Congestion-Controlled Best-Effort Communication for Networks-on-Chip

J.W. van den Brand[1], C. Ciordas[2], K. Goossens[1] and T. Basten[2]
[1] NXP Research, [2] Eindhoven University of Technology
contact: jan.willem.v.d.brand@nxp.com

Abstract. *Congestion has negative effects on network performance. In this paper, a novel congestion control strategy is presented for Networks-on-Chip (NoC). For this purpose we introduce a new communication service, congestion-controlled best-effort (CCBE). The load offered to a CCBE connection is controlled based on congestion measurements in the NoC. Link utilization is monitored as a congestion measure, and transported to a Model Predictive Controller (MPC). Guaranteed bandwidth and latency connections in the NoC are used for this, to assure progress of link utilization data in a congested NoC. We also present a simple but effective model for link utilization for the model-based predictions. Experimental results show that the presented strategy is effective and has reaction speeds of several microseconds which is considered acceptable for realtime embedded systems.*

1. Introduction

Modern multimedia applications require extensive computation power. Chips with multiple processing units (IPs) can provide this power. Networks-on-Chip (NoCs) provide these so-called Multi Processor Systems-on-Chip (MP-SoCs) with a scalable and flexible interconnect [8]. Examples of NoCs are Æthereal [10], Mango [4] and Xpipes [3].

NoCs provide communication services to IPs. Communication services with guarantees on throughput and latency (GS) enable predictable system design. Guarantees are given by reserving communication resources in the NoC (e.g. wires and buffers). Although necessary for hard real-time applications, this results in poor resource utilization for applications that require variable-bitrate (VBR) communication. Best-effort (BE) is a communication service with no guarantees on latency and bandwidth. It can give high resource utilization by using unreserved or unused resources. However, BE traffic is prone to network congestion. Æthereal [10] and Mango [4] are examples of NoCs that provide both GS and BE services.

Network congestion has a negative effect on network performance [22]. The problem occurs in packet-switched networks when resources, such as links, get saturated. The resulting performance degradation is experienced by BE network users as an increase of latency and loss of bandwidth. Figure 1 shows a shared link transporting both constant-bitrate (CBR) BE traffic and VBR GS traffic with the reserved bandwidth depicted with a dashed line. BE traffic follows the variations of the GS traffic. It improves resource utilization but at certain moments the shared resource is congested.

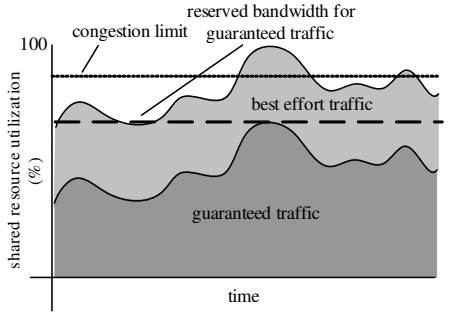


Figure 1. Shared resource without CCBE.

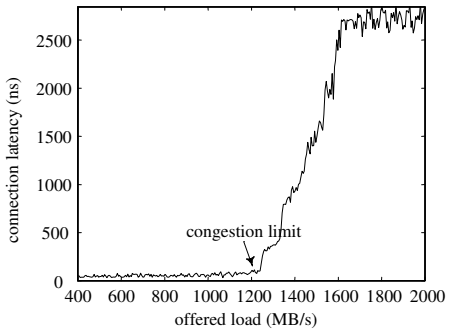


Figure 2. Network latency of an Æthereal BE connection as a function of offered load.

Figure 2 shows network latency of a BE connection as a function of offered load measured for a single connection in a small Æthereal NoC instance. The graph shows that latency is small and almost constant up to a certain turning point after which the latency grows steeply. In this example, the latency saturates at 2600 ns because queuing between IPs and network interfaces is not taken into account.

Networks with BE services should have a strategy to avoid congestion. However, without global knowledge of the network state, such a strategy can never assert that the network does not reach a congested state [22]. Therefore, a network should also have a strategy for resolving congestion. Many strategies for congestion control have been proposed for off-chip networks [1, 15, 12, 22]. On-chip networks pose different challenges. For instance, off-chip environments force networks to allow packet loss and dropping of packets is often used as a means to control congestion [21]. The reliability of on-chip wires and more effective link-level flow-control allows NoCs to be loss-less. This allows use of a simple protocol stack which results in less traffic for the same amount of useful data sent. Therefore, NoC congestion control is a novel problem for the resource constrained on-chip designs.

We propose a strategy for controlling congestion for on-chip networks. The strategy introduces a new communication service level, congestion-controlled best-effort (CCBE), allowing control of offered load based on real-time shared

978-3-9810801-2-4/DATE07 © 2007 EDAA

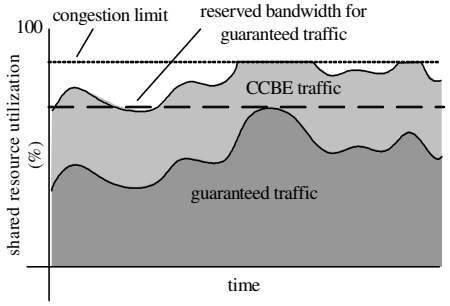


Figure 3. Shared resource with CCBE.

resource utilization measurements. *CCBE connections trade bandwidth for constant and reduced latency.* Figure 3 shows a shared resource with CCBE and VBR GS traffic. It is also possible to combine CCBE with regular BE connections. In such a configuration, regular BE also benefits from the control efforts of CCBE and can be seen as a better quality service because BE IP loads are not controlled.

We use link utilization as congestion measure. Measurements are performed by hardware probes (as proposed in [6]) and are transported to a controller by GS connections in the NoC to assure that this communication is not subject to congestion. The path from controller to IP to communicate the computed loads can be implemented in a similar way.

The controller, a Model Predictive Controller (MPC) [20], determines the appropriate loads for the CCBE connections. It uses a simple link model. Our method requires that routing in the NoC is not dynamic.

The key contributions of this paper are:

i *A new service, congestion-controlled best-effort (CCBE), which bounds latency by controlling NoC load.*

ii *The use of Model Predictive Control for on-chip congestion control.*

iii *A simple but effective model for link utilization.*

The organization of the paper is as follows: related work is discussed in Section 2. In Section 3 we introduce the congestion control method, the controller inputs and its outputs. Then, in Section 4, we quantify the cost of the presented method. Section 5 experimentally demonstrates the effectiveness of MPC for congestion control by showing reaction speeds for different small MPC setups and an MPEG case. Section 6 concludes.

2. Related work

Different solutions for dealing with congestion have been proposed for off-chip networks. For instance, TCP [1] uses a sliding window scheme where packets are allowed to enter the network until packet drop is detected. NoCs are lossless due to the reliability of the on-chip environment; therefore this method can not be used.

Predictive control methods have been presented for off-chip networks because of their ability to deal with uncertain delays. Model predictive control (MPC) is proposed for congestion control for asynchronous transfer mode (ATM) networks in [12]. Buffer filling is modeled in the presented approach. We propose to model link utilization, because link contention is the root cause of congestion.

In [19], a prediction-based flow-control strategy for on-chip networks is proposed where each router predicts future buffer fillings to detect future congestion problems. The buffer filling predictions are based on a router model. The router buffer filling information is used for toggling the sources. Our approach allows both toggling and fluent control of loads offered by IPs.

Dyad [14] deals with congestion by switching from deterministic to adaptive routing when the NoC gets congested. The method can not guarantee that congestion is resolved (i.e. the alternative paths might also be congested); our method always resolves congestion if all BE connections are CCBE.

In [2], an OS communication management scheme is presented that addresses congestion of a BE NoC. The work separates a data from a control NoC to guarantee that control data is not affected by congestion. NI statistics are used as congestion measure. Link-based congestion measurements are more accurate because this is where congestion takes place.

3. NoC congestion control strategy

In this section, we present a novel communication service for on-chip networks. This service, congestion-controlled best-effort (CCBE), controls IP loads to resolve network congestion based on real-time congestion measurements.

We use link utilization as congestion measure, model predictive control (MPC) as controller and IP load as means of control. The principle of CCBE is shown in Figure 4. The figure shows the MPC which gets measured link utilization and desired link utilization as input. Based on these inputs and model-based predictions the controller decides appropriate offered load values for the CCBE connections. These load values go to the cores that use the CCBE service. MPC, the means of control, the congestion measure and the model used for MPC are discussed in this section.

3.1. Model Predictive Control

Model predictive control (MPC) is a technique that combines model-based predictions with actual system measurements [18, 9]. MPC is an optimal control method. These type of controllers are designed by optimizing a cost function and are known for their ability to deal with varying latencies which is critical for our control problem. MPC distinguishes itself from other optimal control methods by solving the optimization problem at runtime. These optimization problems are typically solved by quadratic programming (QP) [20]. We use the MPC from the Matlab MPC toolbox [16] which uses Dantzig Wolfe's method [7] for QP. Stability of MPC can be proved by using a Lyapunov function. See [17] for a detailed discussion on stability of MPC.

In this paper we use a centralized MPC strategy which matches the centralized monitoring service of [6] and copes

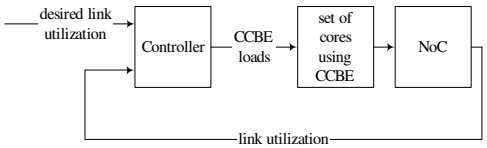
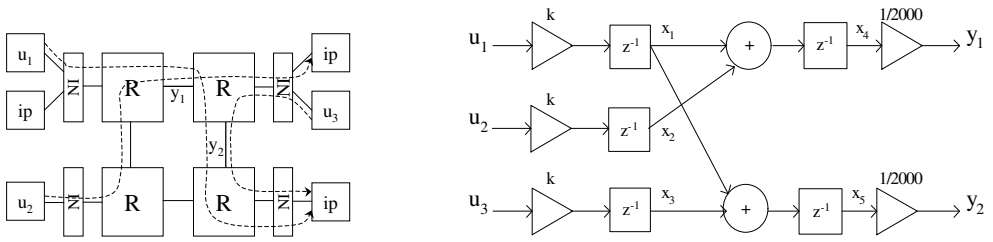


Figure 4. Feedback loop for congestion control.



(a) NoC example. (b) Corresponding flow graph.

Figure 5. Example of utilization model of two links that are shared by three CCBE connections.

well with current NoCs. The use of distributed MPC [5] is regarded as future work and is outside the scope of this paper.

MPCs allow constraints to be specified for controller inputs and outputs, which are the measured link utilizations and the loads offered to CCBE connections respectively (see Section 3.2). Minimum and maximum values can be specified, as well as rise and fall speeds (i.e. how many MBytes/s an IP can raise or drop its load per control interval). MPC takes these constraints into account when making control decisions to ensure that the system will not oscillate.

Control interval and prediction horizon are MPC parameters that affect performance and cost. Each control interval δ the MPC decides on new values for the controlled variables (in our case CCBE offered loads). An important parameter in deciding an appropriate value for δ for NoCs that use time division multiplexing (TDM) is the size of the slot wheel. During a slot wheel, link utilization is highly dynamic. Therefore, δ should at least be a multiple of the slot wheel (e.g. five times the slot wheel). In Section 5, experiments show the influence of δ on the performance of our approach.

During δ, future states are explored over a prediction horizon p. So, each δ, an optimization problem has to be solved by means of QP while considering the effect of decisions over p control steps. These values must be chosen in such a way that computation, area and power cost are acceptable. In practice, choosing $1 \leq p \leq 5$ gives reasonable performance results.

3.2. Means of control

We distinguish two ways of steering network congestion namely control of availability of resources (space) and control of source load (usage). Resource availability can for instance be scaled at runtime by changing the NoC frequency. This is not trivial due to the change in timing behavior of all connections reserved on the NoC. At a source, load can be controlled by using for instance voltage scaling, degrading audio or video quality, or by partially or completely disabling jobs. We choose the option of controlling source loads. The source introduces limits to the amount of control that can be applied. As we saw in Section 3.1, MPC allows constraints to be specified and thus fits well to this means of control.

3.3. Congestion measure

The goal of our congestion control strategy is to bound network latency. Congestion is a resource sharing problem. Links and buffers are the shared resources in packets switched networks. We use link utilization rather than buffer fillings because we think that this is the most direct congestion measure. A link is shared by multiple buffers and lack of space in router buffers is the result of link contention.

Hardware probes, as proposed in [6], are used to measure link utilization. Monitor data is transported from the probes to MPC, by using connections in the NoC. In order to have a reliable system, congestion must have no effect on these connections. Therefore GS connections are used to transport monitor data. This is one of the costs of the proposed congestion control method as further explained in Section 4.

3.4. Network model

MPC uses a model of the controlled system to iteratively compute future behavior. This model must be as simple as possible to minimize the amount of computation for the online QP algorithm. We model link utilization by taking the sum of the loads of the CCBE connections that share the link.

A communication overhead factor k is included for each connection to model the difference between IP load and actual load in the network. For instance, for the Æthereal NoC, BE data is transported through the network as packets. Each packet has a packet header of one word. If a packet size of 36 words is used, $k = 1/36$.

Unit delays following the communication overhead factor model the forward propagation delay from CCBE IP to shared link. The delay from a shared link to the MPC is modeled with a unit delay. By dividing the link load with link bandwidth (2 GBytes/s for Æthereal) we obtain link utilization. Unit delays are used rather than an estimate of propagation delays to keep the model as simple as possible. Estimates improve controller behavior at the cost of a more complex model.

In the model, $\bar{u} = [u_1, u_2, ..., u_m]$ is the input vector which represents the loads of the CCBE IPs. $\bar{y} = [y_1, y_2, ..., y_p]$ is the output vector which represents utilization of the links. $\bar{x} = [x_1, x_2, ..., x_q]$ is the state vector where q equals the number of delays in the model.

In Figure 5(b), two links (y_1 and y_2) from Figure 5(a) are modeled. Connections are represented by dotted lines. Link y_1 shares u_1 and u_2, link y_2 shares u_1 and u_3.

The state space description of the small example of Figure 5(b) is as follows, where n is the discrete time variable, C is the output matrix, A the state-transition matrix and B the input matrix (see for instance [18]):

$$\bar{y}(n) = C\bar{x}(n), \bar{x}(n+1) = A\bar{x}(n) + B\bar{u}(n),$$

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix}, B = \begin{bmatrix} k & 0 & 0 \\ 0 & k & 0 \\ 0 & 0 & k \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

$$C = \begin{bmatrix} 0 & 0 & 0 & 1/2000 & 0 \\ 0 & 0 & 0 & 0 & 1/2000 \end{bmatrix}$$

The state-space model of link utilization is straightforward and its generation can easily be automated from the NoC topology, CCBE connections and routing.

4. Implementation costs

In this section we quantify some of the costs associated with NoC congestion control as proposed in this paper.

We implemented the hardware performance analysis probes. Table 1 shows the area in 0.13 μm CMOS technology for different numbers of monitored links per probe. The area is small compared to an Æthereal router. For instance, a hardware probe that can monitor all ports for link utilization of a six port router has area 0.018 mm^2 which is 10 % compared to the area of a six-port router which is 0.175 mm^2 [10].

#links	**area** (mm^2)
1	0.006
2	0.009
3	0.011
4	0.014
5	0.016
6	0.018
7	0.021
8	0.023

Table 1. Area of hardware probes.

We measure link utilization by accumulating the number of flits that pass a link during a period of time which for our control system is equal to the control interval. The measurement data is sent to the MPC using GS connections in the NoC. Figure 6 shows the load generated by the performance analysis probes as a function of the sample period. For small sample periods, the rate at which monitor data is sent to the MPC is high but the packets are small. Large sample periods result in low rates but require larger packets. We represent the number of flits passing a link during one sample period in bytes. The discontinuity in the figure indicates an extra byte needed for representing the maximum number of flits.

The results show a trade-off between control speed and bandwidth costs. The required bandwidth is small compared to the raw link bandwidth of 2 GBytes/s per link. In the next section we show that good reaction speeds can be obtained for acceptable bandwidth.

In our solution, the MPC controls IP loads. The connections between MPC and IPs is another cost. The required bandwidth for this connection is generally low. It is equal to the number of bits that specify the load divided by the sample period. Bandwidth in the Æthereal NoC is reserved by reserving a certain amount of slots in a slot wheel. Having many low bandwidth connections can result in slot wheel fragmentation. This may need attention when scaling to large numbers of CCBE connections.

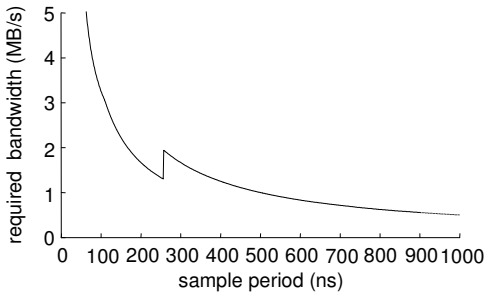


Figure 6. Probe load as a function of sample period.

A final cost is for implementing the MPC. This can be done in software on an embedded processor such as an ARM or via dedicated hardware. The work in [13] shows that it is possible to run MPC for a realistic control problem on a modest FPGA chip. Quantifying the precise cost for implementing MPC for our control problem is regarded as future work. In this context, it is important to observe that the time it takes the MPC to compute new output values is a lower bound for the control interval.

5. Experimental results

In this section we quantify the performance of our NoC congestion control strategy by means of three experiments. First we show the reaction speed of the system by means of pulse responses. Then we show the ability of the system to cope with VBR traffic by means of sine sweeps. Finally, we demonstrate the feasibility of our method by means of a realistic MPEG case study. The NoC and IPs are simulated with a flit accurate SystemC simulator. The MPC controller is modeled in Matlab and a C version is obtained by using the Matlab real-time workshop. The delay of the MPC is not taken into account. The pulse responses and sine sweeps are generated on GS connections by means of traffic generators that are attached to the NoC.

Our BE latency measurements for the Æthereal NoC have shown that 80 % link utilization results in reasonable latencies before the congestion limit in Figure 2. This value is therefore chosen as the target link utilization in the presented experiments. Note that our method works for any target value.

5.1. Pulse responses

For this experiment we use a small NoC consisting of three routers and three network interfaces as shown in Figure 7. The connection from MPC to IP is implemented by a direct link for this test case.

The configuration has two connections, a GS and a CCBE connection. The GS connection offers a constant load to the NoC. The second connection is a CCBE connection which is used for controlling congestion.

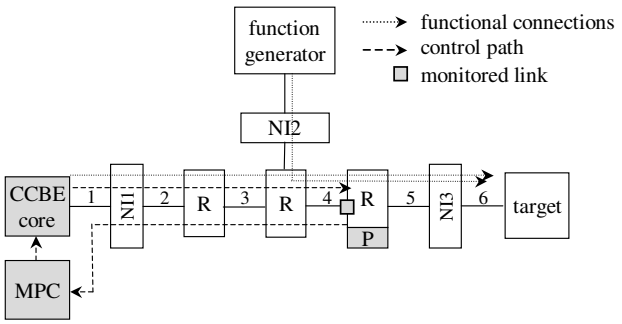


Figure 7. Small setup.

Blocks associated with CCBE are grey. The control path is depicted with a dashed line. The CCBE IPs are controlled by direct communication. Link utilization is monitored at the shared link. The link utilization measure is sent back to the MPC over a connection in the NoC. The configuration has a slot wheel of 8 slots (48 ns).

In the figure, the paths followed by the CCBE and GS connection are shown as a dotted line. In this example, links 4 and 5 (L_4 and L_5) are shared by the two connections. It is sufficient to monitor one of them; we monitor link L_4.

We measured reaction speed by applying a pulse shaped load to the NoC and measuring the time difference between congestion detection and resolved congestion for different control intervals. We use a minimum control interval of 200 ns in this example because smaller intervals result in instable behavior (see Section 3.1).

As expected, the results presented in column 2 of Table 2 show that a small control interval results in a fast reaction speed. However, as shown in Section 4, small control intervals demand higher bandwidth connections and faster QP solving. There is a trade-off between cost and reaction speed.

	reaction speed (μs)		
ctr. int. (ns)	small L_4	MPEG L_6	MPEGL_7
200	4	5	4
400	7	10	8
600	10	12	10
800	13	20	12
1000	15	25	16

Table 2. Reaction speeds for different control intervals in the small and the MPEG example.

5.2. Sine sweep

In the previous subsection the relation between control interval and reaction speed was discussed. The ability of a system to cope with VBR traffic also depends on the control interval of the system.

To see how the system deals with VBR traffic we apply a sine sweep with an amplitude of 100 MBytes/s and a frequency ranging from 2 kHz to 100 kHz to the setup from Figure 7 with a control interval ranging from 200 ns to 1000 ns. We take the frequency at which the sine wave

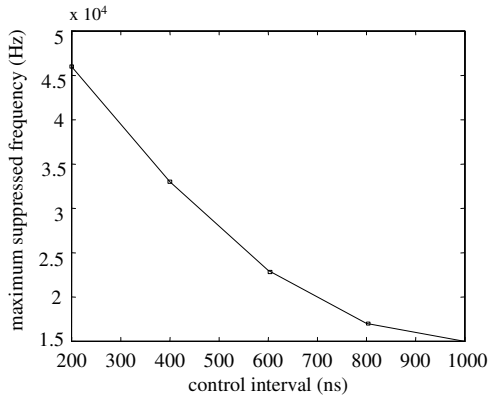


Figure 8. Maximum suppressed frequency as a function of control interval.

is suppressed with 3 dB (factor 0.707) as the maximum suppressed frequency. Frequencies with a time period close to or lower than the reaction speed can not be measured by the system because they are averaged out. Such frequencies will therefore certainly not be suppressed.

Figure 8 shows maximum suppressed frequency as a function of the control interval. The suppressed frequencies are indeed lower than those matching the reaction speed. The maximum suppressed frequency for the shown intervals is slightly higher than 45 kHz. Higher frequencies can be supported by using smaller control intervals but introduce the bandwidth costs presented in the previous section. Appropriate values for the control interval should be chosen based on the desired reaction speed and the expected traffic patterns.

5.3. MPEG example

In this subsection we show that it is feasible to introduce the CCBE service to a realistic system. We use an MPEG2 encoding and decoding system as test case. We show how our approach performs under realistic traffic and with an increased slot wheel (compared to the small example). We measure the increase in link utilization due to the presence of CCBE. Finally, we measure the reaction speeds of the various congested links in the system and we compare the results with those of the previous example.

The MPEG case is taken from [11]. Figure 9 shows the setup for the MPEG case with CCBE. The configuration has a slot wheel of 30 slots (180 ns).

The original case consists of 14 IPs communicating via a shared memory by using 21 bidirectional GS connections through a NoC with a 3x2 mesh topology. Overall link utilization, i.e. the sum of all link utilizations averaged over the number of links, is equal to 24 % for the original configuration. In order to improve utilization we introduce 3 CCBE connections, which for example represent connections of another application using the same platform. These connections follow the paths as displayed in Figure 9 (dashed lines). Hardware probes are attached to each router to monitor all links. The paths of the CCBE connections are presented as dashed lines. The connection for IP CCBE1 traverses three links, none of which is used by other CCBE connections.

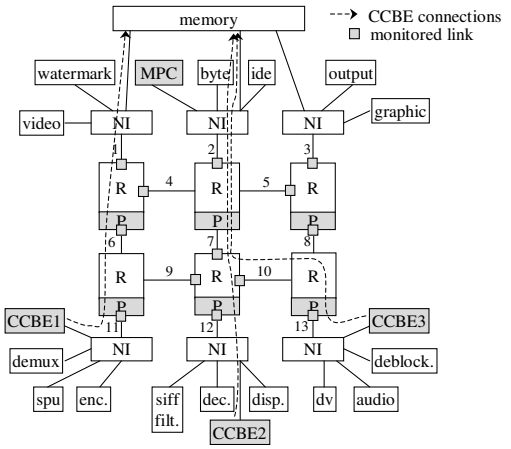


Figure 9. MPEG example.

CCBE2 and CCBE3 share links L_7 and L_2.

The maximum loads of IP CCBE1 to CCBE3 are chosen in such a way that the NoC reaches a congested state. They are: 1000 MBytes/s, 400 MBytes/s and 400 MBytes/s respectively. Attaching these CCBE IPs to the NoC results in an increase of overall link utilization of 16 %. Utilization of link L_{1-3}, L_{6-7}, L_{10} and L_{11-13} are increased significantly by these new loads. However, link L_6 and L_7 are congested (utilization of L_6 equals 100 % and that of L_7 equals 95 %).

We have observed that choosing a control interval of five times the slot wheel size typically results in stable system behavior (see Section 3.1). If this still results in unstable behavior or if smaller control intervals are required, another solution is to constrain rise and fall speeds of CCBE IPs.

We repeat the pulse response experiment by applying the aforementioned loads as pulses to the NoC. To obtain a stable system with the slot wheel of 180 ns but with the control intervals from the previous experiment, we constrain rise and fall speed of the loads of the CCBE IPs to -10 and 10 MBytes/s per control interval. The measured reaction speeds for the two congested links are shown in column 3 and 4 of Table 2 (of Section 5.1). As expected, the reaction speeds for link L_6 are slightly higher than those of the previous experiment (column 2). The reaction speeds for link L_7 are better than those for link L_6. Link L_7 is shared by two CCBE connections. The load of both CCBE IPs are allowed to rise and fall at the same speed as the load of CCBE1 that uses link L_6.

With the MPEG case we have shown that CCBE is feasible for an example with realistic traffic and for multiple shared links and controlled loads in the system.

6. Conclusions

We proposed a congestion control strategy for on-chip networks. The proposed strategy introduces Congestion Controlled best-effort (CCBE) as a new service level. CCBE connections trade bandwidth for constant and reduced latency.

Link utilization is used as congestion measure because link contention is the root cause for congestion. Measurements obtained by hardware analysis probes are sent to a model predictive controller (MPC) which decides CCBE loads based on this information and model-based predictions. MPC is able to cope with the uncertain delays that characterize the NoC congestion control problem. Furthermore, it allows specifications of constraints for the controlled IPs and is scalable in terms of number of controlled IPs and monitored links.

The area of the hardware probes is only 10 % compared to the size of an Æthereal router. Traffic generated by the probes only consumes a small amount of available NoC capacity. Measuring link utilization over a small sample period of 1000 ns only requires 0.3 MBytes/s which is only 0.015 % of the available link bandwidth of 2 GBytes/s.

Experiments show that reaction speeds can be in the order of several microseconds which is generally considered acceptable for realtime embedded systems. An MPEG case study shows that the approach is feasible for realistic systems.

References

[1] G. Almes et al. RFC2581: TCP congestion control. Technical report, Network Working Group, 1999.

[2] P. Avasare et al. Centralized end-to-end flow control in a best-effort network-on-chip. In *Proc. EMSOFT*, 2005.

[3] D. Bertozzi et al. Xpipes: A network-on-chip architecture for gigascale systems-on-chip. *IEEE Circuits and Systems Magazine*, 2004.

[4] T. Bjerregaard et al. A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip. In *Proc. DATE*, 2005.

[5] E. Camponogara et al. Distributed model predictive control. *IEEE Control Systems Magazine*, 22, 2002.

[6] C. Ciordas et al. An event-based monitoring service for networks on chip. *ACM Transactions on Design Automation of Electronic Systems*, 10(4), 2005.

[7] G. Dantzig. *Linear Programming and Extensions*. Princeton university press, 1963.

[8] G. de Micheli et al. *Networks on Chip*. Morgan Kaufmann, 2006.

[9] R. Dorf et al. *Modern Control Systems*. Prentice Hall, 2005.

[10] K. Goossens et al. The Æthereal network on chip: Concepts, architectures, and implementations. *IEEE Design and Test of Computers*, 22(5), Sept-Oct 2005.

[11] K. Goossens et al. A design flow for application-specific Networks on Chip with guaranteed performance to accelerate SoC design and verification. In *Proc. DATE*, 2005.

[12] Y. Gu et al. A predictive congestion control algorithm for high speed communication networks. In *Proc. American Control Conference*, 2001.

[13] M. He et al. Model predictive control on a chip. In *Proc. of the (IEEE) Conference on Control and Automation (ICCA)*, 2005.

[14] J. Hu et al. DyAD - smart routing for networks-on-chip. In *Proc. DAC*, 2004.

[15] S. Mascolo. Classical control theory for congestion avoidance in high-speed internet. In *Proc. Decision and Control Conference*. IEEE, 1999.

[16] Mathworks. Matlab model predictive control toolbox: http://www.mathworks.com/access/helpdesk/help/.

[17] D. Mayne et al. Constrained model predictive control: Stability and optimality. *Automatica*, 36, 2000.

[18] N. Nise. *Control Systems Engineering*. Wiley, 2004.

[19] U. Ogras et al. Prediction-based flow control for network-on-chip traffic. In *Proc. DAC*, 2006.

[20] J. Rossiter. *Model Based Predictive Control, A Practical Approach*. CRC Press, 2002.

[21] A. Tanenbaum. *Computer Networks*. Prentice Hall, 1996.

[22] C. Yang et al. A taxonomy for congestion control algorithms in packet switching networks. *IEEE Network*, 9, 1995.