# Proportionally-Fair Best Effort Flow Control in Network-on-Chip Architectures

Mohammad S. Talebi[1], Fahimeh Jafari[1,2], Ahmad Khonsari[3,1], and Mohammad H. Yaghmaee[2]
[1] *IPM, School of Computer, Tehran, Iran*
[2] *Ferdowsi University of Mashhad, Mashahhad, Iran*
[3] *ECE Department, University of Tehran,Tehran, Iran*
*mstalebi@ipm.ir, jafari@ipm.ir, ak@ipm.ir, hyaghmae@ferdowsi.um.ac.ir*

## Abstract

*The research community has recently witnessed the emergence of Multi-Processor System on Chip (MPSoC) platforms consisting of a large set of embedded processors. Particularly, Interconnect networks methodology based on Network-on-Chip (NoC) in MP-SoC design is imminent to achieve high performance potential. More importantly, many well established schemes of networking and distributed systems inspire NoC design methodologies. Employing end-to-end congestion control is becoming more imminent in the design process of NoCs. This paper presents a centralized congestion scheme in the presence of both elastic and streaming flow traffic mixture. In this paper, we model the desired Best Effort (BE) source rates as the solution to a utility maximization problem which is constrained with link capacities while preserving Guaranteed Service (GS) traffics services requirements at the desired level. We proposed an iterative algorithm as the solution to the maximization problem which has the benefit of low complexity and fast convergence. The proposed algorithm may be implemented by a centralized controller with low computation and communication overhead*

## 1. Introduction

The high level of system integration characterizing Multi-Processor Systems-on-Chip (MPSoCs) is raising the scalability issue for communication architectures. Towards this direction, traditional system interconnects based on shared busses are evolving both from the protocol and the topology viewpoint. Advanced bus protocols acts in favor of better exploitation of available bandwidth, while more parallel topologies are instead being introduced in order to provide more bandwidth [1].

In the long run, many researchers and SoC designers agree on the fact that this trend approaches the Network-on-Chip (NoC) as a solution to the lack of SoCs' Scalability [2].

A NoC system fundamentally consists of three components: switches, Network Interfaces (NIs) and links. The switches can be arbitrarily connected to each other and to NIs, based on a specified topology. They are responsible for routing, switching and flow control logic, as well as error control handling. NIs are responsible for packetization/depacketization and implement the service levels associated with each transaction.

Recently, Quality-of-Service (QoS) provisioning in NoC's environment has attracted many researchers and currently it is the focus of many literatures in NoC research community. NoCs are expected to serve as multimedia servers and are required not only to carry Elastic Flows, i.e. BE traffic, but also Inelastic Flows, i.e. GS traffic which requires tight performance constraints such as necessary bandwidth and maximum delay boundaries.

It's obvious that a network with data services needs some mechanisms 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 problem in NoCs and to the best of our knowledge only few works has been carried out in this field. Congestion control, or equivalently, flow control in NoCs mainly focuses on the resource constrained on-chip designs, with the aim of minimizing the network cost or maximizing network utility while maintaining the required Quality-of-Service (QoS).

## 2. Related Works

Flow control for data networks is a widely-studied issue [3]-[6]. A wide variety of flow control mechanisms in data network belongs to the class of End-to-End control schemes, like TCP/IP, which is mainly based on the window-based scheme. In this methods, routers and intermediate nodes avoid the network from becoming congested by means of packet dropping deterministically (as in DropTail) or

978-1-4244-1694-3/08/$25.00 ©2008 IEEE

randomly (as in RED). Therefore, sent packets are subject to loss and the network must aim to providing an acknowledgement mechanism. On the other On-chip networks pose different challenges. The reliability of on-chip wires and more effective link-level flow-control allows NoCs to be loss-less. Therefore, there is no need to utilize acknowledgment mechanism and we face to slightly different concept of flow control.

So far, several works have focused on this issue for NoC systems. In [7], a prediction-based flow-control strategy for on-chip networks is proposed in which each router predicts the buffer occupancy to sense congestion. This scheme controls the packet injection rate and regulates the number of packets in the network. In [8] link utilization is used as a congestion measure and a Model Prediction-Based Controller (MPC), determines the source rates. Dyad [9] controls the congestion by using adaptive routing when the NoC faces congestion.

In this paper, we focus on the flow control for BE traffic as the solution to a utility-based optimization problem. To the best of our knowledge, none of the aforementioned works have dealt with the flow control problem through utility optimization approach. In our seminal work [10], we have modeled desired BE source rates as the solution to a utility-based optimization problem with general form utility function and aimed at the issue with solving the proposed problem using Newton method. In [11], we also have considered this issue via sum-rate optimization problem and used a different approach to solve the problem. This paper we address the performance analysis of our seminal work [10] with a special utility function which satisfies Proportional Fairness feature and solve the flow control problem using a different approach which leads to low complexity flow control algorithm for BE traffic in NoCs.

This paper is organized as follows. In Section 3 we present the system model and formulate the underlying optimization problem for BE flow control. In section 4 we proceed to the proposed algorithm and discuss about some remarks. In section 5 we solve the optimization problem using an iterative algorithm over its dual and analyze the convergence behavior of it and present the underlying theorem of its convergence. Section 6 presents the simulation results. Finally, the section 7 concludes the paper and states some future work directions.

## 3. System Model and Flow Control Problem

We consider a NoC architecture which is based on a two dimensional mesh topology and wormhole routing. In wormhole networks, each packet is divided into a sequence of *flit*s which are transmitted over physical links one by one in a pipeline fashion. A hop-to-hop credit mechanism assures 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 an optimization problem. For the sake of convenience, 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 $L$ and a set of sources $S$. A source consists of Processing Elements (PEs), routers and Input/Output ports. Each link $l \in L$ is a set of wires, busses and channels that are responsible for connecting different parts of the NoC and has a fixed capacity of $c_l$ packets/sec. We denote the set of sources that share link $l$ by $S(l)$. Similarly, the set of links that source $s$ passes through, is denoted by $L(s)$. By definition, $l \in S(l)$ if and only if $s \in L(s)$.

As discussed in section I, there are two types of traffic in a NoC: Guaranteed Service (GS) and Best Effort (BE) traffic. For notational convenience, we divide $S$ into two parts, each one representing sources with the same kind of traffic. In this respect, we denote the set of sources with BE and GS traffic by $S_{BE}$ and $S_{GS}$, respectively. Each link $l$ is shared between the two aforementioned traffics. GS sources will obtain the required amount of the capacity of links and BE sources benefit from the remainder.

Our objective is to choose source rates with BE traffic so that to maximize the weighted sum of the logarithm of the BE source rates. Hence the maximization problem can be formulated as [12]:

$$\max_{x_s} \sum_{s \in S_{BE}} a_s \log x_s \tag{1}$$

subject to:

$$\sum_{s \in S_{BE}(l)} x_s + \sum_{s \in S_{GS}(l)} x_s \le c_l \qquad \forall l \in L \tag{2}$$

$$x_s > 0 \qquad \forall s \in S_{BE} \tag{3}$$

Optimization variables are BE source rates, i.e. $(x_s,\ s \in S_{BE})$ and $a_s$ is the weight for source $s$. We later on discuss how such a weight determines the priority of source $s$ in resource allocation. The constraint (2) states that the sum of BE source rates passing thorough link $l$ cannot exceed its free

capacity, i.e. the portion of $c_l$ which has not been allocated to GS traffic.

In General, problem (1) belongs to the class of utility-based optimization problems, for which the utility function, $U_s$, is assumed to be logarithmic, i.e. $U_s(x_s) = a_s \log x_s$. Such utility functions, are positive, concave and strictly increasing, as logarithmic function does. There are many choices for utility function, other than logarithmic, with specific features and behavior. We discuss in section V, that logarithmic utility function have nice properties in terms of economic terminology, known as proportional fairness [3].

It is worth to mention that despite the restriction of ourselves to a specific utility function, our work can be easily generalized to arbitrary utility functions, as in our seminal work [10].

With the model above, problem (1) is a convex optimization problem with linear constraints. Hence it admits a unique maximizer [13][14], i.e. there exists an optimal source rate vector, $x^* = (x_s^*, s \in S_{BE})$ that maximizes the objective of problem (1) while satisfying capacity constraints.

Problem (1) is coupled across the network through its constraints. Such a coupled nature, necessitate usage of centralized methods like Interior Point method which poses great computational overhead onto the system [13][14] and hence is of little interest.

In contrast, there are several low-complexity and distributive methods to solve unconstrained problems. Hence, one way to reduce the computational complexity is to transform the constrained optimization problem into its *Dual*, which can be defined to be unconstrained. According to the Duality Theory [13][14], each convex optimization (maximization) problem has a dual, whose optimal solution, called *Dual-Optimal*, leads to best bound (upper bound) of the optimal solution of the main problem. In this respect, the main problem is retroactively called *Primal Problem*. As the dual problem can be defined in such a way to be unconstrained, solving the dual is much simpler than the primal.

For notational convenience, we define:

$$\hat{c}_l = c_l - \sum_{s \in S_{GS}(l)} x_s \tag{4}$$

We also define the source rate vector (for BE traffic) and link capacity vector as $x = (x_s, s \in S_{BE})$ and $\hat{c} = (\hat{c}_l, l \in L)$, respectively. To avoid confusing with summations indices, we define Routing matrix, i.e. $R = [R_{ls}]_{L \times S}$, as following:

$$R_{ls} = \begin{cases} 1 & \text{if } s \in S_{BE}(l) \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Using the abovementioned definitions, problem (1) can be rewritten as:

$$\max_{x_s} \sum_s a_s \log x_s \tag{6}$$

subject to:

$$Rx \le \hat{c} \tag{7}$$

$$x_s > 0 \qquad \forall s \in S_{BE} \tag{8}$$

## 4. Optimal Flow Control Algorithm

In this section, we present a centralized flow control algorithm for BE traffic in NoC systems which controls the BE source rates in favor of problem (1). Later, in section V, we show that solving problem (1) leads to the proposed algorithm, and therefore the algorithm is an iterative optimal solution to it. The proposed flow control algorithm is listed below as algorithm 1.

In the sequel, we make some worth-mentioning remarks. Performance analysis of the algorithm is to be discussed in the next section.

**Remarks:**

**1.** Considering algorithm 1 as a centralized algorithm, we consider a simple controller that can be mounted in the NoC, whether as a separate hardware module or a part of the operating system, which is responsible for running of the algorithm. From computational aspect, such a controller must have the ability of carrying out simple mathematical operations, as in Algorithm 1. Another necessary requirement of the controller, as Output section of the algorithm 1 suggests, is some links e.g. a control bus, to communicate the algorithm output to the BE sources.

Although Algorithm 1 is centralized, it can be easily casted into a distributive one upon introducing low communication overheads. Thus it can be addressed in decentralized scenarios, too. However, due to well-formed structure of NoC Systems, such a centralized algorithm suits for the system and thereafter we only focus on the centralized scheme.

**2.** The proposed flow control algorithm is very similar to End-to-End congestion control schemes in data networks, also known as TCP which are widely used to control BE data flow in the internet. End-to-End schemes use window-based method, i.e. each

**Algorithm 1: Flow Control for BE in NoC**

***Initialization:***

1. **Initialize $c_l$ of all links.**
2. **Set link shadow price vector to zero.**
3. **Set the $\varepsilon$ as the stopping criteria.**

***Loop:***

**Do until** $(\max\left|x_s(t+1)-x_s(t)\right|<\varepsilon)$

1. $\forall l \in L$ **: Compute new link prices:**

$$\lambda_l(t+1)=\Big[\lambda_l(t) -\gamma\left(c_l(t)-\sum_{s\in S_{GS}(l)}x_s(t)-\sum_{s\in S_{BE}(l)}x_s(\lambda(t))\right)\Big]^+$$

2. **Compute new BE source rates as follows**

$$x_s(t+1)=\frac{a_s}{\sum_l R_{ls}\lambda_l(t+1)}$$

***Output:***

**Communicate BE source rates to the corresponding nodes.**

source maintains a window of packets which are transmitted, but not acknowledged. Because the packets in data networks may be lost due to dropping at the routers or link failure, destination should acknowledge the ordered receipt of each packet in the current window. Each source changes its window size in response to congestion signals, i.e. negative acknowledges or duplicates ones, and thereby avoids the network to face congestion. Roughly, the source rate in each round trip (i.e. the way from source to destination and back to the source for acknowledgment), is the ratio of window size to the round trip time (i.e. duration of the trip).

Although flow control in TCP is carried out by means of window updates, however we can derive the corresponding rate updates, too. The proposed flow control algorithm is very similar to rate update in TCP scheme. Such a similarity stems from the similarity in the underlying flow control problem in both schemes. However, it is worth noting that unlike TCP, in algorithm 1 we have not considered any window based transmission and acknowledgement mechanism. This is due to the fact that NoC architecture is lossless, as previously stated in section III, and hence all packets will be delivered successfully and no acknowledgment is needed.

# 5. Performance Analysis: Optimal Solution and Convergence Analysis

In this section, we discuss that solving problem (1) through its Dual, leads to Algorithm 1. Towards this end, we first obtain the Dual of problem (1) and then solve it using Gradient Projection Method [14][15] and derive the abovementioned flow control algorithm. Then, we focus on the convergence behavior and other aspects of the proposed algorithm.

## 5.1. Dual Problem

In this part, we will obtain the dual of problem (1). Using the standard optimization methods [12], the Lagrangian of the problem (1) can be written as:

$$L(x,\lambda)=\sum_s a_s\log x_s-\sum_l\lambda_l(\sum_s R_{ls}x_s-\hat{c}_l) \qquad (9)$$

where $\lambda_l>0$ is the Lagrange Multiplier associated with constraint (2) for link $l$. Usually, $\lambda_l$ is called *shadow price* [12] for the economic interpretation of its role in solving the primal problem through dual.

Regarding the Lagrangian of problem (1), the dual function is defined as [13]:

$$g(\lambda)=\sup_{x_s}L(x,\lambda) \qquad (10)$$

where $\lambda$ is the vector of positive Lagrange multipliers. Thus the dual function is given by:

$$\begin{aligned} g(\lambda)&=\max_{x_s}\sum_s a_s\log x_s-\sum_l\lambda_l(\sum_s R_{ls}x_s-\hat{c}_l)\\ &=\max_{x_s}\sum_s\left(a_s\log x_s-x_s\sum_l R_{ls}\lambda_l\right)+\sum_l\lambda_l\hat{c}_l \end{aligned} \qquad (11)$$

By Karush-Kuhn-Tucker (KKT) Theorem [13], we can obtain optimal source rates, i.e. $x^*=(x_s^*, s\in S_{BE})$. Duality theory states that when the primal problem is convex, strong duality holds and thereby the duality gap is zero [13]. In this respect, the optimal source rate vector, $x^*$, corresponds to the optimal Lagrange multiplier vector, $\lambda^*$ [13]. In other words, if $x$ is a feasible point of the primal problem, which is primal-optimal the corresponding $\lambda$ will be dual-optimal and vice versa. Therefore, at optimality we have

$$\nabla_x L(x,\lambda)\Big|_{(x^*,\lambda^*)}=\mathbf{0} \qquad (12)$$

where $\mathbf{0}$ is a vector with all zero. By taking the derivative of (9) with respect to $x$, we have

$$\frac{\partial L}{\partial x_s}\Big|_{(x_s^*,\lambda^*)} = \frac{a_s}{x_s^*} - \sum_l R_{ls}\lambda_l^* = 0 \tag{13}$$

$$x_s^* = \frac{a_s}{\sum_l R_{ls}\lambda_l^*} \tag{14}$$

Substituting $x_s^*$ into (11) yields

$$g(\lambda) = \sum_s \left( a_s(\log a_s - 1) - a_s \log(\sum_l R_{ls}\lambda_l) \right) + \sum_l \lambda_l \hat{c}_l \tag{15}$$

The dual problem is defined as [13]:

$$\min_{\lambda \geq 0} g(\lambda)$$

therefore, we have

$$\min_{\lambda \geq 0} \sum_s \left( a_s(\log a_s - 1) - a_s \log(\sum_l R_{ls}\lambda_l) \right) + \sum_l \lambda_l \hat{c}_l \tag{16}$$

It is proven that the dual is always convex regardless of convexity or non-convexity of the primal problem [13]. Moreover, it is apparent from (16) that, by ignoring the mild condition on the positivity of $\lambda$, the dual problem is unconstrained. As dual problem is convex, it admits a unique optimal, i.e. a unique minimizer, which can be obtained using iterative algorithms. As the dual problem is unconstrained; solving (16) using iterative methods is much simpler than the primal.

## 5.2. Solving The Dual Problem

In this part, we will solve the dual problem using Projected Gradient Method [13] and derive algorithm 1.

The Projected Gradient Method adjusts shadow prices, i.e. Lagrange multiplier vector, in opposite direction to the gradient of the dual function, i.e. $\nabla g(\lambda)$, as follows:

$$\lambda(t+1) = \left[ \lambda(t) - \gamma \nabla g(\lambda(t)) \right]^+ \tag{17}$$

where $\gamma > 0$ is a constant stepsize, and $[x]^+ \triangleq \max\{x, 0\}$. Since the objective of problem (1) is strictly concave, $g(\lambda)$ is continuously differentiable [13], hence $\nabla g(\lambda)$ exists. Using (15), the $l$-th element of the gradient vector is given by:

$$\frac{\partial g(\lambda)}{\partial \lambda_l} = \frac{\partial}{\partial \lambda_l} \left[ \sum_s \left( a_s(1 - \log a_s - \log(\sum_l R_{ls}\lambda_l)) \right) + \sum_l \lambda_l \hat{c}_l \right] \tag{18}$$

Therefore,

$$\frac{\partial g(\lambda)}{\partial \lambda_l} = \hat{c}_l - \sum_s \frac{R_{ls} a_s}{\sum_l R_{ls}\lambda_l} \tag{19}$$

Regarding (14), (19) can be rewritten as:

$$\begin{aligned} \frac{\partial g(\lambda)}{\partial \lambda_l} &= \hat{c}_l - \sum_s R_{ls} x_s(\lambda) \\ &= \hat{c}_l - \sum_{s \in S(l)} x_s(\lambda) \end{aligned} \tag{20}$$

and the update equation is given by:

$$\lambda_l(t+1) = \left[ \lambda_l(t) - \gamma \left( \hat{c}_l - \sum_{s \in S_{BE}(l)} x_s(\lambda(t)) \right) \right]^+ \tag{21}$$

where $\lambda(t+1) = \left( \lambda_l(t+1),\ l \in L \right)$ and $x_s(\lambda(t))$ is the approximate of $x_s^*$ in time $t$. (14) and (21) together forms the proposed algorithm. Therefore, algorithm 1 is the iterative solution to problem (1).

## 5.3. Convergence Analysis

In this part, we investigate the convergence behavior of the proposed algorithm. As stepsize has an important role in the convergence behavior of the update equation, we mainly focus on the effect of stepsize. The conditions under which Algorithm 1 converges and performance analysis of the algorithm will be obtained with respect to the choice of stepsize.

There are several choices for stepsize, each one belonging to a predefined category and having certain advantages and drawbacks (see [16] and references herein). In the family of gradient algorithm for distributed scenarios, stepsize is usually chosen to be a small enough constant so that to guarantee the convergence of the algorithm. Constant stepsize is robust in the sense of convergence in time-varying conditions and asynchronous schemes. However, it usually has slower convergence rate than time-varying ones. Due to its simplicity and robustness, in this paper we have used a constant step-size.

Before proceeding to the theorem, we first present the fundamental lemma for the gradient optimization algorithms.

**Lemma 1 [14]:** *Consider the unconstrained minimization problem,*

$\min_x f(x)$

*with the minimal* $x^*$. *If* $\nabla f(x)$ *has Lipschitz Continuity property, i.e. there exist* $L$ *such that*

$$\left|\nabla f(x_1) - \nabla f(x_2)\right| \le L\left\|x_1 - x_2\right\|_2 \quad (22)$$

*then the sequence* $x(t)$ *defined as*

$$x(t+1) = x(t) - \gamma \nabla f(x(t))$$

*converges to the neighborhood of* $x^*$ *provided that*

$$\varepsilon \le \gamma \le \frac{2-\varepsilon}{L} \quad (23)$$

*for some* $\varepsilon > 0$,

**Proof:** *See [14].*

The following theorem, determines the condition on the stepsize, under which the Algorithm 1 converges to the neighborhood of the optimal of the problem (16) and thereby that of problem (1).

**Theorem 1:** *The iterative flow control scheme proposed by (14) and (21) converges to a neighborhood of the optimal point of the primal problem (1) provided that*

$$0 < \gamma \le \frac{2\underline{a}}{\overline{c}^2 \overline{L}\overline{S}} \quad (24)$$

*where* $\overline{S}$ *is the length of the longest path used by the sources,* $\overline{L}$ *is the number of sources sharing the most congested link,* $\underline{a}$ *is the minimum weight of sources and* $\overline{c}$ *is the upper bound on link capacities.*

**Proof:** Omitted due to space limit.

## 5.4. Proportional Fairness

Utility function directly influences the policy by which system resources, i.e. bandwidth, are shared among the competing sources. In this respect, in terms of economics terminology, utility function controls the *fairness* among users or sources. Several fairness criteria have been defined in the economics which are applicable to problem (1). Among them are *Max-Min Fairness* and *Proportional Fairness* [3]. In a system with Max-Min fairness, the resources are mainly shared in favor of weak users while in system with Proportional Fairness the resources are shared in proportion to the resource usage of each source. In the latter case, given an optimal source rate allocation $x^* = \left(x_s^*,\ s \in S\right)$ satisfying Proportional Fairness, with any other feasible source rate, say $x = \left(x_s,\ s \in S\right)$, the total proportional net benefit gained by the new source rates is decreased [3], i.e.:

$$\sum_s \frac{x_s - x_s^*}{x_s^*} \le 0 \quad (25)$$

It is proven, systems with proportional fairness that satisfies (25), must have logarithmic utility functions [3], i.e.

$$U_s(x_s) = \log x_s \quad (26)$$

Thus the proposed flow control algorithm, with equal weight factors will be proportionally fair. It is worth to note that the case of heterogeneous weight factors corresponds to another implementation of fairness, the so-called Weighted Proportionally Fair, for which (25) turns to be

$$\sum_s a_s \left(\frac{x_s - x_s^*}{x_s^*}\right) \le 0 \quad (27)$$

In the sequel, we briefly discuss about the effect of weight factors. As previously stated, $a_s$ is the weight for source $s$ in the optimization problem which controls the priority of source $s$ in resource sharing. To gain more insights on the role of $a_s$ in the flow control, we consider a simple network with a single bottleneck link, say link $l'$. Since all other links doesn't saturate, we have $\lambda_l = 0,\ \ l \ne l'$. Using (2) and (14) we have:

$$x_s = \frac{a_s}{\sum_{l \in L(s)} \lambda_l} = \frac{a_s}{\lambda_{l'}}, \qquad s \in S(l') \quad (28)$$

$$\frac{x_i}{a_i} = \frac{x_j}{a_j} = \ldots = \frac{x_n}{a_n} = \frac{1}{\lambda_{l'}} \quad i,j,n \in S(l') \quad (29)$$

$$\sum_{s \in S(l')} x_s = c_{l'} \Rightarrow \lambda_{l'} \sum_{s \in S(l')} a_s = c_{l'} \Rightarrow \lambda_{l'} = \frac{c_{l'}}{\sum_{s \in S(l')} a_s} \quad (30)$$

combining (28)-(29), leads to

$$x_i = \frac{a_i c_{l'}}{\sum_{s \in S(l')} a_s} \qquad \forall i \in S(l') \quad (31)$$

Therefore, (31) shows that in a network with single congested link, the sources passing through the congested link, achieve their rates in proportion to their weights. For networks with multiple congested

links, such an insight might not be easily seen, however weight factors influence the capacity sharing at bottle neck links. In this respect, we can allocate more resources, i.e. link capacity, to some specified sources by assigning larger weights to them.

## 6. Simulation Results

In this section we examine the proposed flow control algorithm, listed above as Algorithm 1, for a typical NoC architecture. In our scenario, we have used a NoC with $4 \times 4$ Mesh topology which consists 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 bit long.

In order to simulate our scheme, some nodes are considered to have a GS data (such as Multimedia, etc.) to be sent while other nodes have a BE traffic. As stated before, GS sources will obtain the required amount of the capacity of links and the remainder should be allocated to BE traffics. Routing policy for BE sources is shown in Fig. 1. We assume that all sources have logarithmic utility function of the form $U_s(x_s) = a_s \log x_s$ where $a_s$ represents the weight factor for source $s$. In the sequel, we present our results in the following parts as below.

One of the most significant issues of our interest is the convergence behavior of the source rates. In this part, we have simulated our scheme using 2 different values for step-size, 1.05 and 0.2, respectively. Weight factor for all sources is assumed to be unity. The convergence behavior of source rates for after 150 iterations is depicted in Fig. 2(a)-(b). Regarding Fig. 2(a), it's apparent that for $\gamma = 1.05$, after 20 iteration steps the source rates will have very little variations, however, from Fig. 2(b) , i.e. for $\gamma = 0.2$, these threshold of iterations will be at least 85 steps.

In order to have a better insight about the algorithm behavior, the relative error with respect to optimal source rates which is averaged over all active sources, is also shown in Fig. 3. Optimal values are obtained using CVX [17] which is MATLAB toolbox for solving disciplined convex optimization problems. Fig. 3 reveals the first step size leads to less than 10% error in average just after about 13 iteration steps, and after 20 steps the average error lies below 5%. However, the second step size would reach the two aforementioned error margins at the expense of iterating for about 60 and 75 steps, respectively. Although not shown in Fig. 3, with much more iteration steps simulation results verify that the average error curve for the smaller step size lies below that of larger step size. However, for practical implementations and real world applications, due to faster convergence speed, larger step size is more appropriate.

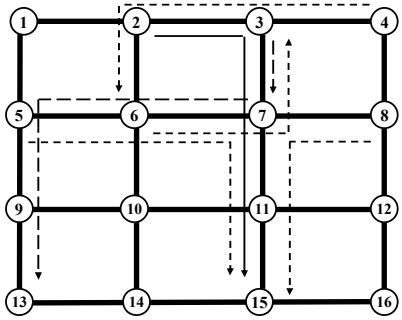
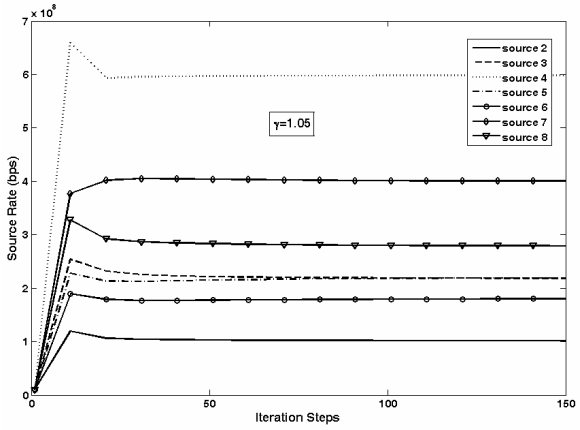
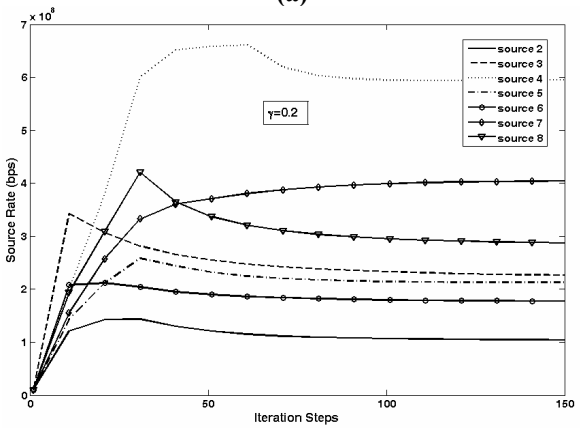



**Fig. 1. Network Topology and Routing Policy**




**(a)**




**(b)**

**Fig. 2. Source rates convergence with symmetric weight factors for (a) $\gamma = 1.05$ and (b) $\gamma = 0.2$**

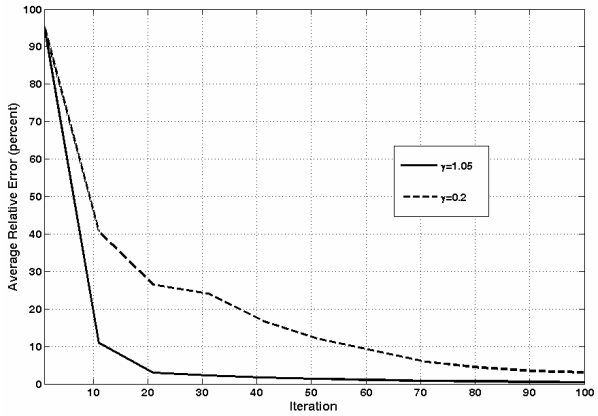



**Fig. 3. Average Relative Error**

## 7. Conclusion and Future Works

In this paper we addressed the problem of flow control for BE traffic in NoC systems. Flow control was considered as the solution to the utility maximization problem which was solved indirectly through its dual using gradient projection method. This was led to an iterative algorithm which can be used to determine optimal BE source rates.

The algorithm can be implemented by a controller which admits a light communication and communication overhead to the system. We have also investigated the convergence behavior of the algorithm. Further investigation about the effect of delay incurred by the proposed algorithm is the main direction of our future studies.

## 8. References


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

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

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

[4] S. Mascolo, "Classical control theory for congestion avoidance in high-speed internet" *Decision and Control IEEE Conference*, 1999, vol. 3, pp. 2709-2714.

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

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

[7] U. Y. Ogras, and R. Marculescu, "Prediction-based flow control for network-on-chip traffic." *In Proceedings of the Design Automation Conference,* 2006.

[8] 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*, 2007, pp. 948-953.

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

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

[11] 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*, 2007, pp. 398-409.

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

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

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

[15] Bertsekas, D. P., and J. N. Tsitsiklis, *Parallel and distributed computation*, Prentice-Hall, 1989.

[16] Boyd, S., *Convex Optimization II Lecture Notes*, Stanford University, 2006.

[17] Grant, M., S. Boyd, and Y. Ye, *CVX (Ver. 1.0RC3): Matlab Software for Disciplined Convex Programming*. Download available at: http://www.stanford.edu/~boyd/cvx.