WCTT bounds for MPI Primitives in the PaterNoster NoC

Alexander Stegmeier
University of Augsburg, Germany
alexander.stegmeier@informatik.uni-augsburg.de

Martin Frieb
University of Augsburg, Germany
martin.frieb@informatik.uni-augsburg.de

Jörg Mische
University of Augsburg, Germany
mische@informatik.uni-augsburg.de

Theo Ungerer
University of Augsburg, Germany
ungerer@informatik.uni-augsburg.de

ABSTRACT

This paper applies several variants of application independent time-division multiplexing to MPI primitives and investigates their applicability for different scopes of communication. Thereby, the scopes are characterized by the size of the network-on-chip, the number of participating nodes and the message size sent to each receiver or received from each sender, respectively. The evaluation shows that none of the observed variants feature the lowest worst-case traversal time in all situations. Instead there are multiple schedule variants which each perform best in a different scope of communication parameters.

Keywords

Real-Time NoC, WCTT, MPI Primitives, TDM

1. INTRODUCTION

As *hard real-time (HRT)* systems must meet deadlines, a *worst-case execution time (WCET)* analysis is necessary to prove the program finishes before its upper timing bound is met. Furthermore, for parallel software the communication between parallel executed threads becomes a relevant issue and processors featuring a large number of cores utilize a *network-on-chip (NoC)* to connect all cores. Several processors connected by a NoC additionally provide distributed memory to all cores and communicate via explicit messages. Thus, the *worst-case traversal time (WCTT)* of those messages is relevant to calculate the overall WCET of parallel applications.

Most existing approaches for calculating a WCTT on NoCs provide application specific traversal times. This strategy causes some disadvantages. If multiple applications run on the same chip in parallel, the applications' WCTTs influence each other. This is because the traffic flows of applications are not isolated against each other within the NoC. Thus, the entire chip and traffic must be considered for the calculation of the application's WCTTs. Another disadvantage occurs when an application already running on the processor needs to be extended or adapted. As changed and added parts of the software affect the legacy component's timing behaviour, the entire application has to be analysed from scratch. Moreover, since the considered application's communication may change, all other applications concurrently running on the same chip must also be analysed to ensure that they still meet their timing constraints or to adapt their communication, respectively.

To overcome these shortcomings, a WCTT estimation independent from traffic generated by nodes which are uninvolved to a considered communication is needed. The *message passing interface (MPI)* is the de-facto standard for message passing in distributed memory systems and encapsulates all communication to primitives applicable independent from applications. Therefore, if they feature a WCTT independent from other traffic, they have to be analysed only once on a specific processor and afterwards can be utilized for multiple programs without further WCTT analysis. Hence, it seems appropriate to provide MPI for message passing in HRT systems based on NoCs. However, an implementation is needed that provides such an upper bound for the WCTT.

There are already concepts available dealing with the issue of upper bounds for traversal times independent to uninvolved traffic. One of them is the *guaranteed service (GS)* for the PaterNoster NoC [8], which forms the basis for this paper. It enables several variants of *time-division multiplexing (TDM)* to calculate an application independent WCTT. So far only the communication of multiple senders transmitting one flit each to a single or multiple receivers has been considered for calculating an upper bound for WCTT [8].

In this paper the calculation rules for the different TDM variants are extended to calculate WCTTs for complete MPI primitives. Based on [8], rules are specified for a communication of one sender to a number of receivers (1:N) and for a communication of a number of senders to one receiver (N:1). Furthermore, concrete implementations of MPI primitives suitable for the applied NoC are chosen and their traffic structure is extracted. Afterwards, the MPI primitives' communications are parsed to 1:N and N:1 communications to deduce rules for determining their WCTT. Besides providing calculation rules several variants of TDM are compared and a statement about their applicability to different scopes of communication structures is given.

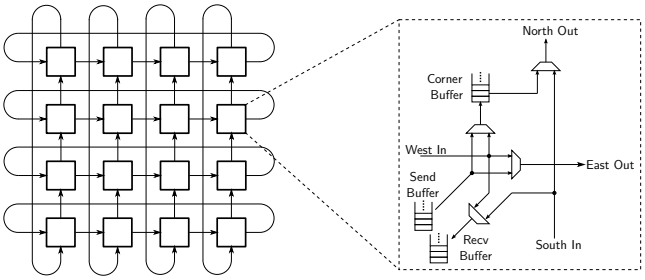


Figure 1: Network structure and router of PaterNoster NoC.

2. RELATED WORK

The Intel Labs presented the Intel *Single-chip Cloud Computer (SCC)* [5] and Scheller [14] investigated real-time programming for this chip. However, the SCC cannot guarantee bounded timing behaviour for each case, as [11] reveals. CompSOC [2] is another platform which focuses on timing-predictability.

There are several methods for GS. The most common ones are rate control for routers or virtual circuit switching applying TDM [3]. Generally, TDM allows each requester to get access to a shared resource for a fixed portion of time within a time interval. The granted time slots are ordered in a way that no conflicts occur when accessing the resource.

There are several approaches to achieve TDM on a specific platform. One option is utilizing a custom schedule, as applied in [3, 19, 15] in different variants. Such schedules are statically defined for the communication of a specific application and require detailed knowledge about the internal communication at design time. Other approaches do not target application specific schedules but apply a generic static schedule independent of the actual communication. Schoeberl et al. [16] propose the use of an All-to-All schedule. This schedule enables each node to send one flit to each other node within one period.

There is a variant of the MPI standard targeting real-time systems [6, 17], but this approach is not very widespread. Sørensen et al. [18] also focus on a time-predictable MPI implementation for a NoC. Thereby, it focuses on end-to-end WCET, but only one TDM schedule is investigated and the NoC size exhibiting 9 nodes is relatively small. In contrast, we concentrate on pure WCTT, but in return explicitly compare different TDM schedules due to their scalability for NoC size, number of participating nodes and message size. Furthermore, Frieb et al. [1] used the results of this paper to calculate the end-to-end WCET of an application.

3. PATERNOSTER NOC

We assume the predicable variation of the PaterNoster NoC [8] to calculate the WCTT bounds. Figure 1 displays the topology of this NoC. Thereby, unidirectional horizontal and vertical rings form a quadratic $n \times n$ torus. Hence, each node can only send flits to the east and north neighbours and receive flits from the south and west neighbours.

We consider a node as a router connected to a local processing element possessing its own memory. In each router flits are forwarded within one processor cycle. When a flit waits in send- or corner-buffer to be forwarded, it can be injected to the NoC if no other flit is forwarded to the same target concurrently.

A strict dimension-ordered routing is applied for delivering flits. At the start of a send action the processing element puts the flit in the send-buffer of the router, where it waits until the insertion to the horizontal ring of the NoC is possible. After insertion in the ring the flit is forwarded bufferless to its target column and stored in the corresponding router's corner-buffer. Here the flit waits until it is inserted in the vertical ring which applies bufferless forwarding as well. When the flit meets its target node it is stored in the receive-buffer where the local processing element can take it for further processing. Subsequently, we consider the WCTT of a flit delivery as the period in time between storing it in the send-buffer and putting it in the receive-buffer of the target. We exclude the time when the flit waits in the receive-buffer because it is mainly influenced by the local WCET of the processing element.

4. GUARANTEED SERVICE WITH TDM

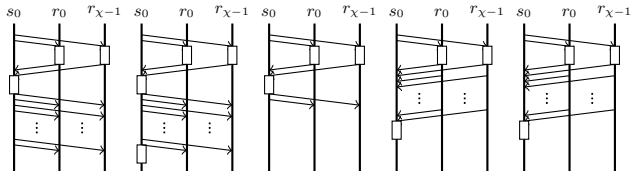
The proposed option for GS is TDM [8]. Applying this method to the considered NoC each send- and corner-buffer gets fixed time slots for forwarding a flit to the horizontal and vertical ring. Thereby, the slots are ordered in a way that no other flit occupies the requested location in the NoC. As there are fixed points in time to insert a flit into the NoC and conflicts are avoided, an upper bound for the time needed to admit and to transport a flit through the NoC can be given. Hence, the WCTT for sending a flit is calculated by adding the admission time t_a and the transportation time t_t of the flit.

Mische and Ungerer [8] present several schedules. These are the schedules called *All-to-All (AA)*, *One-to-All (1A)*, *All-to-One (A1)* and *One-to-One (11)*. The AA schedule is the same as the schedule presented in [16] (see section 2). The 1A schedule allows each node to send at most one flit per period but each node may receive one flit from each other node within one period. The counterpart to 1A is the A1 schedule which allows each node to send one flit to each other node but restricts the total number of sent flits by allowing each node to receive at most one flit in a period. Finally, the 11 schedule restricts the communication by allowing each node to send at most one flit and each node also may only receive one flit per round. To enable communication with different other nodes, several rounds form a 11 schedule that is periodically repeated.

In contrast to custom schedules the schedules AA, 1A, A1 and 11 limit the traffic of all nodes to an amount that enables a statement about the upper bound without considering uninvolved communication. This approach normally leads to a higher upper bound as for a custom schedule, but guarantees application independent temporal isolation of traffic within the NoC. Thus, only traffic of the regarded communication must be considered for the calculation of an upper bound and no other application parts or the placement of the application within the NoC has to be taken into account. As these schedules guarantee a WCTT which is independent from the position of participating nodes in the NoC, no complex method for an optimal placing of tasks is needed.

5. DESIGN OF MPI PRIMITIVES

MPI [9] as de-facto standard for message-passing in distributed memory systems provides the most common com-



(a) broadcast (b) scatter (c) barrier (d) gather (e) reduce

Figure 2: communication structures of MPI collectives implemented for PaterNoster NoC

munication primitives. Thus, we implement a basic subset of MPI for the communication on the PaterNoster NoC. The concept of groups and communicators and also basic point-to-point and collective communication are implemented. Conversely, e.g. no derived data types are allowed and we consider the network is free of hardware faults.

Logically a MPI primitive can be split in several 1:N and N:1 communications. The implementation of such multicast communications can algorithmically take place in different ways, classified as unicast-based and path-based [21]. [7, 12, 13] describe several different unicast-based and path-based algorithms.

Path-based solutions rely on sending one flit to multiple destinations keeping a copy of the flit at the passed target node. These solutions require hardware extension for sending the flit to multiple destinations and copying it at the destination node. Since the PaterNoster NoC does not support such hardware, path-based algorithms are not considered subsequently. Unicast-based algorithms follow another approach. They compose several unicast messages to form a multicast. As they are based on unicast messages no additional hardware is needed and therefore, they are appropriate for the usage in PaterNoster NoC.

Several algorithms exist for unicast-based collective communication [12, 20, 4]. Examples are separate addressing and recursive doubling. Separate addressing is a very simple way to implement a 1:N communication. It performs an unicast from the sender to each receiver and needs N communication steps. However, [10] states that separate addressing can be efficient for short messages and small group sizes. Thus, as a first step, we concentrate on separate addressing and leave other algorithms to future work.

The MPI standard knows several collectives which are implemented for the PaterNoster NoC. These collectives are barrier, broadcast, gather, scatter and reduce. The communication structures exhibited by the PaterNoster implementations utilizing separate addressing are shown in figure 2. Each of them can be subdivided in 1:N and N:1 communication phases and local execution phases. Broadcast begins with a 1:N communication (sending the first flit) followed by a local execution (storing the flit and generating an *acknowledgement (ACK)*) in each receiver. Afterwards, a N:1 communication is performed to send the ACKs back to the sender which in turn processes the received flits locally and delivers all remaining flits by performing a 1:N communication for each flit. The ACKs are needed to check if the receivers are ready to receive further data. With small differences the same holds true for scatter and barrier communication. As scatter sends different data to each receiver and also data for itself, the sender additionally must perform a local execution to copy the fraction of data intended to itself after all flits are sent. The second phase of sending flits to the receivers is bounded to one flit for barrier.

For gather, the node which will receive all data starts with sending an ACK to all participating nodes (1:N) indicating that it is ready to receive data. After receiving and processing the ACKs all senders begin with transmitting the flits to the receiver applying multiple N:1 communications. For reduce, the structure in terms of communication is the same as for gather.

6. CALCULATION OF WCTT BOUNDS

Since the WCTT bounds of primitives implemented with separate addressing are composed of 1:N and N:1 communications, the bounds of these communications are determined first. Thereby, the calculation is based on [8]. There are small differences compared to the time of the original work, because the additional flit needed for the one-ported buffers was not considered originally. Afterwards, selected primitives are investigated in more detail. The other collectives are omitted because of their similarities in communication structure with other collectives and due to space reasons.

6.1 1:N and N:1 communication

As stated in section 3, n indicates the dimension of the considered NoC, which exhibits n^2 nodes. Furthermore, let χ be the number of receivers (1:N) and the number of senders (N:1), respectively. We use f for the number of flits in one message. The time needed to perform one round is indicated as T. Within one round it must be possible to transport a flit to each node of the currently considered ring. Since each hop from one node to the next one takes one cycle and the maximal distance in a ring is $n-1$ hops, the round takes at minimum $n-1$ cycles. Additionally to transportation, the one-ported fashion of the buffers has to be considered. As it is possible to insert flits to buffers (corner-buffer and receive-buffer) from two directions, the length of a round needs to be extended by one cycle to enable the avoidance of conflicts. Hence, one round needs n cycles.

To meet the constraints needed for performing an AA schedule, [8] states that the admission time t_a must be set to $\frac{n^2(n+1)}{2}$. Additionally, in AA a flit stays maximal $\frac{n^2}{2}$ cycles in corner buffer and is transported n cycles per ring. Thus, AA's transportation time t_t is set to $\frac{n^2}{2}+2n$. Since the schedule prohibits sending multiple flits from the same sender to the same receiver, the admission time has to be considered for each flit f. Therefore, the WCTT for a 1:N communication utilizing AA is

$$WCTT_{AA} = \frac{n^2(n+1)}{2} \cdot f + \frac{n^2}{2} + 2n \qquad (1)$$

The argumentation also holds true for N:1 communication with AA and hence, exhibits the same WCTT.

The schedules A1, 1A and 11 require equal time for transportation, particularly maximal one round for each direction (horizontal and vertical) and thus, t_t equals $2 \cdot T = 2n$. As the corner buffers are only used to store flits arrived early to wait for the next round, the time a flit stays in a corner buffer is already included. In contrast to the transportation time, the admission time differs for the considered schedules. While 1A and A1 are characterized by different admission times for 1:N and N:1 communication, the 11 schedule requires equal time for both cases.

Since each flit must be sent to all χ receivers (1:N), in 11 it takes χ rounds until one flit is fully delivered to all destinations. Therefore, the admission of f flits takes $t_a = T \cdot \chi \cdot f = n\chi f$ cycles:

$$WCTT_{11} = n\chi f + 2n \tag{2}$$

In the 1A schedule a node may only send one flit per period but is allowed to receive n^2 flits in a period consisting of n rounds. As a round takes $T = n$ cycles, a period lasts $n \cdot T = n^2$ cycles. When considering 1:N communication, each flit has to be sent to all χ receivers. Hence, χ periods are needed to deliver one flit. Thus, a 1:N communication utilizing 1A takes $t_a = n \cdot T \cdot \chi \cdot f = n^2\chi f$ cycles for admission. In summary the WCTT is

$$WCTT_{1A}^{1:N} = n^2\chi f + 2n \tag{3}$$

In contrast a flit can be received from all senders within one period. Therefore, the admission takes $t_a = n \cdot T \cdot f = n^2 f$ cycles for N:1 communication:

$$WCTT_{1A}^{N:1} = n^2 f + 2n \tag{4}$$

The A1 schedule behaves vice versa to 1A. A node can send n^2 flits and receive only one flit per period. Hence, the schedule exhibits admission times vice versa to 1A for 1:N and N:1 communication. It needs $n \cdot T \cdot f = n^2 f$ cycles for admitting 1:N communication and $n \cdot T \cdot \chi \cdot f = n^2\chi f$ cycles for admitting N:1:

$$WCTT_{A1}^{1:N} = n^2 f + 2n \tag{5}$$

$$WCTT_{A1}^{N:1} = n^2\chi f + 2n \tag{6}$$

6.2 MPI Primitives

The basic primitive in MPI is the point-to-point communication, which involves exactly two nodes. These nodes are the sender and the receiver of a message. Thus, for the calculation of a WCTT bound, a point-to-point communication can be seen as a special case of 1:N or N:1 communication with $\chi = 1$.

The WCTTs of the provided collectives base on the 1:N and N:1 communication. Hence, it is possible to split the collectives in several 1:N/N:1 communications. Subsequently, the execution times for executed code is assumed to be 0 cycles because we focus on traversal times.

According to section 5, a broadcast starts with a 1:N communication of one flit. Afterwards, the ACK is sent back by an N:1 communication and lastly, all remaining flits ($f - 1$ flits) are delivered applying 1:N communication. In the worst case a 1:N or N:1 communication is completely finished before the subsequent delivery begins. Thus, and as we neglect local WCETs, the WCTT of broadcast can be bounded by summing up the WCTTs of all utilized 1:N and N:1 communications, particularly $WCTT^{bcast}(f) = WCTT^{1:N}(1) + WCTT^{N:1}(1) + WCTT^{1:N}(f-1)$. Applying the equations of section 6.1 the WCTTs for the different schedules are:

$$WCTT_{11}^{bcast}(f) = n\chi(f+1) + 6n \tag{7}$$

$$WCTT_{1A}^{bcast}(f) = n^2(\chi f + 1) + 6n \tag{8}$$

$$WCTT_{A1}^{bcast} = n^2(f + \chi) + 6n \tag{9}$$

$$WCTT_{AA}^{bcast} = \frac{n^2(n+1)}{2}(f+1) + \frac{3n^2}{2} + 6n \tag{10}$$

When ignoring the local execution in cores, the WCTT of broadcast and scatter are the same. Furthermore, barrier can be seen as broadcast sending two flits to all participants. Therefore, scatter and barrier are subsequently not investigated in more detail.

Like with broadcast the WCTT of gather is calculated by summing up the WCTTs of performed 1:N / N:1 communications. The collective begins delivering the ACK via 1:N communication and proceeds with gathering the flits of all participants using N:1. Hence, the WCTTs of gather for the different schedules are as follows:

$$WCTT_{11}^{gather} = n\chi(f+1) + 4n \tag{11}$$

$$WCTT_{1A}^{gather} = n^2(f + \chi) + 4n \tag{12}$$

$$WCTT_{A1}^{gather} = n^2(\chi f + 1) + 4n \tag{13}$$

$$WCTT_{AA}^{gather} = \frac{n^2(n+1)}{2}(f+1) + n^2 + 4n \tag{14}$$

The collective reduce exhibits the same communication structure as gather. Like scatter and barrier, the collectives reduce is not further examined, due to similarities in communication structure.

7. EVALUATION

We evaluate each schedule using the formulas for 1:N and N:1 to get a basic understanding of each schedule. Afterwards, the MPI collectives are investigated.

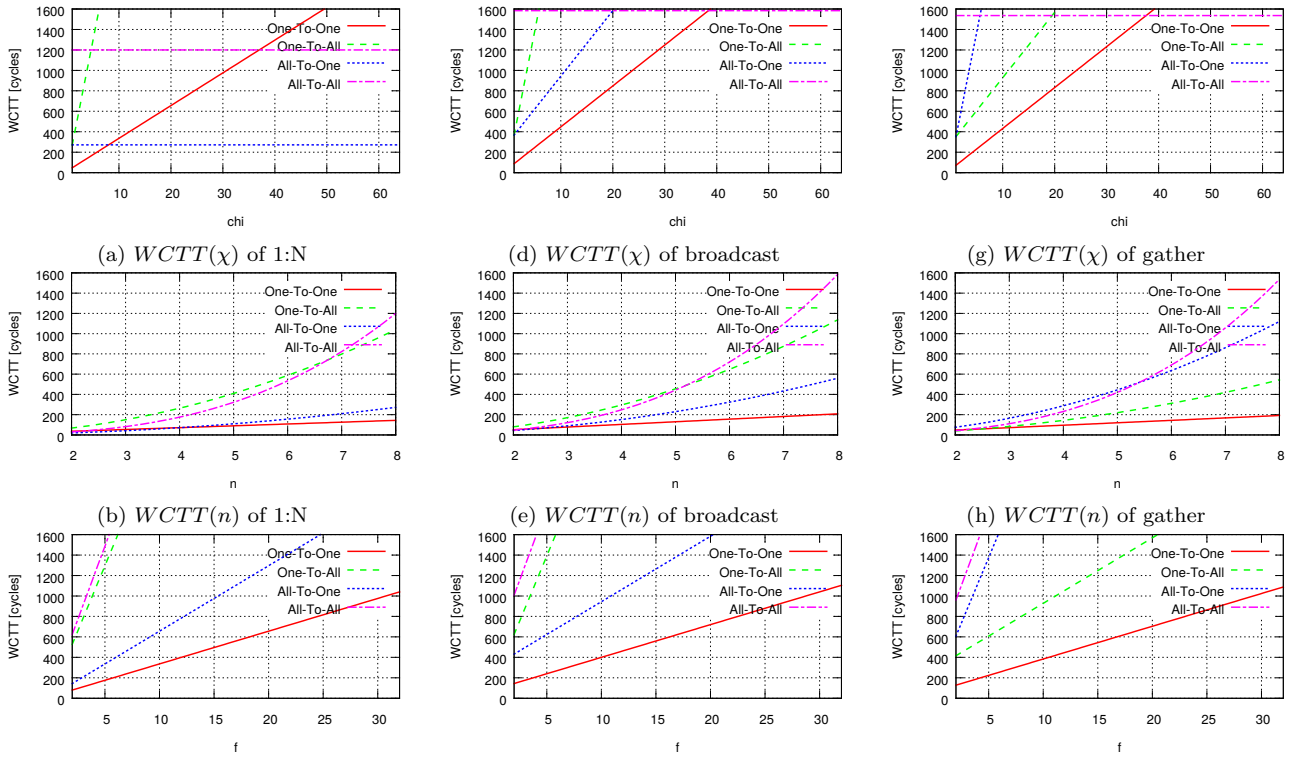
For the evaluation we assume different group sizes (χ), NoC sizes (n^2) and message lengths (f). Two of the mentioned parameters will be fixed and the schedules compared as a function of the third parameter. Finally, a statement about the applicability of the schedules is given.

7.1 1:N and N:1 communication

Figures 3a, 3b and 3c show the behaviour of considered schedules for 1:N communication. As already stated, the AA and 11 schedules exhibit the same behaviour for both kinds of communication, while 1A and A1 behave differently.

Figures 3a displays the schedules' behaviour as a function of χ. Thereby, the dimension of the NoC is fixed to a size $n = 8$ and the delivered message comprises 4 flits. The WCTT of AA and A1 is constant for all group sizes, while the schedules 11 and 1A increase linearly with χ. However, despite AA is not increasing, its WCTT exhibiting 1199 cycles is the highest for $\chi < 5$ and the second highest time for $5 \leq \chi < 37$. 1A performs worse than AA for $\chi \geq 5$ due to a high gradient cause by a factor of n^2 and 11 features a higher WCTT for $\chi \geq 37$. However, the 11 schedule performs best for group sizes smaller than n but is beaten for larger group sizes by A1, which features a WCTT of 272 cycles for all χ.

The behaviour as a function of n is shown in figures 3b. The groups size is fixed to $\chi = 4$ and the same message length is applied as for the investigation of χ. The AA schedule increases fast as it is influenced by n^3, whereas 11 raises arithmetically. 1A and A1 feature a WCTT raising faster than 11 but slower than AA, as both are characterized by n^2. However, the 1A schedule raises faster than A1 schedule. The A1 schedule performs best for $n < 4$. For larger and



(a) $WCTT(\chi)$ of 1:N (d) $WCTT(\chi)$ of broadcast (g) $WCTT(\chi)$ of gather

(b) $WCTT(n)$ of 1:N (e) $WCTT(n)$ of broadcast (h) $WCTT(n)$ of gather

(c) $WCTT(f)$ of 1:N (f) $WCTT(f)$ of broadcast (i) $WCTT(f)$ of gather

Figure 3: WCTT of 1:N (left column), broadcast (middle column) and gather (right column) communication. Each function varies one parameter while the other parameters are fixed to $\chi = 4$, $n = 8$ and $f = 4$, respectively.

equal group sizes, the schedule featuring the lowest WCTT is 11.

Figure 3c shows, that the WCTTs of all schedules increase arithmetically with the number of flits. However, there are big differences in their gradients. The highest WCTT of all schedules exhibits AA. This holds true for all sizes of a message. Even though its performance compared to the other schedules would be better for larger group sizes than displayed, it is always worse than utilizing 11. However, A1 performs better than 11 as soon as χ raises above a size of n.

N:1 communication exhibits the same WCTTs for all schedules as 1:N communication, except that 1A and A1 behave exactly vice versa.

In summary, AA schedule is not suitable for performing pure 1:N or N:1 communication. It is intended to perform a all-to-all communication within one period. Since the considered communications send n times less flits than all-to-all, a lot of free bandwidth remains unused and causes a huge overhead in time. The schedules 1A and A1 are optimised each for one kind of communication, 1A for N:1 and A1 for 1:N. They are able to perform their optimised communication within one period and therefore, are stable due to different group sizes. However, the opposite communication causes significant overhead and performs even worse than AA for moderate and large group sizes. In contrast, 11 fits to both communications, but its WCTT is dependent on the group size. Hence, the schedule performs best for $\chi < n$ in all cases compared to the other schedules and exhibits still competitive traversal times for a moderate χ. Nevertheless, its behaviour is becoming infeasible for large group sizes.

7.2 MPI Primitives

Figures 3d, 3e and 3f show the behaviour of considered schedules for a broadcast communication and figures 3g, 3h and 3i present the performance of the gather collective. Both collectives are based on 1:N and N:1 communication and represent a large range of collectives, as each collective primarily focusing on a different communication structure. Broadcast mainly utilizes 1:N communication, while gather is dominated by N:1 communication. Due to this communication structure broadcast features behaviour similar to 1:N and gather is related to N:1 communication.

As AA and 11 show no differences between 1:N and N:1 communication, they also feature the same tendencies for broadcast and gather communication. Compared to the 1:N or N:1 communication the AA's and 11's WCTT of broadcast and gather is only risen by the ACK flits, which cause one additional admission time and one or two transportation time periods in each schedule. In contrast, the schedules 1A and A1 show significant differences compared to pure 1:N or N:1 communication, since they need to perform both kinds of communication for each collective even though they are optimized for kind of these communication structures. As a result neither 1A nor A1 is independent of the group size any more.

As already stated the main difference for the WCTTs as functions of χ are the numbers of A1 which are not constant any more. Furthermore, 11 raises faster with the group size for broadcast than for 1:N and AA exhibits with 1584 cycles

a larger constant WCTT for broadcast than for 1:N. There are also differences in the WCTT as a function of n. Since the additional ACK takes fewer time for 1A than for AA, 1A performs slightly better than AA when broadcast and pure 1:N communication are compared. In contrary, A1 performs worse than 11 for broadcast compared to 1:N, again due to different costs for the ACK. Comparing broadcast and 1:N for different message sizes (Figure 3c and 3f) it should be noted that all schedules except 1A behave slightly worse for broadcast. Thereby, the difference increases with the number of flits sent.

All statements given for the comparison of broadcast and 1:N communication hold true for the comparison of gather and N:1 except that 1A and A1 must be exchanged. A comparison of broadcast and gather reveals no significant differences except that 1A and A1 behave vice versa to each other. Both schedules feature a higher WCTT than 11 for the investigated MPI primitives in all possible situations. Furthermore, 11 and AA exhibit the same performance for 1:N and N:1 dominated communication structures. Thus, a statement about the applicability of the different schedules can be given for all observed MPI primitives.

For most situations the 11 schedule features the lowest WCTT. It exhibits the lowest growth rate for an increasing n and f, respectively. However, disadvantages arise for large group sizes χ, hence it increases with the number of group members, while schedule AA remains constant. Therefore, the broadcast WCTT of 11 exceeds the one of AA for $\chi > 37$ when sending four flits to each receiver in a NoC of size $n * n = 8 * 8 = 64$ nodes. Thereby, the break even point when the WCTT of AA becomes lower than the one of 11 is dependent from the NoC size (n) and the message size (f). Generally speaking the size of χ at this point becomes lower with an decreasing number of n or f, respectively. Thus, in summary AA performs best for a combination of a small NoC size, message size and a large group size. In other cases 11 exhibits the lowest WCTT. The schedules 1A and A1 are both competitive for either N:1 or 1:N communication, but MPI primitives require mostly both communication structures which neutralizes their positive characteristics.

8. CONCLUSIONS

In this paper we calculated the WCTTs of MPI primitives and compared different schedules of TDM. Separate addressing has been chosen for the MPI implementation and broadcast and gather as representatives for the communication structure are investigated in detail. Thereby, 1:N and N:1 communication acted as basis for calculating the WCTTs.

The results show that in our case the WCTT depends on group size, NoC size and message size. The comparison of the schedules exhibit that for the MPI primitives the lowest WCTT feature 11 and AA schedule, respectively. Even though 1A and A1 schedules show low WCTTs for ether N:1 or 1:N, they are not competitive any more for primitives including both communications.

9. REFERENCES

[1] M. Frieb, A. Stegmeier, J. Mische, and T. Ungerer. Employing MPI Collectives for Timing Analysis on Embedded Multi-Cores. In *16th International Workshop on Worst-Case Execution Time Analysis (WCET)*, 2016.

[2] K. Goossens, A. Azevedo, K. Chandrasekar, et al. Virtual execution platforms for mixed-time-criticality systems: The CompSOC architecture and design flow. *ACM SIGBED Review*, 10(3):23–34, 2013.

[3] K. Goossens and A. Hansson. The Aethereal Network on Chip After Ten Years: Goals, Evolution, Lessons, and Future. In *47th Design Automation Conference*, 2010.

[4] T. Hoefler, T. Mehlan, et al. A survey of barrier algorithms for coarse grained supercomputers. 2004.

[5] Intel Labs. SCC external architecture specification (EAS). Technical report, Intel Corparation, 2010.

[6] A. Kanevsky, A. Skjellum, and A. Rounbehler. MPI/RT-an emerging standard for high-performance real-time systems. In *31th Hawaii International Conference on System Sciences*, pages 157–166. IEEE, 1998.

[7] P. K. McKinley, Y. jia Tsai, and D. F. Robinson. Collective communication in wormhole-routed massively parallel computers. *Computer*, 28(12):39–50, Dec 1995.

[8] J. Mische and T. Ungerer. Guaranteed Service Independent of the Task Placement in NoCs with Torus Topology. In *22Nd International Conference on Real-Time Networks and Systems*, RTNS '14, pages 151–160. ACM, 2014.

[9] MPI-forum. MPI: A Message-Passing Interface Standard Version 3.0, 2012. available at http://www.mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf.

[10] S. Oral and A. D. George. Multicast performance modeling and evaluation for high-speed unidirectional torus networks. *Microprocessors and Microsystems*, 28(9):477–489, 2004.

[11] W. Puffitsch, E. Noulard, and C. Pagetti. Mapping a multi-rate synchronous language to a many-core processor. In *19th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 293–302, 2013.

[12] D. F. Robinson, P. K. McKinley, and B. H. Cheng. Optimal multicast communication in wormhole-routed torus networks. *IEEE Transactions on Parallel and Distributed Systems*, 6(10):1029–1042, 1995.

[13] D. F. Robinson, P. K. McKinley, and B. H. Cheng. Path-based multicast communication in wormhole-routed unidirectional torus networks. *Journal of Parallel and Distributed Computing*, 45(2):104–121, 1997.

[14] J. Scheller. Real-time operating systems for many-core platforms. *Mém. de mast. Toulouse, France: ISAE/ONERA*, 2012.

[15] M. Schoeberl. A Time-Triggered Network-on-Chip. In *2007 International Conference on Field Programmable Logic and Applications*, pages 377–382, Aug 2007.

[16] M. Schoeberl, F. Brandner, J. Sparsø, and E. 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.

[17] A. Skjellum, A. Kanevsky, Y. S. Dandass, J. Watts, S. Paavola, D. Cottel, G. Henley, L. S. Hebert, Z. Cui, and A. Rounbehler. The Real-Time Message Passing Interface Standard (MPI/RT-1.1). *Concurrency and Computation: Practice and Experience*, 16(S1), 2004.

[18] R. B. Sørensen, W. Puffitsch, M. Schoeberl, and J. Sparsø. Message passing on a time-predictable multicore processor. In *18th International Symposium on Real-Time Distributed Computing (ISORC)*, pages 51–59. IEEE, 2015.

[19] 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.

[20] R. Thakur and W. D. Gropp. Improving the performance of collective operations in MPICH. In *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, pages 257–267. Springer, 2003.

[21] Y.-C. Tseng, D. K. Panda, and T.-H. Lai. A trip-based multicasting model in wormhole-routed networks with virtual channels. *IEEE Transactions on Parallel and Distributed Systems*, 7(2):138–150, Feb 1996.