Throughput-Fairness Tradeoff in Best Effort Flow Control for On-Chip Architectures

Fahimeh Jafari*†, Mohammad S. Talebi†, Mohammad H. Yaghmaee*, Ahmad Khonsari‡†
and Mohamed Ould-Khaoua§¶
* *Ferdowsi University of Mashhad, Mashhad, Iran*
† *School of Computer Science, IPM, Tehran, Iran*
‡ *ECE Department, University of Tehran, Tehran, Iran*
§ *Department of Electrical and Computer Engineering, Sultan Qaboos University, Oman*
¶ *Department of Computing Science, University of Glasgow*
Emails: jafari@ipm.ir, mstalebi@ipm.ir, hyaghmae@ferdowsi.um.ac.ir, ak@ipm.ir, mohamed@dcs.gla.ac.uk

Abstract

We consider two flow control schemes for Best Effort traffic in on-chip architectures, which can be deemed as the solutions to the boundary extremes of a class of utility maximization problem. At one extreme, we consider the so-called Rate-Sum flow control scheme, which aims at improving the performance of the underlying system by roughly maximizing throughput while satisfying capacity constraints. At the other extreme, we deem the Max-Min flow control, whose concern is to maintain Max-Min fairness in rate allocation by fairly sacrificing the throughput. We then elaborate our argument through a weighting mechanism in order to achieve a balance between the orthogonal goals of performance and fairness. Moreover, we investigate the implementation facets of the presented flow control schemes in on-chip architectures. Finally, we validate the proposed flow control schemes and the subsequent arguments through extensive simulation experiments.

1. Introduction

With the advance of the semiconductor technology, the enormous number of transistors available on a single chip allows designers to integrate dozens of IP (Intellectual Property) blocks together with large amounts of embedded memory. Such IPs may be CPU or DSP cores, video stream processors, high-bandwidth I/O, etc. Dedicated links may lead to an irregular architecture which is difficult to reuse. Also, shared-medium busses do not scale well, and do not fully utilize potentially available bandwidth. As the feature sizes shrink, and the overall chip size relatively increases, the interconnects start behaving as lossy transmission lines. Line delays have become very long as compared to gate delays causing synchronization problems between cores. This trend only worsens as the clock frequencies increase and the features sizes decrease.

One solution to these problems is to treat systems on a chip implemented using the Networks-on-Chip (NoC) paradigm. Multiple concurrent connections provide much higher bandwidth in Networks. Also, regularity enables design modularity, which in turn provides a standard interface for easier component reuse and better interoperability. Overall performance and scalability increase since the networking resources are shared [1]. Due to the rapid growth of the number of processing elements in NoCs [2], employing efficient policies for flow control has become an inevitable subject in the design of NoCs to provide the required Quality of Service (QoS). A NoC must have network level flow control in order to avoid congestion in the bottleneck links.

Recently, QoS provisioning in NoC's environment has attracted many researchers and currently is the focus of many literatures in NoC research community. NoCs are expected to serve as multimedia servers and are required to carry both Best Effort (BE) and Guaranteed Service (GS) traffics. It's trivial that such a networked architecture with data services should have some policies to avoid congestion. Congestion Control in data networks is known as a widely-studied issue over the past two decades. However, it is still a novel issue in NoC and to the best of our knowledge only a few works have been carried out in this field.

In this paper, we focus on the flow control for BE traffic as the solution to an optimization problem. In our previous work [3], we have modeled desired BE source rates as the solution to a utility-based optimization problem with a general form utility function and solved the problem using Newton method. In [4], we also considered this issue via Rate-Sum optimization problem and used a different approach to solve it. In this paper, we mainly focus on the two extreme cases of the utility optimization approach which lead to different performance and fairness properties. In the first view, in this paper we present two flow control mechanisms for BE traffic in NoC which are derived for the extreme cases of utility definition in [3]. These flow control schemes exhibit different fairness and performance

978-1-4244-3750-4/09/$25.00 ©2009 IEEE

properties. In another view, investigating and balancing the tradeoff between such conflicting properties is the other contribution of ours in this paper.

The organization of the paper is as follows. In Section 2 we will briefly review the most significant works on this subject. In Section 3, we present the system model and problem formulation. In Section 4 we focus on the Rate-Sum problem and propose a flow control for BE as an iterative solution to it. In Section 5 we consider the Max-Min problem along with the concept of the Max-Min fairness and present another BE flow control mechanism. Section 6 is devoted to the discussion about the fairness and performance tradeoff for the presented flow control mechanisms and presents a remedy to balance between them. Section 7 presents the simulation results and discussion about them. Finally, the Section 8 concludes the paper.

2. Related Works

In this section, we briefly review the most significant works focused on this issue.

Flow control for data networks is a widely-studied issue [5]-[7]. A wide variety of flow control mechanisms in data network belongs to the class of End-to-End flow control schemes, like TCP/IP, which mainly act based on the window-based protocols. In this method, sent packets are subject to loss and the network must aim to provide an acknowledgment mechanism. On the other hand, On-chip networks pose different challenges. The reliability of on-chip wires and more effective link-level flow-control makes NoC a loss-less environment. Therefore, there is no need to utilize acknowledgment mechanisms and we face to a slightly different concept of flow control.

So far, several works have focused on this issue for NoC architectures. Dyad [8] controls the congestion by using adaptive routing when the NoC faces congestion. However this method can not guarantee that congestion is solved (i.e. the alternative paths might also be congested). In [9], a prediction-based flow-control strategy for on-chip networks is proposed in which each router predicts the buffer occupancy to sense congestion. In [10] link utilization is used as a congestion measure and a Model Prediction-based Controller (MPC), determines source rates. The authors in [10] state that their work is more complete than [9] because in [9] the router buffer filling information is used for toggling the sources while their approach allows both toggling and fluent control of loads offered by IPs.

To the best of our knowledge, none of the aforementioned works have dealt with the problem through utility optimization approach. As mentioned in [11], our approach has little control overhead than [10] because in [10] the link utilization measurements are periodically performed by hardware probes and are transported to a controller by GS connections while in our approach control packets are sent to the controller only when a flow enters to the network or exits from it. Besides, since we do not need any hardware probes, costs of hardware implementation is reduced.

3. System Model and Flow Control Problem

We consider a NoC architecture with wormhole switching. In wormhole switching networks, each packet is divided into a sequence of *flits* which are transmitted over physical links one by one in a pipeline fashion. A hop-to-hop credit mechanism guarantees that a flit is transmitted only when the receiving port has free space in its input buffer. We also assume that the NoC architecture is lossless, and packets traverse the network on a shortest path using a deadlock free XY routing [2].

We model the flow control in NoC as the solution to a utility-based optimization problem. We turn the aforementioned NoC architecture into a mathematically modeled network, as in [12]. In this respect, we consider NoC as a network with a set of bidirectional links $\mathcal{L} = \{1, 2, \ldots, L\}$ and a set of sources $\mathcal{S} = \{1, 2, \ldots, S\}$. A source consists of Processing Elements (PEs), Routers and Input/Output ports. Also, each link $l \in \mathcal{L}$ has a fixed capacity of c_l bits/sec and is a set of wires and channels that are responsible for connecting different parts of the NoC. We denote the set of sources that share link l by $\mathcal{S}(l)$. Similarly, the set of links that source s passes through, is denoted by $\mathcal{L}(s)$. By definition, $s \in \mathcal{S}(l)$ if and only if $l \in \mathcal{L}(s)$.

As presented before, two classes of traffic are considered in a NoC: Guaranteed Service (GS) and Best Effort (BE). For notational convenience, we represent BE and GS traffic rates by x_s and y_s, respectively. The two classes flow over a link by sharing its capacity as following: GS traffic will obtain the required amount of link capacity and BE traffic can benefit from the remainder.

Our objective is to choose source rates with BE traffic so as to maximize the sum of utilities of BE sources. We assume that source s when transmitting BE packets at rate x_s bps, achieves a utility equal to $U_s(x_s)$. Thus, the optimization problem can be formulated as below:

$$\max_{x_s} \sum_{s \in \mathcal{S}} U_s(x_s) \tag{1}$$

subject to:

$$\sum_{s \in \mathcal{S}(l)} x_s + y_s \leq c_l; \qquad \forall l \in \mathcal{L} \tag{2}$$

$$x_s > 0; \qquad \forall s \in \mathcal{S} \tag{3}$$

where optimization variables are BE rates, which in vector form are denoted by $\mathbf{x} = (x_s, s \in \mathcal{S})$ and belong to $\mathfrak{R}_+^S$. ($\mathfrak{R}_+^S$ denotes nonnegative real).

The constraint (2) simply states that the sum of BE rates passing through link l cannot exceed its free capacity, i.e. the portion of c_l which hasn't been dedicated to GS

traffic. Constraint (2) is equivalent to the case in which the maximum capacity of links can be used. Such an assumption may not hold in general unless the buffering space at each router tends to infinity. Although, such an assumption may seem restrictive, it will be useful in the sense that it yields the upper bound of the achievable source rate. Moreover, we argue that one of the advantages of applying a flow control mechanism is to efficiently allocate source rates so as to better utilize the maximum capacity of links. Therefore, using flow control, we can avoid buffer shortage by efficiently limiting the injection rates of nodes, and therefore, we shift from a buffer-restricted regime towards a link capacity-limited regime.

Problem (1) is such a general form is a Utility Maximization Problem. Based on the convex optimization theory, for such a problem to have a unique optimal point, U_s should be positive, concave and strictly increasing [12]. Several candidates for U_s exist for which the aforementioned conditions hold. Amongst them, presumably α-Fair functions are the most significant ones as they have nice properties in terms of economically fair behavior.

In this paper, we consider problem (1) with the class of α-Fair utility functions, defined as below [13]:

$$U(x,\alpha) = \begin{cases} \frac{x^{1-\alpha}}{1-\alpha} & \alpha \neq 1 \\ \ln x & \alpha = 1 \end{cases} \tag{4}$$

where $\alpha > 0$ is a parameter. With the aforementioned choice of utility function, problem (1) is a convex optimization problem with linear constraints. Hence it admits a unique maximizer [14][15]. For notational convenience, we define:

$$\hat{c}_l = c_l - \sum_{s \in \mathcal{S}(l)} y_s \tag{5}$$

Although $\hat{c}_l$ denotes the usable link capacity, with a slight abuse of definition, hereafter we will refer to $\hat{c}_l$ as the link capacity. Moreover, similar to BE rate vector, we represent link capacity vector as $\hat{\mathbf{c}} = (\hat{c}_l, l \in \mathcal{L})$. Finally, to avoid confusing with summations indices, we define Routing matrix as $\mathbf{R} = [R_{ls}]_{L \times S}$, where R_{ls} is defined as:

$$R_{ls} = \begin{cases} 1 & \text{if } l \in \mathcal{L}(s) \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

In this paper, we mainly focus on the two extreme cases; i.e. $\alpha = 0$ and $\alpha \to \infty$ which lead to performance tradeoffs in rate allocation. In the sequel, we first focus on the case of $\alpha = 0$ for which problem (1) reduces to the so-called Rate-Sum Maximization and then investigate the case of $\alpha \to \infty$ for which problem (1) converts to the so-called Max-Min problem.

4. Rate-Sum Flow Control

In this section, we consider the case of $\alpha = 0$ and solve the accordingly-derived Rate-Sum Maximization problem in an iterative manner. We finally present a flow control algorithm for BE source rates based on the iterative solution.

Algorithm 1: Maximum Rate-Sum Flow Control Algorithm for BE

Initialization:

1. Initialize $\hat{c}_l$ of all links.
2. Set source rate vector to zero.
3. Specify an appropriate value for ϵ.

Loop:

Do until $(\max_{s \in \mathcal{S}} \mid x_s(t+1) - x_s(t) \mid < \epsilon)$

1. $\forall s \in \mathcal{S}$: Compute new source rate: $\mathbf{x}(t+1) = \mathbf{x}(t) + \gamma(t)\mathbf{u}(\mathbf{x}(t))$

where $\gamma(t)$ can be selected as $\gamma(t) = \frac{a}{b+t}$ and

$$\mathbf{u}(\mathbf{x}(t)) = \begin{cases} \mathbf{1} & \sum_{s \in \mathcal{S}(l)} x_s(t) \leq \hat{c}_l, \quad \forall l \\ -\mathbf{R}^T \mathbf{e}_{l'} & \sum_{s \in \mathcal{S}(l)} x_s(t) > \hat{c}_l, \quad \exists l' \end{cases}$$

Output:

Communicate BE source rates to the corresponding nodes.

Regarding problem (1), it's apparent that by substituting $\alpha = 0$, problem (1) can be rewritten as:

$$\max_{x_s} \sum_{s \in \mathcal{S}} x_s \tag{7}$$

subject to:

$$\mathbf{R}\mathbf{x} \leq \hat{\mathbf{c}} \tag{8}$$

$$x_s > 0; \qquad s \in \mathcal{S} \tag{9}$$

We will solve this problem using Gradient Method for constrained problems [14]. The Projected Gradient Method for constrained problems is very similar to the original one which only applies to unconstrained problems [14][15]. Therefore, the update equation to solve problem (7) is given by:

$$\mathbf{x}(t+1) = \mathbf{x}(t) + \gamma(t)\mathbf{u}(\mathbf{x}(t)) \tag{10}$$

where $\gamma(t)$ is a diminishing step-size rule which satisfies specific conditions [14]. One typical example is $\gamma(t) = a/(b+t)$, where $a > 0$ and $b \geq 0$, which we have used in this paper. $\mathbf{u}(\mathbf{x}(t))$ is given by:

$$\mathbf{u}(\mathbf{x}(t)) = \begin{cases} \mathbf{1} & \sum_{s \in \mathcal{S}(l)} x_s(t) \leq \hat{c}_l, \quad \forall l \\ -\mathbf{R}^T \mathbf{e}_{l'} & \sum_{s \in \mathcal{S}(l)} x_s(t) > \hat{c}_l, \quad \exists l' \end{cases} \tag{11}$$

where $\mathbf{e}_{l'}$ is the l'-th unit vector of $\Re^L$ space which is zero in all entries except the l'-th at which it is 1.

Equations (10) and (11) together form an iterative algorithm to solve Rate-Sum Maximization problem (7). Algorithmic realization of the proposed flow control algorithm for BE traffic is listed as Algorithm 1.

5. Max-Min Flow Control Problem

In this section, our focus is on the case of $\alpha \to \infty$. First, we show that the corresponding problem can be construed as a Max-Min problem which can be solved using *Progressive Filling* algorithm [16]. Finally, Based on this algorithm, we present a flow control algorithm for BE source rates. Considering problem (1), it's apparent that when α tends to the infinity, we have

$$\max_{x_s} \lim_{\alpha\to\infty} \sum_{s\in\mathcal{S}} \frac{x_s^{1-\alpha}}{1-\alpha} \tag{12}$$

subject to:

$$\mathbf{Rx} \leq \hat{\mathbf{c}} \tag{13}$$

$$x_s > 0; \qquad s \in \mathcal{S} \tag{14}$$

Problem (12) in such an extreme case is disobedient. However, the following theorem states that it can be reduced to the well-known Max-Min optimization problem.

Theorem 1: The maximization problem (12) reduces to the Max-Min optimization problem, as below

$$\max_{x_s} \min_{s\in\mathcal{S}} x_s \tag{15}$$

subject to:

$$\sum_s R_{ls} x_s \leq \hat{c}_l; \qquad \forall l \in \mathcal{L} \tag{16}$$

$$x_s > 0; \qquad \forall s \in \mathcal{S} \tag{17}$$

Proof: Proof is omitted due to space limit.

Max-Min optimization problem is a widely-studied problem formulation in resource management scenarios such as rate allocation in data networks. This is mainly due to an important property, which is inherent in the Max-Min problem, and discriminates it from the others. The optimal solution to the max-min problem, if exists, admits a specific type of fairness characteristic known as Max-Min Fairness (MMF), which will formally be defined in the sequel.

Definition 1: (Max-Min Fair [17]) A feasible rate allocation $(x_s, s \in \mathcal{S})$ is said to be Max-Min Fair (MMF) *if and only if an increase of any rate within the domain of feasible allocations must be at the cost of a decrease of some already smaller rate. Formally, for any other feasible allocation* $\mathbf{y}$*, if $y_s > x_s$ then there must exist some s' such that $x_{s'} \leq x_s$ and $y'_s < x'_s$.*

Depending on the network topology, a Max-Min fair allocation may or may not exist. However, upon its existence, it is unique (see [17] for proof). In what follows the condition under which the Max-Min rate allocation exists will be stated. Before we proceed to this condition, we must define the concept of bottleneck link.

Definition 2: (Bottleneck Link [17]) A link l is said to be a bottleneck for source s if and only if:

1) *link l is saturated; i.e. $\sum_s R_{ls}(x_s + y_s) = c_l$*
2) *Source s on link l has the maximum rate among all sources passing through this link.*

Intuitively, a bottleneck link for source s is the link which limits x_s.

Theorem 2: A MMF rate allocation exists if and only if every source has a bottleneck link.

Proof: See [17] for proof.

Algorithm 2: Max-Min Fair Flow Control Algorithm for BE

Initialization:

1. Initialize $\hat{c}_l$ of all links.
2. Define:
   a. $\mathcal{T}$ as the set of sources not passing through any saturated link.
   b. $\mathcal{B}$ as the set of saturated links.
3. Set source rate vector to zero.
4. Initialize $\mathcal{T} = \mathcal{S}$ and $\mathcal{B} = \varnothing$.

Loop:

Do until ($\mathcal{T} = \varnothing$)

1. $\Delta_s = \min_{l\in(\mathcal{L}-\mathcal{B})}[(c_l - \sum_{s\in(\mathcal{S}-\mathcal{T})} R_{ls}x_s(t))/\sum_{s\in\mathcal{T}} R_{ls}]$
2. $x_s(t+1) = x_s(t) + \Delta_s, \quad \forall s \in \mathcal{T}$
3. Calculate new bottleneck links and update $\mathcal{B}$.
4. $\forall s \in \mathcal{T}$; if s passes through any saturated link then $\mathcal{T} \Leftarrow \mathcal{T} - \{s\}$

Output:

Communicate BE source rates to the corresponding nodes.

The most famous and simplest algorithm to solve the Max-Min problem (15) is the well-known Progressive Filling Algorithm [16].

We would like to employ the progressive filling algorithm as an iterative solution to the max-min problem (15). Similar to (7), we finally would like to utilize it as a BE flow control mechanism in NoC. The modified version of the progressive filling as a BE flow control mechanism is listed below as algorithm 2.

Algorithm 1 and 2 can be used as centralized flow control mechanisms for BE sources in NoC. In this regard, we consider a simple controller that can be embodied by the NoC, whether as a separate hardware module or as a part of its operating system, which is responsible for running the algorithms. From computational complexity point of view, such a controller must have the ability of carrying out simple mathematical and logical operations, as in Algorithm 1 and 2. Another issue worth considering is the mechanism with which the controller communicates with sources. Since we would like source rate information being communicated without delay and loss, we send them by GS connections to assure that this communication is not subject to congestion.

6. Throughput-Fairness Tradeoff

So far, we have presented two different flow control schemes for BE traffic in NoC. The first one, i.e. Rate-Sum scheme, has been designed to maximize the aggregate source rates. With a slight abuse in the definition of throughput in lossless scenarios, as in NoC, we partly interpret the aggregate of source rate as the throughput of the system. In this respect, Rate-Sum flow control scheme might be construed as one whose aim is to maximize the throughput while simultaneously satisfying link capacity constraints. On the other hand, Max-Min scheme is responsible for maintaining Max-Min fairness among source rates while satisfying capacity constraints.

Any discussion of rate allocation must address the two conflicting issues:

1) *Throughput* as the measure of the efficiency of the network performance.
2) *Fairness* to guarantee that network resources have been allocated to transmitting sources in accordance to a specified fairness metric.

Conflicting with each other, the two mentioned issues are in the extremes of performance spectrum which cannot be simultaneously obtained. Indeed, with the exception of few trivial networking scenarios, there isn't any rate allocation mechanism that can simultaneously realize both optimal fairness and optimal throughput. In fact, any such mechanisms can be seen as providing a tradeoff between throughput and fairness metric. Roughly speaking, boosting one of them will be at the expense of alleviating the other.

One efficient and straight-forward way to establish a tradeoff between throughput and fairness is to introduce weight factors to the underlying optimization problems, i.e. by replacing x_s with $w_s x_s$ in problem (7) and (15). w_s is the weight assigned to source s which determines its priority of rate allocation with respect to other sources. Intuitively, each source upon increase of his weight will obtain more network resources.

As the focus of problem (7) (problem (15)) is on throughput maximization (maintaining max-min fairness) under capacity constraints, the perception of such tradeoffs could not be attained directly. In order to provide more insights, we employ several well-known measures which are defined based on the statistical properties of a rate allocation. In fact, they are defined to quantify performance and fairness factors for a rate allocation vector, regardless of its underlying allocation strategy. Such statistical measures can be further used to compare the inherent tradeoff in different flow control schemes.

Due to space limit, we only introduce the most significant one, i.e. *Jain Fairness Index* [18], which is a widely-addressed index for measuring the fairness maintained amongst the individuals of a rate allocation scenario. Jain Fairness Index, which hereafter will be abbreviated as JFI, is defined as [18]:

$$\text{JFI} = \frac{\left(\sum_{s=1}^{S} x_s\right)^2}{S \sum_{s=1}^{S} x_s^2} \tag{18}$$

It can be proven that JFI for positive rate vectors always falls within [0,1] interval. JFI can be interpreted as a positive fraction which reflects the efficiency of fairness maintained between rate elements. Unity and $1/S$ correspond to the most fair and the least fair cases, respectively.

JFI and throughput together can be used as two simple and efficient measures for quantifying the tradeoffs between performance (throughput) and fairness in any rate allocation scheme. In the next section, we utilize JFI as a fairness measure for different examined scenarios.

7. Simulation Results

In this section we examine the proposed flow control algorithms for a typical NoC architecture. Using MATLAB, we have simulated a NoC with 4×4 Mesh topology consisting of 16 nodes communicating using 24 shared bidirectional links; each one has a fixed capacity of 1 Gbps. In our scheme, packets traverse the network on a shortest path using a deadlock free XY routing. We also assume that each packet consists of 500 flits and each flit is 16 bits long.

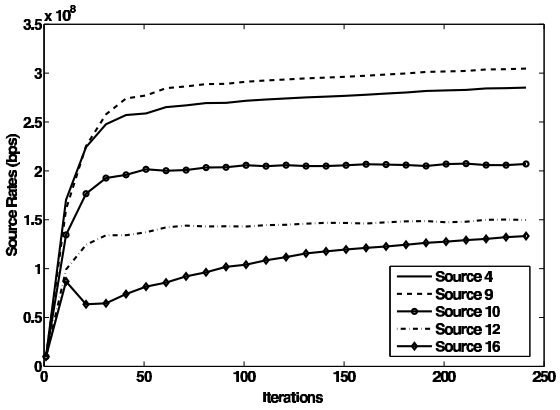

Figure 1. Source Rates vs. Iteration Steps for Rate-Sum

In order to simulate our scheme, some nodes are considered to have a GS-type traffic (such as Multimedia, etc.) to be sent to a destination while other nodes, while others have a BE traffic.

7.1. Comparison Between Rate-Sum and Max-Min

We obtained source rates using proposed algorithms in MATLAB. The evolution of source rates versus iteration steps for both Rate-Sum and Max-Min Fair flow control

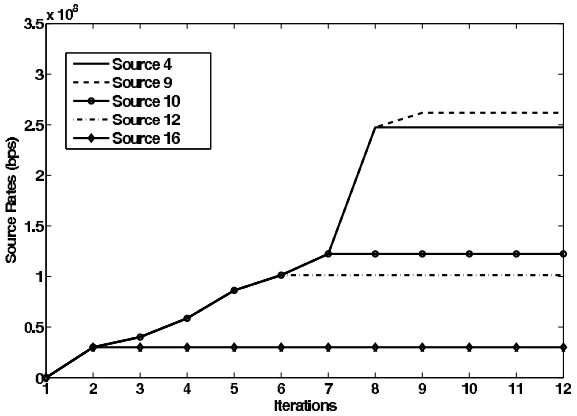

Figure 2. Source Rates vs. Iteration Steps for Max-Min

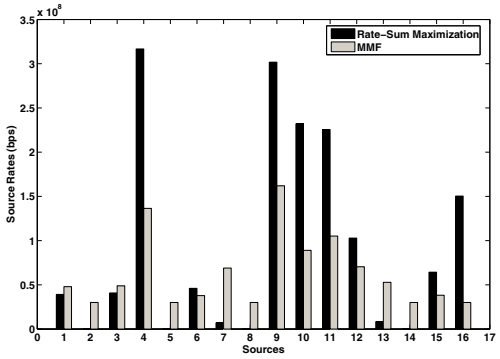

Figure 3. Comparison between Rate-Sum and Max-Min

schemes are shown in Fig. 1 and Fig. 2, respectively. These figures show that after passing enough iteration steps, the proposed algorithms converge to their steady state points. For the sake of convenience in comparing the two schemes, steady state source rates for all sources are depicted in Fig.

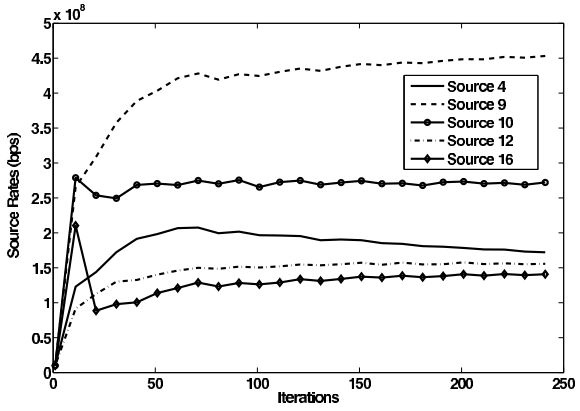

Figure 4. Source Rates vs. Iteration Steps for Weighted Rate-Sum with w_1

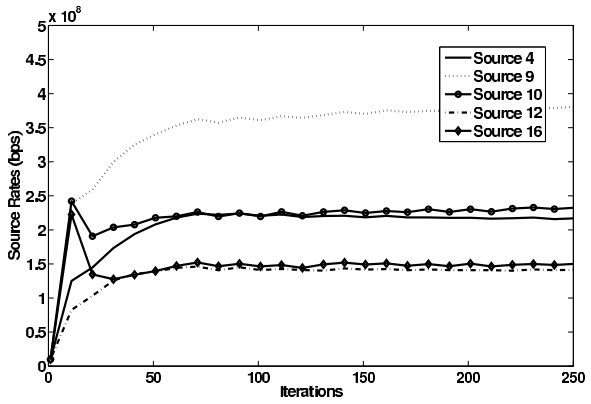

Figure 5. Source Rates vs. Iteration Steps for Weighted Rate-Sum with w_2

3. Comparing Rate-Sum and Max-Min in Fig. 3, it's evident that although Rate-Sum criterion aims at maximizing the sum of source rates, there is no guarantee for the rates of weak sources, i.e. those which achieve very small rates. Indeed, in many scenarios with Rate-Sum flow control, such sources will obtain as small as zero. On the other hand, the weakest source in Max-Min scenario obtains about 0.3 Gbps. Also, it is clear that the variance of Max-Min scheme is evidently less than that of Maximum Rate-Sum (MRS) scheme, which in turn implies the inherent fairness in the Max-Min rate allocation.

From Table 1 we realize that rate allocation with Maximum Rate-Sum criterion, yields greater aggregate rate than Max-Min Fair. However, as discussed above, Max-Min Algorithm guarantees that the rate allocation is Max-Min fair, and hence the minimum source rate wouldn't be greater with any other feasible rate allocation and therefore rate allocation is carried out in favor of such weak sources. On the contrary, Maximum Rate-Sum has no guarantee for such sources and as a result, the weakest source, has achieved as small as zero.

7.2. Influence of Weight Factors

In order to have much more flexibility to balance between throughput and fairness, we introduce two weight factors, w_1 and w_2 to determine the priority of resource allocation. Due to space limit, values of weight factors have been omitted. Such weight factors can be appropriately derived so as to designate network resources (link capacities) in favor

Table 1. Comparison between MRS and MMF

	Max-Min Fair	Rate-Sum
Latest Rate	0.310×10^8	0
Sum of Source Rate	10.079×10^8	15.349×10^8

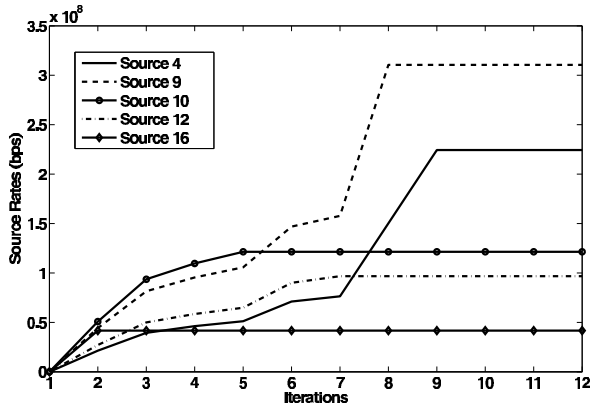

Figure 6. Source Rates vs. Iteration Steps for Weighted Max-Min with w_1

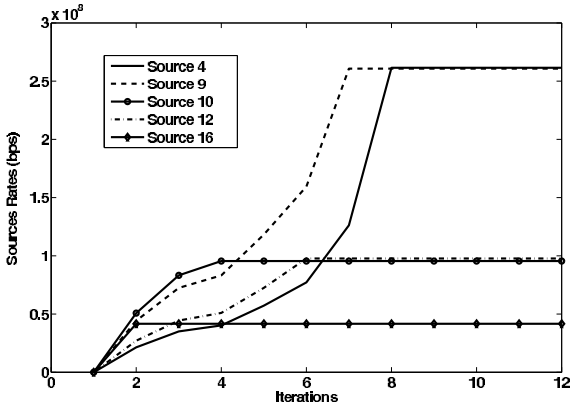

Figure 7. Source Rates vs. Iteration Steps for Weighted Max-Min with w_2

of throughput or fairness. In this respect, we have considered four additional scenarios featuring Weighted Rate-Sum (WMRS) and Weighted Max-Min (WMMF) schemes. The corresponding rate allocations are depicted in Fig. 4-7. From these figures, it is apparent that different weight factors have led to different rate allocations.

7.3. Fairness Metrics

To compare the results of the above mentioned schemes in more detail, we have considered four parameters featuring the merit of the different schemes as following:

1) *Variance of source rates with respect to mean value*
2) *Jain's fairness Index (JFI) [18]*
3) *Min-Max ratio [18]*

The Min-Max ratio is defined as (19).

$$\text{Min-Max Ratio} = \frac{\min_{s\in\mathcal{S}} x_s}{\max_{s\in\mathcal{S}} x_s} \tag{19}$$

The aforementioned parameters for MMF, WMMF (with two different weights), MRS and WMRS (with two different

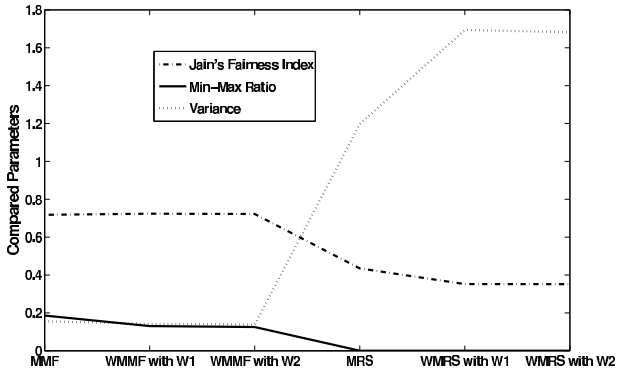

Figure 8. Different Parameters for Different Scenarios

weights) schemes are depicted in Fig. 8. It is apparent that using MMF and WMMF schemes, the variance of source rates are considerably less than MRS and WMRS, which denote the intrinsic fairness in these mechanisms with respect to MRS and WMRS mechanisms. Smaller variance results in the larger Min-Max Ratio and JFI; therefore MMF and WMMF schemes have greater Min-Max Ratio and JFI.

8. Conclusion

In this paper we addressed the problem of flow control for BE traffic in NoC architectures. We considered two extreme cases of the family of α-Fair utility maximization problems, whose solutions led to two iterative flow control algorithms for BE traffics. These extreme cases were 0-Fair and ∞-Fair, which are semantically connected to the throughput-optimal and fairness-optimal scenarios, respectively. These schemes aim at achieving two extreme goals: the first one, MRS aims at maximizing rate-sum (throughput) of the system, while the second, MMF aims at allocating resources in favor of weak sources. We focused on the concept of weight factors to remedy the problem of weak sources in MRS and the problem of throughput inefficiency in MMF. Simulation experiments validated that introducing appropriately-assigned weight factors, could efficiently compromise between throughput and fairness, making proposed flow control algorithms suitable for realistic scenarios.

References

[1] L. Benini, and G. DeMicheli, "Networks on Chips: A New SoC Paradigm," *Computer*, vol. 35, no. 1, pp. 70-78, 2002.

[2] W. J. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," *Design Automation Conference*, pp. 684-689, 2001.

[3] M. S. Talebi, F. Jafari, and A. Khonsari, "A Novel Flow Control Scheme for Best Effort Traffic in NoC Based on Source Rate Utility Maximization," *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, pp. 381-386, 2007.

[4] M. S. Talebi, F. Jafari, A. Khonsari, and M. H. Yaghmaee, "A Novel Congestion Control Scheme for Elastic Flows in Network-on-Chip Based on Sum-Rate Optimization," *International Conference on Computational Science and its Applications*, pp. 398-409, 2007.

[5] F. P. Kelly, A. Maulloo, and D. Tan, "Rate control for communication networks: Shadow prices, proportional fairness, and stability," *Operational Research Society*, vol. 49, no. 3, pp. 237-252, 1998.

[6] Y. Gu, H. O. Wang, and Y. Hong, "A predictive congestion control algorithm for high speed communication networks," *American Control Conference*, vol. 5, pp. 3779-3780, 2001.

[7] C. Yang and A. V. S. Reddy, "A taxonomy for congestion control algorithms in packet switching networks," *IEEE Network*, vol. 9, no. 4, pp. 34-45, 1995.

[8] J. Hu and R. Marculescu, "DyAD - smart routing for networks-on-chip," *Design Automation Conference*, pp. 260-263, 2004.

[9] U. Y. Ogras and R. Marculescu, "Prediction-based flow control for network-on-chip traffic," *Design Automation Conference*, pp. 839-844, 2006.

[10] J. W. van den Brand, C. Ciordas, K. Goossens, and T. Basten, "Congestion-Controlled Best-Effort Communication for Networks-on-Chip," *Design, Automation and Test in Europe Conference*, pp. 948-953, 2007.

[11] F. Jafari,M. S. Talebi, A. Khonsari, and M. H. Yaghmaee, "Best Effort Flow Control Mechanisms in on-Chip Architectures," Technical Report, TRCS2008-30, IPM, School of Computer Science, 2008.

[12] S. H. Low, and D. E. Lapsley, "Optimization Flow Control I: Basic Algorithm and Convergence," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 861-874, 1999.

[13] J. Mo, J. Walrand, "Fair End-to-End Window-Based Congestion Control," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 556-567, 2000.

[14] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, 1999.

[15] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

[16] D. P. Bertsekas and R. Gallager, *Data Networks*, Prentice-Hall, 1992.

[17] J. Y. Le Boudec, "Rate adaptation, Congestion Control and Fairness: A Tutorial," Ecole Polytechnique Federale de Lausanne (EPFL), 2001.

[18] R. Jain, D. Chiu, and W. Hawe, "A Quantitative Measure of Fairness And Discrimination For Resource Allocation In Shared Computer Systems," DEC Research Report TR-301, 1984.