Traffic Prediction for NoCs using Fuzzy Logic

Gervin Thomas Ben Juurlink
Technische Universität Berlin
Department of Computer Engineering and Microelectronics
Embedded Systems Architectures
Berlin, Germany
Email: {gthomas,juurlink}@cs.tu-berlin.de

Dietmar Tutsch
Bergische Universität Wuppertal
Automation / Computer Science
Wuppertal, Germany
Email: tutsch@uni-wuppertal.de

***Abstract*—Networks on Chip provide faster communication and higher throughput for chip multiprocessor systems than conventional bus systems. Having multiple processing elements on one chip, however, leads to a large number of message transfers in the NoC. The consequence is that more blocking occurs and time and power is wasted with waiting until the congestion is dissolved. With knowledge of future communication patterns, blocking could be avoided. Therefore, in this paper a model is introduced to predict future communication patterns to avoid network congestion. Our model uses a fuzzy based algorithm to predict end-to-end communication. The presented model accurately predictions for up to 10 time intervals for continuous patterns. Communication patterns with non-continuous behaviors, such as fast changes from peak to zero, can also be predicted accurately for the next 1 to 2 time intervals to come. The model is a first step to predict future communication patterns. In addition, some limitations are identified that must be solved in order to improve the model.**

I. Introduction

Increasing the clock frequency to increase performance is no longer an option due to, amongst others, energy consumption, heat developments, and the enormous costs for new technologies [1] [2]. To increase the performance of a chip, processor vendors integrate more cores on one die. The current trend is that the number of cores on a chip multiprocessor (CMP) increases with every new generation and so parallel computing has become more important than ever. The increasing number of cores requires a communication system different from a conventional bus system, since a bus quickly becomes the bottleneck of the system. One approach is to employ a Network on Chip (NoC). With the ongoing trend to increase the number of cores on CMPs, the NoC becomes an essential part of the system.

There are many NoC topologies such as meshes, trees, multistage interconnection networks (MINs), and many more. NoCs have several advantages such as scalability and modularity. The optimal network configuration depends on the application that is running because every application produces different traffic patterns and, moreover, these patterns may change over time. The NoC should realize communications with minor congestion or, if possible, free of congestion. Otherwise, the communication between cores may become the bottleneck. Several researcher [3] [4] have proposed reconfigurable networks to establish communication paths without congestion. The challenge of establishing congestion-free communication depends on the applications that run on the system. Often, congestion arises because several cores send messages at the same time and all messages must be routed through the same network. If two or more messages arrive at the same time at the same switching element and compete for the same link, only one can pass while the others must wait. This situation could be avoided if, before the communication starts, it is already known how much data each core will send and to which core. In that case the routing in the network could be realized with minor congestion by changing the routing algorithm. As another example, if the NoC has a reconfigurable structure, disjoint or lightly loaded paths through the network could be established.

This work presents a method to predict end-to-end communication patterns. Our method is based on a fuzzy algorithm. The prediction method searches for similar pattern in the communication history and predicts based on that information the next data point. By taking the newly predicted data point into account and applying this technique several times, several future steps can be predicted. The method is validated with a chaotic time series and with some real traffic traces obtained on a multicore system.

This paper is organized as follows. Section II describes related work. Section III provides a motivational example and Section IV describes the model that is applied. Section V describes the fuzzy based algorithm that is used to predict end-to-end traffic. Results are presented in Section VI. Finally, Section VII summarizes the paper and presents some directions for future work.

II. Related Work

Huang et al. [5] proposed a table-driven predictor to predict communication in NoCs. Like us, they predicted end-to-end traffic without taking intermediate switches into account. Their method, however, only predicted one future time interval. The predicted amount of communication is either zero or the current quantity. The technique was evaluated by running a modified block LU decomposition kernel on Tilera's TILE64 platform. Kaxiras and Young [6] used coherence communication prediction in distributed shared-memory systems to detect data that is needed by several processors and to deliver the data as soon as possible. Their approach is also table-driven.

Duato and Lysne [7] [8] have proposed a methodology for

deriving procedures for dynamic and deadlock-free reconfiguration between routing functions but did not used any prediction technique.

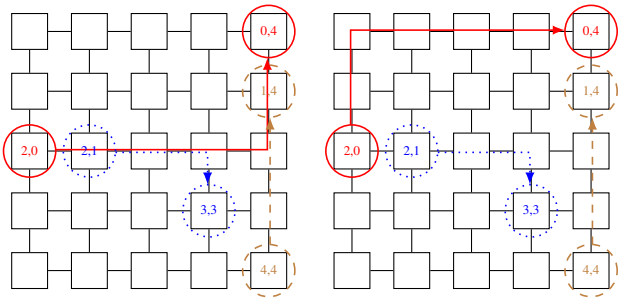
Ahmad [3] introduced a dynamically reconfigurable NoC architecture for reconfigurable Multiprocessor system-on-chip. Hansson and Goossens [4] introduced a library for NoC reconfiguration for dynamically changing the interconnections in dependency of the modules connected to the ports. Both works, however, did not investigate how traffic prediction could improve the reconfiguration of the network.

Chen et al. [9] used a fuzzy based predictive traffic model to avoid congestion at high utilization while maintaining high quality of service in ATM networks. This prediction model was only applied to ATM networks. Pang et al. [10] used a fuzzy traffic predictor and also applied it to ATM traffic management. Results have been presented only for one-step prediction in contrast to our model which predicts several time steps. Otto and Schunk [11] applied fuzzy logic successfully to load forecasting for electric utilities. They did not apply it to other problems, however. Ogras and Marculescu [12] proposed a flow control algorithm to predict switch-to-switch traffic. This prediction is decentralized and based on the information the routers receive directly from their neighbors. From the prediction the number of packets injected in the network is controlled. Brand et al. [13] presented a congestion control strategy based on a Model Predictive Controller which controls the offered load. This method requires that routing is not dynamic, however, in contrast to our model.

The approach presented in this paper differs from and improves upon the ones mentioned above as follows. First, our approach uses a fuzzy based algorithm while previous approaches use a table-driven predictor or flow control algorithm. Second, our method predicts several future time steps which allows to avoid congestion or low utilization in a more flexible way. For example, reconfiguration of a network takes some time and is gainful only when the sum of the reconfiguration time and message transfer time after the reconfiguration is shorter than the transfer time without reconfiguration. It is therefore necessary to predict several time steps ahead to be more flexible for reconfiguration.

III. Motivational Example

Communications that take place at the same time is the reason for blocking in the network. Assume, for example, a mesh NoC topology with 5×5, nodes as depicted in Figure 1. In both figures it is illustrated that node $(2,0)$ communicates with node $(0,4)$ (indicated by solid lines). With dimension-order (xy) routing the communication is established by first routing the message horizontally followed by routing it vertically. Additional messages that are sent simultaneously and need to cross the same links as the first message cannot reach their destination and a congestion occurs. Such an example is shown in Figure 1(a), where an additional message transfer should be established between nodes $(2,1)$ and $(3,3)$ (indicated by dotted line) as well as the nodes $(4,4)$ and $(1,4)$ (indicated by dashed line). These communication cannot



(a) Congested communication (b) Uncongested communication

Fig. 1. Reducing congestion by rerouting communications

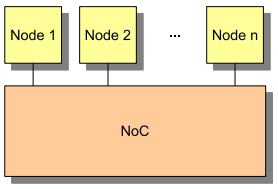


Fig. 2. System as a black box

take place until the first communication releases the switching elements.

With traffic prediction it could be known a priori that the above mentioned nodes want to communicate. With this knowledge a different routing decision could be taken. Alternative routing paths are shown in Figure 1(b). The first communication between nodes $(2,0)$ and $(0,4)$ (indicated by solid line) could be realized by dimension-order (yx) routing which first routes the message vertically and then horizontally. With this new routing decision the other nodes $(2,1)$ and $(3,3)$ (indicated by dotted line) as well as $(4,4)$ and $(1,4)$ (indicated by dashed line) can communicate in parallel. This example illustrates the advantages of traffic prediction to realize blocking free communications. We remark that deadlocks could arise due to the new routing decision, but this is not the main focus of this work, since it can be solved using other techniques such as virtual channels [14].

IV. End-to-end Traffic Prediction

Normally it is important to know the specific NoC topology to be able to analyze it. In order to generalize our method we do not consider the specific network topology. Instead, our goal is to predict end-to-end communication. This means that we do not consider the switching elements between the nodes. It is also irrelevant which type of components (e.g. core, memory, I/O) is connected to the NoC. Every component is simply seen as a node. The NoC is considered as black box to which several node are connected.

The structure of the model is depicted in Figure 2. For the communication between nodes it is important to know which nodes want to exchange information between them and when. Therefore the point of time at which communication takes places and the amount of data that is transmitted are needed.

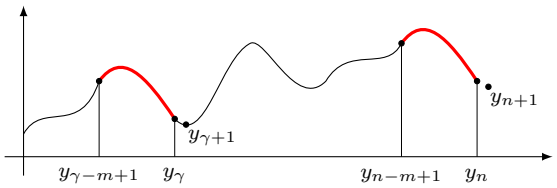


Fig. 3. Search for similar pattern in the history

With these assumptions the problem of predicting traffic in NoCs is similar to predicting a time series.

V. Fuzzy based Traffic Predictor

The proposed traffic predictor is based on [11] and uses fuzzy logic, introduced by Zadeh in 1965 [15]. Fuzzy logic has no strict assignment of elements to sets like binary logic. Instead, every element has a *degree* of membership to a set. This degree is represented by a value between 0 and 1. To be able to apply fuzzy logic to a specific problem such as the prediction of a time series, a fuzzy system must be constructed. The construction consists of three steps:

1) **Fuzzyfication**: In this step the degree of membership of the input values is assigned to fuzzy sets. The degree of membership is given by $\mu : X \to [0, 1]$, where X is the set of input values. So every input value is mapped to a value between 0 and 1.
2) **Fuzzy-Inference**: In this step the output values from the membership function are linked with several different functions to generate an output set.
3) **De-Fuzzyfication**: In this step a numerical output value is generated from the output set.

The above mentioned steps are used to predict a time series. To do so several time steps from the past are needed. The idea behind the algorithm is to consider the latest $m \quad (m < n)$ data points from the time series $Y = (y_0, y_1, \cdots, y_n)$ and then search for some similar patterns in the past. We refer to m as the *pattern length*. The time series Y has $n + 1$ data points. To determine similarity between patterns, fuzzy logic is used. If there are some similar patterns in the past, the algorithm forecasts the next step by interpreting these patterns. This method is depicted in Figure 3. The last data points between (y_n) and (y_{n-m+1}) are compared with pattern from the history communication. If there is a pattern of pattern length m in the past that is very similar to the latest one, like the pattern between (y_γ) and $(y_{\gamma-m+1})$, the algorithm predicts, that the next future point (y_{n+1}) is also very similar to the point that follows the past pattern $(y_{\gamma+1})$. The latest m data points correspond to a sub vector $Y[n-m+1, n] = (y_{n-m+1}, \cdots, y_{n-1}, y_n)$ and this vector is used as a window. That window vector is subtracted iteratively from the past data points, so that in total $j = n - m + 1$ difference vectors $D^{(n-m-i)} = (d_0^{(n-m-i)}, d_1^{(n-m-i)}, \cdots, d_{m-1}^{(n-m-i)})$ are obtained $(i \in [0, n-m])$, where $D^{(n-m-i)}$ is given by

$$D^{(n-m-i)} = Y[n-m-i, n-1-i] - Y[n-m+1, n]. \tag{1}$$

The superscripts indicate the different difference vectors. All elements of the calculated difference vectors are mapped using the membership function $\mu : X \to [0, 1]$ to a value between 0 and 1 which shows the similarity to the original data points from the past. In this work the triangular function, given by

$$\mu(x) = \begin{cases} 1 - \left|\frac{x}{w}\right|, & \text{if } |x| < |w| \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

is used as membership function. In this expression w is the width of the membership function. The width is a degree of how much the latest data points differs from those in the past and can be set by the user. If the difference is too high, the membership function generates the output value 0, which means there is no similarity. Applying the membership function is the first step (fuzzyfication) from the fuzzy system. All j difference vectors are now weighted based on to their similarity. This is done by multiplying all memberships of all elements of a difference vector, as given by the following equation

$$\beta^{(n-m-i)} = \prod_{k=0}^{m-1} \mu\left(d_k^{(n-m-i)}\right). \tag{3}$$

In this equation $d_k^{(n-m-i)}$ is element k of difference vector $D^{(n-m-i)}$. So every difference vector is now reduced to a scalar value which reflects the similarity of the patterns from the past to the last m data points. This step corresponds to the fuzzy-inference step.

From these weights we calculate the next future data point by performing a weighted sum of all past data points. This is done by the following equation

$$y_{n+1} = \frac{\sum_{\gamma=0}^{n-m} \beta^{(n-m-\gamma)} \cdot y_{n-\gamma}}{\sum_{\gamma=0}^{n-m} \beta^{(n-m-\gamma)}}. \tag{4}$$

This step corresponds to de-fuzzyfication.

The steps explained above predict the next future data point. To predict several data points, the algorithm can be reapplied including the predicted data point.

To predict future data points, we need several data points from the past. The more data points are available, the higher the possibility is to find a very similar pattern in the past and increase the accuracy of the algorithm. The disadvantage of using all data points from the past is that the calculation time and memory requirements increase. So a trade-off must be made between the number of considered data points and the accuracy of the algorithm. This trade-off depends on how often some communication patterns repeat. If some communication patterns repeat very often, fewer data points are needed than with less repetitive patterns. The effect of the history length as well as the pattern length m on the accuracy of the proposed algorithm is investigated in Section VI.

VI. Experimental Results and Analysis

In this section experimental results are provided using two types of inputs. First, a chaotic time series will be used as input to the proposed algorithm. Thereafter, traces from a real MPI application will be used.

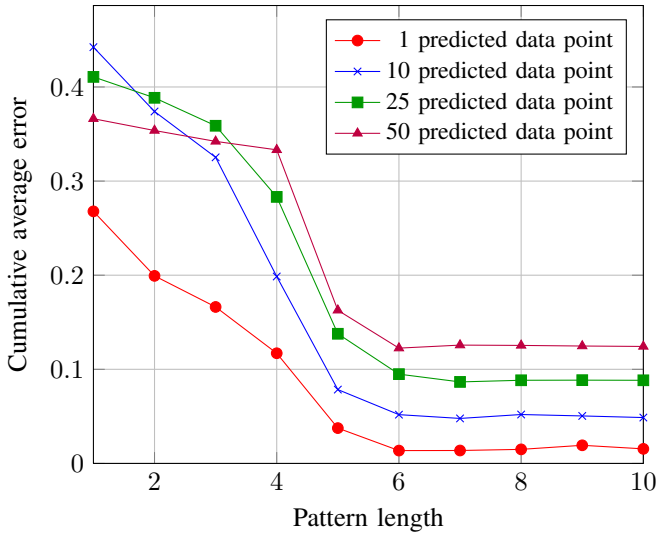


Fig. 4. Cumulative average error as a function of the pattern length

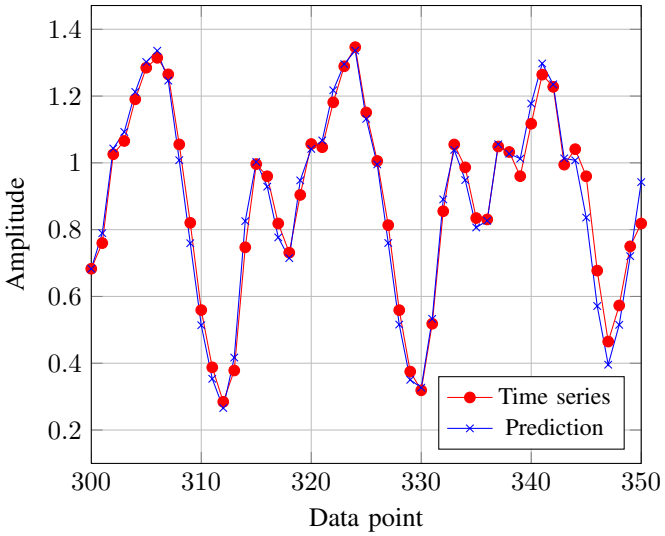


Fig. 5. Generated and predicted data points (history length is 300)

A. *Mackey-Glass*

First the proposed algorithm is tested with a chaotic time series given by the Mackey-Glass differential equation [16]:

$$\frac{dx}{dt} = \beta \cdot \frac{x_\tau}{1 + {x_\tau}^n} - \gamma \cdot x. \tag{5}$$

To generate a chaotic time series from this equation, the parameters are set as follows: $\beta = 0.2$, $\gamma = 0.1$, and $n = 10$. In this equation x_τ represents the value of the variable x at time $(t-\tau)$. The first 600 data points are calculated by solving the differential equation.

The pattern length m is an important parameter for the accuracy of the proposed algorithm. Therefore, the impact of the pattern length on the accuracy of the algorithm is investigated first. To perform this investigation the pattern length varied from 1 to 10 and the history length is set to 300. Furthermore, the algorithm is used to predict different numbers of data points. For every number of predicted data points the cumulative average error is calculated. The cumulative average error after n data points is the average error of the first n data points. Figure 4 depicts the cumulative average error as a function of the pattern length. The results show that the cumulative average error decreases when the pattern length is increased up to a length of 7. Therefore, all further investigations with the Mackey-Glass time series, the pattern length m is set to 7.

Figure 5 compares the predicted data points to the data points generated by Equation (5) for up to 50 predicted data points. The history length is set to 300. Thus the algorithm only considers the last 300 data points to make a prediction. The width w of the membership function, Equation (2), is set to 0.3. The predicted data points differs only slightly from the generated data points. The average error in Figure 5 is less than 4.5%.

The history length is another important parameter for the accuracy of the proposed algorithm. Therefore the impact

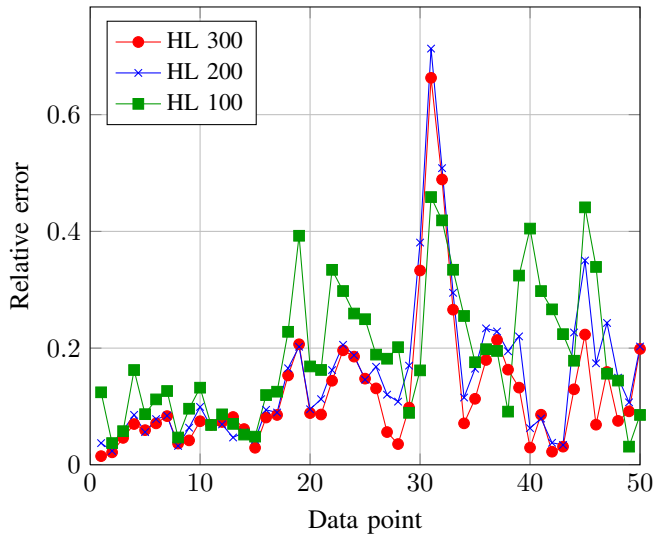


Fig. 6. Average error for 50 predicted data points with different history lengths (HL)

of the history length on the accuracy of the algorithm is investigated in more detail. To perform this investigation, the starting point for the prediction is set to four different values, and from there on 50 data points are predicted. Afterwards the average error for every predicted point is calculated. This experiment is performed for different history lengths. Figure 6 depicts the average error for each predicted data point and Figure 7 depicts the cumulative average error after a certain number of data points have been predicted. The cumulative average error after n data points is the average error of the first n data points in Figure 6. In both Figure 6 and Figure 7, different history lengths of 100, 200 and 300 have been used. These results show that the algorithm has the ability to predict up to 10 data points with high accuracy (5.2% error) for a chaotic time series with a history depth of 300. The high

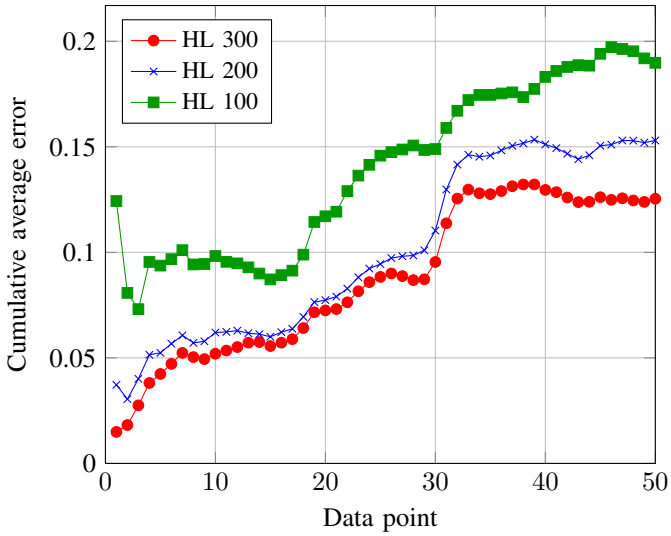


Fig. 7. Cumulative average error for different history lengths (HL)

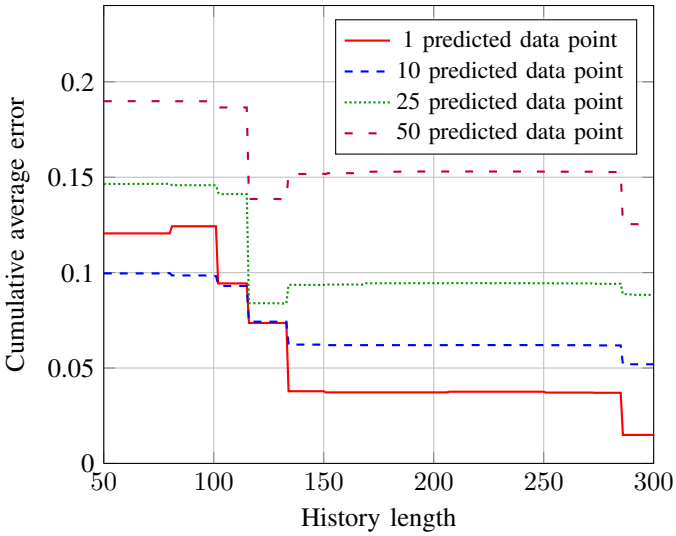


Fig. 8. Cumulative average error as a function of the history length

accuracy is achieved because of the continuous data points. The results also shows that the error increases when the history length is reduced. When 10 future data points need to be predicted the cumulative average error increases to 6.2% for a history length of 200, and to 9.8% for a history length of 100. There is a direct dependence between the first predicted values and the error propagation for succeeding data points. Predicted data points are used for the prediction of the succeeding data point. The error accumulates from data point to data point so error propagation happens. With a longer history lengths the error for the first predicted value is smaller and only a minor error propagation takes places. The algorithm miss predicts a peak in step 31 for all history lengths, which results in an error peak depicted in Figure 6.

Figure 8 depicts the cumulative average error as a function of the history length. The history length is varied from 50 to 300. The four lines correspond to different number of predicted data points (1, 10, 25, 50). The figure shows that the first accuracy improvement occurs when the history length is between 100 and 150. Afterwards the history length has no considerable influence on the accuracy of the predicted data points up to a history length of around 280. Another improvement of the prediction accuracy takes place beyond a history length of 280. It can be seen that for 25 and 50 predicted data points, using a history length of e.g. 125 yields better predictions than using longer history lengths. We cannot fully explain this behavior, but expect this is due to the ”period” of the chaotic time series.

Figure 9 depicts normalized prediction accuracy for different history lengths. The accuracy is normalized with respect to the shortest considered history length (50). This figure shows that the history length has a huge impact on the prediction accuracy. When one data point is predicted, the accuracy is improved by a factor of 8 when the history length is increased for 50 to 300. However, the impact of the history length decreases when the number of predicted data points increases.

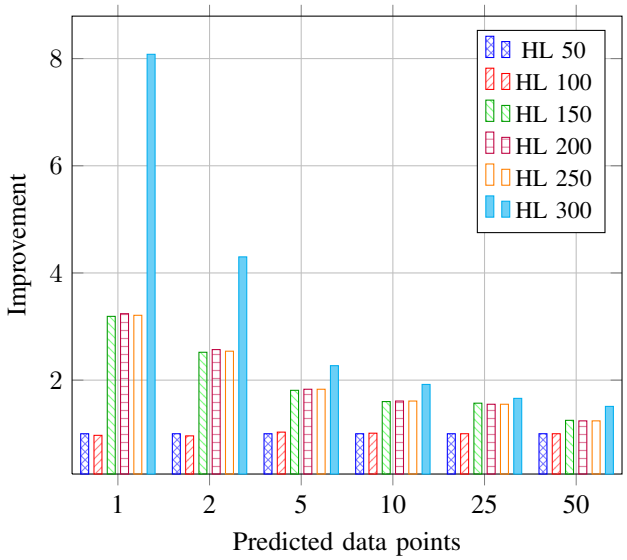


Fig. 9. Prediction accuracy normalized to the prediction accuracy obtained for a history length (HL) of 50 for different numbers of predicted data points.

When 50 data points are predicted, the improvement is reduced to a factor of 1.5.

B. MPI Application

1) Traffic Trace: To validate the proposed algorithm on real traffic patterns, traffic traces from real applications are needed. To generate these traces the application Meep [17] is used. Meep is a free finite-difference time-domain (FDTD) simulation software package developed at MIT to model electromagnetic systems. The program has the ability to to parallelize a problem with the Message Passing Interface (MPI) which is used for the communication between processes. For our purpose Meep is used with OpenMPI [18].

Every communication is recorded using the MPItrace tool [19]. This tool traces the basic activities in an MPI program and

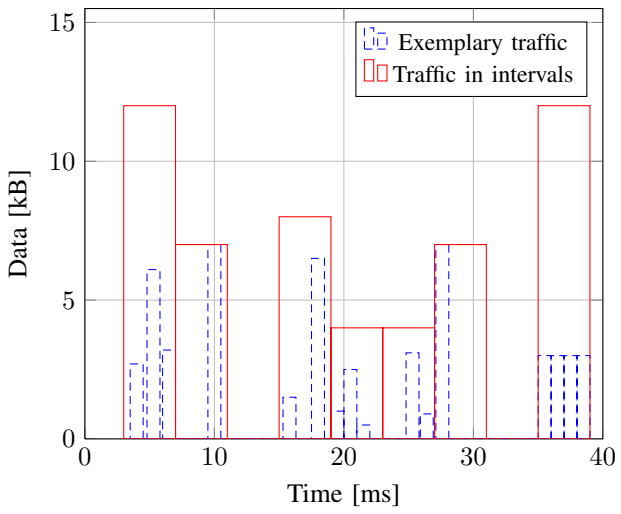


Fig. 10. Non-equidistant MPI communication patterns become equidistant

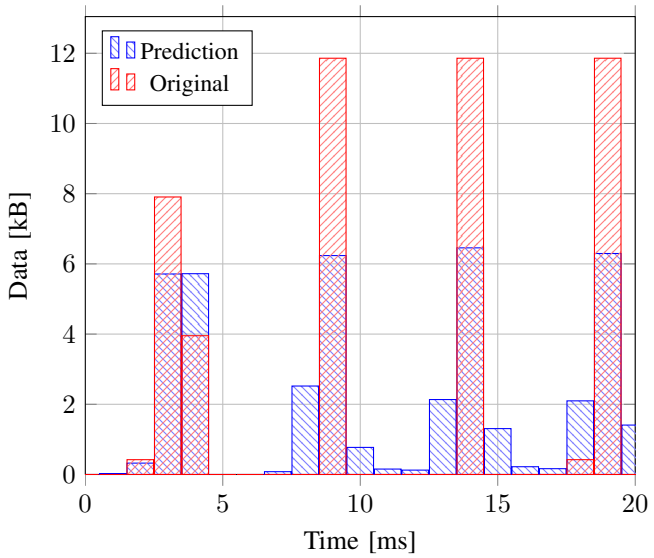


Fig. 11. Prediction of MPI communications (first behavior)

generates a Paraver [20] trace file. This trace file includes information about thread states and communication. For our purpose only the communication information between MPI processes is relevant. Therefore a short script has been implemented that separates the communication information from the rest. A drawback of the MPItrace tool is, however, that the start and end time of every communication correspond to the time when the MPI functions (send and receive) are called respectively return. This calls the logical communication. It is not possible to determine when the real (physical) communication takes places. How we dealt with this problem is described in the next section.
The application Meep has been running on a server with 2 processors with 4 cores each.

2) MPI Analysis: The dashed bars in Figure 10 depicts exemplary the amount of data sent by an arbitrary core over time. The figure shows that the gap between two transferred messages and also the amount of data that is sent vary. This leads to a prediction problem with two unknown variables, since the problem is not only to predict how much data is sent but also at which time. There is a large difference between not knowing if a certain data point in time exists or to know that there is a data point whose value may be zero. This means that the time between two communications in the time domain are not equidistant, which introduces problems for the proposed algorithm. The introduced algorithm cannot deal with this problem because there are too many unknown variables. The algorithm can predict only one unknown variable over an equidistant scale, for example, the amount of data that is sent at a fixed point in time. For that reason the problem must be reduced in order to be able to apply the proposed algorithm.
To reduce this problem, time is divided into fixed sized intervals. In every time interval, the amount of data that is sent is summed up. The solid bars in Figure 10 depict the amount of data sent in each time interval. The advantage of time intervals is that now the time axis no longer has non-equidistant time steps. Therefore the above mentioned problem with two unknown variables has been reduced to a problem with one unknown, namely to predict the amount of data that will be sent. With this technique the problem that the tracing tool provides only traces with logical communication information is also reduced. It can be assumed that the physical communication will take place shortly after the logical so the assumption is made that most communications will start in the corresponding time interval, provided the size of the interval is not too short. For communications that take places at the end of a time interval it, cannot be determined if they are assigned to the correct time interval. This side effect will be neglected.
The proposed algorithm should predict up to 20 prospective time intervals. The width of the membership function, Equation (2), is set to 5.5 and the pattern length m is set to 7. When predicting the MPI communication traces two behaviors have been observed, which are depicted in Figure 11 and Figure 12. The first data point numbered zero in both figures is the real one. Therefore the measured and predicted values match. The prediction starts in time interval 1. In time intervals where no bars are visible the communication volume during this interval is zero.
Both figures show the amount of data sent in each time interval. The dashed bars depict the real data and the solid bars the predicted. The first behavior is depicted in Figure 11. The time at which a communication takes places is predicted with high accuracy, but the predicted amount of data is below the actual amount. The second behavior is depicted in Figure 12. In this case the amount of transmitted data is predicted better than in the case depicted in Figure 11. The algorithm also, however, predicts communication peaks where no peaks are. This can be seen in time interval 12.
Table I shows the absolute average error in kB for both figures after several predictions steps. It is not possible to present the relative error because several data points are zero which

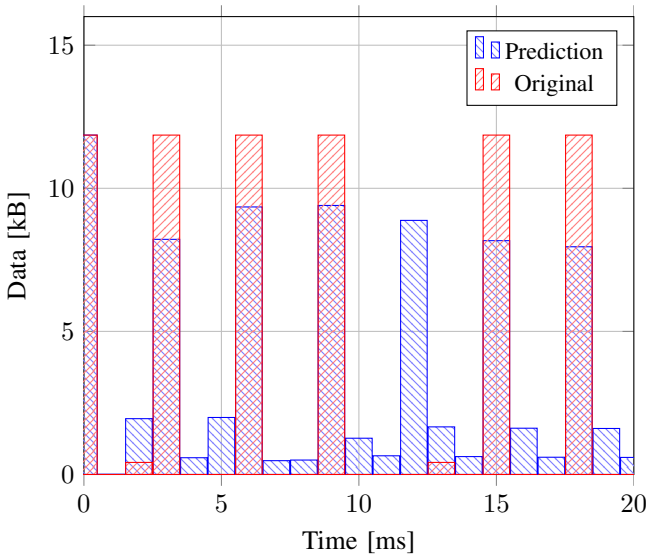


Fig. 12. Prediction of MPI communications (second behavior)

TABLE I
ABSOLUTE AVERAGE ERROR IN KB

Steps	1	2	5	10	20
Avg. Err. (Fig. 11)	0.008	0.768	1.551	1.498	1.920
Avg. Err. (Fig. 12)	0.246	1.596	2.442	2.427	2.656

would lead to a division by zero. Table I shows that a 1-step prediction can be performed with high accuracy for both behaviors. For the first behavior also a 2-step prediction has a small error. After that the error increases but stays nearly constant up to step 10. Thereafter the error increases more and more because of error propagation.

The miss predictions for both shown behaviors arise from the relative distance between two data points in contiguous time intervals, especially when a peak communication is followed by the absence of communication or vice versa. Such jumps mislead the proposed algorithm so that a wrong estimation is produced. The communication patterns from the history are weighted incorrectly so an error arises which influences coming data points.

To interpret the presented results it must be taken into account that only static simulations are performed. Data points that occur far into the future are predicted with previously predicted values. That means that no new data points are taken into account so that the error propagates. In a real system the time goes on and new data points are produced and so the history is updated. In that case more real data points could be used to predict the next data points, which would lead to slower error propagation. This is identical to a prediction with few future time intervals. Moreover, the size of the time intervals must also be taken into account. The size of a time interval is set by the user so that one interval could correspond to many clock cycles. Based on the problem one or two predicted intervals could be sufficient.

VII. Conclusions and Future Work

In this paper a model has been proposed to predict end-to-end traffic in NoC-based multiprocessor systems. The model predicts end-to-end communication, so intermediate switching elements are not considered. A fuzzy based algorithm is employed that searches for similar traffic patterns in the history to predict prospective data points. The prediction is performed for time intervals. Experimental results have been provided for a chaotic time series as well as real traffic patterns obtained by tracing an MPI application on a multiprocessor system. The accuracy of the prediction depends mainly on the behavior the traffic patterns that should be predicted. Chaotic patterns with continuous behavior can be predicted with high accuracy for up to 10 data points. Traffic patterns that are non-continuous, jumping from high communication volume to zero communication or vice versa, can also be predicted accurately, but only up to two steps ahead. Accurately predicting two data points can be sufficient, however, because the prediction is performed for time intervals and one interval consist of many clock cycles.

As future work, we plan to validate the proposed method on a NoC system. To do that the proposed algorithm must be integrated into a NoC simulator. This step allows us to investigate the NoC system speedup due to traffic prediction. Furthermore, this step is important to check how the prediction of future data points influences the NoC system. Also the computational complexity of the model must be analyzed and optimized in order to be able to integrate the proposed model in NoC systems. On a real NoC system, it could be validated how many time steps must be predicted in order to improve the system performance. Furthermore, the prediction accuracy of the algorithm could be improved. In particular the amount of data that is transferred could be predicted with higher accuracy, since the predicted amount is currently below the actual amount.

References

[1] D. Geer, "Chip Makers Turn to Multicore Processors," *Computer*, vol. 38, no. 5, pp. 11 – 13, May 2005.

[2] P. Gepner and M. Kowalik, "Multi-Core Processors: New Way to Achieve High System Performance," in *Parallel Computing in Electrical Engineering, 2006. PAR ELEC 2006. International Symposium on*, 2006, pp. 9 –13.

[3] B. Ahmad, A. Erdogan, and S. Khawam, "Architecture of a Dynamically Reconfigurable NoC for Adaptive Reconfigurable MPSoC," in *Adaptive Hardware and Systems, 2006. AHS 2006. First NASA/ESA Conference on*, 2006, pp. 405 –411.

[4] A. Hansson and K. Goossens, "Trade-offs in the Configuration of a Network on Chip for Multiple Use-Cases," in *Networks-on-Chip, 2007. NOCS 2007. First International Symposium on*, May 2007, pp. 233 – 242.

[5] Y. Huang, K.-K. Chou, C.-T. King, and S.-Y. Tseng, "NTPT: On the End-to-End Traffic Prediction in the On-Chip Networks," in *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, 2010, pp. 449 –452.

[6] S. Kaxiras and C. Young, "Coherence communication prediction in shared-memory multiprocessors ," in *High-Performance Computer Architecture, 2000. HPCA-6. Proceedings. Sixth International Symposium on*, 2000, pp. 156 –167.

[7] J. Duato, O. Lysne, R. Pang, and T. Pinkston, "A Theory for Deadlock-Free Dynamic Network Reconfiguration. Part I," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 16, no. 5, pp. 412 – 427, May 2005.

[8] O. Lysne, T. Pinkston, and J. Duato, "A Methodology for Developing Deadlock-Free Dynamic Network Reconfiguration Processes. Part II," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 16, no. 5, pp. 428 – 443, May 2005.

[9] B.-S. Chen, Y.-S. Yang, B.-K. Lee, and T.-H. Lee, "Fuzzy Adaptive Predictive Flow Control of ATM Network traffic," *Fuzzy Systems, IEEE Transactions on*, vol. 11, no. 4, pp. 568 – 581, 2003.

[10] Q. Pang, S. Cheng, and P. Zhang, "Adaptive fuzzy traffic predictor and its applications in ATM networks," in *Communications, 1998. ICC 98. Conference Record.1998 IEEE International Conference on*, vol. 3, Jun. 1998, pp. 1759 –1763 vol.3.

[11] P. Otto and T. Schunk, "Fuzzybasierte Zeitreihenvorhersage," *Automatisierungstechnik*, vol. 48, pp. 327–334, 2000, In: German.

[12] U. Y. Ogras and R. Marculescu, "Prediction-based Flow Control for Network-on-Chip Traffic," in *Proceedings of the 43rd annual Design Automation Conference*, ser. DAC '06. New York, NY, USA: ACM, 2006, pp. 839–844. [Online]. Available: http://doi.acm.org/10.1145/1146909.1147123

[13] J. van den Brand, C. Ciordas, K. Goossens, and T. Basten, "Congestion-Controlled Best-Effort Communication for Networks-on-Chip," in *Design, Automation Test in Europe Conference Exhibition, 2007. DATE '07*, 2007, pp. 1 –6.

[14] W. Dally and C. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," *Computers, IEEE Transactions on*, vol. C-36, no. 5, pp. 547 –553, May 1987.

[15] L. Zadeh, "Fuzzy Sets," *Information and Control*, vol. 8, pp. 338–353, 1965.

[16] L. Glass and M. C. Mackey, "Oscillation and Chaos in Physiological Control Systems," *Science*, vol. 197, pp. 287–289, 1977.

[17] S. G. Johnson, J. D. Joannopoulos, and M. Soljai, "Meep," 2006. [Online]. Available: http://ab-initio.mit.edu/wiki/index.php/Meep

[18] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R. H. Castain, D. J. Daniel, R. L. Graham, and T. S. Woodall, "Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation," in *Proceedings, 11th European PVM/MPI Users' Group Meeting*, Budapest, Hungary, September 2004, pp. 97–104.

[19] H. S. Gelabert and G. L. Snchez, "MPItrace - User Guide Manual," 2010. [Online]. Available: http://www.bsc.es/plantillaA.php?cat_id=492

[20] *Paraver - Parallel Program Visualization and Analysis tool*, Version 3.1 ed., Barcelona Supercomputing Center - Centro Nacional de Supercomputacin, October 2001. [Online]. Available: http://www.bsc.es/plantillaA.php?cat_id=493