.

In on-chip interconnects without appropriate quality-of-service (QoS) mechanisms, resources are not reserved in advance, i.e., transmission are scheduled as soon as they arrive, and all traffic receive equal treatment. Because of that, some interference from BE traffic may lead to missed deadlines by SRTTs or HRTTs.

On the other hand, there is usually no advantage in completing an HRTT earlier than required

since design process (e.g. safety standards) usually insist on the satisfaction of all timing constraints even if unnecessarily stringent, i.e., as long as an application completes by its deadline, its response time is not important [93, 104]. This property could be used to slow-down SRTTs and HRTTs for accommodating BE traffic. Finally, the majority of best-effort applications, although not timing critical, is still latency-sensitive. Their performance degrades with higher latency as presented in Figure 1.2. An example for this are applications running on processors with caches, which, although rarely, must frequently fulfill temporal requirements to profit from low average-case latencies for improved resource (processor) utilization. Consequently, they profit from higher interconnect performance and higher SoC resource share.

This properties leads to contradictory requirements whenever the interconnect is shared between different classes of real-time traffic.

The integration of HRTTs, with SRTTs and BEs in the same chip requires enforcing *sufficient independence* between components, i.e., assuring that the behavior of HRT is independent from the behavior of SRT and BE senders. However, this often compromises the performance of SRTTs or BEs. Quality-of-Service mechanisms for on chip interconnects usually prioritize hard real-time senders over best-effort traffic and soft real-time traffic. Hence, BE traffic suffers from high latency although time critical traffic has no to little benefit from reduced latency.

Moreover, worst-case dimensioning required for the deployment of HRT senders leads frequently to resource over-provisioning. However, during regular work of the system such extreme conditions may rarely occur. This results in a significant drop of average utilization, i.e., underutilized resources. Consequently, an efficient co-execution of such diverse application types is still an open research question with possibly high engineering and economic impact, i.e., is critical to the success of NoCs on the market.

1.2 Safety Standards and Certification

As already discussed, although not all real-time systems are safety critical, the majority of safety-critical systems have real-time constrains. Safety critical systems are system where a malfunction can cause "unacceptable risk of physical injury or of damage to the health of people, either directly, or indirectly as a result of damage to property or to the environment" following the definition from [3]. Therefore, since several years, one may observe accumulation of industrial and research efforts towards standardization of the safety life cycle for electronic products.

This process is already at an advanced stage in the case of the avionics industry and the domain of industrial automatics. In the avionic domain, the development and validation of software is regulated by the standard DO-178b [2] and of underlying hardware components by DO-254 [1]. During a rigorous validation process the producers must prove compliance of their products to the requirements and production methods enforced by these standards, what plays a decisive role for aircraft approval by certification authority such as the Federal Aviation Administration (FAA). A similar process is also enforced for the development of heavy machinery (IEC 62061) and system for process industries (IEC 61511), railway (IEC 62279) and power plants (IEC 61513).

In case of automotive industry, safety standards have been introduced relatively late.[1]. The major

[1]The reasons for these are: 1) consequences of a car crash are considered to be minor in comparison with an avionic accident; 2) the liability constrains enforcing certain quality from automotive manufacturers

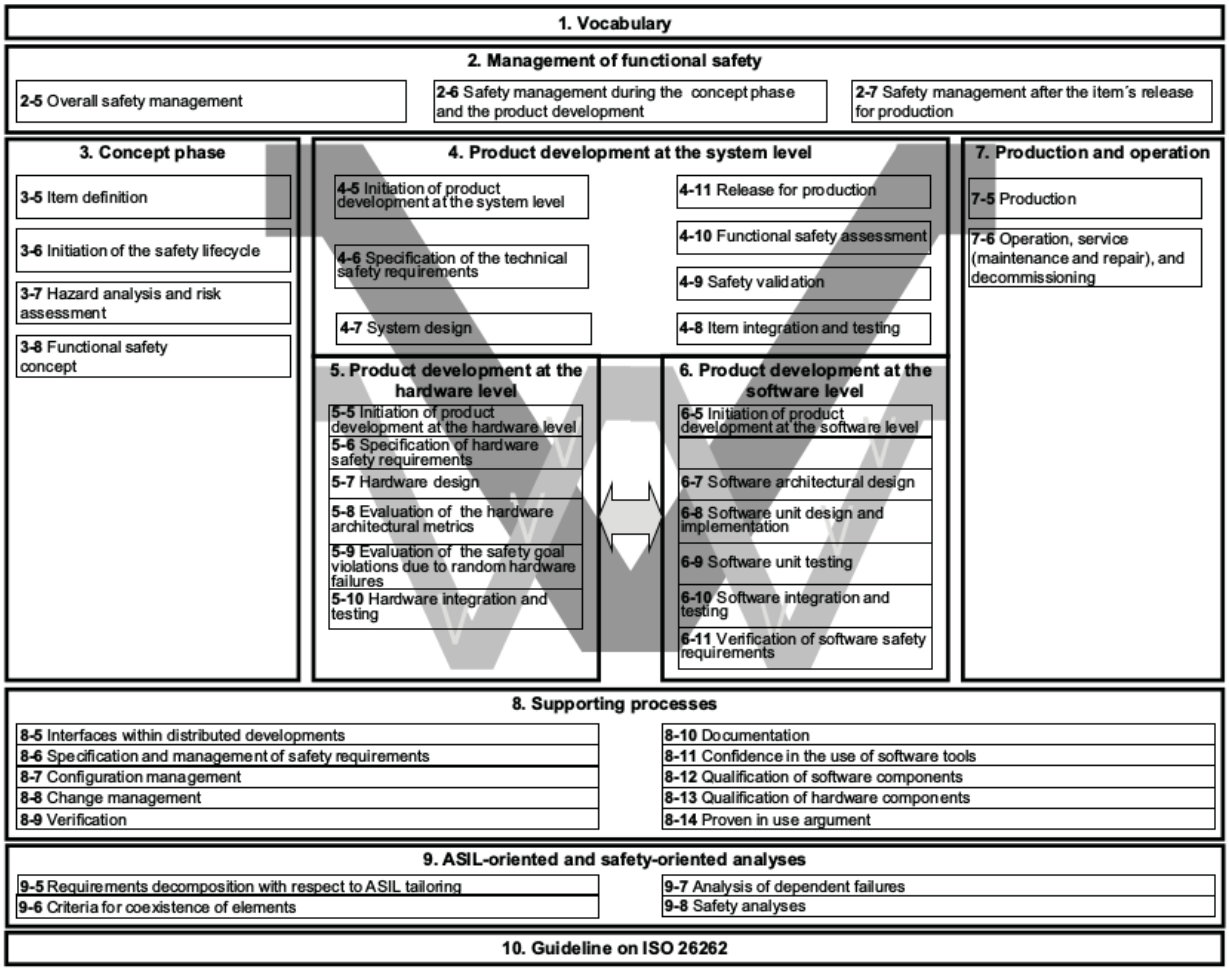


Figure 1.3: Simplified V-Model according to ISO26262 [44]

industrial efforts has led to the establishment of the ISO26262 [44] standard derived directly form the IEC 61508 [3] standard document in 2011. The IEC 61508 introduced by the International Electrotechnical Commision (IEC) proposed a generic approach for the safety life cycle of all systems comprised of electrical and/or electronic elements including programmable elements. Its main goal was to establish a foundation, which can be adjusted for different branches of the industry, e.g., rail, machinery, power plants. Therefore, the ISO26262 constitutes in many aspects a straightforward extension of this approach adjusted for the purpose of the automotive domain. However, there are also differences. For instance, the original IEC 61508 considered low volume fabrication of components, whereas ISO26262 is focused on large quantities, i.e., serial production.

According to ISO26262 safety is defined as the absence of unreasonable risk. Solving of this problem in case of electrical and electronic devices requires incorporation of the appropriate measures throughout the design process. Consequently, ISO26262 focuses and refines the V-Model [44] presented in Figure 1.3. It offers a top-down approach towards to the engineering problem, where first the requirements must be delivered assessing: a) the specification of the intended functions and their interactions necessary to achieve the desired behavior of the system (functional concept) as

well as b) quantification of the tolerable risk. Although the former task is straightforward the latter can be demanding. It usually requires formalized methods and processes for achieving the assurance level required by standards and certification authorities. This is done through a model driven design, where the producer must provide not only the product but also sufficient data to verify and test the design (artifact") independently.

Therefore, networks-on-chip, whenever used for communication between safety critical IPs such as automotive functions, are or will be, depending on the safety-critical domain, the subject of regulation through standards and certification procedures for assuring their correct functioning. In this context, not only the possibly high average performance and low costs play a critical role but also, even more importantly, the ability to prove adherence to the safety requirements. This adds another complexity layer to the design process and requires tracability w.r.t to real-time properties, e.g., application of formal analysis methods such as Real-Time Calculus [99], Network Calculus [62] or Compositional Performance Analysis [40].

The following of this section details which of the ISO26262 requirements for the data communication are applicable to Networks-on-Chip.

1.2.1 Sufficient Independence

There are multiple sources of possible hazards in a MPSoC which differ in the severity and exposure. The process of integrating several ECUs in the same system-on-chip leads to a setup in which not all integrated IPs might have the same impact on the system safety (i.e. criticality) and thus a *mixed-criticality* system. For instance, some of them may not be designed considering users safety, e.g., worst-case behavior. Consequently, their behavior cannot be trusted. In the following, these IPs (hardware or software) will be called *non-critical* and they constitute the majority of BE traffic. Moreover, even safety critical IPs may have different impact on users safety and therefore require different certification efforts. In the automotive context, ISO26262 distinguishes between four different Automotive Safety Integrity Levels (ASIL A-D) defining the relative level of risk-reduction provided by a safety function. Each of them defines more restrictive risk analysis and safety goals.

For such mixed-criticality systems, safety standards require certification of the whole system to the highest relevant safety level (e.g. highest ASIL) or temporal and spatial separation. For instance, the IEC 61508 states ”If the safety integrity requirements for these safety functions differ, unless there is sufficient independence of implementation between them, the requirements applicable to the highest relevant safety integrity level shall apply to the entire E/E/PE safety-related system.”. Similarly, in ISO26262 it is defined ”If the embedded software has to implement software components of different ASILs, or safety-related and non-safety-related software components, then all of the embedded software shall be treated in accordance with the highest ASIL, unless the software components meet the criteria for coexistence in accordance with ISO 26262-9:2011, Clause 6.”, where Clause 6 proposes "In the case of the coexistence of sub-elements that have different ASILs assigned or the coexistence of sub-elements that have no ASIL assigned with safety-related ones, it can be beneficial to avoid raising the ASIL for some of them to the ASIL of the element. When determining the ASIL of sub-elements of an element, the rationale for freedom from interference is supported by analyses of dependent failures focused on cascading failures”. The freedom of interference is later defined as ”absence of cascading failures between components that could lead to the

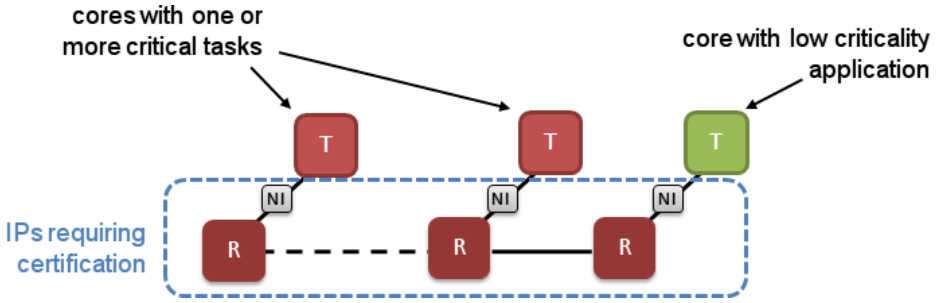


Figure 1.4: NoC in setups with mixed-criticality must be certified according to the traffic with highest criticality.

violation of [some] safety requirements". By applying these rules to the system-on-chip, the parts of the HW and RTE, which are always used, must be certified to the highest relevant safety level. For all other components "sufficient independence" must be implemented, see Fig. 1.4. As non-critical tasks cannot be trusted (e.g. unknown execution time, activation frequency, communication volume) network interfaces must separate the critical network from non-critical tiles. Additionally, routers must provide a predictable upper bound on the worst-case interference between concurrent transmissions.

1.2.2 Communication Faults and Real-Time Properties

According to the ISO26262 the communication of safety-related data must be protected at runtime against effects of faults which may lead to failures of the system. These faults include transient faults, e.g. single event upsents such as bit-flips from electro-magnetic radiation leading to corrupted data, physical damage as well as lack of sufficient independence between tasks. Consequently, the ISO26262 provides a list of faults, presented in Table 1.1 regarding the exchange of information which must be considered in case of an interconnect for certification purposes. In this context, the end-to-end protection defines a set of mechanisms which allow a reliable detection of these faults and appropriate countermeasures. In the NoC context, some of these faults directly refer to the real-time metric for the on-chip interconnect, e.g., a delay of information or blocking access to communication channel. Others relate to the protection of packets done directly in routers, e.g., without a backpreassure signal/flow control (informing about full buffers in ports) packets can be overwritten what inherently leads to corruption of information.

Note, that other faults, although not directly related to the temporal metrics, frequently influence temporal predictability indirectly. Transient errors, malfunctioning or malicious senders may introduce uncertainty and dynamics to the system, e.g., sporadic overloads due to re-transmissions or "Babbling idiots". Therefore, their contribution may significantly decrease the worst-case estimation of the system performance.

1.3 Requirements for NoCs with Real-Time and/or Safety-Critical Workloads

NoC architectures are judged by three measures [23]: production cost, transmission latency, and throughput. The two latter factors are performance metrics. Latency denotes the time necessary for

Fault Type	Description
Repetition of information	A type of communication fault, were information is received more than once.
Loss of information	A type of communication fault, were information or parts of information are removed from a stream of transmitted information.
Delay of information	A type of communication fault, were information is received later than expected.
Insertion of information	A type of communication fault, were additional information is inserted into a stream of transmitted information.
Masquerade or incorrect addressing	A type of communication fault, were non-authentic information is accepted as authentic information by a receiver.
Incorrect sequence of information	A type of communication fault, were information is accepted from an incorrect sender or by an incorrect receiver.
Corruption of information	A type of communication fault, which changes information.
Asymmetric information from sender to multiple receivers	A type of communication fault, were receivers do receive different information from the same sender.
Information from a sender received by only a subset of the receivers	A type of communication fault, were some receivers do not receive the information.
Blocking access to a communication channel	A type of communication fault, were the access to a communication channel is blocked.

Table 1.1: Summary of communication faults which must be considered in the design of the automotive systems, from ISO26262 [44]

a packet to traverse the interconnect whereas the throughput defines the amount of data per time unit which can be moved through the network (e.g. in bits per second). Consequently, the majority of NoC architectures target a possibly high *average performance* [23]. In the following, such NoCs will be addressed as common or performance optimized NoCs/architectures.

However, even if the NoC design allows to achieve high utilization at low cost it is still difficult to guarantee system predictability in real-time domains like avionic or automotive. Whenever concurrent transmissions compete for the interconnect resources (wires, buffers) the resulting interference couples the execution of applications running on different cores. The network architecture must assure that this interference is predictable and stays within the assumed limits. Otherwise a sender may not comply with its temporal requirements.

Therefore, providing predictable arbitration between concurrent transmissions is a critical factor for the design of NoCs in SoCs with temporal requirements. Moreover, the support for temporal properties should not cancel out benefits resulting from the application of NoCs, e.g., high efficiency, scalability, flexibility and low production costs. This requires from the designer to adjust the arbitration of interconnect resources for incorporation of dynamics resulting from system integration, e.g., simultaneously hosting different real-time and/or safety traffic classes, incorporation of system dynamics. Mechanisms designed for this purpose should be not only efficient and affordable but also provide a possibility of straightforward verification and testing as required by safety standards. The summary of these requirements is presented in Table 1.2. The following of this chapter provides a detailed discussion and justification.

Code	Reuqirement	Metrics	Full Description
R1	Traffic Types	Real-Time Performance Safety Costs	NoC should provide support for multiple traffic classes e.g. GL, GT, BE
R2	Workload Integration	Real-Time Performance Costs	NoC should provide efficient support for real-time requirements without need to modify legacy code and other IPs
R3	Flexibility	Performance Costs	NoC architecture should be able to support different resource allocation strategies depending on particular setup
R4	Dynamics	Performance Safety Costs	NoC architecture should allow efficient and safe incorporation of dynamics in system behavior
R5	Fairness	Real-Time	NoC architecture should provide fair allocation of interconnect resources whenever RT senders have different requirements
R6	Mixed-Criticality	Safety Performance	NoC architecture should provide as high average performance to the BE senders as possible in mixed-critical setups
R7	Switch-off QoS	Costs	NoC architecture should provide a possibility to switch-off real-time and/or safety mechanisms whenever there are no senders with such requirements deployed
R8	Scalability	Costs	NoC architecture should provide performance (average and worst-case) proportionally to the load at runtime (i.e. work conserving arbitration) and not other static factors e.g. number of senders
R9	Locality	Performance Real-Time Safety	NoC architecture should preserve the order of arriving packets whenever it is necessary
R10	Verification	Safety Costs	NoC architecture should allow efficient formal verification for standartization/certification purposes
R11	Detect NoC HW faults for higher ASIL levels	Real-Time Safety	Errors must be detected in order to initiate countermeasures (for fail-safe and fail-operational behavior).

Table 1.2: Requirements for the contemporary and future NoC architectures for deployment in real-time and safety-critical domains e.g. automotive.

Support Different Traffic Types (R1,R5,R6)

The main purpose of a real-time NoC is to assure that temporal properties of transmissions comply with the timing requirements of the integrated applications. These requirements for real-time traffic are usually defined with the notions of worst-case latency and minimum throughput. The former defines the upper bound on the duration of the communication, e.g., maximum latency of a single packet transmission. The latter refers to the lower bound on the amount of data transferred per unit of time.

Consequently, the distinction between these two classes of safety-critical workloads is visible in many NoC designs. The transmissions are than classified as:

- *Guaranteed Latency (GL)* traffic, requiring strict guarantees for each initiated transmission, e.g., control traffic, synchronizations or interrupts. Note, that usually such senders require low latencies but at the same time transmit short messages, i.e., low communication volume.
- *Guaranteed Throughput (GT)* traffic, requiring strict temporal guarantees per certain data/transmission volume. Consequently, the sender requires service guarantee per whole logical transmission which can be composed from several packets rather per each packet independently. For instance, in case of streaming applications using DMA engines for cyclic transfers it is important that the whole chunk of data is available at certain moment in time. Consequently, the latency of single packets may vary as long as the deadline for the burst is held.
- *Best Effort (BE)* traffic, having no strict requirements with respect to timing, i.e., no strict latency and throughput requirements are known which could endanger the system safety. However, frequently BE traffic profits from lower latencies as it originates from “general-purpose” applications such as typical desktop or infotainment functionalities. These by design profit from low average temporal metric, e.g., low average latency or high throughput. The reason for this is that low latencies allow to increase the utilization of cores and therefore of the whole system. This allows for instance to improve user experience in case of infotainment systems.

These traffic classes constitute the basis for the designs. Note that combinations of requirements are possible, e.g., high latency, low throughput traffic. Moreover, the knowledge and understanding of the workloads may be exploited to optimize the designs for instance safely postpone some the packets from GT transmissions to improve performance of BEs. Consequently, the NoC architecture for real-time workloads should provide support for multiple traffic classes to ease their incorporation. This is reflected in requirement R1 "Traffic Classes" from the Table 1.2.

Moreover, whenever a NoC is used to host multiple applications with different requirements this should be reflected by the interconnect arbitration. First of all, even if different transmissions have hard real-time or soft-real time requirements they must not necessarily be the same. As described in Section 1.1.2, some transmissions may finish later than others and still be on time. Therefore, the arbitration in the NoC should be fair - offer sufficient resources for sender to comply with its requirements but avoid overprovisioning. This is reflected in requirement R5 "Fairness" from the Table 1.2.

As discussed in Section 1.2.1, this is especially important in mixed-critical setups where there is usually no advantage in completing an HRTT earlier than required since safety standards insist on the satisfaction of all timing constraints even if unnecessarily stringent, i.e., as long as an application

completes by its deadline, its response time is not important. On the contrary, transmissions from soft-real time and best effort applications (SRTTs) are rarely required to rigorously meet their deadlines but at the same time they must still comply to the overall real-time performance objectives, e.g., low average latencies. However, many of safety mechanisms compromises performance of BEs and SRTTs what should be avoided. This problem is reflected in requirement R6 "Mixed-criticality" from the Table 1.2.

Integration Efforts (R2,R8,R9)

The previous section explained why application of NoCs for workloads in real-time, safety-critical domains is much more challenging than in the case of general-purpose computing. Note that temporal requirements do not exclude other design goals which may be used in conjunction, as in other general purpose architectures. Consequently, in the ideal case supporting real-time properties should be transparent to other integrated application and not decrease their performance nor require special integration efforts.

Firstly, integration of real-time senders in a NoC should not require extensive modification of IPs, e.g., legacy code or design of hardware components. This is not only important due to the increased production costs, which such modifications could cause, but also because producers frequently apply proprietary components and have no rights and/or possibility to adjust them. Moreover, even slight adjustments in safety-critical domains could cause the costly process of verification and re-certification. These challenges are reflected in requirement R2 "Workload Integration" from the Table 1.2.

Moreover, mechanisms for QoS should not limit the scalability of NoC architectures. Although strict separation of transmissions connected with static allocation allows providing guarantees it usually results in high overhead. For instance, in performance optimized NoCs, to guarantee the safety of critical senders, mapping algorithms try to avoid overlapping of paths used by safety-critical and non-critical, best-effort (BE) traffic, i.e., spatial separation of transmissions. Otherwise, the critical streams may not be able to meet their deadlines. This typically leads to longer (detoured) paths for BE traffic, which degrades the BE performance as these are often latency sensitive [73]. That is, although BE sender are rarely required to rigorously meet all the deadlines, they must still comply to overall real-time performance objectives as low average latencies. At the same time, the static mapping based on the worst-case behavior leads frequently to significant decrease of hardware utilization, especially when safety critical transmissions are conducted sporadically and resources (paths) reserved for them are rarely used. For instance, the arrival of a safety-critical Ethernet frame that must be forwarded on the shortest route through the NoC occurs rarely when compared to cache traffic and memory accesses of BE tasks [5]. Consequently, the performance of the NoC architecture should only be influenced by the initiated load at runtime and not static factors such as the number and type of integrated senders. This challenge is reflected in the requirement R8 "Scalability".

Finally, the NoC design can additionally assure following traffic properties [23]:

- *traffic locality* requires that: i) packets belonging to the same logical connection arrive in the same specific order in which they were injected to the NoC (in-order delivery) and / or ii) that streams from two different senders targeting the same node do not mix in the NoC. This is

especially important in case of state-dependent peripherals, such as DDR-SDRAM memories, for which performance, response time and power consumption depends on the history of previous accesses [56].

- certain *jitter/varying arrival rate* (i.e. arrival rate) as some applications require certain granularity of data to proceed with execution, e.g., the rate of arrival of video frames for a video decoder. The goal is to limit the arrival jitter which could appear in case when non-bursty traffic mixes with a strongly varying in time.

In many deployment scenarios, protecting locality of connections allows to increase the utilization of the pheripheareals and IP components. For instance, this happens in case of latencies of the DDR-SDRAM memories. Indeed, SDRAMs have an internal level of caching, which in standard DDR3 modules amounts to 8kB. Consequently, contiguous and aligned 8kB long transfers fully benefit from the caching. Consequently, these requirements are reflected by the requirement R9 "Locality" from the Table 1.2.

Incorporation of Dynamics (R4)

Traditionally, the problem of interference between concurrent transmissions in real-time NoCs is solved with Quality-of-Service (QoS) mechanisms, which minimize non-functional dependencies at the cost of less efficient communication protocols. Indeed, the frequently applied, static resource allocation makes communication timing straightforward but results in a significant performance decrease.

However, this is contradictory to the requirements of highly dynamic, contemporary applications in real-time domains that demand allocation flexibility and extensive on-chip reactivity, e.g., advanced driver/pilot assistance systems, industrial robotics, autonomous flight/drive systems. For instance, automated driving introduces a transition of decision making from the driver to the vehicle. Decisions and actions must be taken in the context of a dynamic, constantly changing environment, e.g., situation on the road or weather conditions. This is done through a concurrent execution of complex functionalities including high resolution sensor fusion and data processing combined with machine-learning. At least a part of these functions will be implemented on a many-core system to reduce the number of components and cost. Consequently, NoCs must efficiently host new sets of highly dynamic workloads and the behavior of the system may be influenced by many external factors. Tasks may modify their transmission profiles at run-time depending on the arriving sensor data. Moreover, sets of applications may be initiated dynamically, introducing system modes with changing traffic patterns and mapping or workload profiles, e.g., convolution of a neural network used for decision making. Similarly, sensor fusion and data crunching bring up data-dependent execution, resulting in a highly dynamic task behavior and large jitters. Finally, the dynamics can happen due to the transient-errors and mechanisms preventing them such as data re-transmissions or acknowledges. Therefore, the control used for the underlying NoC architecture must protect the system safety against pitfalls resulting from dynamic behavior (e.g. transient overloads, data loss, high transmission latencies and missed deadlines) while delivering the requested performance. This challenge is reflected in requirement R4 "Dynamics" from the Table 1.2.

Efficient Verification (R10)

As discussed in Section 1.2, safety standards enforce separation of the design and specification from implementation. This requires providing methods for verification and testing of the Networks-on-Chip and not only the functioning implementations.

In this context, some design decisions may increase significantly the difficulty of the formal verification. For instance, complex mechanisms for providing simultaneously efficient guarantees for HRTs and average performance for BEs, although feasible frequently lead to a fine-granular traffic classification with the temporal guarantees as a predominant requirement. Consequently, routers and NIs are becoming more complex (e.g. extra header information, translation tables, additional logic) making them more difficult to analyze and formally verify. This increases production costs and effort. On the other hand, it is still important that NoC resources are shared (i.e. allow as many communications as possible) and that transmission latencies are low (i.e. efficient synchronization with minimum impact on senders performance). This additional validation effort should be therefore as low as possible what is reflected in R10 "Verification" from the Table 1.2.

Note that formal verification must account also for the effects of dynamics described in the previous paragraphs. This may be often difficult and contradictory to other mechanisms in the NoC for instance fail-safety. In fact, the load of 50% which is already considered to be significant in case of automotive systems whenever formal guarantees must be provided and the system exposes dynamics in its behavior [82]. Indeed, results from Daimler [82] state: "... it is accepted that the cyclic busload should not exceed 50% (the exact upper bound depends on the OEM)" and later "In networks dominated by event driven communication, the theoretical worst case load then overshoots 100% and may reach 500%.". These claims are supported by Bosch in [113]. Otherwise, OEM will not be able to provide formal guarantees required by standards.

Safety and Real-Time as Features of the Architecture (R3, R7)

In many architectures proposed by academia, support for real-time and safety features plays a central role and therefore other practical properties are neglected. It is important to mention that although the market of the automotive electronics is the fastest growing one among embedded systems [67] it is still covering less than 20% of the overall production. Moreover, real-time workloads are usually in the minority of developed and deployed applications.

Consequently, from the point of view of a manufacturer, the best solution would be to provide a generic platform which can handle simultaneously general purpose, BE applications and, whenever it is necessary, real-time and safety critical features at low cost. This would allow to increase the production series of SoCs and leave it to the integrator if and to what extend these specific requirements should be incorporated in the particular design. Consequently, a future NoC architecture should allow safety and/or predictable timing *as a feature* of the design which may be activated on demand not its sole purpose. This is reflected in requirement R7 "Switch-off QoS" from Table 1.2.

Of course, such approach is only feasible if the overhead resulting from the domain specific mechanisms is low in comparison with the overall production costs of the IP component.

Similar considerations apply to methods/mechanisms used for supporting temporal and safety requirements. The workloads in embedded domains may differ significantly depending on applications. For instance, in some setups the system integrator may only have safety-critical tasks. They

are highly optimized and tested and due to the nature of controlled process therefore transmit its data using regular pattern with a pre-defined transmission size, e.g., feedback loops used in control engineering. However, in other applications such as ADAS they may be a lot of sporadic and dynamic sensor data which, as discussed before, require different arbitration and policies for efficient work. Therefore, the same generic NoC in the optimal scenario should offer designers possibility to select optimal Quality-of-Service mechanism and policy depending on the particular deployment. This challenge is reflected in requirement R3 "Flexibility" from Table 1.2.

Chapter 2: A Survey of Mechanisms for Supporting Real-Time in NoCs

The integration of traffic from multiple senders in the same Network-on-Chip results in setups where concurrent transmissions compete for the interconnect resources such as link bandwidth and buffer space. In the real-time systems the most important challenge is adherence to the temporal requirements of applications which dominates other requirements, e.g., high-average performance. Consequently, real-time NoCs must provide mechanisms for supporting predictable arbitration. This means that such networks should offer a predictable upper bound on the worst-case end-to-end latencies. Moreover, whenever the NoC is running as a part of safety-critical system there is a need to verify that property during the design process and integration, as request by safety standards. For achieving predictable timing, NoC architectures offer two main control features, cf. [23]:

- **flow control** deciding about scheduling and allocation of network's resources, such as channel bandwidth, buffer space, and control state. Flow control is usually realized through control path components, i.e., arbiters, in routers. The duration of the single grant depends on multiple factors. For instance, some requesters may require uninterrupted access to the resource for a number of cycles.
- **admission control** deciding about who and when can initiate the transmission, i.e., when an application can access the NoC. It is usually adjusted in network interfaces of processing nodes, e.g., rate limiters enforcing minimum temporal distance between two consecutive transmissions from the same processing node.

This chapter presents an overview of available methods and mechanisms for assuring temporal properties in a NoC. The presented survey encompasses temporal and spatial isolation methods. As will be shown, this challenge of providing real-time predictability (and safety whenever relevant) is not trivial as these mechanisms must fulfills multiple design goals at once, as described in the previous Chapter 1. For instance, they must assure predictable behavior and performance simultaneously. Moreover, it is necessary to cope efficiently with system's dynamics resulting from application behavior (e.g. activation jitter, transmissions duration) as well as physical environment (e.g. sensor data, off-chip communication). Finally, safety standards for higher safety-levels require verification by formal methods.

2.1 Spatial Isolation of Traffic in Networks-On-Chip

The most straightforward method for achieving temporal predictability in real-time NoCs is spatial isolation, i.e., assigning each of the senders (or sender/traffic classes) its own set of NoC resources, i.e., links and buffers. This is commonly achieved through static load distribution with

oblivious routing [23]. When applied, the paths followed by packets during runtime are independent from the current state of the NoC and determined only by the source and destination of the data stream. In mixed-critical NoCs this allows to limit the interference and fully isolate senders of different criticality either in space and/or time.

Note however, that all transmissions inherently suffer from the main drawback of oblivious routing - the lack of a load balancing during runtime leading to an underutilization of resources and performance bottlenecks. In fact, for every oblivious routing algorithm there is a traffic pattern that causes large load imbalance [23]. Consequently, as it will be discussed in the following of this section, this method although relatively straightforward is contradictory to multiple requirements defined in Section 1.1.1. This is especially problematic in mixed-critical and safety-critical setups as discussed in Chapter 1. The spatial separation forbids sharing of NoC's resources (buffers and link bandwidth) between BE and HRTTs. That is enforced by appropriate mapping and path allocation, e.g., [4, 34]. Although the implementation of this method is relatively simple and provides the requested predictability, it simultaneously decreases both hardware utilization and senders performance in all traffic classes. BE applications are frequently forced to take non-optimal (i.e. longer) routes to avoid interference with HRTTs. NoCs suffer from misbalance in link occupation. For instance, if a HRTT sender does not have any pending transfer its path remains unused even if BE links are heavily loaded, e.g., timing sensitive transmissions from Ethernet port occur rarely when compared to cache traffic of BE tasks [4].

Additionally, strict separation may lead to a segmentation of the NoC and unused/underutilized cores [34]. Moreover, in commercially available NoCs, e.g., Tilera [109] or MPPA [25], nodes are heterogeneous and constructed of processing cores as well as peripherals, e.g., I/O interfaces, memory controllers. As these different hardware units have a fixed position in the system, certain critical communication paths are also fixed and interference between senders with different criticality levels is inevitable (i.e. it cannot be solved by mapping). This endangers temporal predictability of the system and thus its real-time and safety properties. Consequently, the temporal isolation is the only feasible solution in many setups.

Although spatial isolation has significant performance drawbacks it is frequently used in practice due to easy verification. As discussed in [47] or [39], NoC architectures with support for quality-of-service have high production cost.

In [47] an approach based on simplifying the router structure was proposed. As shown experimentaly, by removing the complexity of a baseline router the low-cost router microarchitecture can also approach the ideal latency in on-chip networks. However, as discussed in Section 1.1.1 router simplification may endanger the safety. In [47] the quality of service was implemented through delaying the rate credits that are returned upstream. The evaluation has reported that the proposed architecture can decrease the area by almost 40% and the power consumption by 45% when compared to standard performance oriented solutions.

Such results encouraged research and design efforts for proposing multi-layer networks. These architectures are based on the assumption that under-utilization of the NoC resources can be compensated by simplicity and low costs of router constructions. Consequently, such MPSoCs are supporting several physical NoC layers for spatial isolation.These architectures are popular especially in commercial applications as they simply significantly verification and control of the MPSoC.

The Tile64 NoC [109] from Tilera supports five independent networks which are designed to serve

different traffic classes: 1) user dynamic network (UDN), 2) I/O dynamic network (IDN), 3) static network (STN), 4) memory dynamic network (MDN), and 5) tile dynamic network (TDN). Four of them (1,2,4,5) are realized with the same dynamic router architecture supporting dimension-ordered wormhole-routing. The main goals are to preserve ordering of messages, offer flow control and realiable delivery for short transmissions from BE and safety critical applications (control-oriented). The STN network is designed as circuit switched network for streaming applications. Each streaming sender may set up a route through network and stream directly without interference. This allows achieving separation of bursty from non-bursty traffic.

The MPPA NoC from Kalray [25] offers two layers C-NOC and D-NOC. The first one is designed for worst-case latency sensitive short control messages the second for bandwidth critical data transmissions. The routers in both layers are identical with respect to applied 2D torus topology, and the wormhole switching. The difference appear at device interfaces, and size of router buffers. A similar approach is proposed by [36] where BE and safety critical traffic are separated in two different planes. The layer for HRTTs uses routers supporting the TDM slot allocation done in routers, whereas BE messages are transmitted using a performance optimized router architecture.

Another possible application of spatial isolation is division of the whole NoC into regions, i.e., assuring through predictable routring and appropriate application mapping that traffic from a specific class does not interfere with other traffic, e.g., BE and HRTTs never share links. For instance, KiloNoC [38] architecture isolates shared interconnect resources into one or more dedicated regions called shared regions (SRs). If QoS is necessary within a SR there may be additional hardware support switched on allowing accommodation of senders with. The rest of the network is freed from different temporal requirements. Theses mechanisms are based on prioritization of accesses to virtual channels (buffer queues) in routers. Moreover, there is still the option to switch off QoS in remaining / all regions and by doing so offer maximum performance. Similar approach for Kalray MPPA architecture was followed by [4]. Authors proposed a methodology for deriving mapping allowing containing traffic from certain applications within selected regions i.e. initiated transmissions can reach only selected subset of available nodes and so traffic of different criticalities never interfere.

2.2 Temporal Isolation of Traffic in Networks-on-Chip

Temporal isolation offers the frequently applied solution to the problem of temporal predictability and safety in the majority of NoCs. According to this approach, senders with different requirements w.r.t the interconnect resources are allowed to share links and buffers. Therefore, providing formal timing guarantees for safety-critical traffic in NoCs is usually done through [36]:

- identification of interfering senders,
- temporal synchronization of transmissions competing for shared resources.

In the majority of NoCs, these two actions are considered to be largely orthogonal [23] and therefore separated and designed independently in order to enforce a predictable behavior [69]. The set of interfering senders can be bounded with deterministic or oblivious routing [23] where paths that transmissions may use are selected during the design phase and static during runtime. Synchronization of concurrent accesses in time can be done in routers as well as network interfaces with a

selected temporal arbitration method. There exist two dominant groups of mechanisms for temporal isolation: Time-Division Multiplexing (TDM) or predictable arbitration in routers. The former offers static guarantees which are independent from the behavior of other synchronized senders. The latter offers guarantees which depend on behavior of other senders but this interference can be safely bounded, e.g., prioritization of traffic in routers.

However, the market success of a NoC architecture depends mainly on two factors: on an efficient scheduling methodology that does not sacrifice the performance and on a router/switch design that is competitive in terms of area and power in comparison with a conventional NoC architecture. As we will argue in the following of this section, these goals are largely contradictory and each of the methods results in hard trade-offs between performance and temporal predictability.

2.2.1 TDM-Based Networks-on-Chip Architectures

The first group of mechanisms allowing temporal isolation in a NoC is based on the paradigm of Time Division Multiplexing (TDM). In TDM approach, each sender receives, in a cyclic order, a dedicated time slot for an exclusive access to the NoC [63]. Consequently, TDM-based systems are relatively easy to implement and analyze.

Implementation of TDM in a NoC can be done differently w.r.t link sharing, routing, connection setup, end-to-end flow control and different connection types. An extensive survey of TDM-based NoC architectures is available in [95]. The following concentrates on the most significant ones. The simplest and most straightforward approaches for time-division-multiplexing (TDM) in a NoC utilize circuit switching. In [90] the whole system is considered to be a globally shared resource. Consequently, the NoC architecture provides directed virtual circuits with the same bandwidth for connections between nodes. The establishment of a dedicated communications channel (circuits) through the network is conducted for a predefined amount of time (time slot) and repeated cyclically. A similar approach was proposed by [64]. This NoC architecture offers TDM-based arbitration applied for the usage of virtual channels. Consequently, time slots are assigned based on path and VC pairs. Both works have shown that circuit switched TDM allows to save resources such as buffers due to the fact that once granted access to NoC packets can proceed with full speed as there is no interference.

However, although TDM-based NoC architectures allows easy implementation and providing timing guarantees, they suffer from severe drawbacks. Firstly, the introduced scheduling is static and non work-conserving i.e. the worst-case latencies depend on the number of synchronized senders and duration of their time slots and not their actual activity during runtime. This causes scalability issues i.e. a high number of synchronized senders (even if they use the NoC sporadically) results in long TDM-cycle and pessimistic guarantees. Moreover, if transmissions are not perfectly synchronized with cyclic resource allocation scheme average latencies are very close to the worst case even when the system is not highly loaded [39]. Therefore, in order to achieve high average performance dedicated solutions are necessary, such as an offline generated schedule statically applied to the whole NoC [90]. This is especially visible at the presence of dynamic such as activation jitter or variable resource usage (e.g. length of the transmission). Consider the example depicted in Fig. 2.1 composed of two applications running in a NoC-based MPSoC, arbitrated using TDM. App1 is composed of three tasks (A, B and C) with precedence constraints and App2 is composed of

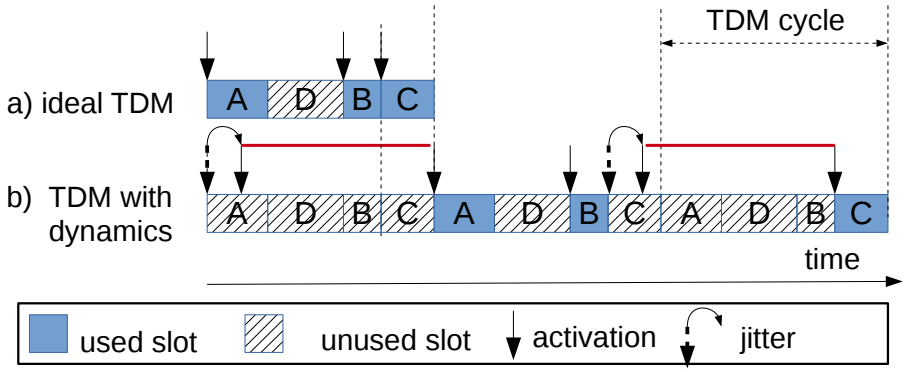


Figure 2.1: Effect of jitter on the total latencies of the applications synchronized with the TDM.

one task (D). If the behavior of the system would be static and predictable, running tasks could be synchronized using TDM cycles, see Fig. 2.1.a), exploiting their periodic activation scheme where the start of an application transmission occurs in a cyclic way whenever it is granted a time slot.

This assumption is non-realistic in majority of setups. The cyclical repetition of the senders behavior must match the cyclical repetition of the TDM cycles. In the case of sporadic (e.g. coprime) repetition rates of activations and/or bursty memory access patterns of the applications, optimization of the TDM-cycle only be possible with extremely long TDM-cycles or with extremely short cycles (the divider of 1), as described in the next section. Therefore, even with strict cyclical activation in a fully predictable environment, full resource utilization cannot be guaranteed in practice due to the nature of underlying workloads.

Moreover, whenever tasks expose dynamics in their behavior, even with a small jitter, transmissions are blocked and their execution is delayed for the duration of a whole TDM cycle, see Fig. 2.1.b). The activation of task A arrives with a small jitter and misses its slot, therefore tasks B and C cannot start their execution before the next cycle. Consequently, task B is granted access in the second cycle but due to the an activation jitter of Task C, a third TDM cycle is required for App1 to complete its execution.

Resulting architectures are not flexible and the worst-case guarantees can be easily endangered by modifications of running applications. Changes in the toolchain, in the distribution of the load on the cores or by modifications in the software (adding/removing tasks) as well as the hardware may cause additional jitter making the TDM-schedule and / or safety guarantees effectively infeasible. Finally, if an application does not have any pending transfer then its time slot is wasted. This prevents the slack bandwidth from being redistributed to improve system's performance. Consequently, TDM-based NoC architectures are moving the main design effort from the NoC arbitration to software/on-core layer which must optimize the behavior of the whole system w.r.t NoC arbitration, thus not visible on the NoC-layer. Using legacy code without large modifications leads to substantial performance decrease [39] resulting in unfulfilled design requirements.

TDM-based Mechanisms with reduced adverse performance impacts

Such drastic decrease in the system performance forces designers to optimize setups e.g. by reducing the number of tasks running on the processing nodes, modify the functionality and the

number of tasks i.e. the move from requirements resulting from the nature of controlled processes to requirements enforced by the platform. As the overhead of TDM depends on the duration of the cycle, i.e. the number of applications and the size of their time-slots, and not on the frequency of their accesses to the interconnect, popular solution is to decrease the slot granularity. Using large slots, for instance adjusted for whole DMA transfers from streaming applications, increases the latency of all senders e.g. [90], [90], [37]. Consequently, architectures with small slots, e.g. single packet / flit has been proposed. This allows to shorten the TDM-cycle and decrease the overhead in case of dynamics. The prominent examples are Aethereal [37], dAElite [95] or Argo [46].

Aethereal [37] architecture supports two physical layers (spatial separation) for accommodation of BE and safety critical traffic. BE traffic can be either transmitted using distributed or source routing with conventional packet-switched router architecture or alternatively may use otherwise unused time slots from real-time senders [39]. BE traffic is also used to set up safety-critical connections, i.e. configure the slot table. This feature allows mitigating the overhead for the non TDM-optimized traffic. However, it is done at the substantial hardware cost - architecture in fact require two separate NoCs with different router architectures. The real-time traffic is routed statically and contention-free using router slot tables. Slots are short and adjusted to single flits. Consequently, it is possible to avoid buffering for time critical traffic class. However, NoC requires complex calculation of a slot assignment allowing transmissions without interference and dynamics cannot be efficiently captured in time-critical applications. Consequently, recent studies suggest that aethereal architecture is not very cost-effective [36]. Next, aelite [36] was proposed as a a lightweight version of aethereal. The main goal was to decrease the hardware overhead of the solution by moving the routing from routers to to the network interfaces. Additionally, support for mesochronous networks has been added i.e. NoCs that have a single clock frequency but asynchronous (not in the same phase) clocks. dAElite [94] is another TDM-based Network-on-Chip with support for short slots which are two words long, and can be further decreased to a single word if necessary. Finally, Argo [46] which combines TDM and GALS/Mesochronous arbitration - the network clock of interfaces may have different phases and routers are asynchronous. Additionally, Argo similarly to aforementioned architectures avoids all buffering and (run time) flow control due to lack of interference in the NoC directly.

The common drawback of architectures based on short length of TDM cycle is that small slots lead to a distribution of longer transmissions over several TDM-cycles, even when the remaining TDM-cycles are not used, which drastically decreases performance. Consequently, transmissions are unnecessarily slowed down even if the NoC is empty. Moreover, too short TDM-slots are undesirable when accessing *stateful* shared resources that benefit from spatial locality, such as DDR memories. In case of short TDM-slots, longer transmissions are distributed between multiple TDM cycles and packets from different senders may freely interleave in the memory controller, potentially leading to undesirable timing effects [35]. Therefore, in order to employ short TDM-based setups in safety-critical systems, the sufficient independence between interfering senders must be enforced using *specialized* predictable memory controllers which, to deal with lack of locality, employ a combination of close-page policy and static bank interleaving. This introduces custom hardware increasing costs and power consumption.

Finally, the scalability problems remain i.e. utilization is low as unused slots cannot be hand over and the length of TDM cycle depends on directly on the number of synchronized applica-

tions. For mitigating these scalability issues, multiplexing of time slots between several channels with independent TDM-schedules was proposed in [39]. The effectiveness of such approaches depends directly on the number of independent channels as well as their utilization and it requires additional arbitration effort increasing costs of the design and verification. If there are few heavily utilized channels the performance improvement will be low. Moreover, they rely on static budgets which leads to the same problems as in case of TDM-slot's granularity.

Other approaches, such as SurfNoc [108], PhaseNoc [80], employ optimized TDM scheduling to minimize the latency overhead. This is performed by replacing the cycle-by-cycle TDM-schedule with more flexible solutions e.g. domain oriented waves. Although this allows to decrease negative side-effects of TDM-arbitration, it does not fully eliminate them, providing only a more optimized solution. Moreover, these solutions require much more complex routers thereby drastically increasing costs of the hardware and power consumption.

2.2.2 Traffic Isolation in the Router

Due to aforementioned limitations, the static non work-conserving arbitration is not a feasible solution in many setups with dynamic workload. The alternative solution to TDM is based on two main components: local arbitration in routers and rate control for transmissions initiated by processing nodes. This well known solution from off-chip networks [112] has been extended and adjusted for the purpose of the on-chip interconnects e.g. [18, 43, 14]. In such NoCs, transmissions must acquire output ports in routers locally and independently as they progress through the interconnect. The router ports are the only point where queuing occurs and can be modeled as a queuing server. The arbitration between transmissions is performed locally and independently within each of the routers.

Contrary to the TDM, these QoS mechanisms do no to prevent the interference between senders but rather tend to safely control its effects. Consequently, the offered service depends for a particular safety-critical transmissions depends on the behavior of other streams. For instance, this may not avoid blocking in the router but by selective prioritization it could ensure that a blocked packet on an output port cannot block a link for other arriving packets, i.e., prevent head-of-line blocking.

In the following sections, existing solutions are briefly discussed considering used priority assignment schemes. The description distinguishes between NoCs in which priorities are assigned offline to the transmissions and therefore static at runtime, see Section 2.2.3, and the real-time NoC in which priorities for a sender may dynamically change during the runtime, see Section 2.2.3. This follows the commonly used classification from related research [17], [63].

2.2.3 Performance Optimized NoCs in the Real-Time Context

The majority of the generic performance optimized NoC architectures apply traffic arbitration done locally and independently in routers. However, their main goal is to deliver possibly low average latencies and high average throughput in the NoC combined with some packaging and cost constraints. These criteria are predominant and drive other design choices e.g. topology, routing, and flow-control.

In such performance optimized NoCs there are no distinction between different traffic classes. All ongoing transmissions compete for output ports (link bandwidth) and virtual channels (buffer

space) and receive equal treatment. Therefore, without appropriate quality-of-service (QoS) mechanisms, resources are not reserved in advance, i.e. packets are switched as soon as they arrive. Moreover, some interference cannot be resolved locally by the router's arbiter and requires input from adjacent neighbours. Therefore, verification is difficult as a single operation may involve multiple routers (independent IPs) as well as multiple arbiters within an router e.g. transmission over several routers on a single path with multistage schedulers. This results in a complex spectrum of direct and indirect interferences between data streams which may endanger the system safety [29], [18].

Although recent works, such as [29], [102], from the real-time analysis domain proved that standard NoCs, even with complex two-staged arbitration (e.g. iSLIP) and virtual channels (VCs), are analyzable allowing to compute the worst-case network latency, this analysis is based on several demanding assumptions. Firstly, reasonably tight guarantees can (typically) be provided only assuming the predictable and known beforehand behavior of all potentially conflicting senders. Unfortunately, this is not the case for majority of the general-purpose applications.

Secondly, all traffic should belong to the same class (e.g. RT and / or safety-critical) or must be spatially separated from other senders. In order to design such mechanism, one must take into account that, in SoCs with performance optimized NoCs, nodes are usually heterogeneous and constructed of processing cores as well as peripherals e.g. I/O interfaces, Ethernet or DRAM controllers. For instance, Tilera Tile64 provides 3 Ethernet interfaces and 4 memory controllers while MPPA provides 8 Ethernet interfaces and 2 memory controllers. These different hardware units have a fixed position in the system. Therefore, when applied in real-time setups, certain critical communication paths are also fixed (i.e. defined by the static routing algorithm), e.g., communication from Ethernet controller to DRAM memory. This frequently makes spatial separation unfeasible (i.e. it is impossible to avoid overlap in communication) as already discussed in Section 2.1. Consequently, when applied in real-time systems, performance optimized NoCs suffer from drastic resource overprovisionining or unfulfilled design requirements.

Static Priority Assignments

The work conserving QoS mechanisms in the domain of real-time NoCs frequently utilize static priority preemptive (SPP) scheduling, see Figure 2.2. According to this scheme transmissions from senders with different requirements w.r.t interconnect, i.e. different criticalities, are mapped statically to different virtual channels in routers (VCs) for assuring functional independence on the data layer. Later, the arbitration is conducted locally and independently in each router. Transmissions mapped to different VCs are preempted. The granularity of the preemption depends on the selected router architecture and switching method. For instance, in commonly deployed NoCs with wormhole switching and virtual channel flow control [23] it can be done on the packet or flit level. The scheduling may optimize different NoC's properties such as latency, buffer sizes and utilization [112].

Exemplary NoC architectures following this principle are QNoC [14], Mango [12] or KiloNoC [38]. The QNoC architecture uses four traffic classes: signaling (for inter-module control signals); real-time (representing delay-constrained bit streams); RD/WR (modeling short data access) and block-transfer (handling large data bursts). Isolation of different traffic classes in routers is done using static priority assignment scheme. Routers use wormhole flow control, i.e. buffer management and arbitration are performed on flits. Packets are build of a head flit, multiple body flits and a tail

flit. Packet routing is resolved upon arrival of the header flit. Resources are released after receiving the tail flit. Packets of different priority are stored in separated buffer queues. Each output port schedules transmission of the flits according to the availability of buffers in the next router, the priority (namely service level) of the pending flits, and the packet based round-robin ordering of input ports awaiting transmission of packets within the same service level. Preemptions, interruptions of currently ongoing transmission of packets with lower priority, are done on the flit level .

Important functional limitation of the QNoC architecture is that it does not distinguish between different requirements of RT applications within the same traffic class. Consequently, if different senders belong to the same traffic class they must compete with each other and the conflicts are resolved using round-robin scheduling locally in routers.

Another implementation of this general principle is offered by MANGO [12] implementing virtual-circuits. MANGO ("Message-passing Asynchronous Network-on-chip providing Guaranteed services over Open core protocol (OCP) interfaces") NoC presents an asynchronous architecture where BE and safety critical traffic are separated in each router. Consequently, each router supports two different arbiters, i.e. one for BE and one for GS, and RT and BE are mapped to the subsets of available VCs. The arbitration for BE traffic is done through simple packet switched transmissions using source routing. The RT traffic is using virtual circuits - reserving sets of buffers in different routers for end-to-end connections. Real-time traffic (e.g. SRT'T or HRT'T) is prioritized over BE. Within the same traffic class (BE or RT) a fair (round-robin) or a non-symmetric arbiter can be applied. The latter uses strict priorities to provide non-symmetric latency guarantees.

Similarly to MANGO, KiloNoC [38] applies separation of RT and BE in different virtual channels. Consequently, for accommodating incoming traffic a baseline router features 25 VCs (therefore 25 priority levels). Each of the VCs in a router has a buffer capable of storing one packet build of four flits. Therefore, each router has 750 buffers belonging to different VCs and 3000 flit slots as it supports 30-ports. To limit this substantial hardware overhead it is possible to switch on and off support for the safety critical applications for selected NoC regions, cf. Section 2.1. If QOS support is off, each router's port serves just one VC per packet class and there are two priority levels (request at low priority and reply at high priority).The required per-port buffering is reduced to 70 flits compared to 100 flits in a QOS enabled router (25 VCs with 4 flits per VC).

As confirmed using formal worst-case analysis methods and simulations, e.g. [18], [43], [14] the prioritization of traffic done locally in routers (switches) allows providing safety guarantees for synchronized applications. However, this method has several drawbacks which limits its applicability. Firstly, the improved work-conserving guarantees offered by this arbitration are possible only under the assumption that packets can be forwarded as they arrive [29], [37] i.e. there is no back pressure and no correlation between routers.To guarantee that, larger buffers must be applied compared to TDM. Otherwise, the propagation of blocking leads to cyclic dependencies and either systems are hardly analyzable or no guarantees can be given. Secondly the scalability of these solutions in a mixed-critical context directly depends on the amount of available hardware resources. If the number of VCs is not equal to the number of criticality levels in the system the formal analysis becomes complex and guarantees can be pessimistic or cannot be given at all as it is necessary to aggregate multiple streams within the same class. Therefore, there is a predominant trade-off between real-time predictability (also safety) and the amount of HW resources used for implementation.

However, the major challenge emerges from the constant increase in the number of applications

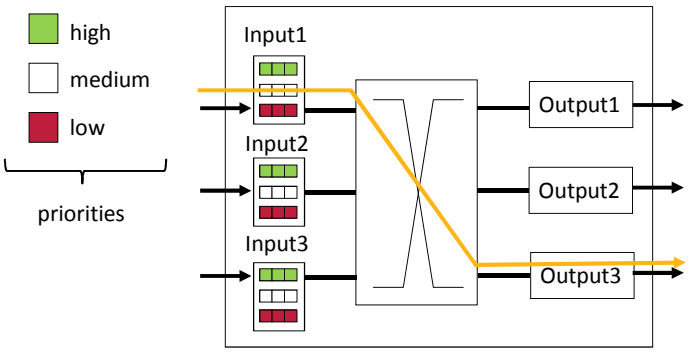


Figure 2.2: NoC router architecture with static priorities assigned to independent buffer queues (virtual channels).

integrated into a single chip such as, the Flight Management System [31] application encompassing multiple tasks with multiple criticality levels. Note, that the number of VCs is independent from the activities of the senders i.e. number and frequency of initiated by them transmissions. This increases additionally the cost of design and power consumption because hardware must be optimized w.r.t to the worst-case behavior of running applications and not their average performance. The worst-case dimensioning is usually overly pessimistic drastically decreasing utilization of the system, as in case of TDM. This effect can be observed in case of KiloNoC. Although theoretically it is capable of handling 25 priority levels in practice it can apply the priority based arbitration to the selected parts of the NoC. This is due to the high hardware overhead resulting from storing streams of different priority in separated VCs and thus buffer queues. Consequently, the overhead is increasing exponentially with number of supported priority levels.

Finally, priority based arbitration is adjusted to the flow-control units (e.g. packets or flits), not to logical transmissions which are frequently composed of multiple packets. This leads to design challenges known from the TDM-based arbitration e.g. locality, in-order delivery, assumed error rate or burstiness which in may setups drastically decrease utilization of peripherals and memories. For instance, as discussed in [56], longer DMA transfers could benefit from locality principle to exploit stateful nature of memory e.g. open bank policy in case of DRAM. Indeed, DRAMs have an internal level of caching, which in standard DDR3 modules amounts to 8kB. Consequently, contiguous and aligned 8kB long transfers fully benefit from the caching. In case of SPP-based routers this property can only be assured for the streams with the highest priority. For all others preemption may happen at any arbitrary moment in time and locality cannot be assured.

Dynamic Priority Assignments

The priority based solutions are frequently foreseen as a solution capable of mitigating the problems of static TDM approaches. However, the SPP-solutions described in the previous section, have a common drawback of treating best effort traffic as a “second-class” citizen. The BE transmissions are preempted as soon as HRRT traffic arrives and therefore suffer from high latency and jitter what is contradictory to their requirements. On the other hand, high criticality traffic is forwarded as soon as it arrives although it has no to little benefit from reduced latency (deadline driven).

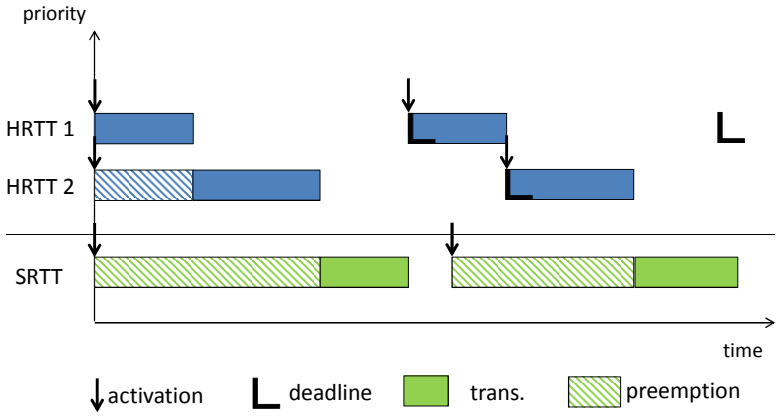


Figure 2.3: Effects of static priority based arbitration done locally in routers on the total latencies of the BE senders in the NoC.

For instance, Figure 2.3 presents a mixed critical system where three applications share the NoC. Two of them are safety critical (HRT'T1 and HRT'T2) and the last one is streaming data (SRT'T). Consequently, each sender has a unique static priority (assigned for instance according to a rate-monotonic scheme) and remaining BEs and SRT'Ts have the same lowest priority. Figure 2.3 illustrates the evolution over time of transmissions in a real- time NoC using a static priority preemptive (SPP) policy, similarly to [14] or [18]. In such systems, SRT'Ts can progress through the NoC only when HRT'Ts are not sending data, therefore they are often unnecessarily delayed. HRT'Ts are scheduled as soon as they arrive, although they usually do not profit from faster execution as long as they are guaranteed to finish before their deadline.

Slack-based resource allocation connected with dynamic priorities is a well known solution from the scheduling theory [63] to this integration challenge, providing high performance for soft-real time applications and BE without violating the timing constraints of hard real-time applications. According to this approach, SRT'Ts are scheduled whenever the execution of HRT'Ts can be safely postponed without causing missed deadlines. This is possible whenever a slack is available, which is the time budget between the worst-case response time of a hard-real time application and its deadline. Only few existing work have considered using slack-based resource allocation in the context of NoCs by applying, locally in routers, dynamic prioritization for different virtual channels.

For instance, "Back Suction" mechanism [28] prioritizes SRT'T and BE traffic over HRT'T. Consequently, the HRT'T is progressing only during idle time when there in no other ongoing transmission as long as the guarantees are not endangered. This is assured by monitoring of the buffer space occupied in routers by HRT'Ts. If buffers are not sufficiently filled, a signal is send to the upstream router resulting in reverse prioritization of the traffic, so HRT'T is prioritized over SRT'T and BE. This signal is propagated upstream towards the source. If the buffers for HRRT traffic are filled again than prioritization is once again reverted.

Another example, is WPMC (Wormhole Protocol for Mixed Criticality) [16] which introduces two modes of work for the NoC. Consequently, different traffic classes are separated in different buffer queues and their behavior is controller before the packets are injected to the NoC. Monitors in NIs observe traffic behavior i.e. message sizes and inter-arrival time. If system adheres to the predefined

behavioral-model, it works in the low-criticality mode in which transmissions of all criticality levels are scheduled based on their typical-case behavior i.e. no differences between different criticalities. When a monitor detects a violation of this behavior, the NoC switches to a high-criticality mode where only streams allocated to VCs of a high criticality are scheduled and all best-effort traffic is dropped by a router to favor the critical packets. This WPMC scheme is extended in [42] for further improving the performance of BEs. The main goal is to allow best-effort traffic to use idling ports of a router even in the critical state instead of dropping it as in [16].

Finally, mechanism proposed in [101] similarly to [42] and [16] prioritizes best-effort traffic over safety-critical traffic in NoC as long as guarantees are not endangered. However, monitoring is done in routers and priority switch is done on demand basing on allowed blocking time (slack). Consequently, it is possible to exploit the latency slack of critical senders, while providing sufficient independence among different criticality levels w.r.t. timing properties. The information about possible slack in contained within the packet header of safety-critical traffic. Later, the slack value is decremented in each router whenever the interference happens. Packets with low slack are values are scheduled before the ones with high slack levels. Additionally, to account for dynamics in the network that typically and blocking along the way as well account for applications (or individual data streams) rather than traffic classes.

The main drawback of all aforementioned mechanisms is the high hardware overhead resulting from a custom router design and the high number of virtual channels (i.e. required buffers) corresponding to the number of priority levels in the system. Moreover, these solutions can significantly increase NoC's complexity as they require to propagate the global state of the NoC to the local arbiters in routers. This is incompatible with the principle of local independent arbitration done in separately routers which requires no correlation between local arbiters and thereby increases the complexity of the worst-case timing analysis. Additionally, the configuration complexity increases. Mechanisms such as [42], [16] or [28] require assignments of particular application to one of existing (supported by NoC architecture) traffic classes. This assignment, when done improperly due lack of well specified traffic behavior or appropriate configuration, can still lead to under utilization of the interconnect as well as decreased performance of BE and SRTTs sender.

2.3 Comparison of Different Mechanisms

In order to discuss the implications of RT requirements on the NoC design, this section presents an evaluation of the selected NoC architectures frequently used for the deployment of real-time workloads. The examination is done in the context of requirements defined in Section 1.3. Its main goal is to provide an insight into design tradeoffs and open research points in this field. The results are presented in Tables 2.1 and 2.2.

Classification

The discussed designs are divided into two groups: general purpose and real-time architectures. NoCs belonging to the former group are performance optimized (all traffic receives equal treatment) whereas the real-time architectures use mechanisms described earlier in this chapter. Moreover, real-time architectures assume that the load from senders is predictable or can be enforced using

rate limiters (RL). RLs define the maximum number of transmissions that a particular sender/processing node can initiate per a given time period i.e. if transmissions arrive too fast there are postponed, cf. Section 1.1.1 for detailed description. The final granular classification is based on flow-control mechanism applied in routers and applied method for assuring temporal predictability. Consequently, following NoC architectures are considered:

- CB-CS - Conventional-Baseline Circuit-Switched is a circuit switched architecture which applies round-robin based scheduling between transmissions
- CB-PS - Conventional-Baseline Packet-Switched is a packet switched architecture which applies single stage round-robin based scheduling between transmissions and single shared buffer queue per port
- CB-PS-VC - Conventional-Baseline Packet-Switched with Virtual Channel is a packet switched architecture which applies multi-stage round-robin based scheduling between transmissions with
  support for multiple virtual channel and thus multiple buffer queues per routers port
- CS-RL Circuit-Switched flow-control and Rate-Limiters in NIs, routers use priority based arbitration including two allocation granularities after which resources must be released: long (logical) transmissions composed of multiple packets, and short transmissions adjusted to the single (or a few) packets
- PS-VC packet switched flow-control and Rate-Limiters in NIs,, support for virtual channels and different router arbiters: round-robin, static priority based and dynamic priority based
- TDM - Temporal Division Multiplexing realized on different slot granularity (short and long slots), including advanced architecture (advn.) optimizing slot allocation to cope with dynamic (e..g. PhaseNoC, SurfNoC)
- Hybrid - frequently referenced architecture with two physical layers: packet switched with round-robin for best-effort traffic and TDM with short slots for RT traffic

Supported Features

The features used for evaluation related directly to the requirements discussion from Section 1. Note, that relevant requirement denotes the principle challenge addressed be the feature, but each feature may simultaneously refer to more than problems in design of real-time NoCs.

- First feature group encompasses, the quality of support for GL and GT traffic understood as conservativeness of the worst-case guarantees. In this case, high relates to guarantees which are not far from the actual performance of the traffic whereas low relates to the significant overprovisioning required for providing the RT support.
- The performance of the BE senders in the mixed-critical scenarios relates to the decrease of the average latencies for this traffic which is resulting from application of safety-critical mechanisms.

- Next, features relate to the dynamic in the sender behavior such as variable activation jitter and variable lengths of transmissions. This allows to assess the additional overhead (delay) which sender himself experiences from the applied real-time mechanisms.
- Later, the influence of this dynamics on other senders in the system is considered. In this context, yes relates to the case when predictable incorporation of dynamics in behavior of one sender may decrease the guarantees of other transmissions through overprovisioning.
- Fairness of arbitration for GL and GT data streams relates to the resource efficient allocation of resources for senders with different real-time requirements e.g. different deadlines, or throughput guarantees.
- Possibility to switch-off QoS and hardware penalty for doing so related to scenario where the real-time NoC should be deployed in the setup with solely general purpose workloads. (safety as a feature of the design not its main goal)
- Support for scalability including to its hardware and temporal overhead relates to the quality of safety resource allocation w.r.t the worst-case guarantees. The real-time NoC should offer an architecture where the guarantees depends on the actual load of the system not other static factors e.g. number of integrated senders (temporal penalty). Consequently, the predictable deployment of many senders with different RT requirements should follow at the low cost (hardware penalty).
- Support for locality relates to setups where packets from transmissions must arrive not only timely but also in the correct arrival order to fulfill the requirements of the peripherals.
- Finally,two last features relate to the complexity and pessimism of the verfication using formal methods, as required by standards for higher safety levels.

NoC Architecture		Real-Time									General Purpose		
Mechanism Type		CS + RL		PS + VC + RL			TDM			Hybrid	CB		
Relevant Requirement	Feature \Arbitration	long trans.	short trans.	static priorities	dynamic priorities	round-robin	long slots	short slots	advn.	TDM+PS	CB-CS	CB-PS	CB-PS-VC
R1	Quality of Support for GL	medium (high avg. latencies depends on trans size)	high (depends on trans. size)	medium (depends on senders prio.; num. of VCs, and number of prios.)	medium (depends on senders prio. and num. of VCs, and number of prios.)	low (depends on rate limiting and num. of senders)	medium (high avg. latencies depends on slot size)	high (depends on slot size)	medium (num. of senders)	high (depends on slot size)	low (depends num.of senders, and msg. size)	low (depends num.of senders)	low (depends num.of senders)
R1	Quality of Support for GT	high	high	medium	medium	medium	high	high	high	high	low	low	low
R6	Performance of BE senders in mixed-critical scenarios	high avg. latencies	medium avg. latencies	high. avg. latencies	low avg. latencies	medium avg. latencies	high avg. latencies	high avg. latencies	high avg. latencies	low avg. latecnies	low avg latecnies	low avg latecnies	low avg latecnies
R2, R3	Sender Penalty for Var. Trans Size	low	low	low	low	low	low	low	low	low	low	low	low
R2, R3	Sender Penalty for Var. Jitter	high	medium	low	low	low	high	medium	low	low	low	low	low
R4, R3	Performance Penalty for others senders in setups with jitter and var. trans	no	no	no	no	no	yes	yes	no	yes	no	no	no
R5, R3	Arbitration Fairness for GL Senders	-	low	low	medium	medium	-	low	medium	low	-	-	-

Table 2.1: Evaluation of different designs for the real-time NoCs w.r.t requirements from Section 1.3, part one.

NoC Architecture		Real-Time									General Purpose		
Mechanism Type		CS + RL		PS + VC + RL			TDM			Hybrid	CB		
Relevant Requirement	Feature \Arbitration	long trans.	short trans.	static priorities	dynamic priorities	round-robin	long slots	short slots	advn.	TDM+PS	CB-CS	CB-PS	CB-PS-VC
R5, R3	Arbitration Fairness for GT Senders	high	high	medium	medium	medium	high	high	high	high	-	-	-
R7	Possibility to switch-off QoS	yes	yes	yes	yes	yes	no	no	no	yes	-	-	-
R7	Hardware overhead in setups with disabled QoS	low	low	medium	high	low	-	-	-	high	none	none	none
R8	Support for Scalability	yes	yes	yes	yes	yes	no	no	yes	yes	yes	yes	yes
R8	Scalability Temporal Penalty	low	low	low	low	low	high	high	medium	medium	low	low	low
R8	Scalability Resources Penalty	low	low	high	high	medium	low	low	high	high	low	low	low
R9	Support for Locality	yes	no	no (yes only for highest prio)	no (yes only for highest prio)	no	yes	no	no	no	no	no	no
R10	Pessimism of formal Verification	medium (low-high)	medium (low)	medium (low)	medium (low)	medium	low	low	low	low	high	high	high
R10	Complexity of Formal Verification	low (high-low)	low (high)	low (high)	high (even higher)	low	low	low	high	low	low	low	low

Table 2.2: Evaluation of different designs for the real-time NoCs w.r.t requirements from Section 1.3, part two.

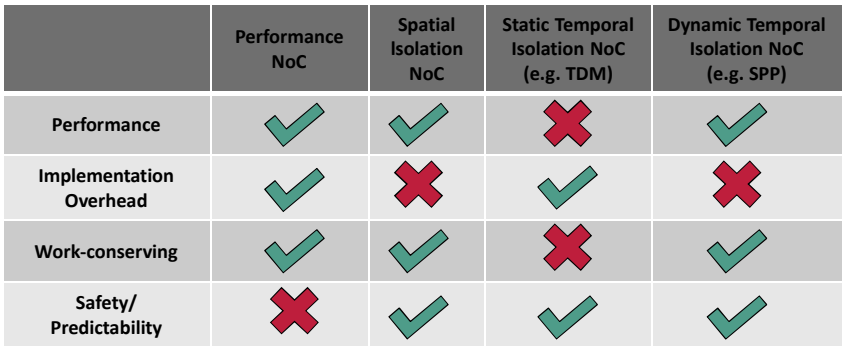


Figure 2.4: Comparison of different NoC architecture with the special focus on safety and performance in the presence of dynamics.

2.4 Conclusions

The conducted detailed survey of mechanisms and architectures can be briefly summarized with Table 2.4. The evaluation has shown that there are hard trade-offs in the design of real-time NoCs (w.r.t performance, implementation overhead, quality of guarantees) and that none of the existing solutions is fully dominating the spectrum of the required solutions.

Commonly used performance oriented NoCs use wormhole-switched architectures with multi-stage arbitration for efficiency and scalability. However, they are not designed to meet temporal requirements but rather to deliver high performance on average. In such networks, ongoing transmissions compete for output ports (link bandwidth) and virtual channels (buffer space). Therefore, without appropriate quality-of-service (QoS) mechanisms, resources are not reserved in advance, i.e. packets are switched as soon as they arrive, and all traffic receive equal treatment. Moreover, some interference cannot be resolved locally by the router's arbiter and requires input from the adjacent neighbours e.g. a joint allocation of the crossbar switch and the router output due to a possible lack of buffers at the output (input- buffered router). This results in a complex spectrum of direct and indirect interferences between data streams which may endanger the system safety [85]. Nevertheless, standard NoC architectures still remain appealing for use since they are affordable, fast and flexible [47].

Isolating traffic in space offers each of the senders a dedicated set of resources. Therefore, it offers maximum performance also at the presence of system dynamics (as this sender is the only one using the given path through the NoC). However, although safe and performant, spatial isolation comes at the substantial price of hardware over-provisioning. Hardware resources are not used whenever applications are not transmitting data what is especially problematic in systems with sporadic workloads and high dynamics. This effects increases the production costs as well as power consumption of the MPSoC.

For mitigating the shortcoming of the spatial isolation, the temporal isolation has been proposed. Two widely established solutions are Time-Division Multiplexing (TDM) (e.g. [37]) and source rate-limiting (e.g. [112]). TDM based architectures consider the network as a single shared resource protected by a global (TDM) arbiter. Each application or transmission is granted, in a cyclic order, a dedicated time slot to have exclusive access to the NoC. The size and number of these slots (i.e. the

TDM schedule) is defined during the system's design phase and optimized for the worst-case behavior.Although TDM offers a relatively simple analysis and implementation, it drastically reduces the system utilization whenever the applications do not behave according to the predefined static scheme used for the schedule. Dealing with transient errors, arrival jitters, changes in communication volume, or conditional executions of synchronized senders, results in average latencies close to the worst-case or overly complex schedules which are hard to calculate and maintain. These effects happen even in lightly loaded systems introducing a major overhead squandering the performance advantage of MPSoCs. Furthermore, the overhead of TDM depends on some static factors i.e., the number of applications and the size of their time-slots, thus is not work-conserving what rises the scalability problems known from the bus-based interconnects. Therefore, although TDM supports independent verification of each communication's runtime behavior, it comes at the significant performance penalty whenever the system exposes dynamics in its behavior.

In contrast to TDM, rate control avoids the drawbacks of a fixed time schedule. This scheme is based on two main components: local arbitration in routers and rate control for transmissions initiated by senders. Hereby, the local arbitration in each router assigns resources independently (link-bandwidth and buffers) to packets traversing the NoC. Rate control is applied in the network interface (NI) of a processing node and limits the maximum number of packets sent per sender. This allows to provide an upper bound on the interference a transmission can experience in the network. Additionally, transmissions can be isolated in dedicated buffer queues in routers (virtual channels - VCs). This enables a priority-based arbitration to further separate senders with different QoS requirements. However, the multi-stage arbitration and resulting transient runtime effects require a complex corner-case analysis, i.e., obtained results are either overly pessimistic or difficult to verify. Additionally, as all the arbiters must be optimized and adjusted to the formally verified worst-case behavior, the performance overhead in highly-dynamic setups is significant. For instance, as a local arbiter cannot control the number of simultaneously transmitting senders, buffers in router's ports must be large enough to accommodate all incoming packets. Otherwise, the packets can be lost (overwritten) or cyclic dependencies between schedulers may endanger the safety (e.g. propagation of blocking). Similarly, the settings of the rate regulators are static during runtime which is contradictory to the required dynamics of the system, i.e., it is not possible to change resource allocations along with the varying system state without loosing safety guarantees. Note that performance penalty increases along with the complexity and inherent dynamics - as in the case of the TDM-based arbitration.

Consequently, the static and dynamic resource allocation schemes, as used in current safety critical systems, are no longer sufficient to provide the needed level of performance without endangering the safety guarantees.

Chapter 3: The QoS Control Layer

This chapter presents a new method for operating a multiprocessor computer system with switched on-chip interconnect, i.e., a Network-on-Chip (NoC). It permits to simultaneously satisfy performance and safety requirements of modern MPSoCs also in the presence of highly dynamic workloads and therefore to overcome the shortcomings of other existing methods.

The proposed solution is based on a global arbitration and synchronization of accesses to shared resources. This is achieved through dynamic adjustments of the admission control locally in network interfaces (NIs) and Quality-of-Service (QoS) arbiters (mechanisms) in routers. The base functionality is delivered through a novel, *global* arbitration layer using key elements of Software Defined Networks (SDN) [60] and adopting them for the requirements of real-time NoCs [52, 56]. The proposed scheme is based on decoupling of the QoS control from traffic arbitration (packet switching and flow control) in routers and separates the NoC in a (virtual) QoS control plane (layer) and a data plane (layer), see Fig 3.1. This is realized using a protocol-based negotiation between senders, i.e., providing a validation method to check if the currently available NoC resources are sufficient before the application is granted physical access to the NoC. This also permits adjustments of Quality-of-Service schemes during the negotiation phase, e.g., different traffic rates depending on the system load. The synchronization can follow through a central scheduling unit, e.g. [52], [56] or directly through broadcast or multicast protocols, e.g., [51]. In both scenarios, the introduced architecture relieves NoC routers from QoS functions, e.g., admission control or maintaining QoS states whether per transmission or aggregate.

The introduced decoupling is appealing in several aspects. The major advantage of the proposed approach comes from the fact that the routers can be oblivious of the QoS functions. Therefore there is no need for custom QoS-oriented NoC extensions which can be costly in terms of area and power e.g. [37], [80]. The introduced solution allows the deployment of a sophisticated contract-based QoS provisioning (e.g. round-robin, priority-based arbitration) without introducing complex and hard to maintain schemes, known from hardware arbiters in real-time routers e.g. [28]. It can even be applied to commercially available, NoC-based architectures, e.g., MPPA, Tile64 in real-time domains. Otherwise, these architectures introduce complex interdependences between senders leading to pessimistic worst-case guarantees or lack of formally assured safety[1] [102], [29]. Furthermore, the arbitration can be done globally using the knowledge about the current state of the system, i.e., which applications are active and which resources are occupied using work-conserving schedulers. Such arbitration allows to efficiently incorporate dynamics in senders' behavior and can offer service guarantees by applying temporal-analysis frameworks, such as Compositional Performance Analysis (CPA) [40] or Network Calculus [62], which allows to capture the dynamics of the system. Additionally, the global arbitration scheme applied in both performance optimized and real-time

[1] For implementation details please refer to Chapter 4

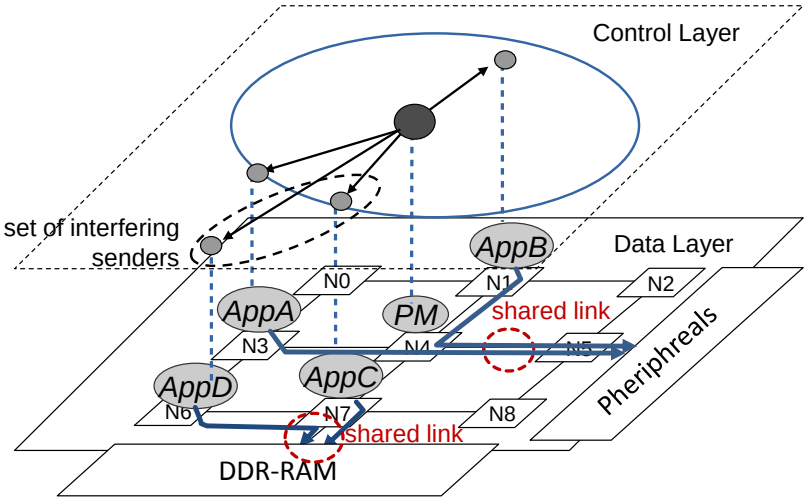


Figure 3.1: Illustration of a QoS control plane and its operations in NoC domain.

architectures allows to improve the utilization of both locally and globally shared resources. For instance, the designer may isolate full transmissions, constructed of multiple packets, for assuring locality of accesses to shared memories or further efficiency improvement. It is also possible to provide feedback about the state of the NoC for effective preemption based on-core schedulers. Consequently, the proposed control layer offers joint benefits integrating features from standard NoCs and real-time NoCs on top of an existing infrastructure. Such hybrid approach offers easy implementation, high flexibility and efficient guarantees even in systems with dynamic workloads.

However, apart from aforementioned benefits, this novel QoS provisioning introduces simultaneously a set of new challenges which must be considered during the system design phase. The protocol based synchronization is done at the cost of additional latency, i.e., temporal overhead. Consequently, the scalability of the architecture and its ability to manage a high number of requestors and many complex scenarios plays a critical role for future industrial deployments. The following sections will discuss a set of mechanisms that address these challenges.

The rest of the chapter is structured as follows: Section 3.2 gives an overview of the related work regarding Software Defined Networks and their applications to Networks-On-Chip as well as safety critical systems. Section 3.3 introduces main concepts of the control layer, followed by an overview of the centralized architecture in Section 3.4. Figure 3.2 presents an overview of the remaining content. In Section 3.6 different resource allocation policies are described which can be achieved with the control layer. Section 3.7 presents the interface to the on-core schedulers offered by the mechanism whereas Section 3.8 interface to memory and peripherals. Finally, Section 3.9 provides a summary.

3.1 Baseline NoC Architecture

The proposed solution is built on-top of an existing NoC architecture. Consequently, it is largely orthogonal to the underlying interconnect. However, as there exist a wide selection of possible solutions (see Chapter 2), in this section we present the underlying NoC architecture (router and NI) used throughput the remainder of this document. Focusing on this architecture simplifies the un-

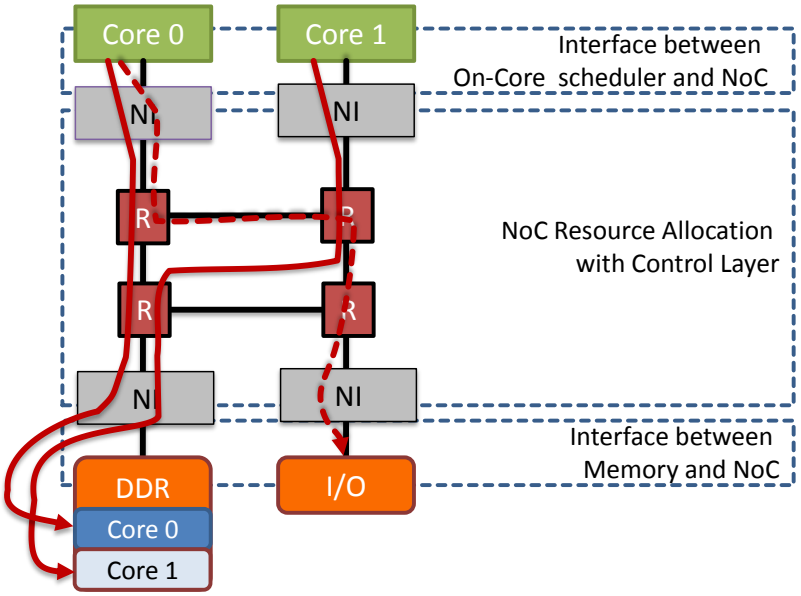


Figure 3.2: Applications of the control layer in the NoC based in the MPSoC.

derstanding of the control layer and its principles of work. However, the description will highlight all key features and requirements which must be considered in general for the application in other NoC architectures. The main goal is to build a solid theoretical basis using the in-depth discussion with the selected NoC example and, without loosing generality, allowing fairly straightforward porting to other architectures.

3.1.1 Router

The baseline router architecture is largely based on the generic architecture proposed by Dally [23], which is frequently cited and applied in related work. Figure 3.3 presents its block diagram. The datapath of the router, is responsible for storing and forwarding the data and is built of the following modules: *input ports* with *VC buffers*, a *switch* and a set of *output ports*. The remaining modules belong to control functionality and are responsible for allocating available resources (i.e. buffers and links / output ports) to concurrent transmissions.

Router arbitration supports prioritized virtual-channels (VCs) (cf. [14]) and wormhole switching (cf. [23]) where packets are decomposed into flits which constitute the granularity of transmission and arbitration. Each packet is constructed of one header flit (including address information), variable number of body flits, and tail flit. An arriving header flit is stored into the appropriate FIFO queue (depending on the priority resulting from statically assigned VC) and triggers the arbitration. Forwarding of the tail flit releases the resources.

Each virtual channel has an assigned fixed static priority. Buffering happens in input ports and buffers for different virtual channels (priorities) may differ in length. The routers schedule transmissions according to:

- the availability of buffers (for a given priority) in the next router (flow control / backpressure signals),

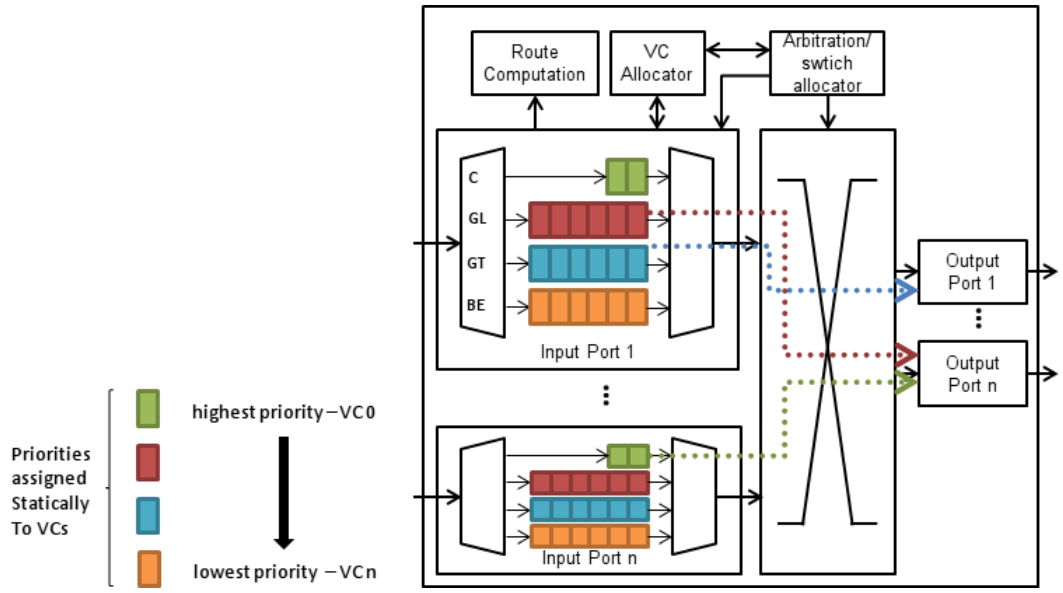


Figure 3.3: The exemplary baseline architecture for the NoC .

- the priority of the particular VC, and
- the packet-based round-robin between the input ports for packets using the same VC.

The transmission of a packet is preempted at the flit level as soon as a new packet with higher-priority arrives to the VC. Later, transmission can resume after all higher priority packets are served. We assume a deadlock-free, deterministic, source routing policy.

3.1.2 Network Interface

For assuring preditcable access patterns of integrated senders, network interfaces use rate control [106], [112], [26]. It is a commonly applied safety measure in many real-time setups to safely bound the number of initiated transmissions in NoC (accesses) as well as their length. According to this method, at each source node, a monitor - **rate limiter (RL)** - regulates the traffic which can be injected usually through two parameters: window length (T_w) and bandwidth quota (N_{max}).

At each cycle the rate limiter compares the length of a pending packet to the available bandwidth quota plus the amount of data (e.g. packets or flits) sent during the previous period equal to the window length (in cycles or ms). If not greater, the packet can be injected to the NoC otherwise it must wait until the budget will replenish. Figure 3.4 presents an example illustrating the principle of this flow regulation at the source. X axis depicts the trace of accesses from a node to the NoC whereas Y axis depicts the accumulated number of finished transmissions. In the figure, within each time window (T_w) there may be only three ($N_{max} = 3$) conducted transmissions. A second burst of three activations arrives too early, is postponed, and may be executed only after the first window finishes. Recent research, e.g. [75], has also shown that fine granular representation of activation patterns through minimum distance functions is possible. The fine grained rate control offers better latency compared to classical periodic (coarse grained) monitors and construction with user-specified constant overhead (in terms of necessary processing time/power and memory).

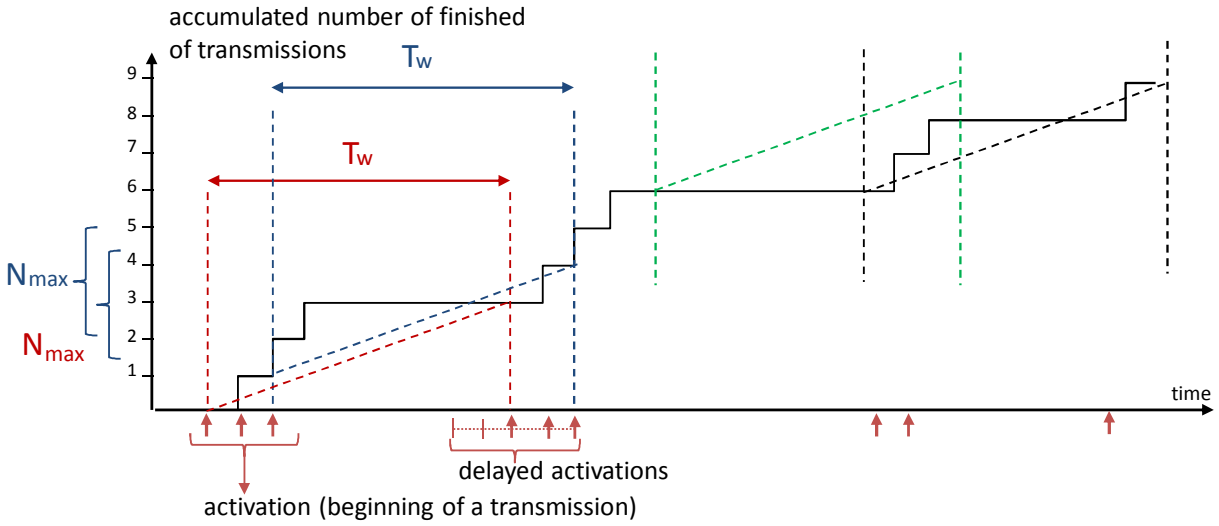


Figure 3.4: Rate control for providing safety guarantees in real-time NoCs.

Consequently, extending Network Interface with monitors using rate control is a well known solution for supporting quality-of-service in NoCs e.g. [107], [96], [13], [29]. This method owns its popularity to relatively easy implementation (no need of custom router architecture) as well as flexibility (may cover different activation patterns, no need of "predictable execution models"). Therefore it is largely suitable for BE traffic from processors with caches and as such already available in many NoCs on the market, e.g., MPPA from Kalray [25], FlexNoC from Arteris.

However, this solution has also a drawback - it is trading performance for temporal guarantees, i.e., performance penalty. The rates are adjusted statically according to the worst-case scenario during design or boot time, i.e., assuming that all senders are running simultaneously with maximum possible transmission arrival rates. Therefore, this approach is not work conserving and sacrifices the interconnect utilization whenever senders expose dynamics in the execution time, release jitter or communication volume and the system is not highly loaded, i.e., whenever senders expose dynamics number of activations, execution time.

Additionally, in more complex setups (e.g. many senders per node, mixed-criticality) a NI may support address translation. The working is similar to the functionality of MMU/MPU units[2], see Figure 3.5. Consequently, the mapping of the local physical address within a tile (processing node) is done with the usage of address translation tables. Each record in such a table may include the route, which a transmission should take, access rights at the target (Read or Write), and the physical address at the remote node.

Such mechanism has several important benefits from the safety perspective. Firstly, it allows to isolate the selected parts of NoC, what is especially important for spatial isolation of traffic and mixed-critical setups. Secondly, it allows to separate address spaces for different tiles for providing backwards compatibility and higher flexibility of integrated software. Finally, allowing overlapping address spaces can be used for accommodation of the core-to-core communication.

[2]Note, that functioning of such unit in the NI is orthogonal to the work of a MMU/MPU units running in the tile.

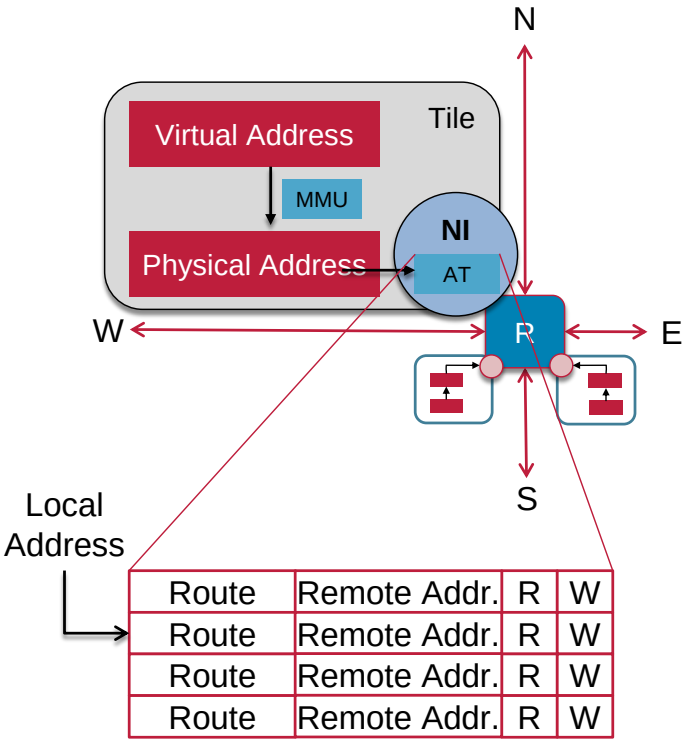


Figure 3.5: Address Translation in NI for providing safety guarantees in real-time NoCs.

3.1.3 Traffic Characteristic

This section discusses the typical NoC traffic patterns resulting from the design constrains in the real-time and safety-critical domains. Consequently, for achieving temporal predictability and shorter worst-case execution/response times (which can be proven with formal methods), designers distinguish between two traffic groups: short and long transmissions. The former group consists of memory accesses, typically cache-lines, from applications compiled with regular tool chains. They originate from both, real-time and best-effort, traffic classes with a typical metric of worst-case and average latency of the single packet. The latter group contains messages composed from several packets which are usually originating from safety cirtical and streaming applications or optimized to run on a specific hardware (*control-oriented* real-time applications). The typical criterion is worst-case or average throughput for HRTTs and SRTTs appropriately. In the following of this section, we describe both groups in detail.

Long Transfers and Timing-Critical Traffic

The main goal for systems with hard real-time requirements is to decouple on-core interference from scheduling of accesses to shared resources, e.g., shared main DDR-SDRAM memory. This allows easier verification through fairly straightforward analysis of the temporal behavior. Designers of applications in real-time domains, such as automotive and avionic, frequently achieve this goal through predictable on-core execution models (e.g. [77]). Figure 3.6 presents a “classic" example of this approach - “read in the beginning and write at the end” which is applied in both avionic and

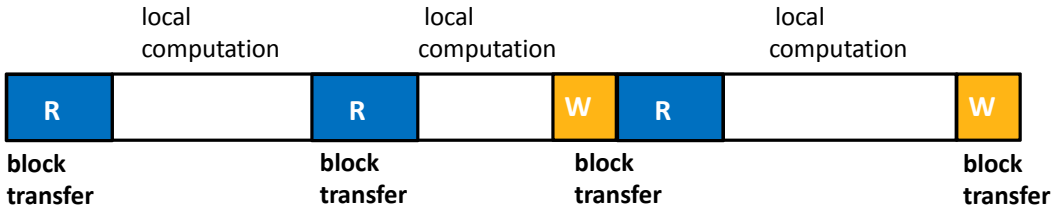


Figure 3.6: Model enforcing predictable on-core execution in safety critical domains [33].

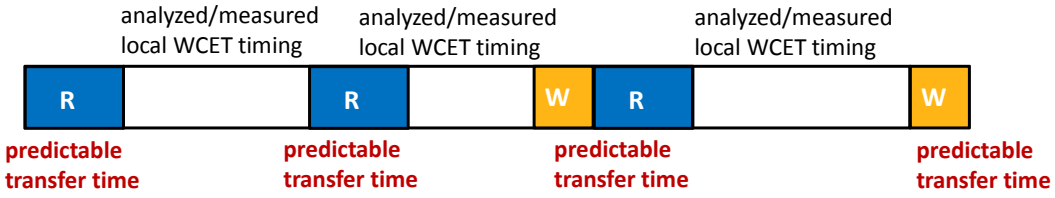


Figure 3.7: Predictable timing through idependent reservation of NoC and on-core resources [33].

automotive domains [76, 78]. According to it, the on-core execution of an application can be divided into three phases: memory read block (data *Acquisition*), actual execution (*Execution*) and memory write block (*Restitution*). This requires decoupling execution from communication performed early during software design phase. As a consequence, a white-box approach is applied which imposes an execution model such as the discussed predictable execution model [77], [89].

During the first phase a task or application accesses the external memory for receiving data and all necessary code. Later, during execution phase, the application/task can proceed sporadically accessing the shared resource to ensure data consistency. Finally, it may write the results at the end for further usage or integration by other system components.

Note however, that frequently not all data can be loaded efficiently. Tasks must still conduct these ”sporadic”accesses to main memory (e.g. loading records from database in the case of drive management). Therefore, the underlying NoC architecture should always be capable of supporting combination of long and short accesses, which leads to the classes described above.

This analysis of temporal behavior for verification is presented on Figure 3.7. The problems of on-core and off-core resource reservations are decoupled, i.e., are orthogonal. The worst-case timing can be obtained by measuring/summing up two factors: local (on-core) worst-case execution time and accesses to shared resources, e.g., the NoC. Therefore, the designer must prove through measurements and analysis that there exist an upper bound on the transfer latency. Efficiency of this evaluation can be supported by design, e.g., on-core or peripherals schedulers and architecture. The next sections discuss these choices for NoCs in detail.

The second group of applications requiring throughput guarantees (soft- and hard-) are signal-processing and streaming applications. As discussed in [23] requirements of such senders are frequently directly related to user perception, e.g., video and audio codecs, or resulting from certain standards like DVB, DAB, TSN, AVB.

Consequently, timing-critical traffic from both hard and soft real-time senders exhibits frequently predictable access patterns, e.g., periodic with jitter. Moreover, the transmissions are longer and

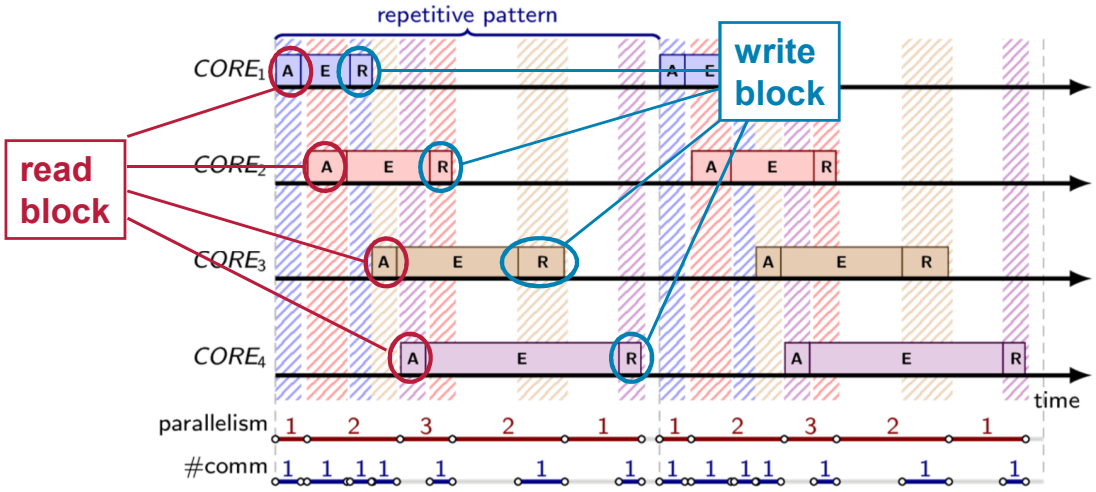


Figure 3.8: Static repetition in multicore, results from CERTAINTY project [20].

consists of multiple packets, e.g., video frame transmission for SRTT or memory read in predictable execution model. This promotes the usage of DMA engines whenever possible, e.g., MPPA architecture from Kalray. Of course, efficiency of such approaches is limited by the size of local memory (which is a critical factor in on-chip design) as well as communication cost resulting from interference with other senders using the NoC.

Here, the huge benefit is the predictable NoC access pattern which is highly repetitive and regular. In many setups, this regularity allows to improve utilization and performance of resources. Figure 3.8 presents an example from CERTAINTY project [20], where tasks are activated with an offset what allows pipelining of DMA transfers. Moreover, some peripherals, such as DDR-SDRAM memories, are state dependent, i.e., their response time depends on the history of previous accesses. Therefore, clustering of transmissions combined with a known memory controller behavior allows to exploit this behavior, i.e., assure a known access sequence, for decreasing response latency as well as improving power usage and utilization.

Short Transfers and Best-Effort Traffic

Short messages (e.g. single packet or flits) are usually originating from control-oriented safety-critical senders as well as best effort applications running on processors with caches. They are often very sensitive to latency, i.e., the reaction time is critical for worst-case or average performance [23, 28]. Cache miss increases the Worst-Case Execution Time of a task running on the processor and thus the load of a processor. Since it is only possible to conservatively limit the number of misses (see overview paper [110]) the miss-times must be included in the WCETs several times. Therefore, they conservatively increase the worst-case load. Furthermore, the characteristic of traffic from general-purpose, best-effort applications is usually not known, what may endanger the deadlines (i.e. temporal properties) and as consequence also functional safety in safety-critical domains. If aggregated traffic demands exceed, even for a short period of time, an available bandwidth (e.g. due to unpredictable bursts from cache-based execution or coherency protocol) it may endanger safety

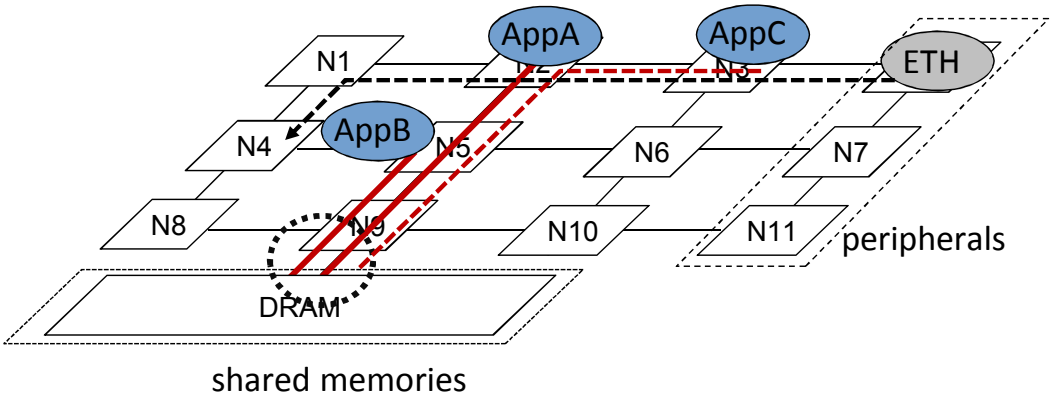


Figure 3.9: Exemplary scenario where propagation of blocking could endanger deadlines and as consequence also functional safety in NoC through the effects of buffer saturation and backpreassure.

of other applications using NoC resources.

Figure 3.9 presents an exemplary scenario in which three applications (AppA-AppC) sharing the same DRR-SDRAM memory share the interconnect resources with traffic from an Ethernet controller (ETH). Consequently, if accumulated traffic from AppA and AppB exceeds the capacity of memory controller or available links bandwidth it postpones also AppC. As AppC shares links with ETH, ETH can also be blocked what causes a "domino effect". Therefore, ETH becomes dependent of both AppA and AppB although these senders do not directly share resources. This propagation of blocking [106], [23] (also known as saturation tree or head-of-line blocking) can quickly and drastically decrease performance of the whole network leading to pessimistic guarantees or unfulfilled design requirements. Formation of a saturation tree can be additionally accelerated by other architectural features. For instance in case of wormhole switching, buffers in routers are smaller than a packet, such that a blocked packet can block several senders in multiple routers on the same path. Additionally some modules may occasionally operate at a slower speed, e.g., a variable speed coder, encoder or storage device.

The popular and frequently applied method to solve these problems is traffic shaping using rate limiters, which is discussed in detail in Section 3.1.2.

3.2 Software Defined Networking

The main concept of the proposed resource manager and control layer is derived from the global management scheme for off-chip interconnects - Software Defined Networking (SDN). This section briefly revises and summarizes related research based on [98] and [60]. Key concepts of SDNs rely on division and separation of interconnect mechanisms into control and data planes. The data plane is responsible for actual forwarding of transmitted data, e.g., packets or frames. The control plane controls, through protocol based synchronization, behavior of routers and adjusts it depending on the global state of the system.

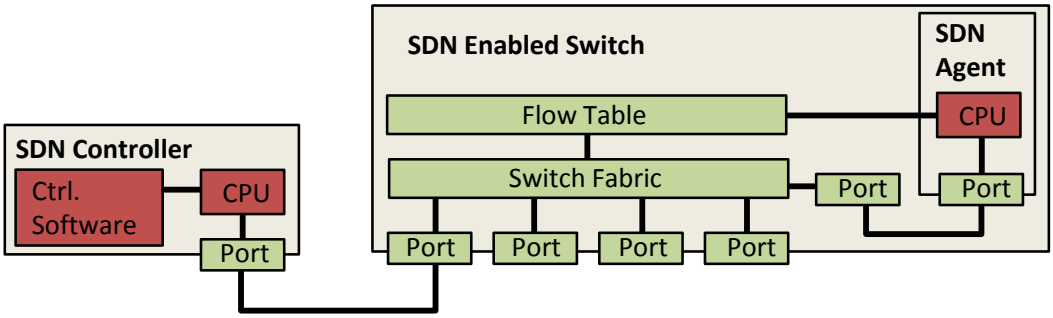


Figure 3.10: Main SDN components in a switch [98].

3.2.1 SDN Principles and Real-Time Systems

The exemplary SDN scheme, taken from the Ethernet domain, is presented in Figure 3.10. Components belonging to the data plane are marked with green whereas the control plane is marked with red.

Traditionally in a SDN the control layer is build of a centralized SDN controller and agents in the network switches (red components in Figure 3.10). The data plane encompasses remaining functionality of the switch (green components in the figure). The role of the SDN agent within a switch is to monitor the communication passing the switch, synchronize it with the SDN controller and, if necessary, update/modify the rules used for traffic arbitration. For doing that it uses a flow table inside the switch which stores settings for forwarding operations performed by the switch's data plane. For instance, each incoming traffic stream (frame or packet) can be identified by source and destination MAC addresses and/or port numbers. Consequently, the flow table may contain settings deciding about the output port to which it should be forwarded.

The SDN controller is usually implemented in form of a software library (also referred to as network operating system) running on a dedicated core. The communication between SDN agents and the controller follows in the same network in which communication is done. *SDN interface* define the communication protocol used for synchronization between the SDN controller and the switches/routers.

A plethora of mechanisms supporting SDN exists for off-chip networks with a predominant focus on Ethernet. A comprehensive overview of SDNs, including standardization, interfaces, controllers, network programming languages, and applications is given in [60]. The major research focus is placed on optimization with respect to average system performance. Consequently, the schemes are either hardly analyzable or no guarantees can be given. For instance, the most prominent SDN interface OpenFlow [71] relies on TCP-based communication. Therefore, it suffers from complex handshake and flow control schemes which significantly increase the worst-case latency of real-time traffic [98] which in case of Ethernet is typically UDP-based.Nevertheless, SDN Ethernet protocols have been analyzed in [8] and [98].

The challenges in porting of the SDN principles to the real-time and/or safety critical domains, such as automotive and avionic, will be discussed using an exemplary topology from the automotive domain which is presented in Figure 3.11. In this context, the role of the SDN controller would be to manage and control arbitration settings (e.g. routing or rate control) in switches depending on

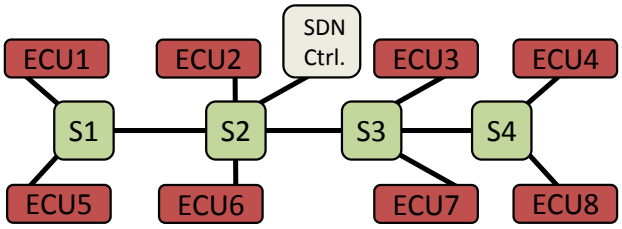


Figure 3.11: An exemplary topology from the domain of automotive Ethernet enhanced with a SDN controller.

information about global state of the entire network, i.e., who is sending and which resources are occupied. The re-configuration includes admission control and run-time reconfiguration. The former allows limiting the link load and therefore network congestion by limiting accesses (blocking or rate control) for selected senders. The latter can be used for recovery from faults, e.g., through bypassing a failed switch or link.

A centralized single point of synchronization through the SDN controller could be a performance bottleneck. However, it has also significant advantages especially in the safety critical domains. Firstly, it allows to simplify synchronization protocol, e.g., (rapid) spanning tree protocol or shortest path bridging. On the contrary, in case of decentralized system a lengthy and complex protocols for assuring system coherence are frequently needed. It is so because the communication participants must firstly agree on a state of the network before they can take actions. In case of a single SDN controller all actions are serialized in this node what simplifies the process of worst-case verification.

Application of a SDN in safety critical system requires providing strict timing requirements. Consequently, the time necessary for admission control and re-configuration including actions of the SDN controller and agents must be bounded in time (worst-case behavior). Verification in such real-time systems is frequently done with formal worst-case analysis methods and/or simulation-based approaches. The disadvantage of the latter is lack of guarantee that the simulation exposed included all corner cases leading to worst-case behavior. On the other hand, the former gives safe upper bounds on a system's worst-case behavior which can be however slightly pessimistic (includes scenarios which wont happen in practice).

In [8] a formal analysis of the communication in a SDN utilizing OpenFlow protocol has been proposed using the network calculus framework. This work models the behavior of an SDN switch in terms of delay and queue length boundaries. Next it provides the analysis of the buffer length of SDN controller and SDN switch by modeling them as single resources (network calculus servers). The work has proved that indeed it is possible to provide a formal upper bound on the transmission latencies in a SDN network.

In [98] authors proposed an alternative SDN analysis focusing on automotive Ethernet using the Compositional Performance Analysis framework. Switches have been modeled considering their internal structure (ports, multiple traffic queues etc.). Moreover, for both, SDN controller and SDN agents, limited amount of processing resources (CPUs) has been assumed. Moreover, in contrast to [8], for the worst-case timing derivation also latency of network re-configuration has been considered. This refer to temporal delay required by the SDN controller to reconfigure the network by distributing new forwarding rules to individual switches and wait for their acknowledgment.

Finally, overhead of introducing SDN has been evaluated, i.e., impact of SDN traffic on non-SDN traffic.

Finally, some related research, e.g. [66], [111] considered using queueing theory for performance evaluation. These methods, although useful for identifying bottlenecks, offer only probabilistic performance guarantees given by queuing theory. Therefore, they are not capable of deriving worst-case metrics required in real-time safety critical domains.

At the moment of writing there is no related research considering application of SDN in Networks-On-Chip considering real-time objectives.

3.2.2 SDN mechanisms in NoCs

Existing NoC architectures, implementing the concept of SDN, concentrate on the reconfiguration of independent NoC switches to optimize the average performance by adjusting the scheduling arbitration during the runtime. Consequently, they offer no guarantees for the worst-case behavior. In the following, the most significant contributions are described in detail.

In [22] authors present a software defined on-chip network (SDNoC) in which control logic is software controlled, and applications are able to configure the network for improving the average latencies. SDN agents in the proposed architecture of SDNoC are embedded within routers and used to exchange control messages. Consequently, routers are significantly more complex than the baseline implementation. The control capabilities include routing and link load. The former can be used for application of fail-over mechanisms. These allow switching to a redundant or standby hardware component upon the failure. SDNoC has no central management unit. Resource allocation is done directly by sending applications by adjusting packets headers. However, this limits the applicability in hard real-time domains, for instance no solutions for conflict resolving between requests from different senders exists. In [87] authors present an evaluation of the SDNoC architecture. - a NoC which is directly applying SDN principles for off-chip to on-chip interconnect. Consequently, switches are enhanced with agents and configuration tables which decides about routing whereas the controller is running on a selected network node. This however leads to significant hardware overhead as router complexity increases with the number of senders and synchronization scenarios. Finally, in [11] SDNoC is realized in form of a hybrid hardware/software approach. Firstly, there is spatial isolation between control network and data network (physically separated). Next, SDNoC is controlled by a centralized network manager (NM) which is running on a selected NoC node in form of software library. Usage of the SDNoC allows to take advantage of the global state for reduced buffering in the data layer.

There are significant differences between the solution which is proposed in this work and aforementioned approaches. The majority of the related work applies a straightforward approach for porting the off-chip SDN ideas and mechanisms to on-chip interconnect. Consequently, the SDN-controller adjusts the settings of routers as it is done in the off-chip networks. This allows authors to directly port the mechanisms but disregards the specifics of the on-chip interconnects. As discussed in [47] and [23] the constrains imposed on on-chip interconnect differ significantly from off-chip networks. As a result, direct adoption of the principles from conventional SDN off-chip network switch microarchitecture results in an overly complex design. Such NoCs increase not only production costs (in terms of area and power) but even more importantly latencies of traffic

(typically memory accesses). This results in significant performance drops, and therefore simpler approach to on-chip networks is frequently considered, e.g., [47], [23], [37].

Consequently, contrary to the related architectures, the control layer proposed in this work controls traffic at the source, i.e., checks the availability of the interconnect resources before the sender can obtain physical access to the interconnect. Therefore, no router modifications are necessary and agents can be implemented as a straightforward extension of the rate control mechanism available in many architectures. For implementation details please refer to Chapter 4. Moreover, the described solution is from the beginning focused on providing worst-case guarantees for synchronized senders. Currently, in the context of NoCs, client - server admission control schemes are only applied to improve average latencies of synchronized transmissions by limiting the access rates to the frequently requested peripherals such as DDR-SDRAM memories, e.g., [106], [50].

3.3 The Main Concepts of the Control Layer

In real-time systems using NoCs, synchronization between interfering transmissions can help to run the network with significantly less resources and still be able to guarantee temporal properties of the system and, whenever required, its safety. This includes handling the effects of both backpressure and head-of-line blocking in routers, where switch arbitration between packets/flits is performed [50], [106]. In order to provide service guarantees, e.g., an upper bound on the worst-case latencies or guaranteed throughput, the architecture must allow bounding direct and indirect interference between *interfering transmissions* - transmissions which overlap in at least one router on their path from source to destination and therefore are sharing NoC resources, i.e., link bandwidth and/or buffers in routers. For doing so, applications are grouped in *synchronization scenarios*, i.e., sets of senders which may mutually influence their execution times, e.g., through concurrent accesses to shared interconnect resources. Assuring temporal guarantees for synchronization scenarios, i.e., offering a *predictable NoC*, requires reserving enough resources so that traffic requirements from all senders are meet, cf. Chapter. 1.

For achieving this goal, it is required to start with decoupling of the admission control / QoS configuration from system execution, using key principles of Software Defined Networks. This is reflected by the architectural measures introduced by the proposed *control layer*. The existing NoC is treated as a *data layer* (communication domain) responsible for switching of particular packets. The *control layer* (model domain) responsible for a safe *admission control* is build on top of it. Admission control is understood as a validation process performed at runtime before a communication is established (or its requirements are allowed to be changed) to see if current resources are sufficient for the particular transmission. Consequently, the main functionality of the control layer encompasses model-based analysis methods which are used to establish the settings for monitoring and isolation mechanisms. Both layers, the control and data layer, are coupled with a protocol based synchronization, i.e., contracts, allowing *resource reservations* for senders.

In case of a switch-based interconnect, these reservations must include all routers on the end-to-end path through the NoC which often have to encompass more than one network resource. Consequently, for each interfering transmission one may define a *virtual resource* - set of real resources belonging to different network components which must be selected and allocated to satisfy its requirements. This is accomplished through *connections* managed by the introduced control

layer, i.e., contract-based end-to-end resource reservations for a given sender-receiver pair on the selected path with a selected QoS provisioning. The QoS provisioning defines the minimal set of requested parameters of connections (reservations), namely at least the type and time response depending on the particular synchronization scenario. In case of interfering transmissions, the latter translates to the volume of data to be transmitted (e.g. MBs or GBs) or rate settings for traffic shapers (rate limiters).

Connections for each synchronized sender must be negotiated and established dynamically at runtime. The negotiation is based on the global state of the system, i.e., *coordinated reservations*. The *global state of the system* is defined by the number of currently running applications and their current requirements w.r.t. shared resources (Network-on-Chip). Note, that both aforementioned factors can change during runtime depending on the dynamics of the system itself as well as the physical environment in which system is used, e.g., situation on the road, cf. Chapter 1. An exemplary design is shown in Fig. 3.12 whereas the control layer may be build from multiple *analysis engines* responsible for different aspects of resource allocation and admission control. The analysis can be done for optimizing different systems aspects of system's work (e.g. safety, security or performance) and provides an initial input to the model exploration module. Moreover, the analysis engines can assign resources according to pre-defined and static allocation schemes (e.g. Time-Division Multiplexing) as well as dyncamic and work conserving schedulers (e.g. round-robin, priority based policies or even dynamic priority assignments). The *model exploration* component is optional and used to perform design-space exploration within the system model to find a feasible solution whenever the requested contract can not be established as well to adjust system to unexpected events, e.g., robustness against faults of the components. Consequently, it introduces to a design, if necessary, full adaptivity.

As the main goal of this work is to introduce a deterministic response time for the programs executed on the computer system, the main focus lies on the predictable resource allocation schemes that are defined during the system design phase. Consequently, the main goals of the arbitration enforced by the control layer for NoCs can be defined as follows:

- avoiding contention in (NoC) buffers,
- dividing bandwidth between interfering senders according to their requirements,
- adjusting the settings during the runtime to cope with dynamics in the behavior.

The following details on how to safely adjust these models during runtime reacting to the changing global state of the system. In the majority of existing NoCs, *predictable resource reservations* are controlled trough data layer introducing:

- admission control done locally in nodes, e.g., traffic limiters, performance counters,
- arbiters deciding about allocation of resources to streams in routers,
- mixed-solution combining both aforementioned methods.

Summarizing, the behavioral model defined by connections for a particular sender on a particular node is negotiated with the control layer and later enforced with the data layer using aforementioned mechanisms for predictable resource reservations.

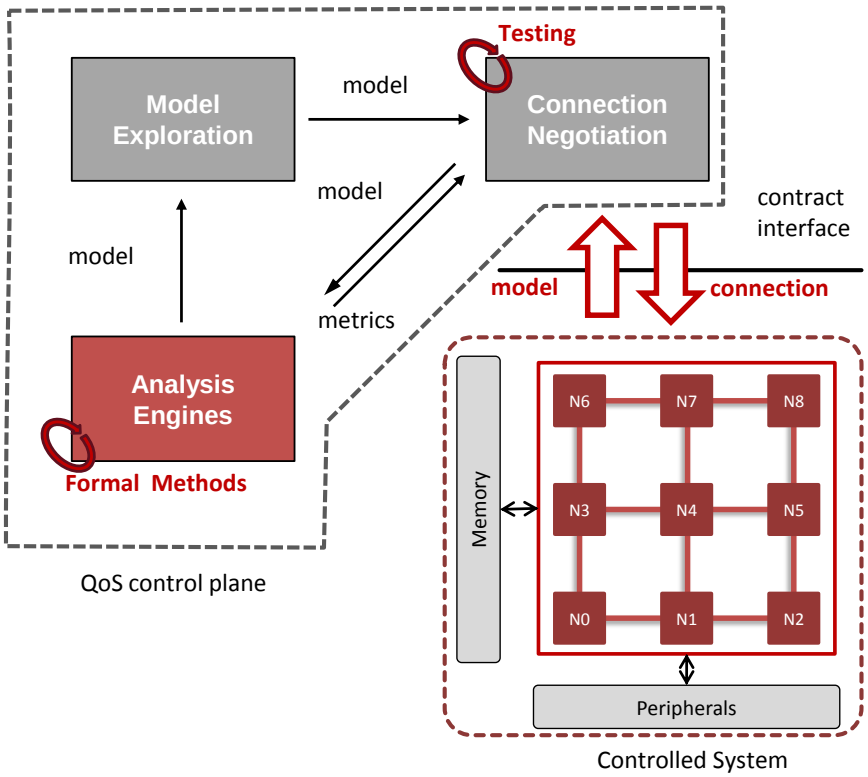


Figure 3.12: Model domain architecture for adaptive arbitration in a NoC.

Note, that the proposed scheme, based on decoupling of admission control from system execution, opens multiple implementation possibilities. Clearly, the data layer encompassing independent, local resource schedulers must be implemented on each resource in the system which can be accessed by more than one communication participant (application / processing node / tile). However, for the control layer there are several negotiation schemes possible since it is orthogonal to the underlying system and may be implemented independently.

3.4 Overview of the Architecture

The dynamic QoS requires safe modification of resource allocation at runtime for adjusting the QoS settings of the NoC to the changing, global state of the system. This requires the consideration of the local state of the core (number and execution profile of currently active applications defining core's requirements w.r.t NoC) as well as to the global state of the NoC (number and profiles of active interfering senders defining possible interference from interconnect). Although evaluation of the former "on-core" factors can be done locally on the same node, information about the latter "off-core" factors requires further, global synchronization done by the QoS control plane. The essence of the centralized implementation of the QoS control plane is a resource management (RM) unit and clients.

Figure 3.13 presents an overview of the architecture which, from technical perspective, introduces *connection oriented network services* [97] adjusted for providing a predictable and safe behavior. In order

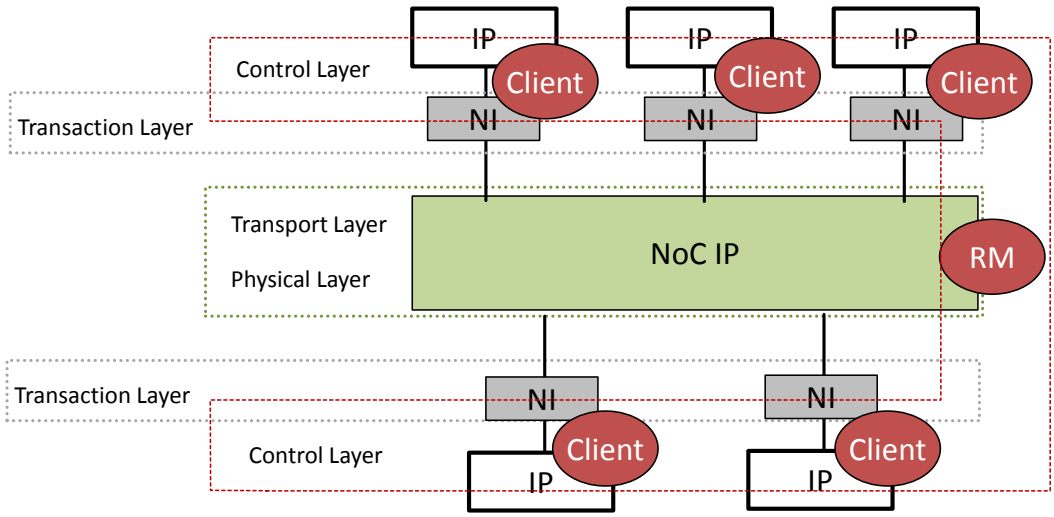


Figure 3.13: New modules introduced by the control layer forming the session layer of NoC communication.

to use a connection-oriented network service, the communication participant (e.g. IP) must firstly establish a connection[3], as discussed in the previous Section, then use the resources and release at the end.

In a SoC multiple senders often compete for the same resources. Resulting interference may endanger the real-time properties of the system and / or safety. This raises a need for an arbitration unit which in a predictable and safe manner decides about an order and duration of connections (transmissions). This is the main function of the **Resource Manager (RM)** - the scheduling unit used for establishing and managing all aspects of connections. However, assuring the predictable behavior of RM is not enough for providing temporal guarantees. Note that malfunctioning or malicious senders may easily influence work or the RM. For instance, wrongly configured applications could send too many requests delaying or blocking the work of the arbitration unit. They could also wrongly/maliciously configure their connections, e.g., ask for too much resources (blocking other senders) or occupy them for long period of time (or ever never releasing them).

Due to their importance, these problems must be addressed whenever temporal predictability is required. Otherwise no real-time guarantees can be given. Moreover, it is enough that one from the synchronized senders is malfunctioning and it may compromise the safety of whole system. This is especially important in mixed-critical setups where integrated in a SoC applications and functions may have different impact on safety and therefore require different certification efforts (e.g. ASIL levels from ISO26262 [44]).

For dealing with these problems, we introduce **clients** - the admission control units (local supervisors) running locally on each processing node in the NoC. Clients are responsible for: i) establishing connections with RM (issuing correct messages to RM for appropriate actions of senders and processing node); ii) preventing non-authorized accesses to the NoC iii) adjusting local admission setting in NI, e.g., rate settings for rate limiters or MMU/MPU address translation tables, based on the configuration messages from RM; iv) prevent non-authorized or too frequent accesses; v)

[3]A circuit is another frequently used term for a connection with associated resources

release the NoC resources (inform the RM whenever an application terminates); and vi) prevent unbounded NoC accesses - release the NoC resources (inform the RM whenever a connection takes to long and/or terminate too long connections).

Figure 3.13 presents the client and RM, forming the session layer (in red color). The client modules can be implemented as hardware extensions of the NI for high performance or as a software module running on the IP core (however also in this case some extension of the NI could be necessary). Similarly, RM can realized fully in hardware, i.e., as an independent HW IP connected to the NoC, or as a software IP running on one of the nodes. The main advantages of the software realization is flexibility. The complexity of both units depends on the complexity of selected synchronization protocol and will be discussed in the next sections.

Separating the clients and RMs from software running on processing nodes is especially important in safety-critical setups where NoC is treated as a shared resource. Recall, that in such setups it is necessary to either certify the whole system (including all applications) to the highest relevant safety level or decouple the resource arbitration from senders for providing sufficient independence [44], [3]. As assuring adherence to standards is a costly and demanding process, the latter is the preferred solution in majority of setups. This can be achieved through clients and RM which can be certified/designed independently to the highest relevant safety level.

3.5 Synchronization with the Control Layer

In general, the process of global synchronization can be divided into the following phases: *initialization*, *reservation*, *usage*, *release*. Fig. 3.14 presents the method in using a motivational example: the application (sender) issues a request to RM for providing access to a DDR-RAM. Firstly, the client must detect the access to the Noc from application (IP) and block it until the connection will be granted by RM. Next, the client must issue a request to the RM for establishing of a proper connection, Fig. 3.14.a). After receiving, evaluating and processing of the request, the RM can grant the access to the resource - DRR-SDRAM memory, Fig. 3.14.b). Upon arrival of the grant message from the RM, the client unblocks the sender and the connection may become active, Fig. 3.14.c). The Sender might have access to the resources for a predefined connection length (e.g. number of issued packets or time), depending on the actual implementation. After the transmission finishes or the connection length has been reached, the client must terminate the resources (and eventually block the sender). This is done with the appropriate release message issued to the RM, Fig. 3.14.d). The resources are considered to be free again after the release message has been processed by the RM.

In the following of this section we will discuss the process of synchronization in detail. This description will be used for presenting different versions of the protocol depending on the temporal requirements of senders.

Fig. 3.15 provides a generalized version of the synchronization workflow. Note that the actual steps and their details may vary according the selected protocol and implementation. In the figure, a sender conducts an access to a virtual resource composed of two real hardware resources, i.e., connection (e.g. path through a NoC with a predefined QoS) and an exclusive access to a selected hardware module (e.g. DDR-SDRAM controller). In order to satisfy this request, the RM firstly conducts the evaluation of the available resources w.r.t. the global system state. This is done based

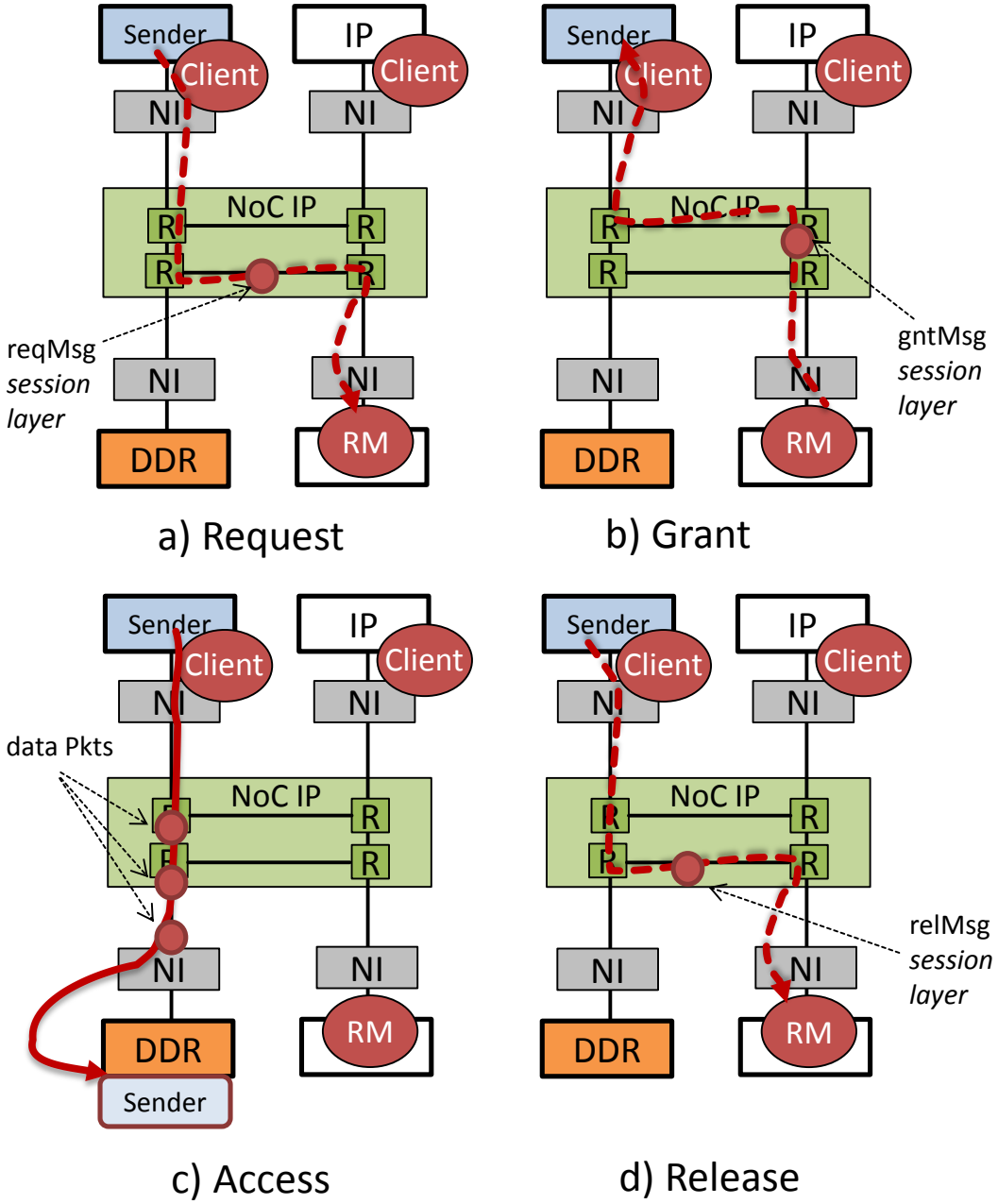


Figure 3.14: New modules introduced by the control layer forming the session layer of NoC communication.

on the defined system model and selected arbitration policy. Next, it locks a hardware module for the requesting sender and reconfigures the settings of the NoC (e.g. traffic shapers / rate limiters in egress ports of other senders) for fulfilling the requested connection parameters. The main goal is to assure that every requesting sender receives the maximum available service for the given state of the system, as well as the transition between system states (i.e. mode changes) are safe and reliable, i.e., avoid sporadic overloads. After transmission is finished resources must be once again released for other senders. Note, that the complexity of the protocol, i.e., each of the reservation phases, depends directly on the resource allocation scheme applied by the control layer (designer) as well

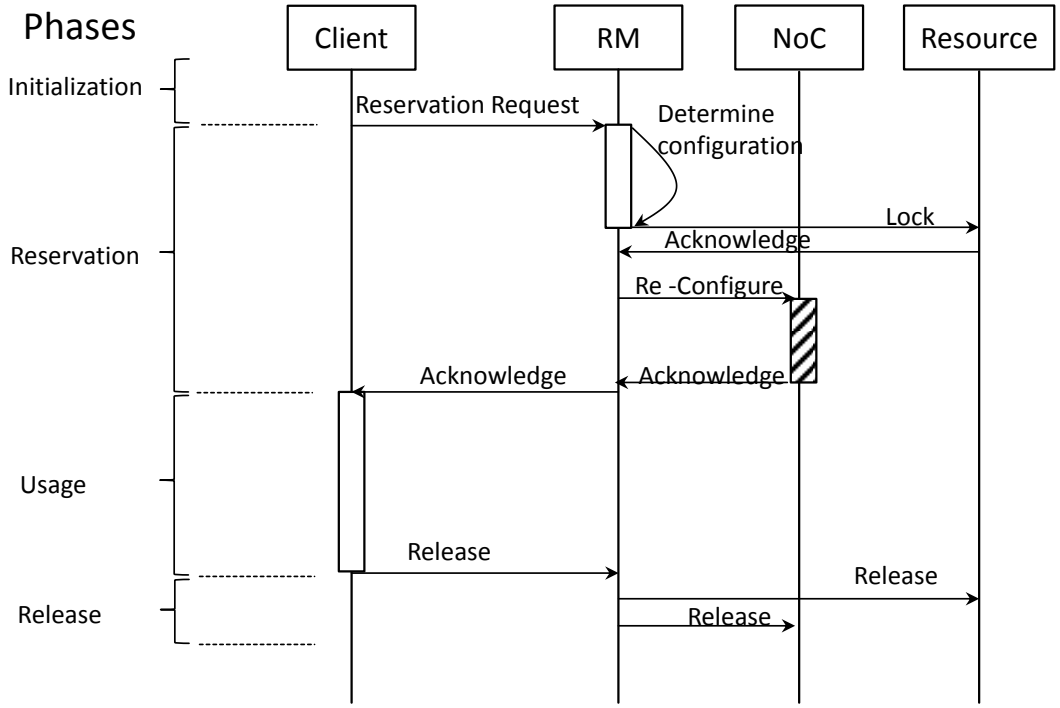


Figure 3.15: Exemplary workflow of the control layer specifying synchronization phases.

as on the concrete architectural features of the selected, underlying system architecture.

The following sections will provide an overview of the necessary actions, covering all aspects of these phases in the scope of conducted operations and participating elements of the system. The detailed information about the possible implementation methods and required tools will be provided in Chapter 4.

3.5.1 Phase One: Initialization

Firstly, the access from the synchronized sender must be detected and evaluated to provide information about the QoS requirements for the requested connection. Later, this information must forwarded to the central scheduling unit - the Resource Manager.

The prerequisite is HW support for the admission control, i.e., NIs/Tiles should support at least monitoring for rate control. This assumption is realistic as rate limiters are provided by the majority (if not all) of contemporary MPSoCs, e.g., R-Car [83], Cell [79], KiloNoC [38], IDAMC [100], MPPA [26]. It is also supported by the considered baseline NoC architecture. The main difference w.r.t. existing architectures is the need to re-configure the rate settings at runtime. These values are currently set only once during the boot-up phase of the SoC or hard coded in the system settings. Consequently, adjusting them at runtime may require an additional extension of the interface. An MMU or MPU would be required if the NoC should provide fine granular protection, e.g., distinguish between different address ranges. Both can be found in the majority of new controllers, such as the R-Car from Renesans but also in the Cell processor. Therefore, the synchronization can be initiated by:

- NI through monitors (Rate Ctrl + MPU/MMU) ,

- Tile with Rate Control (implemented in HW or OS-SW)

The accuracy/granularity of synchronization constitutes the main trade-off. In the first scenario, a monitor in the NI can recognize only the addresses accessed by the tasks running on the tile. As a result, distinguishing between specific senders may be difficult. Therefore, synchronization can be applied either for the whole tile (e.g. monitoring of joint workload resulting from traffic initiate by a group of tasks) or for the specific address ranges (whenever MPU/MMU protection is supported by the NI). The second scenario considers setups where the tile provides additional support for identifying senders and processing messages from RM. This could be done by extensions of SW (e.g. dedicated syscalls) or HW (e.g. interrupts) IPs. At the cost of this additional resource overhead, it is possible to identify specific tasks running on tile, or with even higher granularity, specific actions conducted by tasks (e.g. concrete DMA transfers).

The logic responsible for synchronization with RM, which will be referred to in the following as client, can be developed as SW or HW IP. The software deployment decreases the size of the NI. It requires only an extra interface for programming rate limiters and MMU/MPU. On the other hand, the HW implementation allows better performance.

Complexity and goals of clients depend on the resources which are available for implementation as well as characteristics of the running senders and therefore vary between setups. Independently from the method of deployment (HW or SW), the actions of the client are transparent to the sender.

3.5.2 Phase Two: Reservation

Each request, before it will be granted, must pass several processing steps defining the internal working of the Resource Manager. These steps are presented in Fig. 3.16. Firstly, the new request is evaluated with respect to the availability of resources depending on the current state of the NoC, i.e., the number of running senders and conducted transmissions. Later, RM must check if the requested connection can be realized in the current system configuration assuming the selected arbitration method, i.e., conduct an *admissibility test*. For this purpose, the RM requires an up-to-date model of the NoC and a set of rules - the arbitration policy. The former model provides a description in the form of a data structures with information about resources, e.g. links and buffers in NoC, their capacity and utilization. The main goal is to check which NoC resources are occupied by senders and what are the QoS requirements of currently running transmissions. The arbitration policy must decide about the order of allocation of requests. For instance, all requests can be treated equally (round-robin policy) or a priority based preemptive or not-preemptive schedules can be used as well as time-based allocations (e.g. time-division multiple accesses TDMA). Example of such is synchronization of DMA transfers protecting locality of memory accesses demanding:

- only one transmission from the synchronization scenario can be active at a time,
- the change of an active transmission is possible only after the previous one has finished or it is preempted due to a transmission request with a higher priority,
- the RM must prevent starvation of requestors.

If there are enough resources to handle the request, the RM may allocate them to the requester and notify the client (or sender) about the success. If this is not the case, the management unit may

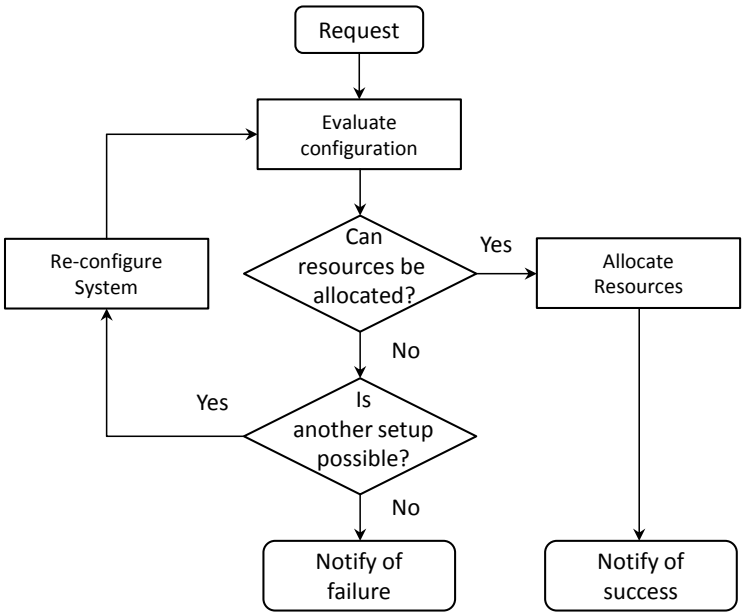


Figure 3.16: Steps required for the re-configuration of QoS control plane and its operations in NoC domain.

look for another possible setup/configuration, e.g., adjust properties of other clients so that the requirements of all senders can be met. If reservation requests are provided with a priority-based arbitration, then the RM must enforce the proper order of re-configurations during the allocation phase. This can be done in both preemptive or non-preemptive manners. If the former method is selected, RM can deprive senders with lower priorities from resources as soon as the hi-priority request arrives. Notice, that this operation inherently causes the change of system mode and in result may cause a sporadic overload situation which endanger system safety [74],[52]. For instance, although RM may send a control message to block certain nodes, the flits/packets injected before arrival of this message must still leave NoC and could possibly interfere with higher priority senders. Therefore, designers must take such effects into account during the design of control layer protocol and assure safe and efficient mode changes, for details refer to Sections 3.6 and Chapter 4. If a non preemptive method is selected, the higher priority request must wait for ongoing lower priority transmission to finish, but this blocking happens only once.

In situations when there are not enough NoC resources to handle the request, the RM may actively reject the transmission or inform the requester about the foreseen length of the blocking, see Section 3.7. This allows building an interface between on-core and resource scheduling, e.g., on-core preemptions. Similarly, RM may re-order (or prioritize) requests basing on the properties of resources which they are targeting, e.g., stateful nature of DDR-SDRAM scheduler. Details on the interfaces to on-core and resources scheduler are provided in Sections 3.7 and 3.8. Finally, RM must communicate its decisions (appropriate regulator settings) to all the involved senders/clients with control messages.

3.5.3 Phase Three: Usage

The resource usage starts after receiving the acknowledge for a request. Clients modify the regulator settings *only* after receiving the settings from the RM depending on the current *system mode* communicated by the RM. Depending on the selected implementation the acknowledge may allow:

- *entire transmissions constructed from multiple packets* for instance a DMA transfer.
- accesses to the NoC with a pre-defined rate (settings for rate-controllers) for cache based traffic.

The settings during the usage phase may change at runtime. Firstly, application may be blocked to allow transmissions with higher priorities to progress with their execution. Secondly, the allowed access rate may be increased or decreased depending on the global state of the system. In case of cache lines, this effectively throttles (accelerates or slows down) on-core execution of senders.

Finally, for some applications the settings may be pre-defined (during the design phase) and constant during the runtime, i.e., they stay for the whole execution duration in the usage phase. For instance, the designer may decide to keep the rates for some applications on a constant level (without endangering their deadlines) in some or all modes, in order to limit the number of necessary synchronizations and re-configurations. Additionally, the fully dynamic reconfiguration may be conducted only for selected senders or usage scenarios, e.g., fail-over recovery assigning more bandwidth to a select sender or data flow in case of a need for re-transmission caused by a transient error.

3.5.4 Phase Four: Release

The one of the most important advantages of the proposed resource allocation scheme is the fact that resources can be released whenever they are not used by a sender. This allows to introduce dynamics which improves not only the utilization of the system but also, even more importantly, allows to reach better formal guarantees in the majority of setups.

The release method depends strictly on the requesting method and has similar advantages or drawbacks. Apart from release being done at the end of transmission, e.g., signalized by the last packet, flit or interrupt, the release may be enforced by monitor after a predefined timeout. This is done for assuring a safe upper-bound on the resource usage sessions, i.e., avoid infinite accesses which could be caused by malicious or malfunctioning applications.

3.5.5 Summary

The mechanism provides a QoS abstraction of the underlying NoC (data plane) allowing a path oriented approach in which both the per-hop behaviors of routers and the end to end properties of communication can be unified. This allows safe and efficient resource reservations but requires knowledge about the global state of the system, i.e., number of simultaneously running senders, their current state and QoS requirements which may change during runtime. Therefore, for establishing and adaptively managing connections, clients must negotiate reservations. This is done through exchange of messages between nodes with interfering senders. These communication is

forming a synchronization protocol allowing propagating information about the state of a connection as well as current QoS requirements for a particular sender. Note, that the latter parameter can change dynamically and the contract based negotiation allows safely adjusting the QoS of the platform.

Several negotiation schemes are possible with two pre-dominant patterns: direct communication between clients or communication through a single scheduling unit (Resource Manager). This problem translates to the classic synchronization dilemma differentiating between centralized and decentralized resource assignment scheme. Trade-offs between both approaches are well researched and described in the existing literature, e.g., [48], [72], [9]. Application of one from these schemes is usually setup dependent and influenced by the number of synchronized senders, their mapping, parameters of the transmissions (length, duration etc.) as well as concrete architecture (e.g. propagation latency of the single control message). In the next section, a couple of exemplary resource arbitration protocols will be used to familiarize the reader with the mechanism and present the possible trade-offs.

However, the main advantage of the mechanism, common to both the synchronization schemes, is that it makes the QoS functions in the routers oblivious for achieving the real-time and safety guarantees. The admission control and adaptive management of NoC state is fully controlled by the introduced QoS control plane making the system potentially more efficient. This allows to adaptively optimize the arbitration to the changing global state of the system for accommodating arriving workloads as well as reacting to possible errors.

3.6 Resource Arbitration with Control Layer

One of the most significant competitive advantages resulting from the decoupling of the admission control from the underlying NoC infrastructure is the wide spectrum of the resource arbiters (allocation schemes) which can be implemented with the control layer. The control layer may implement non-work conserving (e.g. time-division multiplexing) as well as work-conserving arbiters (e.g. round-robin arbitration) covering the majority of the sharing schemes known from the literature e.g. [63], [84]. The proposed method may also introduce different priorities for independent critical senders depending on their requirements w.r.t. the NoC resources (static priority preemptive or priority-based non-preemptive allocation schemes) as well as to release the resources for best-effort senders in mixed-critical setups. Additionally, the arbitration can be done at different levels of granularity, cf. Section 3.1.3. It can be applied on the transmission basis, e.g., bulk DMA transfers build from multiple packets, or at the task level, e.g., on-core activations of tasks deciding about the traffic control settings for cachelines on a particular node. Moreover, the control layer can apply different isolation techniques to achieve the aforementioned goals, i.e., temporal isolation where paths through network are statically assigned to senders and access are controlled in time as well as spatial allocation paths for a single sender can change during the runtime.

This flexibility allows the designer to extend the capabilities of a selected NoC-based platform without the need for modifications of routers. For instance, the introduced solution permits implementation of a TDM-based arbitration on top of a NoC with the performance oriented arbiters in routers such as iSLIP arbiter [70]. Consequently, the control layer permits increasing the spectrum of system applications, e.g., adjusting it for providing guarantees in real-time domains, without

need for re-design or costly hardware extensions.

In the following chapters the main focus will be placed on the centralized implementation of the control layer. Although a centralized single point of synchronization through a single RM unit may seem to be a performance bottleneck, it has also significant advantages when it comes to safety and formal verification in case of SDNs, see [98]. Firstly, it allows to simplify the synchronization protocol, e.g., (rapid) spanning tree protocol or shortest path bridging. On the contrary, in case of decentralized system a lengthy and complex protocols for assuring system coherence are frequently needed. Indeed assuring that all senders agree on the global state of the network before they can take actions, increases complexity and volume of communication. Moreover, simpler clients and RM decrease the costs of verification and certification what plays critical role in many real-time applications from e.g. avionic and automotive domains. However, this applies only to setups which do not require a complicated controller, as mentioned above.

This section presents multiple examples of synchronization protocol for supporting different allocation schemes with the control layer. Note that, the protocols are adjusted so that they may support sharing of the same VC between transmissions also from senders with different criticality levels while preserving service guarantees. This can be used to decrease the number of required VCs in a system or to increase the number of hosted applications. The majority of the proposed protocols derives their efficiency from work-conserving scheduling which tires to keep interconnect resources busy for a maximum period of time.

3.6.1 Time-Driven Scheduling

Static isolation is a popular method for managing contention in a NoC. It allows through time slot assignment providing service guarantees independent from other tasks, cf. [63], [84]. Consequently, each task can occupy the interconnect for a given, pre-defined amount of time. After fully using its time budget, the sender must release the resource and allow the next task to proceed with transmission. The budgets are replenished in cyclic order. The proposed control layer can implement both popular variants of time-driven scheduling: *time-division multiple access (TDMA)* and *round robin (RR)*.

Time-Division Multiple Access

As discussed in Chapter 2 the time-division multiplexing (TDM) approach constitutes the commonly applied solution for real-time systems and is a dominant standard in the avionic domain. According to this scheme each application is granted a dedicated time slot during which it acquires exclusive access to the interconnect. Transmissions are performed in a cyclic, static order forming a TDM cycle. Consequently, the designer may enforce full temporal isolation between independent transmissions, i.e., guarantees for running applications are independent from activities of other senders running in the system.

Fig. 3.17 presents the workflow of such a solution. In the considered example, two senders (A and B) form a TDM-cycle in which they acquire access to the NoC alternately. The RM sends the messages - *gntMsg*- granting access for a selected core for a predefined amount of time[4] - its time slot (slots

[4]Note that NIs must have ability to measure time (e.g. time window for rate limiters). However, also a setup is possible where only RM has timer. In this case, RM must firstly block/preempt currently ongoing transmission from the

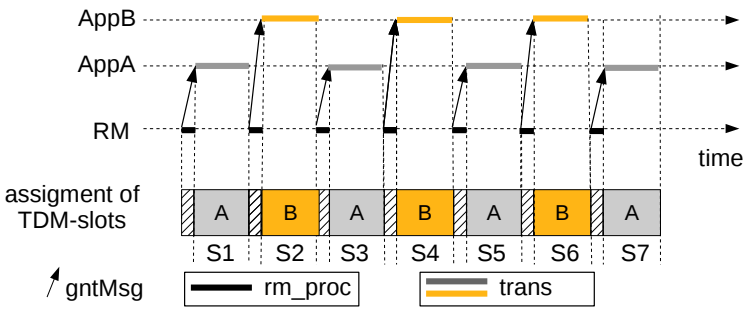


Figure 3.17: Workflow of the TDM-cyclus enforced using RM in a NoC with protocol overhead (rm proc) and durations of two different transmission (trans).

S1-S8). The time necessary for the RM to generate the message and its propagation latency in the NoC constitute the overhead of the scheme, when compared to the traditional TDM deployments done in hardware.

However, the main advantage of the proposed solution is constituted by its applicability to the majority of commercially available NoCs, such as MPPA or Tile64. The proposed RM-based TDM admission control can be introduced "on-top" of existing, commonly used wormhole-switched NoCs with multistage arbitration. On the contrary, existing NoCs supporting time driven scheduling require custom, and frequently complex, router architectures, e.g., PhaseNoC [80], SurfNoC [108], Aethereal [37]. Please, refer to Chapter 2 for detailed discussion. Moreover, TDM scheme when implemented with the control layer can be flexible and easily extended to incorporate elements of round-robin work-conserving scheduling. In the following we show some examples for possible improvements based on [55].

Round-Robin Based Performance Optimized Arbitration

Although TDM-based static resource allocation scheme allows an easy implementation and provides timing guarantees, it also results in average latencies which are very close to the worst case even when the system is not highly loaded [39]. This is mainly due to the traffic from general-purpose applications that hardly ever follows a constant, predictable pattern assumed by TDM schemes.

The RM-based control layer helps to overcome this challenge through an implementation of the selected round-robin based resource allocation scheme, well known from the real-time literature [63]. The description of possible protocols extension and its working is based on [56], [57]. Similar to TDM-based solutions, the interconnect is considered to be the global resource shared in time. The transmissions are granted in the cyclic-repetitive order as in case of TDM. However, the slots of not active senders can be released (re-allocated) to faster serve pending requests from active senders, i.e., therefore introduced arbitration policy is work conserving. Therefore, the proposed architecture introduces work-conserving arbitration. A round-robin scheduler avoids unnecessary idle times and offers a more compact schedule, i.e., it tries to keep the NoC resources busy whenever there are ready pending transmissions. Consequently, round-robin based control layer allows improved average performance without decreased safety guarantees, i.e., it offers the same worst-case

previous slot and later send gntMsg, as described in the Sections.

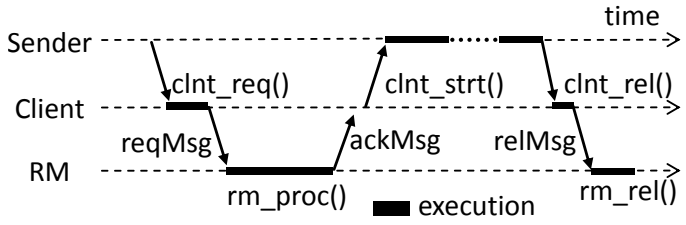


Figure 3.18: Workflow of the RR-based admission control in a NoC implemented with the control layer.

performance/guarantees as TDM.

Workflow The communication protocol for round-robin based allocation of resources is realized using three control messages: *reqMsg* (request), *relMsg* (release) and *ackMsg* (acknowledge), cf. the workflow in Fig. 3.18. The corresponding client sends a request message to the RM (*clnt_req*()). The RM is equipped with a queue for storing pending requests from clients. If the queue is empty, the RM must wait for a new request to arrive. Otherwise, the scheduler decides about which request from the queue should be served first (*rm_proc*()). The access are granted in the predefined cyclic order but unused slots (for which there are no pending requests) are skipped. After that, the *RM* must notify the selected sender for service with the *ackMsg*. After receiving the *ackMsg*, the communication may start (*clnt_strt*()). Once granted, the connection holds until the end of the transmission or the abortion through the client based on a predefined timeout used to prevent unbounded connection times. When the client detects the end of the transmission (e.g. based on its time-budget or injection of the last flit) it issues a *relMsg* to the RM (*clnt_rel*()). As soon as the *relMsg* arrives, the RM considers the resource to be free again (*rm_rel*()).

RR-based Control Layer vs TDM The proposed solution allows to overcome the main drawbacks of the TDM based arbitration. Firstly, the introduced control layer decreases average latencies, i.e., temporal overprovisioning, of the applications which are not optimized w.r.t. the TDM-cycle due to the work-conserving arbitration. As presented in Fig. 3.19.a) , in a system with a TDM based arbitration, whenever tasks expose dynamics in their behavior, e.g., even with a small jitter, transmissions are blocked and their execution is delayed for the duration of a whole TDM cycle. On the contrary, when RR-based arbitration and control layer are applied, the transmissions can be scheduled whenever the NoC is free, see Fig. 3.19.b). This is due to the introduced work-conserving arbiter which tries to process the requests as soon as they arrive. Consequently, the control layer significantly improves the average latencies especially in systems which are not fully loaded/utilized. Consequently, the arbitration of big scenarios with multiple senders exposing dynamics in their activation patterns (e.g. modes of work, interrupt signals) is done without the need for complex and hard to maintain TDM-cycles. Additionally, the proposed mechanism allows to maintain the locality of memory accesses [77] through the isolation of a whole transmission. This improves performance and decreases power consumption of DRAM memory modules which constitute the most common hot-module in MPSoCs.

Secondly, the proposed solution allows to compact the access schedule for the interconnect. Whenever the interconnect is not heavily loaded, see Figure 3.20 the unused slots can be omitted (re-allocated) for pending requests. This is especially useful for longer transmissions composed of multiple packets, e.g., DMA transfers which can progress faster on average without idling between

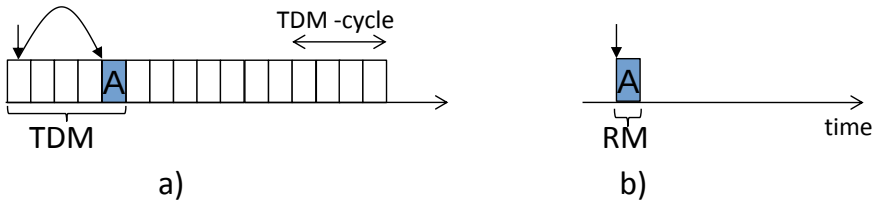


Figure 3.19: Incorporation of dynamics in activation patterns in a system with a) TDM-based arbitration and b) RR-based control layer.

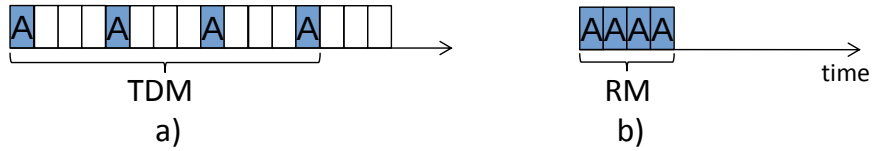


Figure 3.20: Blocking w.r.t to the actual load of the system for a) TDM-based arbitration and b) RR-based control layer.

consecutive TDM cycles.

3.6.2 Static Priority Based Arbitration

This section presents an alternative approach in which control layer is adjusted for providing efficient and safe guarantees in mixed-critical real-time systems based on [58]. For this purpose, RM conducts a global, priority based scheduling.

The synchronization between clients and resource managers is accomplished using five control messages: *reqMsg* (request), *relMsg* (release), *ackMsg* (acknowledge), *blckMsg* (preempt communication) and *resMsg* (resume communication). The workflow is depicted in Figure 3.21.

Clients issue request message *clnt_req* whenever supervised sender are trying to access resources. If there is no pending request, *RM* must wait for a new request to arrive. If a request cannot be granted, due to an ongoing higher-priority transmission, it is stored in a request-queue. When the resource is free again and the request-queue is not empty, *RM* selects the request m_i with the highest priority. *RM* notifies the appropriate client with the *ackMsg* which unblocks the granted transmission m_i. If a resource is occupied, i.e., there is an ongoing transmission, *RM* must monitor all arriving requests and compare their priorities against the priority of the currently ongoing transmission. The comparison and monitoring is done in the arrival order. If the priority of the request is lower than the priority of the ongoing transmission it is stored in the queue, otherwise *RM* conducts a preemption. In order to preempt an ongoing transmission, the *RM* sends a *blckMsg* message to the client supervising it. Preemptions may be nested, i.e., transmissions which previously preempted an ongoing transmission can also be preempted. Therefore, a *RM* must store information about ongoing and preempted transmissions. Moreover, RM sends the *ackMsg* with a delay to ensure that packets from previously ongoing transmission have definitely left the NoC, i.e., provide logical isolation. After receiving the *blckMsg*, client blocks the next packet from the ongoing transmission as soon as possible. The preemption is performed at the packet level. Because

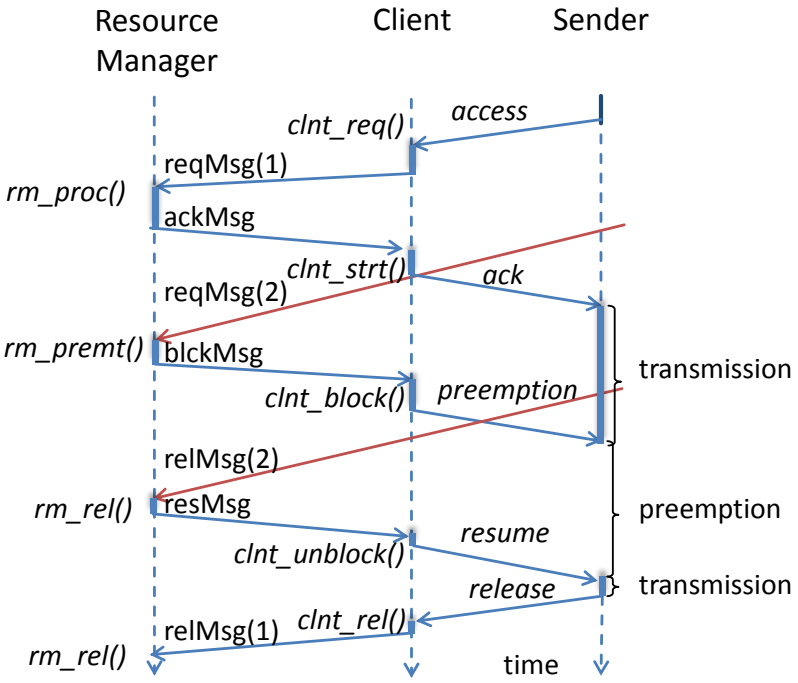


Figure 3.21: Workflow of the SPP-based synchronization protocol in a NoC utilizing the control layer.

preemptions are happening on the same VC, the flit level is unacceptable due to the properties of the wormhole routing (cf. [23]). Higher preemption granularity, e.g. multiple packets to ensure the locality of a transmission, is also possible after accounting for an additional latency. When the client detects the end of a transmission (e.g. based on its time-budget/timeout or injection of the last flit), it issues a *relMsg* to the appropriate *RM*. As soon as the *relMsg* arrives, the RM considers the resource to be free again. If there is a pending preempted transmission with lower priority, *RM* resumes its execution after sending a *resMsg* to the appropriate client. Otherwise, a new request is selected from the queue. As soon as the *resMsg* arrives, the client unblocks corresponding preempted transmissions.

RM-based SPP vs SoA Solutions. As discussed in Chapter 2 traffic prioritization offers an alternative towards synchronizing transmissions in Networks-on-Chip (NoCs) [18],[43]. The main disadvantage of this scheme is high resource demand, i.e., streams of different priorities must be isolated in different traffic queues. Consequently, the number of VCs must be equal to the number of criticality levels in the system and therefore must increase accordingly. The constant increase in the number of applications integrated into a single chip, e.g., Flight Management System [31], requires the number of VCs to be equal to the number of criticality levels and to increase accordingly, otherwise the system is not predictable [36]. The proposed scheme based on the control layer allows efficient sharing of the same buffer queues in routers by streams in mixed-criticality systems.

Additionally, formal verification of priority based routers can be complex especially in case when backpreassure effects (see Chapters 1 and 2) could lead to the propagation of blocking and cyclic dependencies between streams. The control layer avoids this problems using the global arbitration scheme. The blocking is moved from the NoC to the cores, therefore i) it is possible to significantly reduce the size of buffers [58] ii) one could re-use the blocking time on-core for execution of other

tasks, what is is in detail descried in Section 3.7. Control layer allows also to guarantee locality of memory accesses what is not possible in case of priority based scheduling in routers.

3.6.3 Dynamic Priority Based Arbitration

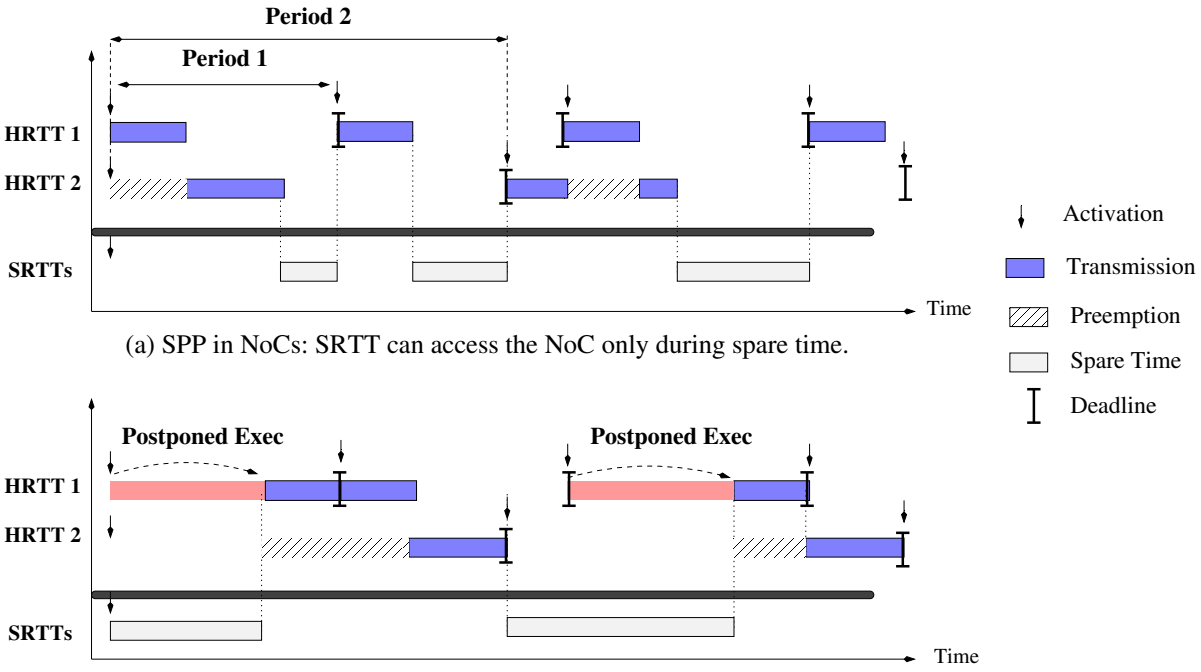
The strict priority-based scheduling done locally in routers decreases the performance of best effort senders. These receive usually lowest possible priority [28],[58] and are scheduled only when there is no other pending transmission. Consequently, they suffer from high latencies and jitter.

Slack-based resource allocation is a well known solution from the scheduling theory [63], [24] to this integration challenge, providing high performance for best-effort and soft-real time applications without violating the timing constraints of hard real-time applications. According to this approach, BEs and SRT'Ts are scheduled whenever the execution of HRT'Ts can be safely postponed without causing missed deadlines. This is possible whenever a slack is available, which is the time budget between the worst-case response time of a hard-real time application and its deadline. Applying this principle to NoC significantly increases the performance of soft real-time and best effort data streams which is particularly useful for general-purpose latency sensitive applications running on processors with caches [104, 101, 105].

This section proposes an extension of the previously introduced priority-based protocol for the control layer. The description is based on [49]. Consequently, this approach allows RM to safely *delay* the execution of HRT'Ts, whenever it is possible, in order to improve the performance of SRT'Ts.

A prerequisite is offline computation of time budget for each HRT'T a called *slack* (see Chapter 4, Section 4.1.1), which defines the maximum delay an HRT'T can experience without endangering its deadline. Later, the RM monitors the slack budgets of *currently ongoing* HRT'Ts and dynamically adjusts the priority of SRT'Ts accordingly. SRT'Ts get the highest priority whenever there is an available slack and the lowest priority whenever there is no slack. Note that delaying the execution of an HRT'T with the highest priority also delays all currently preempted HRT'Ts with a lower priority. Therefore, this delay can only occur if enough time (i.e slack) is available for all HRT'Ts to meet their deadlines.

Workflow The implementation of arbitration based on dynamic priorities is realized using five control messages: *reqMsg* (request), *relMsg* (release), *ackMsg* (acknowledge), *preMsg* (preempt) and *resMsg* (resume). Upon its arrival a request message (*reqMsg* the RM, each request is classified as SRT'T or HRT'T and stored in the appropriate queue. FIFO is used for SRT'Ts and a priority queue for HRT'Ts. As described previously, the RM monitors all HRT'Ts and keeps track of the amount of available slack. When an SRT'T is granted access to the NoC, the RM decrements in real time the slack budgets of all active pending HRT'Ts, i.e., it considers all requests in the HRT'T queue. The slack budget is renewed at each request arrival (*reqMsg*) from an HRT'T. If there are no pending SRT'Ts or the slack budget of at least one HRT'T has been exhausted, HRT'Ts transmissions regain access to the NoC and are scheduled according to the standard static priority preemptive (SPP) policy. Note that, when SRT'Ts have access to the NoC, the RM must release the resource early enough to guarantee that HRT'Ts start on time (i.e. not after the slack budget has been consumed). When the NoC is free, the RM notifies the appropriate client with an *ackMsg* for the pending transmission to start sending data. If the NoC is occupied and the RM receives a new request with a higher priority,



(a) SPP in NoCs: SRTT can access the NoC only during spare time.

(b) Slack Based Allocation: postponing HRTTs allow early SRTT progress in the NoC

Figure 3.22: Comparison between SPP and slack-based resource allocation in NoCs.

the RM must preempt the currently ongoing transmission. The preemption is realized in the same manner as in case of the protocol for arbitration with static priorities, i.e., *preMsg* for blocking the client and *resMsg* for resuming it. Recall, the high-priority sender must start its transmission with a delay (by *ackMsg*) to ensure that packets from previously ongoing transmission are not present in the NoC. Finished transmissions are confirmed by clients through sending the *relMsg* to the RM which removes the corresponding request from the head of the queue. Next, if there is a pending preempted transmission with a lower priority, the *RM* resumes its execution after sending a *resMsg* to the appropriate client. Otherwise, a new request is scheduled.

Comparison against SoA Only few existing works, e.g. Backsuction mechanism [28, 101, 105], have considered using slack-based resource allocation in the context of NoCs by applying, locally in routers, dynamic prioritization for different virtual channels. However, the main drawback is the high hardware overhead resulting from a custom router design and the high number of virtual channels (i.e. required buffers) corresponding to the number of priority levels in the system. Moreover, these solutions drastically increase NoC complexity as they require to propagate the global state of the NoC to the local arbiters in routers. This is in conflict with the principle of non-blocking routers which requires no correlation between local arbiters and thereby increases the complexity of the worst-case timing analysis.

Consider the example depicted in Figure 3.22, a NoC shared between a set of HRTTs and SRTTs. It constitutes a mixed-critical system where each HRTT has a unique *static* priority (assigned for instance according to a rate-monotonic scheme) and remaining SRTTs have the same lowest priority. Figure 3.22-(a) illustrates the evolution over time of transmissions in a real-time NoC using a static priority preemptive (SPP) policy [18],[43].

RM permits to safely delay the execution of HRT'Ts, whenever it is possible, in order to improve the performance of SRT'Ts. [5] SRT'Ts get the highest priority whenever there is an available slack and the lowest priority whenever there is no slack. Note that delaying the execution of an HRT'T with the highest priority also delays all currently preempted HRT'Ts with a lower priority. Therefore, this delay can only occur if enough time (i.e. slack) is available for all HRT'Ts to meet their deadlines. This is illustrated in Figure 3.22-(b), where HRT'Ts are postponed, to allow SRT'Ts to start earlier. Consequently, the introduced control layer manages to drastically reduce the latencies of SRT'Ts, compared to Figure 3.22-(a), without endangering the deadlines of HRT'Ts.

Finally, this complex scheduling does not requires custom router design. Indeed, the proposed solution can be applied on-top of simpler routers e.g. in performance optimized NoCs.

3.6.4 Adaptive path allocation

In the majority of existing NoCs guarantees for safety critical (SC) senders are achieved at the cost of decreased BE performance. To assure predictability, designers apply static resource allocation schemes - oblivious routing - where communication paths are solely defined by source and destination and do not change during runtime, see Chapter 2. This results in known, well defined sets of interfering senders and allows the implementation of the strict spatial separation (BE do not share paths with SC) or temporal isolation (BE are blocked whenever SC are sending) for providing guarantees. However, such static path allocation scheme significantly decreases hardware utilization and performance. Indeed, if oblivious routing is applied, non- uniform traffic patterns can induce large load misbalances giving suboptimal throughput [23], thus resulting in the overly pessimistic worst-case guarantees for SC and unfulfilled design requirements for BE senders [106].

In order to fully exploit the multidimensionality of a NoC's topology, this section introduces a protocol for the operation of control layer which applies different path allocation methods, i.e., taxonomies, depending on sender's criticality. The description is based on [51], [54]. For handling SC traffic control layer applies oblivious routing (static path allocation) whereas BE traffic will use multiple paths from the source to destination - an adaptive load distribution. Whenever NoC resources are not used, BE traffic is allowed to use the shortest (optimal) path to its destination, even if it overlaps with links used by critical senders. Upon activation of the SC sender, the control layer releases NoC resources by redirecting the BE transmissions to an alternative route, i.e., BE senders use detoured paths (non-optimal) only when critical senders are actively using resources. This reduces the impact of SC transmissions on the average BE performance and allows gradual degradation of service i.e. instead of experiencing full blocking BE senders experience slightly higher latencies on the detoured path for some packets and consequently lower average latencies.

Workflow Ensuring worst-case guarantees requires:

- to identify interfering senders and
- to provide temporal synchronization of concurrent transmissions.

The former can be achieved by statically assigning a set of paths for each BE application with one of available allocation methods, i.e., the set of possible paths for a BE sender does not change during

[5] Recall that frequently in real-time systems systems, SRT'Ts can progress through the NoC *only* when HRT'Ts are not sending data, therefore they are often unnecessarily delayed. HRT'Ts are scheduled as soon as they arrive, although they usually do not profit from faster execution as long as they are guaranteed to finish before their deadline.

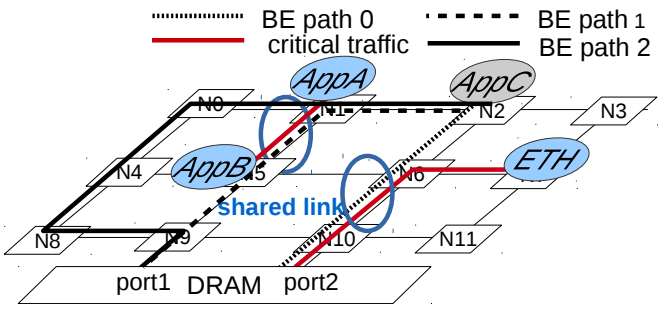


Figure 3.23: An exemplary setup detailing possibilities of spatial isolation in a NoC.

runtime. In principle there are no limitations in alternative route selection, besides the rule that if a link in the path is shared with a SC sender, this sender must be capable of accepting the protocol overhead, what will be discussed later in this section. However, if all detoured paths are shared with SCs and all SCs are active then the BE sender will be blocked similarly to temporal isolation. The latter condition is achieved through decoupling admission (the control layer) and flow control (the data layer) in the NoC.

Upon beginning and termination of a transmission from SC sender, RM sends control messages to all interfering with him BEs: *actMsg* (SC activation), *relMsg* (SC release). These messages propagate the global NoC state defined by the currently active SC applications and determine the paths for the BE senders, i.e., after receiving these messages, appropriate paths used by SC are blocked or released for BE transmissions. After each mode change, the BE sender must use the shortest path available for it - a path without any active SC sender. Finally, to preserve predictability (i.e. isolation) of SC senders, RM must account for the delay resulting from time necessary to deliver control messages to BE senders and for the packets from BE transmissions to leave the interconnect on the selected path. This overhead is acceptable due to the slack of SC senders which is the time budget between the worst-case transmission latency and its deadline. This allows a trade-off analysis providing estimation of the overhead resulting from the global arbitration, see Chapter 4.

Comparison against static routing. The main advantage offered by the solution is the work-conserving resource allocation scheme allowing improved average performance of the best effort senders. Consider an exemplary setup with the mixed-critical workload presented in Figure 3.23. As path sharing between BE and SC may endanger the latter, the traditional safety approach would require traffic from application AppC to take the longest path (path 3). Consequently, AppC will not share any resources (links, buffers) with SC sender and spatial isolation will be achieved. However, this decreases its average performance. It must constantly follow the longer route even when SC senders are not active.

When control layer is applied, clients intercept transmission requests from SC senders (e.g. Ethernet ETHo and AppA) and enforce paths for BE senders accordingly (e.g. AppC) based on the received control messages, i.e., currently active SC senders. Consequently, if the NoC is free, the BE sender (AppC) is allowed to use the shortest path (Patho) towards DRAM. Upon activation of critical sender ETHo (arrival of Ethernet frame) RM is sending the control messages (*ackMsg*) to all BE applications sharing resources (links and buffers) with AppC. After receiving this message BE sender must redirect its traffic to the longer, detoured path (Path 1). If the second critical sender (AppA sending to

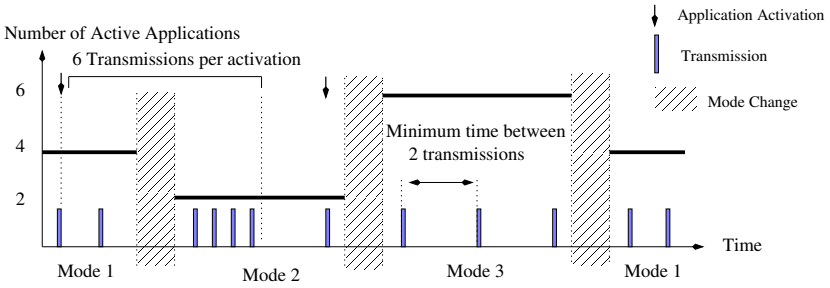


Figure 3.24: A schematic description of the traffic injection rate for a given application. The rate varies according to the system mode.

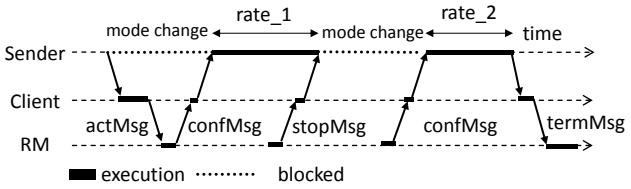


Figure 3.25: Workflow of an rate control mechanism synchronization with the RM.

AppB) is activated the procedure repeats and AppC must take the longest path (Path2). The paths are occupied by the SC sender until the end of the transmission/task execution or an abortion by the supervisor based on a predefined timeout. When the supervisor detects the end of a SC transmission (e.g. based on its time budget or injection of the last flit), it directs the relMsg to involved BE senders. As soon as Patho is free, AppC is allowed to use it once again. As illustrated in Figure 3.23, thus allows the detoured transmissions to progress faster using the free hardware resources (links and buffers), instead of experiencing full blocking by exploiting the NoC's dimensionality.

3.6.5 Adaptive Rate Control

As described in Chapter 2, traffic shaping using rate control is a well known method for achieving quality-of-service in real-time systems. The goal of this approach is to bound/enforce NoC access patterns and therefore interference which must be resolved through arbiters in NoC routers. The description is based on [53].

As in previously described protocols, tasks / applications must inform RM whenever they are activated or terminated on processing nodes. Using this information, the RM may decrease or increase the injection rates for a particular node, as depicted in Figure 3.24, dynamically depending on the current system mode. Each mode is defined by the number of currently active applications, and determines the minimum time separating every two transmissions issued from the same application.

Workflow The communication is realized with four control messages, see Figure 3.25: activation (*actMsg*), termination (*terMsg*), stop (*stopMsg*) and configuration (*confMsg*).

Firstly, the corresponding client sends an activation message to the RM for the activated task on a processing node. The activation and termination messages are processed by the RM in their arrival order. Each of them initiate the *transition* of the system to a different mode. In order to ensure pre-

dictability, these transitions are done sequentially, i.e., the new one may start only after the previous re-configuration is fully completed. Additionally, the RM must assure that the transition between modes is safe. Therefore, before changing the rates, the RM sends to the clients supervising active applications, a stop message (*stopMsg*) to block all accesses to the NoC from the corresponding node. Clients then waits for the *confMsg* communicating the current system mode. Consequently, the RM sends *confMsg* with a delay to ensure that packets from previously ongoing transmissions have left the NoC, i.e., provide logical isolation. After receiving the *confMsg*, clients adjust the rate and unblock transmissions. Additionally, *confMsg* initiates activated applications (ones which just issued *actMsg*) which can then use the NoC.

Comparison with rate control done locally in nodes. The major drawback of the traditional approach for rate control is that the rates are adjusted statically according to the worst-case scenario, i.e., assuming that all senders are running simultaneously with maximum possible transmission arrival rates. Therefore, this approach is not work conserving and sacrifices the interconnect utilization whenever senders expose dynamics in the execution time, release jitter or communication volume and the system is not highly loaded. Note that performance penalty increases along with the complexity and inherent dynamics – as in the case of the TDM-based arbitration.

RM may decrease or increase injection rates for a particular node dynamically depending on the number of concurrently active applications. The mechanism is capable of enforcing symmetric and non-symmetric guarantees. Consequently, it allows to enforce behavioral models for different data streams as well as to adapt at runtime to the currently active NoC load. The proposed approach allows to decrease both temporal and hardware overhead while significantly improving the overall performance of the system and meeting the strict timing constraints.

3.7 Interface Between Cores and NoC

The client forms an interface between the on-core execution of tasks and applications and their accesses to NoC. Therefore, its actions can be divided into a) control of interference which a Tile may exert on the NoC b) provide Tile with information about the status of the NoC, i.e., if this resource is busy or free for accommodating the new incoming traffic. The following subsections describe these main functions in detail.

3.7.1 Resource Control in NI

Because on each core there may be multiple senders requiring different connection settings, client must be capable of distinguish between them. Consequently, connections must be described accurately in terms of specific parameters that can be negotiated, e.g., senderId, route (path) through the network, settings of rate limiters and access right (e.g. read/write transactions). Typically, the sender should produce a flow specification proposing the parameters it would like to use. However, in the considered context of the real-time and safety critical systems the characteristic of applications with real-time and safety requirements is usually known beforehand and thoroughly tested, cf. Section 3.1.3. This settings are determined during the design phase and encoded/programmed at the chip startup (bootup) in real-time and safety mechanisms. An example of such mechanism are address translation tables and rate limiters which are extending the features of the NI from the

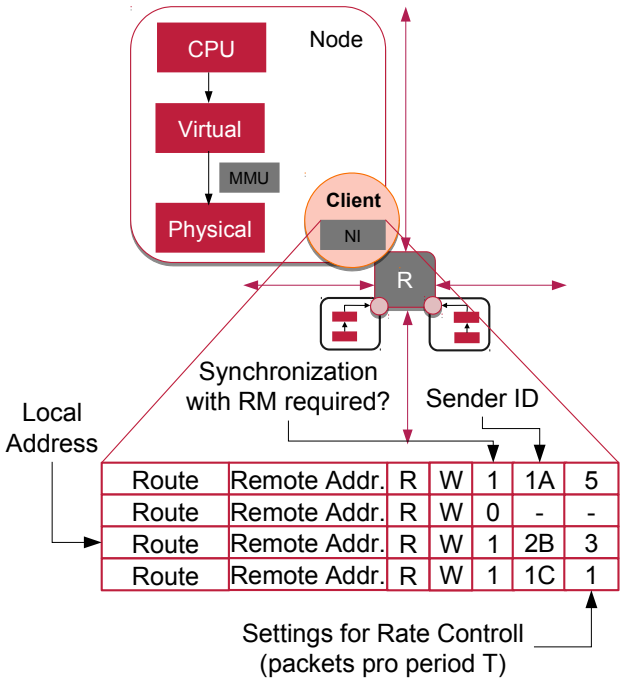


Figure 3.26: Synchronization data for client stored in address translation tables for address mapped NI.

baseline architecture.

Therefore, implementation of client can follow as an extension of these mechanisms. An exemplary client deployment is presented in Figure 3.26. It follows in a system with a memory mapped NoC, where NI introduces transparent address translation, where local (for processing node) physical address is converted into a remote address (on another processing core) and forwarded using NoC. Consequently, senders which are executed do not require any knowledge about the operation of the interconnect. The actual translation (including proper NoC addressing) is done by the configurable address translation module in the NI. This module is equipped with tables containing information about route and destination (remote address). Implementation of the client is done through straightforward extension of the mechanism. Firstly, the new column is added to assess if the synchronization with RM is necessary. Later, information must contain sender ID for identification of the synchronization scenario (assuming multiple of them). Finally, QoS settings are stored as number of packets per base time period. Note, that the last field can be omitted at the cost of the higher protocol overhead, i.e., control messages may contain settings but they will be longer.

3.7.2 NoC Support for Suspensions

In the majority of safety critical systems, suspension-based locking protocols, e.g., MPCP, OMLP, FMLP, are used to efficiently and safely coordinate accesses to shared resources. However, existing architectures do not support such arbitration for Networks-on-Chip (NoCs) although they must resolve conflicts between concurrent transmissions. Enabling suspension-based locking for applications using interconnect resources requires not only predictable latencies of resource operations, e.g., transmission times, but also providing feedback about the global state of the interconnect. This

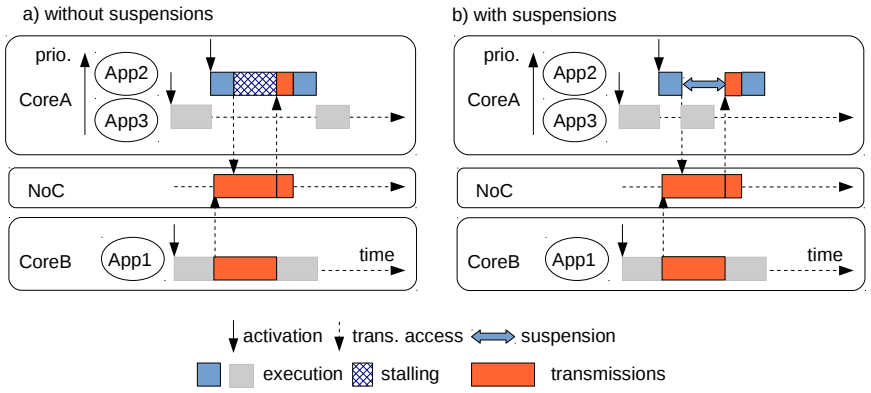


Figure 3.27: Example: Effects of conflicting accesses to the NoC resources from tasks running on different cores in a system with and without support for suspensions.

is relatively simple in classic bus-based architectures, where senders can directly receive feedback about the occupancy of the interconnect from the bus controller using control lines. However, existing Networks-on-Chip (NoCs) do not provide such information, although they must resolve conflicts between concurrent transmissions.

Indeed, in the majority of NoCs the arbitration between transmissions is done independently and locally in each router. Consequently, the state of the interconnect is unknown to the on-core scheduler during runtime, i.e., it assumes the NoC is always ready. Whenever a task accesses the NoC, its processor is stalled (e.g. busy waiting) for the entire duration of a transmission including blocking from other senders. This leads to i) a decreased processor utilization and ii) overly pessimistic worst-case guarantees as the network blocking propagates to the cores affecting the timing of other (low priority) tasks.

Existing real-time NoCs e.g. [37, 91, 56] can provide an upper bound on the transmission latencies using Quality-of-Service (QoS) mechanisms, such as TDM or dynamic prioritization. However, the decreased processor utilization resulting from the lack of feedback about the state of the interconnect has not been considered until now in the design of on-chip communications.

In commonly used wormhole-switched NoCs with the multi-stage arbitration, ongoing transmissions compete for output ports (link bandwidth) and virtual channels (buffer space). As the arbitration between interfering streams is conducted independently and locally in routers, each network's node has only information about the local status e.g. information from the flow-control mechanism about the state of the buffers in the neighboring router (back-pressure). Consequently, along with the progress of the transmission through the interconnect it must acquire several resources with independent arbiters and the information about the state of the NoC is unavailable. Because of that, some interference cannot be resolved locally by the router's arbiter and requires input from the adjacent neighbors e.g. a joint allocation of the crossbar switch and the router output due to a possible lack of buffers at the output (input-buffered router). Consequently for the on-core sender the state of the transmission in unknown during the execution e.g. its stalling the core during the blocking.

An exemplary scenario is depicted in Fig 3.27.a) where three tasks (App1-3) on two cores (CoreA

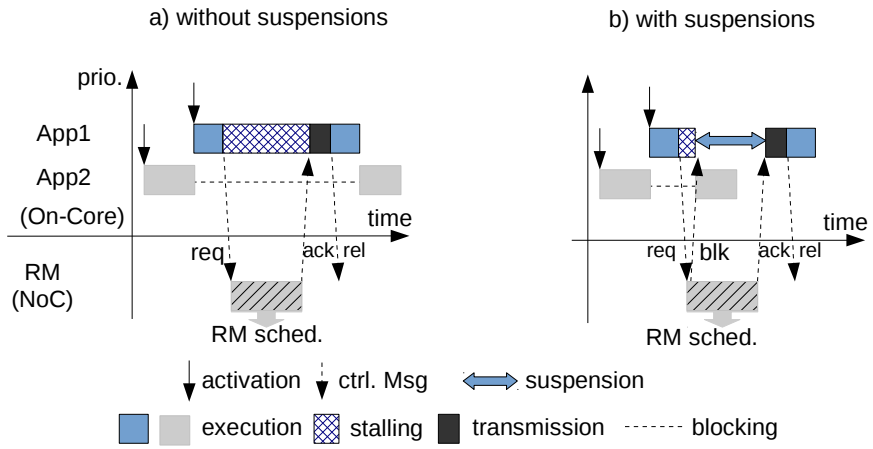


Figure 3.28: Workflow of the RM-based admission control in a NoC without and with the support for suspension-based locking of inter- connect resources.

and CoreB) share the path in a NoC with priority-based arbitration between VCs e.g. [18]. The application's number denotes its priority, i.e. App1 has the highest priority and App3 the lowest. App1 conducts its transmission as first. Next, App2 is activated and blocks (i.e. preempts) App3. App2 initiates its transmission but it is blocked in the routers as its transmission's inherited priority is lower than the priority of the transmission from App1. As App2 does not know how long the blocking will take it is stalling the core, thus App3 is also blocked and can resume its execution only when both App1 and App2 finished their transmissions.

An alternative, and desired, solution is suspension-based locking which allows to suspend an active application, i.e., App2, waiting for a resource to let other (ready) applications, such as App3, execute even if they have a lower priority, see Fig. 3.27.b). Therefore, the blocking time of App2 remains unchanged but is moved from the NoC to the core, which allows to re-use the waiting/stall time to increase performance and improve guarantees for other tasks. This requires the feedback on the state of the NoC resources (e.g. number of currently active senders and duration of their transmissions) which is not supported by the majority of existing NoC architectures, cf. Chapter 2.

3.7.3 Control Layer Support for Suspensions during NoC Accesses

This section presents a straight forward extension of the control layer protocol which allows to orchestrate both computation and communication in real-time multicores by supporting suspension-based locking for transmissions from tasks sharing the NoC i.e. accesses to NoC resources.

Supporting the suspension based mechanism requires the extension of the synchronization protocol. The description of this extension is based on [59]. Consider the same example as previously. Figure 3.28-(a) illustrates the evolution over time of the task execution when the control layer is applied to arbitrate between interfering transmissions on the shared NoC. Whenever a scheduled task running on the processor is trying to start a transmission, its request is trapped by the NI/scheduler and the task is stalled. The corresponding NI/scheduler sends a request message reqM sg to the RM to obtain an access to the network. The RM is equipped with a queue for storing pending requests from clients/NIs. Consequently, the RM has a knowledge about the global state of the interconnect,

i.e., which transmissions/tasks are active and which resources (links and buffers) are occupied. The scheduler decides about which request from the queue should be served first. After that, the RM must notify the selected application for service with the ackM sg. After receiving the ackMsg, the communication may start. Once granted the access to the NoC, the execution of the stalled task on the processor is resumed and the connection holds until the end of the transmission. The end of the transmission is signaled with a relMsg issued to the RM. As soon as the relMsg arrives, the RM considers the NoC to be free again.

For achieving suspensions, RM must have the capability to propagate the information about the state of a particular transmission, e.g., if it is blocked and for how long. The extension is relatively simple as it requires only introducing a new block message -msgBlk- to the synchronization protocol. Whenever a request for a particular transmission arrives and the NoC is busy, the RM issues the msgBlk message, as depicted in Figure 3.28- (b), to the requesting node. After receiving this information, the on-core scheduler suspends the requesting task. Similarly, whenever the transmission is re-scheduled by the RM, the arrival of the msgAck must be forwarded to the on-core scheduler.

3.8 Interface Between NoC and Memory

The worst-case timing of running new complex applications in real-time and safety critical domains, e.g. autonomous driving, is mainly determined by the latency of accesses to a complex memory hierarchy. Consequently, during the system's runtime applications typically access one or more main memory controllers, scratchpad memories, and even one or more levels of cache memories. Some of the elements of the hierarchy are shared only between tasks running on a particular processing node. This work refers to them as local/on-chip memories and for their arbitration one may apply rules known from existing multicore setups e.g. [15]. However, in MPSoCs selected components of the hierarchy can be jointly used by plurality of tasks running on different processing nodes. This applies especially for high-speed external memories, such as DDR SDRAMs. SDRAMs are often required as on-chip SRAMs cannot provide adequate capacity in a cost-effective manner [7].

3.8.1 Classic Approach for Safe Handling of Accesses to SDRAMs

As already discussed, the major challenge comes from the need of solving a joint arbitration of multiple resources (on-core schedulers, NoC routers and memories). However, even SDRAMs themselves constitute a major problem in safety critical design due to their stateful nature. Indeed, the latency of a single memory operation depends not only on the amount of interfering requests (e.g. from other senders on different nodes), but also on the commands required to serve them [32]. This makes temporal analysis challenging and results in a variable access times making available bandwidth to/from external memory dependent on the traffic history [7].

For instance, given that our NoC admission control enforces the locality of transfers, it is possible to map consecutive physical SDRAM addresses to the same bank/row in a SDRAM device, i.e., one can employ a contiguous address mapping, as depicted in Figure 3.29. This allows to decrease significantly memory response time as consecutive CAS commands that target the same bank row can be executed faster than those that do not. This is because there is no need for pre-charging and activating a row.

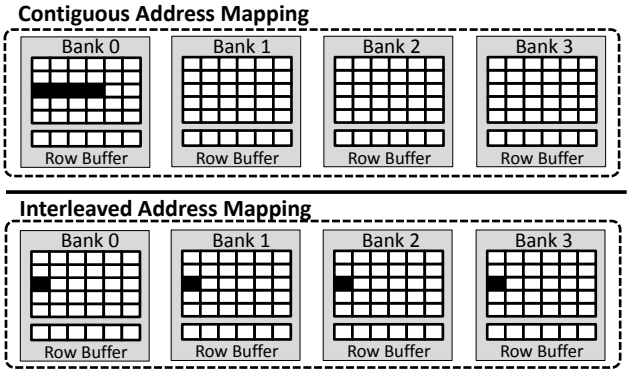


Figure 3.29: Mapping of four consecutive addresses to SDRAM banks in two different mapping mapping configurations (in a system with 4 banks).

Consequently, the majority of available, performance optimized memory controllers (for instance First-Come First-Served arbitration) are either not sufficiently flexible to manage NoC's traffic or are hardly analyzable w.r.t their temporal properties e.g. due to sophisticated features, such as support for preemption and reordering [86]. For instance, in NoCs with large TDM slots (that exceed the granularity of the DRAM controller), DRAM locality is also enforced and, consequently, the same standard SDRAM controller can be employed. However, as described in Chapter 2, large TDM slots have the disadvantage of increasing the latency of all requestors in a system, including those who generate non-SDRAM traffic. Such shortcoming is tackled by employing smaller TDM slots or priority based arbitration in routers. In such setup, safe usage of a performance optimized SDRAM controller would require matching the granularity of the SDRAM with the granularity of the NoC. Due to high granularity of accesses allowing improved scheduling (e.g. contiguous and aligned 8kB long transfers fully benefit from the caching in DDR3) this is either not feasible or drastically decreases utilization of the interconnect.

The alternative would be to map consecutive addresses to different banks, the so called interleaved address mapping. Bank interleaving is attractive when the locality of incoming accesses is not enforced as it allows to hide the latency of the row buffer management. Therefore, researchers and designers proposed dedicated memory controllers exploiting these effect. An example of non-trivial predictable SDRAM controller are Dedicated Close Page-Controllers (DCPC) that employ static bank interleaving and closed-page policy (automatically closing the row buffer after using it) e.g. [7]. However, although DCPC makes response time independent of the access history it also significantly increases power consumption [21].

Consequently, the designers in safety critical domains are confronted with a hard trade-off between cheap, safe but hardly analyzable performance oriented memory controllers or custom made, expensive and power hungry dedicated controllers. The next section describes how the introduced control layer mitigates these shortcoming and allows to protect the locality of memory transfers (low memory latency) without a need for custom memory controllers (lower costs and power consumption).

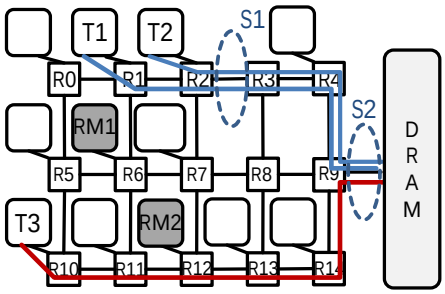


Figure 3.30: Multiple RMs to mitigate the interference on the NoC and memory.

3.8.2 RM-based Admission Control for SDRAMs

The proposed admission control mechanism allows to control interference on the NoC side, but also on the shared SDRAM side. Indeed, since memory traffic constitutes important part of the traffic in an MPSoC system, controlling the interference on the NoC and the memory becomes a main issue in real-time multicore systems. The proposed admission control mechanism allows to consider a holistic approach: NoC and shared SDRAM, as follows: i) it preserves the locality of memory accesses since the access to the NoC is allocated to the entire transmission which allows to fully benefit from the open row policy and to optimize the performance of the system, in addition to ensuring predictability, ii) it mitigates the management of interference between the NoC and the memory controller.

The mechanism is explained using the example in Figure 3.30. Three applications are accessing the shared DRAM memory through the interconnect. Data streams triggered by applications T1 and T2 interfere on the NoC since they are sharing the same path in the network, and therefore form a synchronization scenario S1 managed by the RM1. The data stream triggered by application T3 is using a different path on the network, but is sharing the input to the DRAM controller. Therefore, this traffic stream should also be synchronized using the RM2 with the rest of the traffic in the network - synchronization scenario S2. Note that depending on the tolerable synchronization overhead, a *single* RM can be used to synchronize all the traffic in the NoC accessing the DRAM memory. In this case, whenever an application is granted an access by the RM, it acquires an exclusive access to both resources: the NoC, as well as, to the SDRAM memory. In this case, no predictable memory controller is required in the system since a temporal isolation from other streams is guaranteed by the RM both at the NoC and SDRAM level. Consequently, the control layer moves the arbitration from resource controllers to RM allowing joint arbitration without further need for custom hardware extensions.

3.9 Summary

This chapter introduced a novel Quality-of-Service mechanism for networks-on-chip - the control layer. This novel method uses as basis principles of Software Defined Networks but extends them for purposes of real-time Networks-On-Chip.

The control layer decouples the admission control, which is incorporated on-top of the existing architectures, from the flow-control conducted locally and interdependently in routers. The QoS is

achieved through online adaptation of admission control in nodes based on contract-based negotiation between senders, i.e., providing a validation method to check if the currently available NoC resources are sufficient for the change in QoS before the application becomes physical access to the NoC. The major advantage this approach is that the routers are relieved of the QoS functions, therefore there is no need for custom QoS-oriented NoC extensions. Furthermore, our solution allows the deployment of a sophisticated contract-based QoS provisioning (e.g. round-robin, priority-based arbitration) without introducing complex and hard to maintain schemes, known from static arbiters. Consequently, the QoS control plane allows to dynamically adapt the NoC to the changing system's behavior, mode or environment which can be influenced by on-chip as well as off-chip factors. Moreover, through implementation of the arbitration based on the global state of the network (number of running senders and used by them resources), the control layer introduces efficient interfaces to on-core schedulers as well as peripherals. In the former case, it provides information about state of the transmission (e.g. duration of blocking) which could be improved for improved arbitration of local resources e.g. suspension based scheduling of the CPU time. In the latter, it allows to control order and duration of, i.e. locality, of accesses to selected shared resources. This is especially important in case of state-based schedulers where the history of accesses decides about latency e.g. DDR-SDRAM. Consequently, introduced approach allows achieving safety along with online adaptivity for high performance real-time system with dynamic communication requirements.

Chapter 4: Realization of the Control Layer in NoC

The control layer employs a formal, model-based verification for proving the adherence to real-time and/or safety constraints whenever a dynamic adaptation of resource allocation is necessary. From the conceptual point of view, the mechanism is based on a mathematical model providing QoS abstraction of the underlying NoC (data plane). The implementation worklow for a successful deployment of the proposed scheme is shown in Figure 4.1 and can be divided in the following steps:

Step 1: Evaluation of the underlying MPSoC w.r.t. timing properties.

Step 2: Development of the RM protocol/architecture for the selected MPSoC.

Step 3: Evaluation of the introduced protocol w.r.t. temporal overhead.

Step 4: Assessment of the necessary platform extensions w.r.t. implementation overhead.

In a first step (**Step 1**), it is necessary to verify to what extend the underlying MPSoC architecture is already suitable for the desired real-time and safety-critical workloads. In this context, the predictability of a NoC denotes how accurately it is possible to formally evaluate (i.e. calculate or "predict") the observed run-time behavior of the interconnect (e.g. latency of transmissions) without (full) knowledge of run-time workloads (e.g. deployed traffic, senders behavior), cf.[30]. This includes evaluation of the following NoC features:

- (flow-control) arbiters in routers,
- mechanisms for admission control in NI.

Note, that predictability of the NoC does not guarantee the successful deployment of the control layer but just provides information about how difficult it is to compute the timing bounds for the NoC and how hight these guarantees are.Tightness here refers to overestimations which may happen during the analysis. For instance, designer in real-time domains frequently consider only one potentially unrealistic) scenario which can be worse than any observed combination of traffic at runtime, cf.[30]. The main trade-off lies between the quality of analysis results and complexity of the computation. Usually, by increasing the complexity of the analysis one may improve its results e.g. provide models which mimic the behavior of the NoC and traffic with the higher accuracy.

For this purpose, the initial evaluation uses profiles of senders (i.e. benchmarks) and models of the underlying NoCs hardware (i.e. routers and network interfaces). The profiles of senders (behavioral models) can be extracted for real-time and safety critical applications from their specifications. Their behavior and characteristics are usually well described and tested due to the certification (Section 1.1). Examples for the characteristics are the maximal and minimal frequency and number of

initiated transmissions as well as upper limits (i.e. deadlines) for allowed transmission latencies, which do not violate the safety of the system.

The non safety-critical, best-effort (BE) senders constitute a special group of applications which can be integrated within a NoC-based system. They typically do not provide safe application profiles, e.g., general purpose applications running on processors with caches. For simulation of these applications, the synthetic benchmarks can be used (e.g. bursty access patterns using Markov chains) or traces of memory accesses.

The mapping of applications plays also an important role. For designing the control layer, one must consider that in many existing NoC-based MPSoCs, nodes are heterogeneous and constructed of processing cores as well as peripherals, e.g., I/O interfaces, Ethernet or DRAM controllers. The different hardware units have a fixed position in the system. Therefore, when applied to real-time setups, certain critical communication paths are also fixed and enforced by the routing algorithm, e.g., communication from an Ethernet controller to the DRAM memory. Consequently, many of the performance bottlenecks which may occur in the systems cannot be easily solved through mapping or routing modifications.

The results from *Step 1* provide the designer with following outcomes:

- real-time models of the underlying NoC architecture (QoS parameters in routers and NIs which can be controlled with RM and clients) and sender profiles (e.g. worst-case latencies, deadlines, slacks etc.)
- synchronization scenarios, i.e., sets of interfering senders requiring synchronization (cf. Chapter 3)
- Quality-of-Service settings for each synchronization scenario assuring predictable system behavior, e.g., rates enforced by NIs

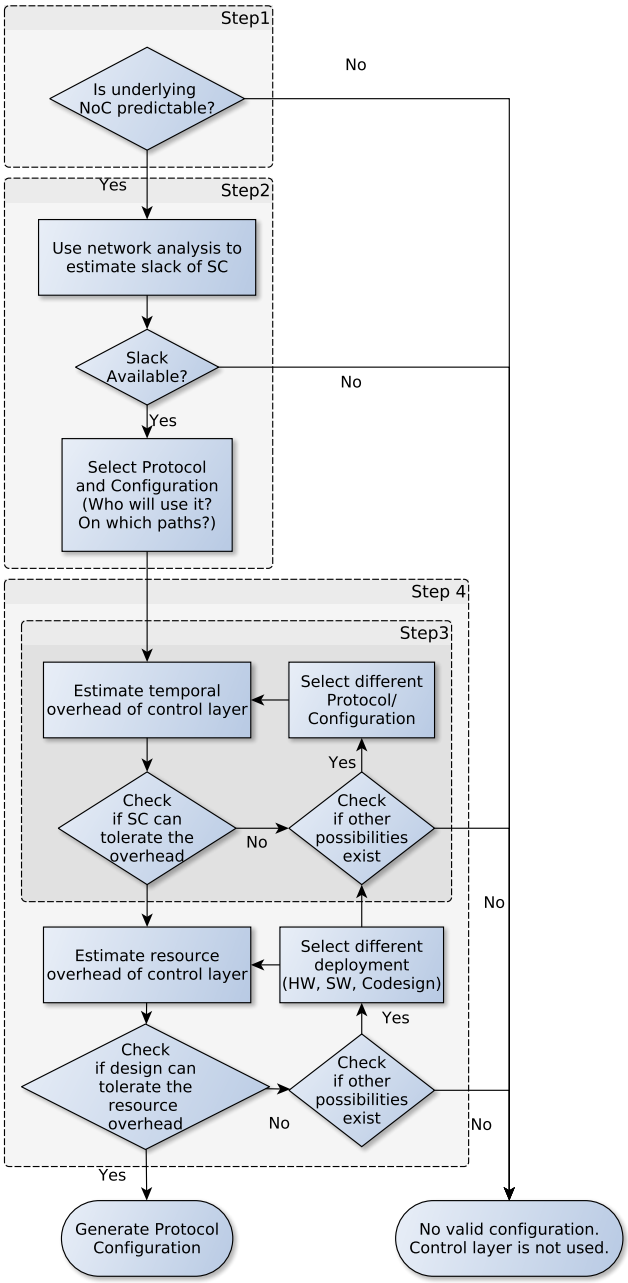


Figure 4.1: Design steps necessary for the implementation of the control layer.

In **Step 2** the design of an appropriate synchronization protocol and control layer elements, i.e., clients and RM, will be done. In this step, it is necessary to identify parts of the NoC architecture which can/must be controlled by RM, e.g., rate limiters, admission control or settings of schedulers in routers. Note, that RM can usually control a subset of this parameters without additional hardware overhead, i.e., there is no need for a new interfaces or APIs. For instance, in many architectures parameters for rate limiters are already propagated using control messages during the system startup. However, it may happen that some of the QoS parameters, although theoretically adjustable, would require slight modifications of existing infrastructure, e.g., extending a router with a configurable flow table similarly to SDN. With knowledge about the elements that can be adjusted it is possible to design the RM protocol, following the arbitration guidelines from Chapter 3. The protocol description should include the workflow, definitions of control messages, and arbitration in RM. The average performance can be evaluated in a simulator (cf. Section 4.1.2) and the worst-case verification derived with formal analysis frameworks (cf. Section 4.1.1).

In **Step 3** the derived protocol architecture will be evaluated w.r.t. the introduced overhead. The QoS control plane, as many other mechanisms for QoS, introduces additional latency resulting from the synchronization. This overhead must be compared to the gain in guaranteed performance achieved using a RM-based arbitration. The results from related research (cf. Chapters 2 & 3) have shown that a self-aware RM control not only offers higher NoC's performance/utilization but, even more importantly, shorter, formally guaranteed latencies for realistic levels of utilization. The reduced latency can typically compensate for the overhead. Consequently, the utilization of the control layer enables solutions for otherwise infeasible configurations. Additionally, Step 3 introduces an iterative process in which throughout gradual improvements a compromise between protocol efficiency and complexity (i.e. introduced overhead) can be found. The final outcome of this stage of the design constitutes detailed requirements for the implementation of the platform extensions required by the control layer. The most important ones are: the number of synchronization scenarios, the frequency of initiated messages for estimating size of the queues in RM and clients, the maximum latency for message processing done by RM and clients, and the maximal mode change latency (if applicable). Here, once again the introduced design tools can be used, i.e., worst-case analysis and simulation (cf. Section 4.1).

Finally, in **Step 4** the actual implementation of the clients and RM is done, using the outcomes of previous design stages. The resource overhead resulting from the integration of the QoS control plane must be considered in the context of a concrete MPSoC. For instance, clients and RM modules can be realized in software (e.g. stand-alone libraries, extensions of the existing OS/RTEs) as well as in hardware (e.g. standalone modules, extensions of NIs). The former offers full flexibility, the latter maximum performance. Note that hybrid solutions (hardware/software co-design) re-using existing chip components are also possible. For instance, as the complexity of a central RM unit is higher than the complexity of a client, it can be implemented in form of a stand-alone processing node. In order to do so, the designer may use supervisor nodes available in many MPSoCs. The process of modules generation for the control layer (RM and clients) may be automatized allowing EDA tools for protocol configuration and design. The implementations of the control layer are flexible and may follow on different virtualization layers requiring variable amount of the system resources. Consequently, Step 4 requires iterations during which different possible setups must/can be evaluated in order to find a compromise between efficiency and overhead. In some cases also

regression to Step 3 is necessary as introduced temporal overhead strongly depends on the method of implementation.

4.1 Design Environment

The design of a multiprocessor system-on-chip is a complex process as on-chip networks integrate a plethora of programmable ECUs as well as other intellectual property (IP) components and peripherals. For evaluation of possible setups (e.g. NoC based MPSoCs running specific benchmark) there are in principle two tools available: *simulation* and *analysis*. The former is the state-of-the-art solution frequently used for verification of MPSoC's performance, e.g., Mentor Graphics Seamless-CVE , Axys Design Automation's MaxSim. Existing tools support cycle-accurate simulation of a complete hardware and software system. Although simulation is time consuming, it allows to evaluate simultaneously (the same environment and benchmarks) the functioning and performance. However, effectiveness of this approach depletes along with increasing complexity of the simulated system. For high number of stimuli (different arbiters, complex traffic patterns etc.) its difficult to assess if the observed/measured results covered all possible corner-cases, i.e., actual worst-case scenario. Therefore, existing simulation tools only offer quick and rough average system estimates but do not reliably cover system-level corner cases.

On the contrary, formal analysis, as e.g. described in [102], offers full corner-case coverage and worst-case performance bounds for critical performance parameters. However, also this technique has a disadvantage: the offered guarantees are frequently pessimistic, i.e., it may cover scenarios which never happen during runtime. This can lead to significant differences between formally analyzed worst case timing and observed worst case timing in simulation or measurement. In order to avoid resource overprovisioning and still be able to provide worst-case guarantees both tools are frequently used in conjunction.

4.1.1 Temporal Analysis Framework

From the conceptual point of view, the introduced NoC control employs a formal, model-based verification to prove the adherence to constraints whenever a safe adaptation is necessary. Consequently, the mechanism is based on a QoS abstraction of the underlying NoC (data plane) allowing a path oriented arbitration approach. For assuring safety, the calculation of the settings is done at design time based on system-level, worst-case timing analysis methods. The analysis uses profiles of senders and models of the underlying NoC's hardware. The modeling of the underlying arbiters in routers can be conducted with any of the available frameworks for the formal analysis of NoCs e.g. [29],[62], [91].

The profiles of real-time and safety-critical senders can be easily extracted as the behavior and characteristics of real-time applications are usually well specified and tested. Examples for the characteristics are the maximal and minimal frequency and number of initiated transmissions (e.g. amount of transmitted data) as well as upper limits (i.e. deadlines) for transmission latencies, which do not violate the safety of the system. This models can be later translated into the settings for the rate controllers on each processing node, i.e., the system must assign for each real-time and/or safety critical sender a sufficient bandwidth share to reach its deadlines and requested throughput.

Using these inputs, the worst-case analysis derives the information about the schedulability of integrated workloads, e.g., if all real-time transmissions meet theirs deadlines. This is done through maximization of the response time, denoting the duration between the beginning (activation) and termination of each transmission. Moreover, the analysis provides also indications about the resources required for achieving these results, e.g., maximal number of outstanding packets in routers buffers.

This initial results are used as a foundation for incorporation of the control layer in the design. The online adaption of the system behavior with the control layer will induce additional overhead to the system. Regarding the timing overhead, it is possible to distinguish between two major sources of overhead resulting from the control layer: (i) control messages (i.e. transmission latency, interference), and (ii) complexity of used protocol (e.g. the different system behavior due to the protocol, frequency of mode changes). These factors must be accounted for evaluation of the systems worst-case performance. Safe application of the control layer is possible whenever all real-time senders belonging to the synchronization scenario have enough slack to compensate protocol overhead. The slack is basically the difference between the maximum time needed to traverse the network and the allowed deadline for traversing the network. The formal NoC analysis of the underlying architecture is used as an input to the analysis of the RM arbiter and the control layer. The analysis models/frameworks for different synchronization protocols are provided in the related work, e.g., round-robin [56], static priorities [58], dynamic priorities [49]. As these are based on well known principles that are also used by commercial tools (e.g. SymTA/S), it is possible to integrate/provide extensions for formal verification of the control layer in existing, commercial toolchains/work flows.

Note, that the worst-case latencies of transmissions synchronized with RMs depend directly on the activities of other senders sharing the resources. Therefore, it is possible to apply temporal-analysis frameworks, such as Real-Time Calculus [8] or Compositional Performance Analysis (CPA) [40], which use abstractions of the worst-case possible access traces (event models), to capture the dynamics of the system behavior. This allows to reduce the pessimism in setups which do not fully utilize the interconnect resource. The main goals of the formal analysis are:

- evaluation of the profit resulting from the control layer w.r.t. the worst-case performance of senders
- evaluation of the temporal overhead resulting from the transmissions of the control messages and the selected arbitration scheme

Through iterative runs and design space exploration the designer should search for the configuration and synchronization method which allows to achieve schedulability of the system as well as improve its performance. As a result of the worst-case analysis the designer can obtain following information:

- worst-case performance of the NoC with the applied control-layer,
- settings for the synchronization protocol and senders which allows meet the temporal requirements.

Recall, that providing the formal guarantees is predominant requirement for safety-critical domains. The effectiveness of the simulation for evaluation of the worst-case behavior of complex

Simulator	Framework	Topologies	Open Source	Hetero -genity	Synchronous / Asynchronous
BookSim [45]	C++	mesh /torus/trees	Yes	No	Synchronous
Noxsim [19]	SystemC	All	Yes	No	Synchronous
Nigram [61]	SystemC	All	Yes	No	Synchronous
Wormsim [41],[68]	C++	mesh/torus	Yes	No	Synchronous
Garnet2.0 [6]	C++ (gem5)	All	Yes	Yes	Synchronous
Sicosys [81]	C++	Mesh/Torus	Yes	No	Synchronous
Nostrum [65]	SystemC	Mesh/Torus	Yes	No	Synchronous
HNOCS [10]	Omnet++	All	Yes	Yes	Both

Table 4.1: Comparison of NoC simulators.

setups is limited. The main problem results from necessity of providing the stimuli to cover all system-level corner cases.

4.1.2 Simulation Tool

Network simulators are the most frequently used tool belonging to the Electronic Design Automation (EDA) designer's toolset for developing electronic systems. On the contrary to prototype testing, simulation can be applied at all abstraction levels. These features are particularly useful during the design of the protocol of the control layer. Consequently, for the protocol design a transaction-level modeling (TLM) approach is used allowing to separate the design of the communication protocol (e.g. details of communication, channel, number of messages, scheduling policy of RM and client, mapping of RM) from the design and implementation of functional units (clients and RM) or of the communication architecture (e.g. router and network interfaces).

The incorporation of the control layer is can be done through extension of existing TLMs imulation tools for NoCs, see Table 4.1, as it is independent from the underlying NoC architecture (cf. Chapter 3). As the choice is largely arbitrary, it is mainly defined by functional properties of the environment, e.g., flexibility, code re-usability, popularity among research community and industrial partners as well as availability of libraries as open-source software.

For simulation purposes, it is necessary to generate the basic underlying NoC architecture (using the given functional and non-functional requirements). Running the NoC simulation based on this with benchmarks should provide the designer with information about the bottlenecks of the design, e.g., heavily loaded links, fully occupied buffers, large routers (many ports). In the context of real-time and safety critical domains, performance (latency, throughput) verification is a major step, covering:

- worst-case performance with synthetic benchmarks for covering of designs corner cases,
- average performance with realistic workloads (and/or synthetic benchmarks).

This initial evaluation should provide the designer with important input about the design: which senders/nodes have unfulfilled requirements w.r.t. timing and which network components (e.g.

routers, NI) may suffer from over provisioning for providing the temporal guarantees (e.g. too many packets in buffers, to wide links required).

Finally, the underlying design may be enhanced with the control layer. This requires incorporation of the following elements (models) to the (TLM) simulation tool:

- extensions of NIs/processing nodes with client logic,
- interfaces necessary for network reconfiguration in NIs,
- deployment of the RM unit (logic)
- communication channel for the control messages (e.g.).

The settings for the control layer (clients, RM and the synchronization protocol) may be obtained through iterative runs and design space exploration. The designer should search for a configuration and synchronization method which allows to achieve schedulability of the system as well as improve its performance. Alternatively, simulation may be used for validation of the results obtained from the formal verification. The main advantages of the simulation framework are:

- rapid development of the main concepts of the operation of the control layer, e.g., placement of the RM, structure of the protocol, size and number of control messages;
- deriving requirements and basic architectural principles for the RTL design of main elements constructing the control layer: placement and scheduling principles of the RM, work principles of clients, size of the buffer queues for RM and clients, size of the state machines etc.;
- verification and validation of the main principles of the system design.

4.1.3 Tool for Protocol Configuration

The initial design phase done with the analysis and transaction-level simulation, introduced in the previous sections, allows designer to evaluate and verify the protocol architecture resulting from the selected resource arbitration method. This work is done on a higher abstraction level (e.g. transactions and contracting) for the purpose of an early exploration of a variety of inputs and possible strategies for resource assignments. As a result, the set of constrains for the low-level design must follow. Examples for the characteristics are: number of synchronization scenarios, maximal and minimal frequency and number of initiated transmissions, states of RM, size of messages as well as upper limits (i.e. deadlines) for transmission latencies, which do not violate the safety of the system. Consequently, a tool can be provided which makes this process (semi) automatic and generates the code of the clients and RM depending on selected and evaluated resource allocation scheme.

This allows introduction of a tool for (semi) automatic generation of the IPs required for the deployment of the control layer, which could extend the capabilities of other EDA toolchains, e.g., tools from Synopsys and Xilinx. The automated design flow, including design evaluation and enhancements, would allow optimized real-time solutions without the need for specialized know-how from the designer. It is critical for enabling "safety as a feature" of the future NoC designs and simplifying the process of system verification.

Chapter 5: Conclusion & Future Directions

The control layer is capable to efficiently accommodate real-time and safety requirements in modern NoC architectures through small extensions of the underlying infrastructure. The main design goal of the mechanism is to achieve temporal predictability without compromising other benefits resulting from application of NoCs in a SoC, e.g., scalability, modularity, flexibility and supreme performance. Moreover, the control layer may enable real-time features also in NoC architectures optimized for average performance (latencies and/or throughput), which, although economically appealing, introduce complex multistage schedulers and therefore are hardly capable of providing formal service guarantees.

The rest of this chapter discusses the proposed mechanism in the context of requirements from Section 1.3. This evaluation summarizes the features, enhancements and limitations of the baseline NoC architecture (with a Static Priority Preemptive scheduler and Rate limiters) with and without the control layer.

5.1 Evaluation against requirements

Section 1.3 described the set of functional and non-functional requirements which NoCs should support for hosting workloads from real-time and/or safety-critical domains. The next paragraphs provide a detailed discussion of how to achieve these goals with the help of the control layer.

R1 Traffic Types

R1 refers to support for multiple classes of real-time traffic, e.g., GL, GT, BE. This goal cane be achieved with the control layer, because:

- the control layer can implement different arbitration methods for different synchronized senders, groups of senders, or even selected nodes in the NoC (parts of the network)
- clients are capable of providing fine-granular support for different admission control mechanisms (rate limiters, address translation tables) what allows to treat differently the short (cache-based) and long (DMA-based) transmissions
- the mechanism permits flexible, efficient and safe incorporation of scenarios where multiple different traffic types share the interconnect. Different resource allocation schemes can be applied to accommodate these traffic classes without the need for modification of underlying router architectures.

R2 Workload Integration

R2 demands from NoCs providing efficient support for real-time requirements without need to modify legacy code and other IPs. The proposed mechanism allows to achieve this goal as the synchronization between the RM and clients can be fully transparent to the legacy code and be implemented as an extension of the underlying NoC infrastructure, e.g., extensions of the NIs. Note, that the accuracy/granularity of synchronization constitutes the main trade-off, as discussed in Chapter 3. Clients must distinguish between different transmissions which requires synchronization with RM. In case of the coarse grained solution client's arbitration is applied to the accumulated workload (traffic from all senders running on this tile). This allows to keep client complexity at a relatively low level as there is no need to recognize and store information about individual transmissions/connections initiated in the tail. The fine-grained integration of workload requires distinguishing between initiated transmissions from individual senders. This can be achieved through extensions of the address translation units (similar to MMU/MPU) in NI. However, for achieving highest efficiency some adjustments of the code running on core my be necessary. An example, would be a new system call which through writing of the appropriate value to the clients register would allow to precisely identify sender. Note, that such extensions usually will require only minor modifications of OS/drivers for tasks running on the tile.

R3 Flexibility

The proposed solution allows to achieve flexibility of the design through support for different resource allocation strategies depending on a particular setup. As discussed, switching between different strategies for resource allocation in NoCs is relatively simple with the control layer. It requires only reprogramming of the RM module which for this purpose can be delivered in software or as a microcoded processor. Note, that its also possible to apply different resource allocation policies within regions of the same NoC. For instance, for selected nodes static priority based scheduling can be applied whereas for other regions strict temporal isolation of senders with TDM. The arbitration can be fine-granular and limited to a specific set of resources (e.g. path through the NoC, virtual channel) or mode of systems work (e.g. performance optimized for regular work and priority based for emergency situations).

Finally, application of different resource arbitration strategies does not require modifications of routers. Therefore, it is possible to offer one chip with a generic router architecture and switch-on the real-time / safety features only in case of concrete deployments, i.e., offer safety and real-time as a feature for specific design.

R4 Dynamics

The mechanism allows to efficiently and safely handle dynamics in system behavior. Firstly, it offers support for work-conserving resource arbitration schemes. Therefore, the workloads are processed as soon as they arrive (or NoC resources are released) minimizing the penalty for senders with variable transmission times or changes in the amount of transmitted data. Furthermore, our solution allows the deployment of a sophisticated contract-based QoS provisioning without introducing of complex and hard to maintain management schemes, known from static arbiters. Consequently,

the QoS control plane allows to dynamically adapt the NoC to the changing system behavior, mode or environment, which can be influenced by on-chip as well as off-chip factors, e.g.:

- in-field software integrations, including upgrades, software modifications, conditional, or mode dependent functions
- transient errors resulting from technology downscaling raising the susceptibility of the hardware
- self-optimization w.r.t. aging (temperature), power consumption, response time, or resource utilization
- predictable fail-operational behavior providing for minimum performance overhead error recovery procedures without functional hazard

Recall, that adjustments of scheduling policy can also be done in already deployed chips, as they usually require only a reconfiguration of the RM unit. This allows to optimize, modify and update the real-time and safety policy along with updates of running applications (workloads). This can be hard or even not possible in case of other QoS mechanisms which are usually integral (hard-coded) part of routers in NoC. Finally, the control layer may provide self-aware arbitration where RM adjusts its policy basing not only on the predefined scheduling algorithm but also considering the monitoring data (e.g. temperature, access patterns). However, although such approach allows far-reaching optimization for achieving high performance, it may be hard to certify such systems due to its complexity. Therefore, this challenge is left for future investigation.

R5 Fairness

The control layer allows NoC infrastructure to provide fair allocation of interconnect resources whenever RT senders have different requirements.

Firstly, the introduced mechanism supports different resource allocation schemes, e.g., round-robin, static and dynamic priorities, which could be applied depending on deployed workloads. The RM has a knowledge about the global state of the NoC - which sender is active and which resources are occupied. Using these information, the RM may decrease or increase the allocated Noc resources (e.g. adjust rate limiters) for a particular node. dynamically depending on the current system mode. Each mode is defined by the number of currently active applications, and determines the minimum time separating every two transmissions issued from the same application. The mechanism is capable of enforcing symmetric and non-symmetric guarantees. In the former, the amount of allocated NoC resources decrease uniformly for all synchronized senders along with the increasing number of connections (transmissions) running in parallel. In the latter, the amount of allocated NoC resources depends not only on the current system mode but also on the sender's particular requirements (e.g. deadline, slack, priority) and may differ between tiles and senders. The non-symmetric mode can be used in a mixed-criticality system to maintain the critical application guarantees while reducing best effort traffic. Consequently, the introduced mechanism enforces behavioral models required to meet the critical timing constraints of the system for each application at runtime .

R6 Mixed-Criticality

Interfering HRTTs and BEs senders are synchronized using the RM. This enables to use a dynamic priority arbitration and thus to give BEs access to the NoC whenever there is an available slack and resources are available.

R7 Switch-off QoS

R7 demands from the NoC architecture providing a possibility (mechanism) to switch-off real-time and/or safety mechanisms whenever there are no deployed senders with such requirements. Achieving this goal is straightforward with the control layer. The QoS arbitration introduced by RM and clients is build on-top of the existing routers. Therefore, by switching of clients it is possible to provide the generic NoC functionality. Note, that the hardware overhead resulting from the unused mechanism (clients and RM) is relatively low and most resources may be released for non real-time setups. Recall that clients are created as extensions of the existing mechanisms in the NIs, thus only small extensions of interfaces and logic are necessary. Moreover, the software deployment of RM and clients is also possible. In this, case the NI should only provide an interface for controlling its QoS mechanisms (e.g. programming of rate limiters or adress translation tables) for a client running as a task on a processing node (e.g. extension of the OS). Similarly, the processing node used for deployment of the RM software can be straightforward recovered and applied for other purposes whenever there is no need for real-time and/or safety. Note also that the deployment of the control layer may be limited to selected parts (regions) of the SoC.

R8 Scalability

Requirement R8 demands from NoC architecture to provide performance (average and worst-case) proportionally to the load at runtime (i.e. work conserving arbitration) and not other static factors, such as the number of senders. For doing so, the control layer allows to introduce a work-conserving scheduling of network traffic, i.e., it always tries to keep the network resources (links and buffers) busy whenever there are pending connections. This feature along with capability to run on-top of performance optimized NoC architecture allows to overcome limitations of the strict temporal or spatial arbitration.

However, there are several design trade-offs which one must consider while implementing the control layer directly influencing the scalability of the approach. This is due to the additional latency resulting from synchronization of the clients and RM with the selected protocol.

Each control message may interfere with other control messages during transmission or processing. This overhead depends on the frequency, number and latency of necessary synchronizations and re-configurations. Note that in the considered embedded safety critical systems, the behavior and characteristics of real-time applications are usually well specified and tested, contrary to the off-chip networks where the behavior of nodes is often unknown at design time. Therefore, it is possible to derive a load distribution to estimate this interference (i.e. overhead of the synchronization) and assess if response times within requested bounds. If the overhead is too high, designer have still several mechanisms to control it:

- Settings for some applications may be kept on a constant level in some or all modes, i.e.,

decrease the number of modes of system work. This allows limiting the number of necessary synchronizations and re-configurations.

- In systems where multiple disjoint sets of interfering applications exist the designer may use different RMs to synchronize each of them independently.
- For maximum performance one may connect the clients and RM with dedicated lines (star topology). In this case only processing delay must be considered.
- Adjust the granularity of the synchronization. The overhead decreases, as an absolute ratio, with increasing length of connection as the protocol overhead is constant w.r.t. the transmission/connection duration.
- Decrease the complexity of the protocol or apply other resource allocation scheme.

The latency of control messages is crucial for the performance of the proposed mechanism. In order to minimize the interference with other traffic in the baseline NoC, the control messages are transmitted using VC with the highest priority. More generally, control messages can be allocated to any available VC capable of giving latency guarantees, a dedicated independent control NoC, using an additional bus, e.g. bus-enhanced NoC; or signal lines for maximum performance.

To summarize, the predominant trade-off resulting from global synchronization is between fine granular adjustments (for increasing performance) and both temporal (interference) and hardware (size of clients and RM) overhead.

R9 Locality

R9 demands that NoC architecture should provide the possibility to prevent interleaving of different transmissions. The proposed admission control mechanism allows to achieve this goal through a considered holistic approach: (i) it preserves the locality of memory accesses since the access to the NoC is allocated to the entire transmission (ii) it allows to adjust order of granted transmissions (connections) for mitigating the management overhead resulting from arbiter in the peripheral (interference between the transmissions in the resource scheduler). This allows for instance RM to provide resource oriented arbitration for decreasing the performance overhead between the NoC and the memory controller as well as improving the (formally proven) worst-case guarantees. The detailed discussion of interface between the control layer and DDR-SDRAM is provided in in Section 3.8.

R10 Verification

R10 states that the NoC architecture should support efficient formal verification for standardization/certification purposes (whenever relevant). The control layer allows to achieve these goals, what has been proven in related publications e.g. [56], [53], [58]. In order to provide the predictability of the system, it is possible to compute a bound on the worst case latency for each sender synchronized with RM. The analysis takes into consideration other interfering senders as well as the overhead of the protocol. The timing relations of the individual transmissions can be abstracted by event models [40] to capture the worst-case and best-case behavior of every possible transmission arrival/activation pattern. Therefore, one may use temporal-analysis frameworks, such as applied

in this work Compositional Performance Analysis (CPA) [40, 27], for capturing the dynamics of the system's behavior with event models.

Additionally, the proposed centralized architecture with RM permits to realize globally optimal network management based on the current network state. This allows to avoid complex (and potentially lengthly) distributed network control protocols and synchronization scenarios which have to account of the network state before they can take appropriate actions, e.g., propagation of blocking, cyclic dependence. Therefore, the control layer frequently allows to provide shorter, formally proven guarantees for end-to-end temporal guarantees in complex SoCs, and or limit the amount of resources (e.g. buffers) necessary for providing such guarantees when compared to other state-of-the-art solutions.

5.2 Evaluation for the baseline NoC architecture

This section provide an evaluation of the baseline architecture (cf. Sec. 3.1). The detailed results are available in Tables 5.1 and 5.2.

The presented mechanism combines the global arbitration for end to end guarantees (whole circuits through NoC) with the local arbitration in routers. This introduces a work-conserving, client-server based, global resource arbitration scheme. This enables support for different protocols fine-granular tuning of the arbitration depending on the deployed workload, e.g., non-preemptive transmissions, efficient handling of mixed-critical setups, improved arbitration fairness or even strict isolation with TDM.

Moreover, the scalablity improves as the mechanism largely decouples the temporal guarantees for synchronized senders from the available hardware resources. For instance, senders of different criticality may share the same virtual channel. Additionally, lack of cyclic dependencies (which frequently appear as a result of local and independent arbitration in routers) can be assured. As an example, the control layer may assure that once granted the connection will proceed without interference from other senders therefore making the buffers in routers oblivious for both average and worst-case performance.

5.3 Summary

The goal of this section is to place the proposed mechanism and its features in the broader context of real-time and safety critical systems. Figure 5.1 presents a summary of the most important finding along with a comparisons to other Quality-of-Service arbitration methods for NoCs. As discussed in the previous chapters, in the vast majority of setups, the control layer is capable of mitigating the challenges and penalties resulting from applications of other QoS schemes. The detailed evaluation is available in related publications e.g. comparison with TDM [56], SPP in routers [58], dynamic priorities [49] or rate-limiting [52]. Below a brief overview of the most important results.

The control layer introduces work-conserving arbitration which allows to achieve high system performance. Indeed, comparison with TDM resulted in up to 80% of improvement in considered scenarios with DMA transfers [56]. Similarly, adjusting dynamically rate control in nodes for handling cache-based traffic resulted in up to 75% of the improvement [52]. The mechanisms offers high flexibility supporting multiple scheduling schemes (time-driven schedulers, static and dynamic pri-

Relevant Requirement	Feature \Mechanism Type	Baseline (PS + VC + RL, static priorities)	RM + Baseline NoC (PS + VC + RL, static priorities)
R1	Quality of Support for GL	medium (depends on senders prio.; num. of VCs, and number of prios.)	high (independent from available NoC resources through different allocation schemes may be directly adjusted to requirements of senders)
R1	Quality of Support for GL	medium	high
R6	Performance of BE senders in mixed-critical scenarios	low(high. avg. latencies)	high (possibility to apply slack-based arbitration) low average latences
R2	Sender Penalty for Var. Trans Size	low	low
R2	Sender Penalty for Var. Jitter	low	low
R5	Arbitration Fairness for GL Senders	low	high
R5	Arbitration Fairness for GT Senders	medium	high
R4	Performance Penalty for others senders in setups with jitter and var. trans	no	no

Table 5.1: Evaluation of the baseline NoC architecture features with and without the control layer, part 1

Relevant Requirement	Feature \Mechanism Type	Baseline (PS + VC + RL, static priorities)	RM + Baseline NoC (PS + VC + RL, static priorities)
R7	Possibility to switch-off QoS	yes (but BE workloads may suffer from the high performance penalty)	yes
R7	Hardware overhead in setups with disabled QoS	medium (but workloads may suffer from the high jitter and out-of-order arrival of packets)	low (most resources from the control layer can be re-assigned for other purposes, possibility to introduce a round-robin based scheduling on top pf routers with SPP)
R8	Support for Scalability	poor (but statically limited to the available HW resources)	good (largely independent from the underyling HW resources e.g. priority-based sharing of the same VC)
R8	Scalability Temporal Penalty	low (if there is enough HW resources)	medium (depends on the protocol overhead number and frequency of synchronized reconfigurations)
R8	Scalability Resources Penalty	high (a VC per priority level)	low (proportional only to the load and independent from resources)
R9	Support for Locality	no (yes only for highest prio)	yes (RM decides about the order of the transmissions therefore non-preemptive priority based schedules are also possible)
R10	Pessimism of formal Verification Static Setups	medium (depends on the HW resources in NoC, num. of VCs and size of buffers, low if there are sufficient resources otherwise high)	low (application of the global scheduling allows to remove cyclic dependencies, efficiently capture dynamics)
R10	Complex Formal Verification Static Setups	low (complexity depends on the amount of available NoC resources low if there are sufficient resources high otherwise)	medium (complexity depends directly on the complexity of the introduced synchronization protocol)

Table 5.2: Evaluation of the baseline NoC architecture features with and without the control layer, part 2

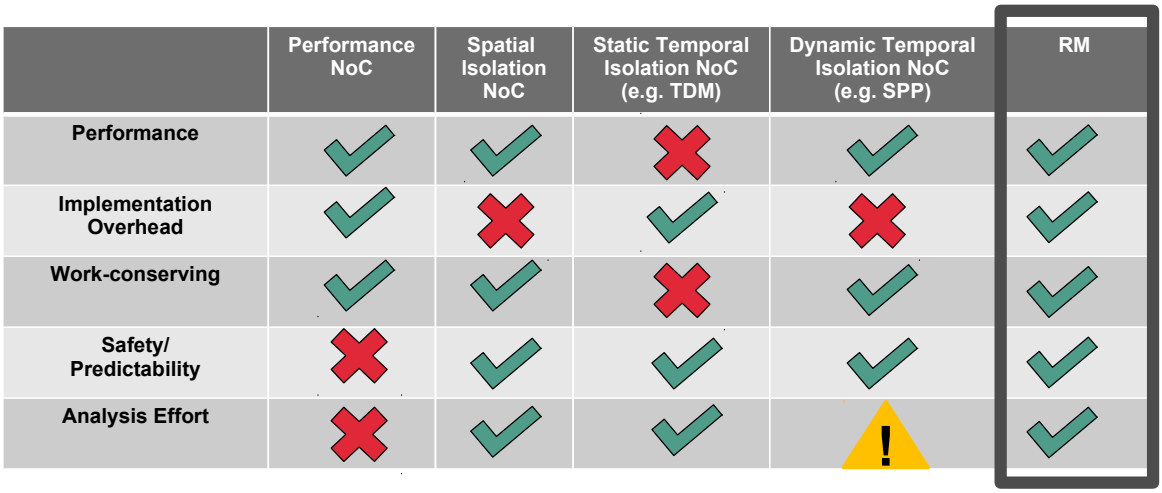


Figure 5.1: Comparison of different QoS mechanisms for NoCs with the control layer. Special focus is placed on safety and performance at the presence of dynamics.

ority based arbitration). Majority of results reported improvements in the range 30% - 70% [56],[58], [49], [52], [51]. Moreover, the introduced interface to peripheral schedulers control layer allows to decrease the latency of the memory accesses while providing safety- up to 70% improvement in case od DDR3-SDRAM [57].

Evaluation from aforementioned works has also shown that the proposed synchronization can be efficiently applied on different granularity levels, e.g. flit, packet, DMA transfer or even whole task activation. This allows to safely adjust the overhead to the acceptable level, e.g. below 5% of the duration of considered NoC access.

Simultaneously, introduced resource management scheme decreases hardware overhead due to the global synchronization. Links and paths can be efficiently shared in time. RM allows to safely limit the load for avoidance of the backpreassure and head-of-line blocking, i.e., no buffer overflows which could endanger safety. Moreover, in highly dynamic priority-based setups, control layer can be used to decouple number of priorities from the available HW-resources, i.e., priority based sharing of the same buffer queue. The work in [56],[58], [49], [52] confirmed that in many deployment setups, including realistic use-cases, the control layer can be efficiently implemented while re-using existing resources (i.e. different NoC layers for control messages, existing resource management cores). Consequently, the hardware overhead is low and feasible for implementation in contemporary and future SoCs Finally, the introduced analysis framework allows not only providing temporal guarantees but even more importantly reaching improved service guarantees in many setups when compared to static resource allocation schemes. This is due to the global point of synchronization simplifying the formal verification and capturing the system wide dynamics. Note, that guarantees can be given for a given architecture without hard predictability and resource (e.g. buffer size) trade-offs as known from other solutions. Consequently, control layer is fulfilling all the evaluation criteria constituting the feasible and appealing alternative for future designs requiring simultaneously high performance and safety.

List of Publications Related to the RM Technology

This appendix provides the list of all publications written by the authors which are disseminating the results of investigations with respect to the control layer technology. *All presented works has been peer reviewed.*

Related to the Report

Adam Kostrzewa, Sebastian Tobuschat, Philip Axer und Rolf Ernst, "Supervised Sharing of Virtual Channels in Networks-on-Chip" in In Proc. of SIES, (Pisa, Italy), Juni 2014.

The article presents the challenges and sketches solutions for global, dynamic and work-conserving arbitration in Networks-on-Chip.

Adam Kostrzewa, Selma Saidi, Leonardo Ecco und Rolf Ernst, "Flexible TDM-based resource management in on-chip networks" in Proceedings of the 23rd International Conference on Real Time and Networks Systems (RTNS), vol. 23, (Lilly, France), pp. 151-160 , 2015.

The article presents the control layer protocol for introducing TDM-based resource management in Network-on-Chip. The discussion is supported by formal worst-case analysis.

Adam Kostrzewa, Selma Saidi und Rolf Ernst, "Dynamic Control for Mixed-Critical Networks-on-Chip" in IEEE Real-Time Systems Symposium (RTSS), (San Antonio, TX, USA), December 2015.

The paper presents the static priority based resource management in real-time Network-on-Chip using the control layer. The work combines the global, work-conserving scheduling for the end to end guarantees with the local arbitration in routers.

Adam Kostrzewa, Selma Saidi, Leonardo Ecco und Rolf Ernst, "Dynamic admission control for real-time networks-on-chips" in Design Automation Conference (ASP-DAC) 21st Asia and South Pacific, (Macau, China), Januar 2016.

The work targets the limitations of the TDM-based network management in NoC. Consequently, it introduces an alternative round-robin based scheme through a dedicated control layer protocol.

Adam Kostrzewa, Selma Saidi, Leonardo Ecco und Rolf Ernst, "Ensuring safety and efficiency in networks-on-chip" , Elsevier Integration, the VLSI Journal, 2016.

The article extend the work from ASP-DAC 2016, by considering memory effect of modern DDR-SDRAM chips. Of special interested are temporal properties and response time in combination with locality of the accesses. Consequently, control layer extensions are proposed for forming an interface between the memory and NoC.

Adam Kostrzewa, Selma Saidi und Rolf Ernst, "Slack-based resource arbitration for real-time Networks-on-Chip" in Design, Automation & Test in Europe Conference & Exhibition (DATE), (Dresden, Germany), April 2016.

The work presents the dynamic allocation scheme for BE sender i.e. providing latency guarantees for hard real-time transmissions with minimum impact on performance sensitive best-effort traffic. This is performed using the control layer.

Adam Kostrzewa, Rolf Ernst und Selma Saidi, "Multi-path scheduling for multimedia traffic in safety critical on-chip network" in 2016 14th ACM/IEEE Symposium on Embedded Systems For Real-time Multimedia (ESTIMedia), vol. 14, (Pittsburgh, PA, USA), pp. 1-10, Oktober 2016.

The work considers multi-path traversals of safety-critical traffic with real-time requirements in a system supervised with the control layer.

Adam Kostrzewa, Sebastian Tobuschat, Selma Saidi und Rolf Ernst, "Supporting Suspension-based Locking Mechanisms for Real-Time Networks-on-chip" in 24th International Conference on Real-Time Networks and Systems (RTNS), vol. 24, (Brest, France), pp. 215-224 , Oktober 2016.

Enabling suspension-based locking requires providing feedback about the global state of the interconnect. For this purpose, in the article the control layer extension is proposed forming an interface between cores and interconnect.

Adam Kostrzewa, Sebastian Tobuschat, Leonardo Ecco und Rolf Ernst, "Adaptive load distribution in mixed-critical Networks-on-Chip" in 22nd Asia and South Pacific Design Automation Conference (ASP-DAC), 2017 .

The publication discusses a protocol-based adaptive load distribution which by selectively detouring BE traffic i.e. load balancing, allows to significantly improve NoC's performance without costly hardware extensions.

Adam Kostrzewa, Sebastian Tobuschat und Rolf Ernst, "Self-Aware Network-On-Chip Control in Real-Time Systems", IEEE Design & Test, 2018.

The article discusses the application of the control layer in the context of the new highly complex embedded applications, e.g. autonomous driving, which require simultaneously high performance and safety in highly dynamic setups.

Bibliography

[1] Do-254 - design assurance guidance for airborne electronic hardware, April 2000.

[2] Do-178B - software considerations in airborne systems and equipment certification, December 2009.

[3] IEC SC 65A. Functional safety of electrical/electronic/programmable electronic safety-related systems. Technical Report IEC 61508, The International Electrotechnical Commission, 3, rue de Varembé, Case postale 131, CH-1211 Genève 20, Switzerland, 1998.

[4] L. Abdallah, J. Ermont, J. l. Scharbarg, and C. Fraboul. Towards a mixed noc/afdx architecture for avionics applications. In *2017 IEEE 13th International Workshop on Factory Communication Systems (WFCS)*, pages 1–10, May 2017.

[5] L. Abdallah, M. Jan, J. Ermont, and C. Fraboul. I/o contention aware mapping of multi-criticalities real-time applications over many-core architectures. In *2016 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 1–1, April 2016.

[6] Niket Agarwal, Tushar Krishna, Li-Shiuan Peh, and Niraj K Jha. Garnet: A detailed on-chip network model inside a full-system simulator. In *Performance Analysis of Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on*, pages 33–42. IEEE, 2009.

[7] Benny Akesson, Kees Goossens, and Markus Ringhofer. Predator: A predictable sdram memory controller. In *Proceedings of the 5th IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis*, CODES+ISSS '07, pages 251–256, New York, NY, USA, 2007. ACM.

[8] S. Azodolmolky, R. Nejabati, M. Pazouki, P. Wieder, R. Yahyapour, and D. Simeonidou. An analytical model for software defined networking: A network calculus-based approach. In *2013 IEEE Global Communications Conference (GLOBECOM)*, pages 1397–1402, Dec 2013.

[9] Mario Baldi, Silvano Gai, and Gian Pietro Picco. Exploiting code mobility in decentralized and flexible network management. In *Proceedings of the First International Workshop on Mobile Agents*, MA '97, pages 13–26, London, UK, UK, 1997. Springer-Verlag.

[10] Y. Ben-Itzhak, E. Zahavi, I. Cidon, and A. Kolodny. Hnocs: Modular open-source simulator for heterogeneous nocs. In *2012 International Conference on Embedded Computer Systems (SAMOS)*, pages 51–57, July 2012.

[11] Konstantin Berestizshevsky, Guy Even, Yaniv Fais, and Jonatan Ostrometzky. Sdnoc: Software defined network on a chip. *Microprocessors and Microsystems*, 50(Supplement C):138 – 153, 2017.

[12] T. Bjerregaard and J. Sparso. Implementation of guaranteed services in the mango clockless network-on-chip. *IEEE Proceedings - Computers and Digital Techniques*, 153(4):217–229, July 2006.

[13] Paul Bogdan and Radu Marculescu. Workload characterization and its impact on multicore platform design. In *Proceedings of the Eighth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, CODES/ISSS '10, pages 231–240, New York, NY, USA, 2010. ACM.

[14] Evgeny Bolotin, Israel Cidon, Ran Ginosar, and Avinoam Kolodny. Qnoc: Qos architecture and design process for network on chip. *J. Syst. Archit.*, 50(2-3):105–128, February 2004.

[15] Björn B. Brandenburg and James H. Anderson. The omlp family of optimal multiprocessor real-time locking protocols. *Design Automation for Embedded Systems*, 17(2):277–342, 2013.

[16] A. Burns, J. Harbin, and L. S. Indrusiak. A wormhole noc protocol for mixed criticality systems. In *2014 IEEE Real-Time Systems Symposium*, pages 184–195, Dec 2014.

[17] Alan Burns and Robert I. Davis. Mixed criticality systems - a review, 9th edition. In *Technical Report, University of York, York, UK.*, 2017.

[18] Alan Burns, Leandro Soares Indrusiak, and Zheng Shi. Schedulability analysis for real time on-chip communication with wormhole switching. *Int. J. Embed. Real-Time Commun. Syst.*, 1(2):1–22, April 2010.

[19] Vincenzo Catania, Andrea Mineo, Salvatore Monteleone, Maurizio Palesi, and Davide Patti. Cycle-accurate network on chip simulation with noxim. *ACM Trans. Model. Comput. Simul.*, 27(1):4:1–4:25, August 2016.

[20] ICT COLLABORATIVE PROJECT Certainty. *D8.2 – Preliminary Methodology*. Seventh Network Programme, 2011.

[21] K. Chandrasekar, B. Akesson, and K. Goossens. Improved power modeling of ddr sdrams. In *Digital System Design (DSD), 2011 14th Euromicro Conference on*, pages 99 –108, 31 2011-sept. 2 2011.

[22] Liu Cong, Wang Wen, and Wang Zhiying. A configurable, programmable and software-defined network on chip. In *2014 IEEE Workshop on Advanced Research and Technology in Industry Applications (WARTIA)*, pages 813–816, Sept 2014.

[23] William Dally and Brian Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[24] R. I. Davis, K. W. Tindell, and A. Burns. Scheduling slack time in fixed priority pre-emptive systems. In *1993 Proceedings Real-Time Systems Symposium*, pages 222–231, Dec 1993.

[25] B. D. de Dinechin, D. van Amstel, M. Poulhiès, and G. Lager. Time-critical computing on a single-chip massively parallel processor. In *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1–6, March 2014.

[26] Benoît Dupont de Dinechin, Yves Durand, Duco van Amstel, and Alexandre Ghiti. Guaranteed services of the noc of a manycore processor. In *Proceedings of the 2014 International Workshop on Network on Chip Architectures*, NoCArc '14, pages 11–16, New York, NY, USA, 2014. ACM.

[27] Jonas Diemer, Philip Axer, and Rolf Ernst. Compositional performance analysis in python with pycpa. In *3rd International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, jul 2012.

[28] Jonas Diemer and Rolf Ernst. Back suction: Service guarantees for latency-sensitive on-chip networks. In *Proceedings of the 2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip*, NOCS '10, pages 155–162, Washington, DC, USA, 2010. IEEE Computer Society.

[29] Jonas Diemer, Jonas Rox, Mircea Negrean, Steffen Stein, and Rolf Ernst. Real-time communication analysis for networks with two-stage arbitration. In *Proceedings of the 11th International Conference on Embedded Software, EMSOFT 2011, part of the Seventh Embedded Systems Week, ESWeek 2011, Taipei, Taiwan, October 9-14, 2011*, 2011.

[30] Jonas Fabian Diemer. *Predictable Architecture and Performance Analysis for General-Purpose Networks-on-Chip*. PhD thesis, TU Braunschweig, 2016.

[31] Guy Durrieu, Madeleine Faugère, Sylvain Girbal, Daniel Gracia Pérez, Claire Pagetti, and Wolfgang Puffitsch. Predictable flight management system implementation on a multicore processor. In *Embedded Real Time Software and Systems*, ERTS '14, 2014.

[32] Leonardo Ecco and Rolf Ernst. Improved dram timing bounds for real-time dram controllers with read/write bundling. In *2015 IEEE Real-Time Systems Symposium*, pages 53–64, Dec 2015.

[33] Rolf Ernst. Bringing dynamic control to real-time nocs. In *16th International Forum on MPSoC for Software-defined Hardware*, Nara, Japan, July 11-15 2016.

[34] Mohammad Fattah, Masoud Daneshtalab, Pasi Liljeberg, and Juha Plosila. Smart hill climbing for agile dynamic mapping in many-core systems. In *Proceedings of the 50th Annual Design Automation Conference*, DAC '13, pages 39:1–39:6, New York, NY, USA, 2013. ACM.

[35] Manil Dev Gomony, Benny Akesson, and Kees Goossens. A real-time multi-channel memory controller and optimal mapping of memory clients to memory channels. *ACM Transactions on Embedded Computing Systems (TECS)*, 2014.

[36] K. Goossens and A. Hansson. The aethereal network on chip after ten years: Goals, evolution, lessons, and future. In *Design Automation Conference*, pages 306–311, June 2010.

[37] Kees Goossens, John Dielissen, and Andrei Radulescu. Aethereal network on chip: Concepts, architectures, and implementations. *IEEE Des. Test*, 22(5):414–421, September 2005.

[38] Boris Grot, Joel Hestness, Stephen W. Keckler, and Onur Mutlu. Kilo-noc: A heterogeneous network-on-chip architecture for scalability and service guarantees. In *Proceedings of the 38th Annual International Symposium on Computer Architecture*, ISCA '11, pages 401–412, New York, NY, USA, 2011. ACM.

[39] A. Hansson, M. Coenen, and K. Goossens. Channel trees: Reducing latency by sharing time slots in time-multiplexed networks on chip. In *2007 5th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pages 149–154, Sept 2007.

[40] Rafik Henia, Arne Hamann, Marek Jersak, Razvan Racu, Kai Richter, and Rolf Ernst. System level performance analysis - the symta/s approach. *IEE Proceedings Computers and Digital Techniques*, 2005.

[41] Jingcao Hu and R. Marculescu. Energy- and performance-aware mapping for regular noc architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(4):551–562, April 2005.

[42] L. S. Indrusiak, J. Harbin, and A. Burns. Average and worst-case latency improvements in mixed-criticality wormhole networks-on-chip. In *2015 27th Euromicro Conference on Real-Time Systems*, pages 47–56, July 2015.

[43] Leandro Soares 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.

[44] ISO. Road vehicles – Functional safety, 2011.

[45] Nan Jiang, Daniel U. Becker, George Michelogiannakis, James D. Balfour, Brian Towles, David E. Shaw, John Kim, and William J. Dally. A detailed and flexible cycle-accurate network-on-chip simulator. In *ISPASS*, pages 86–96. IEEE Computer Society, 2013.

[46] Evangelia Kasapaki and Jens Spars. *The Argo NOC: Combining TDM and GALS*. IEEE, 2015.

[47] J. Kim. Low-cost router microarchitecture for on-chip networks. In *2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 255–266, Dec 2009.

[48] John Leslie King. Centralized versus decentralized computing: Organizational considerations and management options. *ACM Comput. Surv.*, 15(4):319–349, December 1983.

[49] A. Kostrzewa, S. Saidi, and R. Ernst. Slack-based resource arbitration for real-time networks-on-chip. In *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1012–1017, March 2016.

[50] A. Kostrzewa, S. Tobuschat, P. Axer, and R. Ernst. Supervised sharing of virtual channels in networks -on-chip. In *Proceedings of the 9th IEEE International Symposium on Industrial Embedded Systems (SIES 2014)*, pages 133–140, June 2014.

[51] A. Kostrzewa, S. Tobuschat, L. Ecco, and R. Ernst. Adaptive load distribution in mixed-critical networks-on-chip. In *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 732–737, Jan 2017.

[52] A. Kostrzewa, S. Tobuschat, R. Ernst, and S. Saidi. Safe and dynamic traffic rate control for networks-on-chips. In *2016 Tenth IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, pages 1–8, Aug 2016.

[53] A. Kostrzewa, S. Tobuschat, R. Ernst, and S. Saidi. Safe and dynamic traffic rate control for networks-on-chips. In *2016 Tenth IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, pages 1–8, Aug 2016.

[54] Adam Kostrzewa, Rolf Ernst, and Selma Saidi. Multi-path scheduling for multimedia traffic in safety critical on-chip network. In *Proceedings of the 14th ACM/IEEE Symposium on Embedded Systems for Real-Time Multimedia*, ESTIMedia'16, pages 37–46, New York, NY, USA, 2016. ACM.

[55] Adam Kostrzewa, Selma Saidi, Leonardo Ecco, and Rolf Ernst. Flexible tdm-based resource management in on-chip networks. In *Proceedings of the 23rd International Conference on Real Time and Networks Systems*, RTNS '15, pages 151–160, New York, NY, USA, 2015. ACM.

[56] Adam Kostrzewa, Selma Saidi, Leonardo Ecco, and Rolf Ernst. Dynamic admission control for real-time networks-on-chips. In *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 719–724, Jan 2016.

[57] Adam Kostrzewa, Selma Saidi, Leonardo Ecco, and Rolf Ernst. Ensuring safety and efficiency in networks-on-chip. *Integration, the VLSI Journal*, 58(Supplement C):571 – 582, 2017.

[58] Adam Kostrzewa, Selma Saidi, and Rolf Ernst. Dynamic control for mixed-critical networks-on-chip. In *Proceedings of the 2015 IEEE Real-Time Systems Symposium (RTSS)*, RTSS '15, pages 317–326, Washington, DC, USA, 2015. IEEE Computer Society.

[59] Adam Kostrzewa, Sebastian Tobuschat, Selma Saidi, and Rolf Ernst. Supporting suspension-based locking mechanisms for real-time networks-on-chip. In *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, RTNS '16, pages 215–224, New York, NY, USA, 2016. ACM.

[60] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, Jan 2015.

[61] M. Gaur V. Laxmi L. Jain, B. Al-Hashimi and A. Narayanan. Nirgam: A simulator for noc interconnect routing and applications' modeling1. In *Workshop on Diagnostic Services in Network-on-Chips, Design, Automation and Test in Europe Conference (DATE'07)*, page 16–20, 2007.

[62] Jean-Yves Le Boudec and Patrick Thiran. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer-Verlag, Berlin, Heidelberg, 2001.

[63] Jane W. S. W. Liu. *Real-Time Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2000.

[64] Z. Lu and A. Jantsch. Tdm virtual-circuit configuration for network-on-chip. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 16(8):1021–1034, Aug 2008.

[65] Zhonghai Lu, Rikard Thid, Mikael Millberg, and Axel Jantsch. Nnse: Nostrum network-on-chip simulation environment. In *In Proc. of SSoCC*, 2005.

[66] K. Mahmood, A. Chilwan, O. sterb, and M. Jarschel. Modelling of openflow-based software-defined networks: the multiple node case. *IET Networks*, 4(5):278–284, 2015.

[67] David Manners. *Auto is fastest growing IC market, says IC Insights*. electronicsweekly.com, (Online on 21.02.2018), 9th November 2017.

[68] R. Marculescu. Worm_sim: a cycle accurate simulator for networks-on-chip. In *Carnegie Mellon University, Website (access 01.11.2017)*, 2017.

[69] R. Marculescu, J. Hu, and U. Y. Ogras. Key research problems in noc design: a holistic perspective. In *2005 Third IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS'05)*, pages 69–74, Sept 2005.

[70] N. McKeown. The islip scheduling algorithm for input-queued switches. *IEEE/ACM Transactions on Networking*, 7(2):188–201, Apr 1999.

[71] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, March 2008.

[72] Kraig Meyer, Mike Erlinger, Joe Betser, Carl Sunshine, Germán Goldszmidt, and Yechiam Yemini. *Decentralizing Control and Intelligence in Network Management*, pages 4–16. Springer US, Boston, MA, 1995.

[73] Naveen Muralimanohar and Rajeev Balasubramonian. Interconnect design considerations for large nuca caches. *SIGARCH Comput. Archit. News*, 35(2):369–380, June 2007.

[74] Mircea Negrean, Sebastian Klawitter, and Rolf Ernst. Timing analysis of multi-mode applications on AUTOSAR conform multi-core systems. In *Design, Automation and Test in Europe, DATE 13, Grenoble, France, March 18-22, 2013*, pages 302–307, 2013.

[75] M. Neukirchner, T. Michaels, P. Axer, S. Quinton, and R. Ernst. Monitoring arbitrary activation patterns in real-time systems. In *2012 IEEE 33rd Real-Time Systems Symposium*, pages 293–302, Dec 2012.

[76] OSEK Group. OSEK/VDX Operating System Specification, 2015.

[77] R. Pellizzoni, E. Betti, S. Bak, G. Yao, J. Criswell, M. Caccamo, and R. Kegley. A predictable execution model for cots-based embedded systems. In *2011 17th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 269–279, April 2011.

[78] Quentin Perret, Pascal Maurère, Éric Noulard, Claire Pagetti, Pascal Sainrat, and Benoît Triquet. Mapping hard real-time applications on many-core processors. In *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, RTNS '16, pages 235–244, New York, NY, USA, 2016. ACM.

[79] D. Pham, S. Asano, M. Bolliger, M. N. Day, H. P. Hofstee, C. Johns, J. Kahle, A. Kameyama, J. Keaty, Y. Masubuchi, M. Riley, D. Shippy, D. Stasiak, M. Suzuoki, M. Wang, J. Warnock, S. Weitzel, D. Wendel, T. Yamazaki, and K. Yazawa. The design and implementation of a first-generation cell processor. In *ISSCC. 2005 IEEE International Digest of Technical Papers. Solid-State Circuits Conference, 2005.*, pages 184–592 Vol. 1, Feb 2005.

[80] A. Psarras, I. Seitanidis, C. Nicopoulos, and G. Dimitrakopoulos. Phasenoc: Tdm scheduling at the virtual-channel level for efficient network traffic isolation. In *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1090–1095, March 2015.

[81] V. Puente, J. A. Gregorio, and R. Beivide. Sicosys: an integrated framework for studying interconnection network performance in multiprocessor systems. In *Proceedings 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing*, pages 15–22, 2002.

[82] S. Quinton, T. T. Bone, J. Hennig, M. Neukirchner, M. Negrean, and R. Ernst. Typical worst case response-time analysis and its use in automotive network design. In *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, June 2014.

[83] Renesans. *R-Car H3 Architecture Overview*. renesas.com, (Online on 21.02.2018), 2017.

[84] Kai Richter. *Compositional Scheduling Analysis Using Standard Event Models*. PhD thesis, TU Braunschweig, IDA, 2005.

[85] E. Rijpkema, K. G. W. Goossens, A. Radulescu, J. Dielissen, J. van Meerbergen, P. Wielage, and E. Waterlander. Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip. In *2003 Design, Automation and Test in Europe Conference and Exhibition*, pages 350–355, 2003.

[86] Scott Rixner, William J. Dally, Ujval J. Kapasi, Peter Mattson, and John D. Owens. Memory access scheduling. In *Proceedings of the 27th Annual International Symposium on Computer Architecture*, ISCA '00, pages 128–138, New York, NY, USA, 2000. ACM.

[87] R. Sandoval-Arechiga, R. Parra-Michel, J. L. Vazquez-Avila, J. Flores-Troncoso, and S. Ibarra-Delgado. Software defined networks-on-chip for multi/many-core systems: A performance evaluation. In *2016 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, pages 129–130, March 2016.

[88] L. Schenato, B. Sinopoli, M. Franceschetti, K. Poolla, and S.S. Sastry. Foundations of control and estimation over lossy networks. *Proceedings of the IEEE*, 95(1):163–187, January 2007.

[89] Bastian Schlich. Model checking of software for microcontrollers. *ACM Trans. Embed. Comput. Syst.*, 9(4):36:1–36:27, April 2010.

[90] Martin Schoeberl, Florian Brandner, Jens Spars, and Evangelia Kasapaki. A statically scheduled time-division-multiplexed network-on-chip for real-time systems. In *Proceedings of the 2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip*, NOCS '12, pages 152–160, Washington, DC, USA, 2012. IEEE Computer Society.

[91] Zheng Shi and Alan Burns. Real-time communication analysis for on-chip networks with wormhole switching. In *Proceedings of the Second ACM/IEEE International Symposium on Networks-on-Chip*, NOCS '08, pages 161–170, Washington, DC, USA, 2008. IEEE Computer Society.

[92] John A. Stankovic, Krithi Ramamritham, and Marco Spuri. *Deadline Scheduling for Real-Time Systems: Edf and Related Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.

[93] John A. Stankovic, Krithi Ramamritham, and Marco Spuri. *Deadline Scheduling for Real-Time Systems: Edf and Related Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.

[94] R. Stefan, A. Molnos, A. Ambrose, and K. Goossens. A tdm noc supporting qos, multicast, and fast connection set-up. In *2012 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1283–1288, March 2012.

[95] R. A. Stefan, A. Molnos, and K. Goossens. daelite: A tdm noc supporting qos, multicast, and fast connection set-up. *IEEE Transactions on Computers*, 63(3):583–594, March 2014.

[96] D. Stiliadis and A. Varma. Latency-rate servers: a general model for analysis of traffic scheduling algorithms. *IEEE/ACM Transactions on Networking*, 6(5):611–624, Oct 1998.

[97] Andrew Tanenbaum. *Computer Networks*. Prentice Hall Professional Technical Reference, 5th edition, 2011.

[98] Daniel Thiele and Rolf Ernst. Formal analysis based evaluation of software defined networking for time-sensitive ethernet. In *Design Automation and Test in Europe (DATE)*, Dresden, Germany, March 2016.

[99] L. Thiele, S. Chakraborty, and M. Naedele. Real-time calculus for scheduling hard real-time systems. In *2000 IEEE International Symposium on Circuits and Systems. Emerging Technologies for the 21st Century. Proceedings (IEEE Cat No.00CH36353)*, volume 4, pages 101–104 vol.4, 2000.

[100] S. Tobuschat, P. Axer, R. Ernst, and J. Diemer. Idamc: A noc for mixed criticality systems. In *2013 IEEE 19th International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 149–156, Aug 2013.

[101] S. Tobuschat and R. Ernst. Efficient latency guarantees for mixed-criticality networks-on-chip. In *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 113–122, April 2017.

[102] S. Tobuschat and R. Ernst. Real-time communication analysis for networks-on-chip with backpressure. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, pages 590–595, March 2017.

[103] S. Tobuschat, R. Ernst, A. Hamann, and D. Ziegenbein. System-level timing feasibility test for cyber-physical automotive systems. In *2016 11th IEEE Symposium on Industrial Embedded Systems (SIES)*, pages 1–10, May 2016.

[104] S. Tobuschat, M. Neukirchner, L. Ecco, and R. Ernst. Workload-aware shaping of shared resource accesses in mixed-criticality systems. In *Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2014 International Conference on*, pages 1–10, October 2014.

[105] Sebastian Tobuschat and Rolf Ernst. Providing throughput guarantees in mixed-criticality Networks-on-Chip. In *2017 30th IEEE International System-on-Chip Conference (SOCC) (SOCC 2017)*, pages 207–212, Munich, Germany, September 2017.

[106] I. Walter, I. Cidon, R. Ginosar, and A. Kolodny. Access regulation to hot-modules in wormhole nocs. In *First International Symposium on Networks-on-Chip (NOCS'07)*, pages 137–148, May 2007.

[107] E. Wandeler and L. Thiele. Interface-based design of real-time systems with hierarchical scheduling. In *12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'06)*, pages 243–252, April 2006.

[108] Hassan M. G. Wassel, Ying Gao, Jason K. Oberg, Ted Huffmire, Ryan Kastner, Frederic T. Chong, and Timothy Sherwood. Surfnoc: A low latency and provably non-interfering approach to secure networks-on-chip. In *Proceedings of the 40th Annual International Symposium on Computer Architecture*, ISCA '13, pages 583–594, New York, NY, USA, 2013. ACM.

[109] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C. C. Miao, J. F. Brown III, and A. Agarwal. On-chip interconnection architecture of the tile processor. *IEEE Micro*, 27(5):15–31, Sept 2007.

[110] Reinhard Wilhelm, Jakob Engblom, Andreas Ermedahl, Niklas Holsti, Stephan Thesing, David Whalley, Guillem Bernat, Christian Ferdinand, Reinhold Heckmann, Tulika Mitra, Frank Mueller, Isabelle Puaut, Peter Puschner, Jan Staschulat, and Per Stenström. The worst-case execution-time problem—overview of methods and survey of tools. *ACM Trans. Embed. Comput. Syst.*, 7(3):36:1–36:53, May 2008.

[111] Bing Xiong, Kun Yang, Jinyuan Zhao, Wei Li, and Keqin Li. Performance evaluation of openflow-based software-defined networks based on queueing model. *Comput. Netw.*, 102(C):172–185, June 2016.

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

[113] Dirk Ziegenbein and Arne Hamann. Timing-aware control software design for automotive systems. In *Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, June 7-11, 2015*, pages 56:1–56:6, 2015.

Glossary

BE best-effort. 2, 4

FAA Federal Aviation Administration. 5

failure Termination of the ability of a system of functional unit to perform a required function.. 8

fault abnormal condition that can cause an element or an item to fail. 8

HRT hard real-time. 2, 3, 5

HRTT hard real-time transmission. 3, 4

IEC International Electrotechnical Commision. 6

QoS quality-of-service. 4

SoC System-on-Chip. 4, 5, 10

SRT soft real-time. 2, 3, 5

SRTT soft real-time transimssion. 3, 4

Adam Kostrzewa, Sebastian Tobuschat, Rolf Ernst
Hans-Sommer-Str. 66
38106 Braunschweig