IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 19, 2019, accepted January 13, 2020, date of publication January 17, 2020, date of current version February 26, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2967459

A Fine-Grained Source-Throttling Method for Mesh Architectures

HONGZHI ZHAO[1], (Member, IEEE), NADER BAGHERZADEH[2], (Fellow, IEEE), QIANG WANG[1], AND YONGCHANG WANG[1]

[1]Beijing Key Laboratory of Transportation Data Analysis and Mining, School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China
[2]Department of Electrical Engineering and Computer Science, University of California–Irvine, Irvine, CA 92697, USA

Corresponding author: Yongchang Wang (19112030@bjtu.edu.cn)

This work was supported by the Fundamental Research Funds for the Central Universities of China (Beijing Jiaotong University) under Grant 2019JBM017.

ABSTRACT Source-throttling algorithms aim at adjusting network load appropriately so that high network throughput can be maintained and latency can be decreased efficiently. One challenge of designing a source-throttling algorithm is how to precisely evaluate network congestion status. State-of-the-art source-throttling algorithms for mesh-connected network-on-chip usually evaluate congestion status of the whole network or only that of neighbor routers. They usually appear empirical or locally-greedy. To avoid such problems, we propose a fine-grained source-throttling algorithm. Our main contributions are: 1, we can quantify the impact of throttling action on network routers; 2, we can monitor network traffic more intelligently compared with previous work, only the routers that are most affected by the source's throttling action are used to evaluate congestion status of the RSD network. Based on such measures, most throttling parameters shall not be empirically set but be precisely figured out by a smarter searching algorithm of finding anchor routers as the throttling baseline. Thus, it can solve over-throttling and under-throttling problems more precisely. Time complexity of proposed method is only $O(M * N)$ (M*N is the scale of 2D mesh network). Simulation results show that it has higher throughput, lower fluctuation and lower latency than source-throttling algorithms with empirical parameters such as INC and self-tuned technologies under different network scenarios.

INDEX TERMS Source-throttling algorithm, fine-grained, accurate throttling parameters, over-throttling, under-throttling.

I. INTRODUCTION

2D mesh NoC(Network-on-Chip) has been the most suitable interconnect infrastructure for MPSoCs which provides high computing performance for most scientific and commercial applications [1]. 2D mesh NoC is available for different kinds of traffic patterns. When traffic load is not heavy, NoC can work well. However, as traffic load increases, 2D mesh NoC begins to suffer from network congestion problem. Network congestion leads to quick reduction of NoC network throughput, and the rapid increment of end-to-end communication delay due to contention (i.e. many packets compete for the same routers, which leads to the congestion of router). Severe network congestion even makes the whole NoC unavailable [2], [3]. How to alleviate network congestion is a major challenge for designing NoC. There are two main congestion control methods used to alleviate 2D mesh NoC network congestion. One of them is congestion-awared adaptive routing algorithm [4]–[6]. This method transfers packets from routers with heavy load to routers with light load in order to distribute traffic load into network as evenly as possible. When network load is not heavy, such methods can decrease average end-to-end communication delay and improve network throughput. When network load becomes heavy, tree saturation comes into being and then spreads to the whole network [7], [8]. The whole network congestion will consequently come into being. At this time, routing algorithms don't help alleviate network congestion.

Another kind of congestion control method is source-throttling algorithms [2], [3], [9]–[11]. When network load becomes heavy and network congestion is detected, they can decrease network load by throttling network nodes' packets

The associate editor coordinating the review of this manuscript and approving it for publication was Songwen Pei.

injection. Network congestion can be consequently alleviated or prevented in advance by this approach. On the other hand, network node's packet injection rate will be increased by them to maximize network throughput when no congestion is detected. Thus, by using source-throttling algorithms, network congestion can be avoided and network performance can be maximized.

Compared with routing algorithms, source-throttling algorithms can control network congestion better. However, it is difficult for source-throttling algorithms to precisely control the throttling rate. If it sometimes throttles node's injection rate too much, i.e., over-throttling, network throughput will be low even if no congestion happens and latency is also low in the same time. On the other hand, if it sometimes throttles node's injection rate too little, i.e., under-throttling, congestion will happen and latency will be very high. Network throughput will consequently be decreased dramatically. To maintain high network throughput and low latency, a well-designed source-throttling algorithm should control the throttling rate as precisely as possible to avoid over-throttling and under-throttling problems simultaneously [2], [3], [12].

To precisely control the throttling rate, some throttling parameters such as congestion status should be analyzed precisely [2], [13]. This paper presents a fine-grained source-throttling method for mesh-connected NoC. It is comprised of two key components: a method of analyzing the impact of throttling action on network routers, and a technique of quantifying the critical factors causing network congestion. Compared with some state-of-the-art source-throttling algorithms, it can achieve a more appropriate tradeoff between preventing upcoming new congestion and maintaining high network throughput.

The remainder of this paper is organized as follows. Section 2 reviews related work of throttling algorithms. Section 3 presents a concept of CCR (Congestion-Contribution-Rate) to analyze the impact of source nodes' injection action on on-chip routers in a fine-grained manner. A concept "anchor router" is also presented to quantify the critical factor causing network congestion. Section 4 proposes a fine-grained source-throttling method based on the concepts of "CCR" and "anchor router." Section 5 simulates and compares two other kinds of source-throttling algorithms with proposed method, and then analyze their difference. Section 6 concludes the paper.

II. RELATED WORK

To control the throttling rate as precisely as possible, source-throttling algorithms usually consist of three parts [2], [3], [12]: the first part is a congestion-awareness scheme which detects congestion status of a part or all of the network; the second part is a congestion-estimation mechanism which evaluates the status of network congestion; the third one is a load-adjusting mechanism which sets some parameters and then throttles a source node's injection rate.

The status of network congestion can be detected in different ways such as the number of a local router's busy buffers or neighbor routers' busy buffers [14], [15], time-out value [16], applications' cache miss rate or its variants [17], [18] and so on. According to the range of being aware of network congestion, there are local-knowledge based mechanisms and global-knowledge based mechanisms. Some local-knowledge based source-throttling algorithms are only aware of the congestion status of a single router [3]. For example, the INC algorithm counts a subset (free and useful) of virtual channel buffers in a router to decide whether to throttle or not [2]. Their throttling decision is helpful to alleviate local congestion, but is not efficient for alleviating congestion that has appeared in other network parts [13]. The reason is that some packets from the source node are forwarded through not only the local area but also other network parts. These packets exert their influence on both the congestion status of local routers and that of other routers. That is to say, local-knowledge based source-throttling algorithms usually cause local optimum problem. Congestion status of the whole network easily falls into under-throttling or over-throttling dilemma [2]. The main drawback of such algorithms is their locally-greedy awareness.

Considering the drawback of local-knowledge based mechanism, many source-throttling algorithms are aware of the congestion status of the whole network, i.e., global-knowledge based mechanisms [2], [3], [17]. The simplest available approach of achieving the global congestion status is to evaluate the time-out value which originates from computer networks [12], [16]. It is impossible to quantify the global network status accurately under different scenarios. These constraints prevent reaching efficient solutions [12], [16]. Contrary to other networks, NoC can quantify the whole network congestion status because of its flexibility, fixed topology and other related parameters.

For the case of NoC, evaluating global information with time-out value used in computer network is too complex. Many source-throttling algorithms for NoC scenario can evaluate of the whole network congestion status more accurately than time-out based methods do [2], [18]. For example, Nychis, Fallin, etc. [17] proposed a source-throttling method with application-level awareness for bufferless NoC. They noticed that starvation rate grew super-linearly with network utilization and was a more accurate indicator of the level of congestion than network latency in a bufferless NoC. They used Instructions-per-Flit (IPF) based on PE's L1 cache miss rate to throttle applications' execution. But empirical values were used for starvation rate, threshold upper bounds and threshold lower bounds. Additionally, how far a starvation rate was affected by the congested area was not analyzed. Similarly, a source-throttling mechanism called HAT was proposed. It was application-aware and network-load-aware [18]. It used the sum of all non-network-intensive applications' L1 Cache's MPKI (misses per thousand instructions) to evaluate global network-congestion status. Then it used several global throttling parameters such as applications' cache miss rate, global throttling rate and the congestion threshold "nonIntensiveCap" to throttle network-intensive application's injection rate. There was no quantitative standard to decide whether an application was network-intensive or not. And its parameters were almost all empirically set.

Another global-knowledge based source-throttling method called "self-tuned congestion control technique" [13] used the number of full buffers over the whole network as the global congestion information. Its global congestion information was not empirically evaluated. A self-tuned technique used constant additive increments and decrements to update the throttling threshold value. Compared with source-throttling methods such as HAT [18] whose parameters are all empirical, self-tuned method can evaluated network congestion status more accurately. However, its other parameters such as the tuning period and the bandwidth dropping rate were still empirical. Congestion status of some routers that could not receive packets from the source were also studied as a part of global-knowledge. They did not consider the distribution of full buffers over the network and quantify the impact of a source node's injection rate on congested routers.

State-of-the-art global-knowledge based source-throttling algorithms for NoC attempt to evaluate network congestion status accurately. Many researchers believe that the more accurately network congestion status is achieved and evaluated, the higher probability of efficient throttling effect can be achieved. Based on evaluated results, congestion status is used to compare with a static or dynamic threshold [13]. If the value of congestion status is smaller than the threshold, the packet injection may be increased to improve network throughput using load-adjusting mechanism. Otherwise, some source nodes' injection rates will be decreased.

However, how far and when the injection rate shall be increased or decreased are often empirically set [2], [18]. For example, self-tuned technology used an increment of 1 percent of all buffers for constant additive increments and a decrement of 4 percent of all buffers for constant additive decrements [13]. The reason why 1 percent and 4 percent are better than other percentages was not explained. If network scenario changes, it is difficult for such empirical values to be appropriately updated. In our opinion, such empirical values can be replaced by fine-grained evaluation of the impact of a source node's injection rate on congested routers and of network congestion status.

III. SOME COMMON MISTAKES

As mentioned in above section, evaluating network congestion is the first part of source-throttling method. In our opinion, the more precisely network status is evaluated, the better under-throttling and over-throttling problem can be avoided. To implement fine-grained evaluation of network congestion, we propose a concept of CCR (Congestion-Contribution-Rate) to quantify the impact of a source's throttling on all routers located in the communication range from the source to the destination. CCR depends on an assumption that Manhattan routing algorithms are used for mesh-connected NoC and a concept "RSD" presented in our previous work [19].

A. DEFINITIONS

Definition 1 (RSD): "RSD (Rectangle defined by Source node and Destination node)" is introduced in [19] to cover all the possible Manhattan paths from a source node to a destination node ('RSD Manhattan path' for abbreviation). Every pair of source/destination has its own RSD in a mesh network. All the routers included in a RSD are called "RSD routers." Every RSD Manhattan path is made up of multiple RSD routers. A RSD example can be seen in Fig. 1.

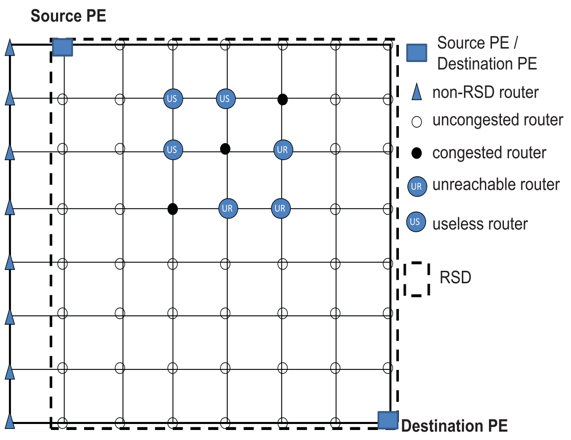

FIGURE 1. An example of useless routers and unreachable routers in a source-throttling scenario.

Definition 2 (Congested Router and Non-Congested Router): If each buffer in an input-port of a router is busy, the router is congested for its upstream router which is directly connected to this input-port. Otherwise, it is a non-congested router.

Definition 3 (Congestion-Around Manhattan Paths and Congested Manhattan Paths): For a source-destination pair in a RSD, there are many Manhattan paths from the source PE to the destination PE. These Manhattan paths on which there aren't any congested routers are called "congestion-around Manhattan paths." Manhattan paths that consist of some congested routers are defined as "congested Manhattan paths."

B. ANALYZING EFFECTIVENESS OF THROTTLING

To quantify the impact of a source's throttling on all routers located in a RSD, it is necessary to analyze the effectiveness of throttling action. It is assumed that adaptive Manhattan routing algorithms are used and packets can be routed around congested routers as far as possible. Given that an adaptive Manhattan routing scheme is used in every RSD, there are two sub-cases: (1) there does not exist any congestion-around Manhattan path in a RSD, and (2) there exist both some congestion-around Manhattan paths and congested Manhattan paths in a RSD. In the first sub-case, all new packets injected by the source PE are received by many congested routers on some congested Manhattan paths. Such congested routers will then affect their upstream routers, i.e., back-pressure [20]. Because of tree saturation, there is a spatial spread procedure in congestion [7], [8]. At this time, any packets injected will drastically deteriorate network congestion in a RSD. 100% throttling rate has to be used by this source PE to avoid such tree saturation.

In the second sub-case, it is assumed that a few of new packets injected can route around congested routers as far as possible by adaptive routing algorithms. When new packets injected are routed along congestion-around Manhattan paths, all intermediate routers on such paths are not easily congested. Whether throttling action is taken or not does not exert a crucial impact on these paths' congestion. In other words, throttling action exerts little impact on such intermediate routers' congestion status. However, when new packets injected are routed along congested Manhattan paths, back-pressure may come into being. At this time, throttling action can help intermediate routers at these paths decrease their load and avoid incoming back-pressure. That is to say, throttling action exerts impact on such intermediate routers' congestion status. Thus, it can be seen that throttling action from the source has various impacts on different routers' congestion status in the same RSD. To quantify such impact, a concept "Congestion-Contribution-Rate" is presented as follows.

Definition 4 [CCR(Congestion-Contribution-Rate and Anchor Router)]: "Congestion-Contribution-Rate (CCR for abbreviation)" is such a probability of a source's injection rate occupied by a RSD router under ideal conditions. It is used to help analyze the impact probability of a source's injection rate on all of its RSD routers under ideal conditions. For example, if a RSD router is not expected to receive packets from the source, its CCR value will be set to zero even if the router has received some packets from the source. $CCR_{ij}(0 \leq CCR_{ij} \leq 1)$ represents the probability occupied by $router_{ij}$ in a RSD under ideal conditions. Obviously, the more CCR_{ij} is, the more packets $router_{ij}$ is expected to receive from the source. Consequently, a source's throttling rate will exerts more influence on the congestion status of $router_{ij}$.

In this paper, CCR_{ij} is used to quantify the impact of a source's throttling rate on $router_{ij}$. How to figure out every RSD router's CCR value is introduced in Section 3.3. If a router locates in multiple RSDs, it has different CCR values from different sources.

For a given source, every router whose CCR value is more than zero is affected by the source's packet injection. That is to say, the source node's injecting action does exert its impact on congestion status of such routers. To achieve better throttling effect, the source's throttling behavior had better aim to routers with high CCR values. If such routers are going to be congested, the source's throttling behavior can quickly alleviate their congestion degree. On the other hand, the source's throttling behavior doesn't work well for those routers with low CCR values even if they are also to be congested. Under extreme cases, the congestion status for routers with low CCR values may still be congested when no packets are injected by the source. The reason is that such routers are congested by packets from other sources.

Different RSD routers may have different CCR values and various congestion status. There are three kinds of RSD routers:

(I) The first kind of routers are those routers that are not congested. Because they are not congested, no throttling has to be taken for them no matter how their CCR values are higher or lower.

(II) The second kind of routers are those that are congested and have higher CCR values. Such routers have received most of packets from the source. If the source throttles its injection rate much, these routers will receive few packets from the source. These routers' congestion degree will be alleviated higher. The larger such a router's CCR value is, the higher efficiency the source's throttling for this router is. A router with the maximum CCR value is called an "anchor router." The basic idea of our proposed source-throttling method in Section 4 is that a source's throttling is taken according to the congestion status of anchor routers. Once they are congested, throttling should be taken.

(III) The third kind of routers are those routers that are congested and have lower CCR values. Such routers have received fewer packets from the current source compared with the second kind of routers. Such routers are still congested because they have received many packets from other sources. It is obvious that the current source's throttling cannot efficiently alleviate such routers' congestion status. Such routers' congestion status will not be the standard of whether the current source's throttling should be taken or not.

Note that an anchor router for a RSD may not be an anchor router for other RSDs. Every source throttles its injection rate only according to congestion status of its own anchor routers. So, to achieve high efficiency of throttling for a source it is necessary to identify where anchor routers of the source are. The following subsection introduces how to identify anchor routers for a source.

C. METHOD OF FIGURING OUT CCR VALUE

It is assumed that fully-adaptive Manhattan routing algorithms based on global-congestion knowledge are used. It is also assumed that if all buffers in an input-port of a router are busy, i.e., congested, the router's upstream router which is directly connected with this input-port cannot forward packets to the router any more. For a congested router, to alleviate its congestion status, it is not expected to receive packets from the source any more. On the other hand, when fully-adaptive routing algorithms are used, it is of high probability for a congested router to be routed around. The congested router will receive few packets from the source. Thus, CCR value of a congested router is set to zero even if it may receive some packets from the source in practice. Setting CCR value of a congested router as zero is only used to analyze the impact of a source's injection action on congestion status of every RSD router.

Furthermore, if a router is not congested but all of its downstream routers are congested, such a router is not expected to receive new packets from the source, too. The reason is that any new packets received by this router has to be forwarded one or more of its downstream congested routers. As mentioned in above section, any congested routers including its downstream routers are not expected to receive packets. Thus, such a router's CCR value is also set to zero although it is not congested. On the other hand, if all of a router's upstream routers are congested and cannot receive packets from the source, the router correspondingly cannot receive packets from the source no matter how the router is congested or not. CCR value of such a router will also be set to zero. To figure out each RSD router's CCR value, it is necessary to label these two kinds of routers whose CCR values are zero even if they are not congested. Definition and method of labeling these two kinds of routers are introduced in the following subsubsection.

1) DEFINITIONS

Definition 5 (Useless Router & Unreachable Router): These two concepts were first presented by Wang for fault-tolerant routing with Manhattan routing schemes [21] and then used by others such as Jiang *et al.* [22] and Zhao and Xue [19], [23]. For useless routers in fault-tolerant scenario, they could not forward packets received to a destination node along any fault-tolerant Manhattan paths because of the existence of faulty nodes located at some specific positions. For unreachable routers in fault-tolerant scenario, they could not receive any packets from a source node along any fault-tolerant Manhattan paths because of the existence of faulty nodes located at some specific positions. Neither an unreachable router nor a useless router can be an intermediate router on any fault-tolerant Manhattan paths.

Obviously, useless or unreachable routers under faulty-tolerant scenario are similar to two kinds of routers as mentioned in the second paragraph of Section 3.3. The reason is that all of them cannot (or not expected to) receive packets from the source even if there are not faulty (or congested) under fault-tolerant (or congestion-avoidant) Manhattan path constraint. So if congested routers are looked as faulty routers, every un-congested RSD router whose CCR value is set to zero can also be regarded as a useless router or an unreachable router. Fig. 1 shows an example of useless routers and unreachable routers used for source-throttling scenario.

For source-throttling scenario using fully-adaptive Manhattan routing scheme, useless routers are not expected to receive packets from the source PE. Unreachable routers are not expected to receive packets from the source. Under ideal conditions, packets injection of a source will not affect congestion status of every useless RSD router, congested RSD router and unreachable RSD router.

Definition 6 (Normal Router): A "normal router" is a router in a RSD that is not congested, useless or unreachable. CCR value of a normal router is more than zero because it can receive packets from the source and then forward them.

There are two typical methods of labeling all useless routers and unreachable routers in a RSD. The first one is proposed in [21] whose time-complexity is much more than the second one called "path-counter method" in [23], [24]. In this paper, we use the second one because of its low time-complexity. For readers' convenience, description about "path-counter method" in [23], [24] are referred in Section 3.3.2.

2) LABELING USELESS ROUTERS AND UNREACHABLE ROUTERS BY PATH-COUNTER METHOD

Definition 7 (Path-Counter): As mentioned in our previous work [23], [24], a concept "path-counter" represents the number of all Manhattan paths which route round all faulty nodes in a RSD. If congested nodes act as faulty nodes, i.e., congested nodes don't store/foward new packets any more, the concept "path-counter" can also represent the number of all Manhattan paths which route round all congested nodes in a RSD, i.e., congestion-around Manhattan paths. For a source in a RSD, when the path-counter value is zero, it means that there are no any congestion-around Manhattan paths.

Definition 8 (Positive Path-Counter and Negative Path-Counter): Based on the concept "path-counter", "Positive Path-Counter" and "Negative Path-Counter" are proposed to label useless router and unreachable router in our previous work [24]. "Positive Path Counter (P-PC for abbreviation)" is defined as the total number of fault-tolerant (or congestion-around in source-throttling scenario) Manhattan paths from (X_s, Y_s) to the current router (i,j) in a RSD. If (X_s, Y_s) is not a faulty (or congested in source-throttling scenario) router, its P-PC is initially set to 1. It indicates that the total number of fault-tolerant (or congestion-around in source-throttling scenario) Manhattan paths from (X_s, Y_s) to itself is 1. If (X_s, Y_s) is a faulty (or congested) router, its P-PC is set to 0. It means that there is no fault-tolerant (or congestion-around in source-throttling scenario) Manhattan path from (X_s, Y_s) to itself.

"Negative Path Counter (N-PC for abbreviation)" represents the total number of fault-tolerant (or congestion-around in source-throttling scenario) Manhattan paths from the current router (i, j) to (X_d, Y_d) in a RSD. If (X_d, Y_d) is not a faulty (or congested in source-throttling scenario) router, its N-PC is set to 1. It means that the total number of fault-tolerant (or congestion-around in source-throttling scenario) Manhattan paths from (X_d, Y_d) to itself is 1. If (X_d, Y_d) is a faulty (or congested in source-throttling scenario) one, its N-PC is set to 0. It means that there are no fault-tolerant (or congestion-around in source-throttling scenario) Manhattan paths from (X_d, Y_d) to itself.

Path-counter proposed in our previous work [24] can label useless routers and unreachable routers well by figuring out every RSD router's P-PC and N-PC value. It is assumed that (X_d, Y_d) is in the X+ and Y+ direction of (X_s, Y_s). P-PC of router (i, j) is represented by "$C(i, j)$" in Eq. (1) [24]. N-PC of router (i, j) is represented by "$C'(i, j)$" in Eq. (2) [24]. If (X_d, Y_d) is in other directions of (X_s, Y_s) and router (i, j) is not a faulty (or congested) node, its P-PC or N-PC can be formulated in a similar way.

$$C(i,j) = \begin{cases} 0, & \textit{if } (i,j) \textit{ is congested}; \\ 1, & \textit{if } (i,j) \textit{ is the uncongested src.}; \\ C(i-1,j), & \textit{if } (i-1,j) \textit{ exists}; \\ C(i,j-1), & \textit{if } (i,j-1) \textit{ exists}; \\ C(i-1,j)+C(i,j-1), & \textit{if } (i-1,j)\&(i,j-1) \textit{ exist}; \end{cases} \tag{1}$$

$$C'(i,j) = \begin{cases} 0, & \textit{if } (i,j) \textit{ is congested}; \\ 1, & \textit{if } (i,j) \textit{ is theuncongested dst.}; \\ C'(i+1,j), & \textit{if } (i+1,j) \textit{ exists}; \\ C'(i,j+1), & \textit{if } (i,j+1) \textit{ exists}; \\ C'(i+1,j)+C'(i,j+1), & \textit{if } (i+1,j)\&(i,j+1) \textit{ exist}; \end{cases} \tag{2}$$

Based on Eq. (1-2), a path-counter method figuring out every RSD router's P-PC and N-PC can be easily deduced. The details of labeling useless routers and unreachable routers can be seen in our previous work [24]. Time complexity of path-counter method is $O(M * N)$ (M*N is the scale of 2D mesh network). According to path-counter method, N-PC value of every useless router is always equal to 0. If a router's N-PC is more than 0, it will not be a useless one. P-PC value of every unreachable router is always equal to 0. If a node's P-PC is more than 0, it will not be a unreachable one. Thus, all useless routers and unreachable routers in a RSD can be labeled.

3) FIGURING OUT EVERY RSD ROUTER'S CCR VALUE

It is assumed that every normal router forwards packets received to all of its normal downstream neighbor routers along Manhattan paths with the same probability. If a downstream neighbor router is not normal, i.e., it is congested, useless or unreachable, its' probability of receiving packets is zero. When an adaptive minimal routing algorithm is used, all normal RSD downstream neighbor routers will receive packets from its upstream routers with the same probability. After labeling all useless routers and unreachable routers in a RSD using path-counter method, every RSD router's CCR value can be represented by Eq. (3). As mentioned in Section 3.3.2, if a router is congested, useless or unreachable, its CCR value is zero. If a router is normal, its CCR value depends on its upstream neighbor routers' CCR values and the numbers of these upstream neighbors' other downstream neighbors. The bigger its upstream neighbor routers' CCR values are, the bigger its CCR value is. Additionally, the bigger the numbers of its upstream neighbors' other downstream neighbors are, the smaller its CCR value is. For a 2-D RSD, every router has at most two downstream neighbor routers, and every router has at most two upstream neighbor routers. $CCR(i, j)$ represents CCR value of the source on $router(i, j)$. The source PE sends 1 unit packets to the destination PE. The detail of figuring out a router's CCR value can be seen in Eq. (3).

$$CCR(i,j) = \begin{cases} 0, & \textit{if } (i,j) \textit{ is congested/useless/unreachable}; \\ 1, & \textit{if } (i,j) \textit{ is a normal router directly connected with the source}; \\ CCR(i,j-1), & \textit{if } (i,j) \textit{ is } (i,j-1)'s \textit{ unique normal downstream neighbor, and it has no other upstream neighbor routers}; \\ CCR(i,j-1)/2, & \textit{if } (i,j) \textit{ is one of } (i,j-1)'s \textit{ two normal downstream neighbor routers, it has no other upstream neighbor routers}; \\ CCR(i-1,j), & \textit{if } (i,j) \textit{ is } (i-1,j)'s \textit{ unique normal downstream neighbor, and it has no other upstream neighbor routers}; \\ CCR(i-1,j)/2, & \textit{if } (i,j) \textit{ is one of } (i-1,j)'s \textit{ two normal downstream neighbor routers, and it has no other upstream neighbor routers}; \\ CCR(i,j-1)+CCR(i-1,j), & \textit{if } (i,j) \textit{ is the unique normal downstream neighbor of } (i,j-1) \textit{ and that of } (i-1,j); \\ CCR(i,j-1)+CCR(i-1,j)/2, & \textit{if } (i,j) \textit{ is the unique normal downstream neighbor of } (i,j-1), \textit{ and it is one of } (i-1,j)'s \textit{ two normal downstream neighbor routers}; \\ CCR(i,j-1)/2+CCR(i-1,j), & \textit{if } (i,j) \textit{ is the unique normal downstream neighbor of } (i-1,j), \textit{ and it is one of } (i,j-1)'s \textit{ two normal downstream neighbor routers}; \\ CCR(i,j-1)/2+CCR(i-1,j)/2, & \textit{if } (i,j) \textit{ is one of } (i,j-1)'s \textit{ two normal downstream neighbor routers, and is also one of } (i-1,j)'s \textit{ two normal downstream neighbor routers} \end{cases} \tag{3}$$

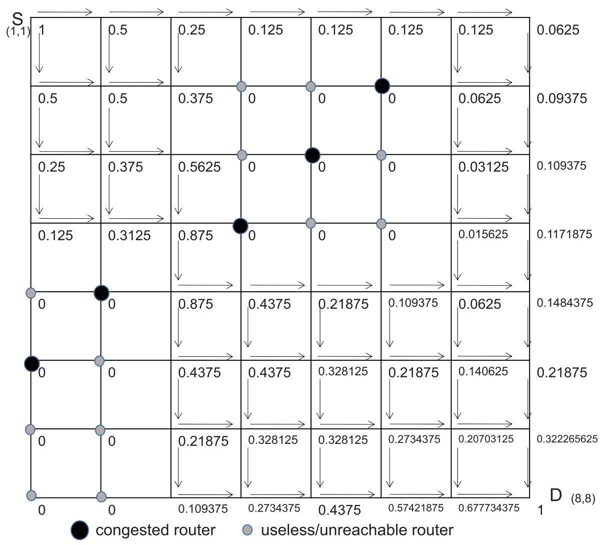

FIGURE 2. An example of figuring out every RSD router's CCR value.

Fig. 2 shows an example of figuring out every RSD router's CCR value in a 8*8 RSD. The coordinate of the source PE and its directly-connected router is (1,1). The coordinate of the destination PE and its directly-connected router is (8,8). It is assumed that the source "S" has injected 1 unit packets into the RSD. No packets are dropped and the destination receives all the 1 unit packets. Under ideal conditions, every router that is congested, useless or unreachable doesn't receive any packets from the source. Their CCR values are all set to zeros. The probability of packets received by every normal router is more than zero.

The source router (1,1) has two normal downstream neighbor routers, i.e, router (1,2) and (2,1). They share equally the 1 unit packets from the source. Each of them receives 0.5 unit packets so that their CCR values are both 0.5. Then router (1,2) distributes equally its packets to its two normal downstream neighbor router (1,3) and (2,2). Router (1,3) has no other upstream router, so its CCR value is 0.25. Router (2,1) forwards equally its packets to its two normal downstream neighbor router (2,2) and (3,1). Router (2,2) receives 0.25 unit packets from router (1,2) and 0.25 unit packets from router (2,1) individually. So its CCR value is 0.5. Other routers' CCR values can be deduced in the same way. It can be seen that routers with the highest CCR values is router (3,4) and (3,5) except the source and the destination. Their CCR values are both 0.875. Both of them are looked as anchor routers. When the source PE begins to throttle its injection rate, packets received by these two anchor routers will decrease much more drastically than that by other normal routers.

Based on Fig. 2, it can be deduced that CCR value of a normal router depends on both its own position and all congested routers' positions. Basically, congested routers' quantity and positions can both affect the congestion status of a network. At the same time, anchor routers change with the number and location of all congested RSD routers. Only quantity of all congested routers which is proposed in some related work such as [17] is not sufficient to evaluate congestion status of a network accurately.

By figuring out every RSD router's CCR value and identifying anchor routers, it is possible to evaluate network congestion more accurately. Then, when and how far a source's throttling is taken can be accurately figured out. All of these steps make up of our proposed fine-grained source-throttling method. Obviously, time complexity of figuring out routers' CCR values is also $O(M * N)$ (M*N is the scale of 2D mesh network).

IV. A FINE-GRAINED SOURCE-THROTTLING METHOD

Basic idea of our fine-grained source-throttling method is that the source's throttling coefficient dynamically changes with anchor routers' congestion status. If the number of busy input-buffers in an anchor router is more than a given threshold, the throttling coefficient should be adjusted.

A. A FINE-GRAINED SOURCE-THROTTLING METHOD

For a RSD communication range, steps of proposed fine-grained source-throttling method are as follows:

Step 1. Initialize a threshold TH, and set the source's initial injection rate IIR;
Step 2. Collect the position of every congested RSD router;
Step 3. Label every useless or unreachable RSD router using path-counter method;
Step 4. Figure out every normal router's CCR value;
Step 5. Find out anchor routers by comparing all normal RSD routers' CCR values;
Step 6. Monitor the congestion status information of every anchor router;
Step 7. Use Eq. (4) to figure out the source's throttling ratio α;
Step 8. Figure out the source's current injection rate CIR using Eq. (5) and begin to throttle the source's injection rate;
Step 9. Go to Step 2 in order to start a new round of throttling.

TH in Step 1 is used to decide whether throttling action shall be taken or not. It is represented by the busy buffer number of a router's single input-port. It is less than the total buffer number of a router's single input-port. Step 2 and Step 3 can identify every congested, useless or unreachable RSD router. Time complexity of these two steps are $O(M*N)$ (M*N is the scale of 2D mesh network) which can be seen in our previous work [24]. Such routers' CCR values are all zeros. The detail of Step 4 whose time complexity is also $O(M * N)$ can be seen in Section 3.3.3. The reason that Step 5 is presented can be seen in Section 3.2. Time complexity of step 5 is also $O(M * N)$. Step 6 an 7 aims at checking whether there exists an anchor router whose *anch* is more than TH or not and then figuring out a throttling ratio. Eq. (4) are introduced in Section 4.2. These two steps show that congestion status of anchor routers are used to evaluate congestion status of the whole RSD network. It is different from several previous work in which congestion status of all routers are used to do it. Because the number of anchor routers is less than the scale of 2D mesh, time complexity of Step 6 is equal to or less than $O(M * N)$. Step 8 and Eq. (5) are also introduced in

Section 4.2. Time complexity of Step 7 and Step 8 is constant. In Step 9, the next round of throttling will immediately start after Step 8 is finished. The method can be implemented by software on the source PE node. Every source PE node runs such fine-grained source-throttling method to figure out its own injection rate. Several parameters of this method such as *TH* are interpreted in Section 4.2. Time complexity of proposed method is only $O(M * N)$ (M*N is the scale of 2D mesh network). The period of proposed method is the sum of latency from Step 2 to Step 9. Note that Step 2 and 6 can collect RSD routers's congestion status by using state-of-the-art side-band network shown in related work [3], [20]. Thus, it will not consume current NoC network bandwidth.

B. FIGURING OUT A THROTTLING RATIO AND CURRENT INJECTION RATE

It is assumed that each input-port of a router has the same number of buffers. As mentioned in Section 3.2.3, the source's injection rate exerts more impact on those RSD routers with higher CCR values. Anchor routers have the highest CCR values over a RSD. Eq. (4) represents the relationship between a source's throttling ratio, i.e., α, and the congestion status of an anchor router, i.e., *anch*. *anch* is the maximum number of busy buffers among all anchor routers' input-ports that can receive packets from the source. *anch*'s upper bound is the total buffer number of a router's single input-port. When *anch* is more than *TH*, the source's injection rate should be decreased or throttled.

As shown in Eq. (4), when *anch* is more than *TH*, it means that an anchor router is going to be congested. Throttling has to be taken to avoid incoming congestion for the anchor router. Otherwise, no throttling has to be taken. *TH* is a precaution mechanism of preventing anchor routers being congested. It is empirically set as a value close to the total buffer number of a router's single input-port.

$$\alpha = \begin{cases} 1 - TH/anch, & if\ anch > TH; \\ 0, & if\ anch \leq TH; \end{cases} \tag{4}$$

After achieving α, the source's current injection rate can be figure out by Eq. (5).

$$CIR = \begin{cases} 0, & if\ the\ source\ router\ is\ congested; \\ IIR * (1 - \alpha), & Otherwise; \end{cases} \tag{5}$$

A source's current injection rate is marked "*CIR*." A source's initial injection rate is marked "*IIR*." As mentioned in Section 3.3, when a router's downstream neighbor router are congested or busy, it should not forward packets to its congested neighbor again in order to prevent the spread of tree saturation. Correspondingly, when a source's directly-connected router is congested, the source PE should not inject any new packets into this router although its injection rate has already been set to a value. Thus, the source's real injection rate is 0 as shown in the first branch of Eq. (5). When a source's directly-connected router is not congested, the source's real injection rate depends on *IIR* and α as shown in the second branch of Eq. (5). Once a source's *IIR* is set, it will not changes at all. The source's current injection rate will correspondingly change with α.

According to Eq. (4-5), RSD congestion can be adjusted as a critical status that *anch* is equal to *TH*. Any further increment of current injection rate will increase the probability of making anchor routers congested. On the other hand, α can also prevent further decrement of source's current injection rate. Throughput in this RSD can be maintained as much as possible. Thus, throttling based on α can avoid over-throttling and under-throttling simultaneously.

Note that if the initial injection rate is very low, current injection rate will be very low, too. Consequently, RSD network throughput is hard to be improved. On the other hand, if *IIR* is set as a high value, the source's directly-connected router is easily congested. At this time, no new packets will be injected into this router and the *CIR* is really 0 according to Eq. (5). That is to say, high *IIR* will not lead to high *CIR*. Thus, *IIR* should be set as a high value in order to maintain high RSD network throughput.

In a word, Eq. (4-5) are used to prevent anchor routers becoming congested in advance. They can be figured out by software running on PE nodes as presented in our proposed method.

C. EXPERIMENT

We simulate three kinds of global-knowledge based throttling algorithms, including proposed method, INC algorithm [2] and self-tuned algorithm [3] in a 8*8 mesh network. INC algorithm is a typical local-knowledge based source-throttling method [2]. It counts a subset (free and useful) of virtual channel buffers to decide whether a source's injection rate should be throttled or not. Self-tuned algorithm is a typical global-knowledge based source-throttling method [13]. It used the number of full buffers over the whole network as the global congestion information. It has better network performance than the INC algorithm [2] does. These two source-throttling algorithms don't evaluate how much a throttling action affects congested routers. Their throttling parameters are almost all empirically set. So we compare these two source-throttling algorithms with ours. To highlight the throttling result, we also simulate the scenario that no throttling action is taken.

In the simulation, Wormhole routing scheme and fully-adaptive routing algorithm are used. Every packet is made up of 5 flits. Every router has 5 input queues. Every input queue uses 2 virtual channels to avoid deadlock. Every virtual channel can store 10 flits. Every input queue can store 20 flits at most. The threshold *TH* is set to 16 flits for an input queue. When the number of received flits by a router's input-port queue is more than *TH*, this input-port queue will not receive flits any more. Random, transpose and hotspot traffic patterns are set. Note that the latency and throughput of un-throttled condition are convergent here. The reason is that every router will not receive flits any

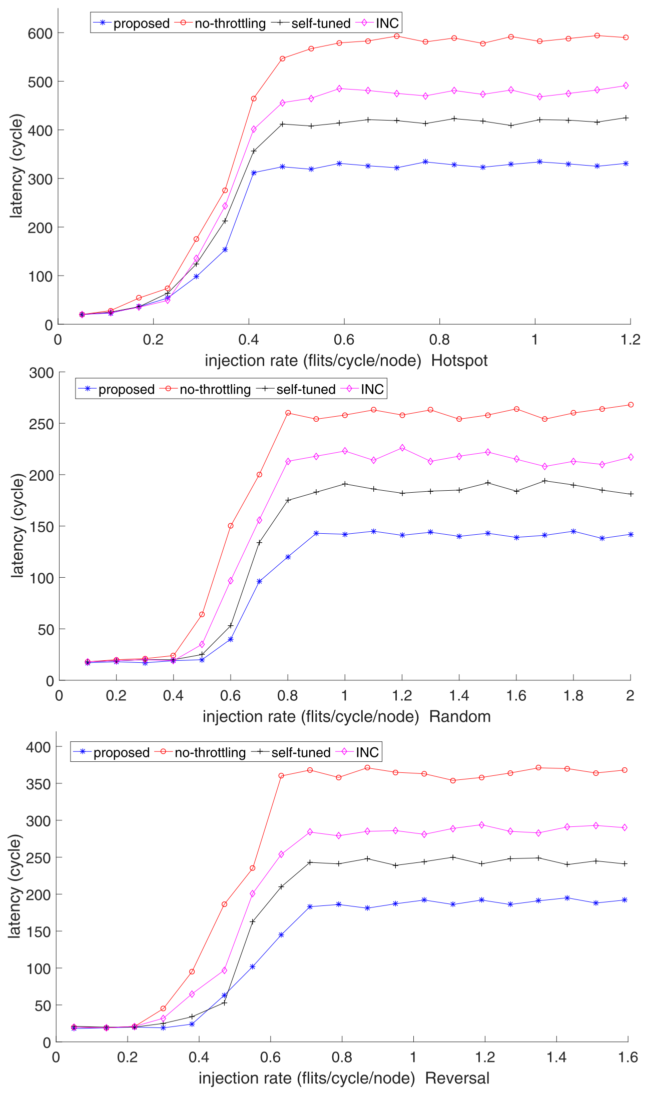

FIGURE 3. Latency under hotspot,random and reversal traffic patterns in an 8*8 2D Mesh.

more when its input-ports queues are all full as mentioned in Section 3.3.

Fig. 3 shows the latency under hotspot, random and reversal traffic patterns in an 8*8 2D mesh network. Under the hotspot traffic pattern, the stable latency of our proposed algorithm is about 21.2% less than that of self-tuned algorithm, 30.8% less than that of INC algorithm, and 43.9% less than that of un-throttled condition, respectively. Under the random traffic pattern, the stable latency of our proposed algorithm is about 23.3% less than that of self-tuned algorithm, 34.6% less than that of INC algorithm, and 46.6% less than that of un-throttled condition, respectively. Under the reversal traffic pattern, the stable latency of our proposed algorithm is about 20.3% less than that of self-tuned algorithm, 33.8% less than that of INC algorithm, and 48.4% less than that of un-throttled condition, respectively. Our proposed algorithm achieves higher performance than other three algorithms do.

Fig. 4 shows the throughput under hotspot, random and reversal traffic patterns in an 8*8 2D mesh network. It can be seen that four throughput-load curves have almost the same top throughput points under the same traffic pattern, i.e.,hotspot, random or reversal. If the load continues to increase, their throughput will drop at different levels and then become slowly stable with different source-throttling methods. Under the hotspot traffic pattern, the stable network throughput of our proposed algorithm is about 15.5% higher than that of self-tuned algorithm, 28.7% higher than that of INC algorithm, and 91.5% higher than that of un-throttled condition, respectively. Under the random traffic pattern, the stable throughput of our proposed algorithm is about 17.6% higher than that of self-tuned algorithm, 25.4% higher than that of INC algorithm, and 102.4% higher than that of un-throttled condition, respectively. Under the reversal traffic pattern, the stable throughput of our proposed algorithm is about 14.1% higher than that of self-tuned algorithm, 25.2% higher than that of INC algorithm, and 92.8% higher than that of un-throttled result, respectively. In a word, throughput with our proposed method owns the lowest drop level compared with other three throttling methods. Our proposed method can usually keep network throughput very close to the top point under different traffic patterns.

Fig. 5 shows the throughput varies with the simulation time under random traffic pattern in an 8*8 2D mesh network. Compared with other three throttling results, our proposed algorithm has the lowest fluctuation of throughput. It means that it can avoid over-throttling and under-throttling better.

Fig. 6 shows the latency varies with the simulation time under random traffic pattern in an 8*8 2D mesh network. Compared with other three throttling results, our proposed algorithm has the lowest fluctuation of latency. It also means that it can avoid over-throttling and under-throttling more accurately.

Our proposed algorithm can achieve better throttling effect than typical INC and self-tuned source-throttling algorithms.

D. ANALYSIS

1) LESS FLUCTUATION OF THROUGHPUT AND LATENCY

There are two key features in our proposed method. The first one is that it can quantify the effectiveness of throttling action on RSD routers. As mentioned in Section 3.2, some RSD routers are affected more by a source's throttling action while others are not. CCR is used to accurately quantify such effectiveness difference. The second one is that it does not use the congestion status of the whole network but uses a concept "anchor router" to evaluate network congestion. Consequent throttling aims not at the whole network but at anchor routers in a RSD. In some previous work such as [2], congestion status of some routers which are affected little by the source's injection rate are also counted into congestion status of the whole network. Consequently, part of throttling which is for congestion status of such routers will not work well to avoid over-throttling and under-throttling. Compared with these work, throttling aiming at anchor routers will work better. That is the reason why our proposed method shows less

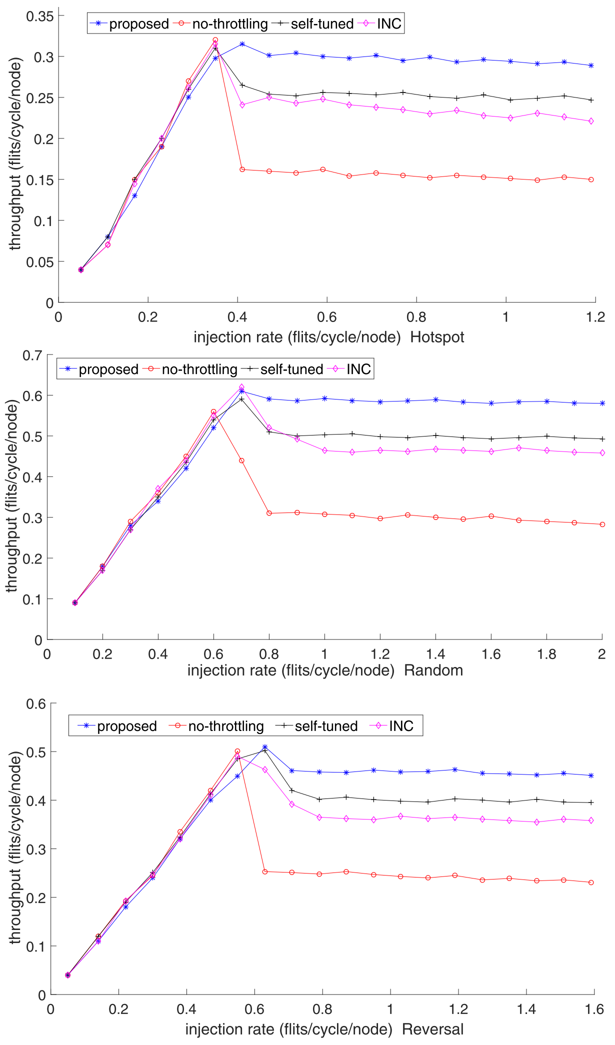

FIGURE 4. Throughput under hotspot,random and reversal traffic patterns in an 8*8 2D Mesh.

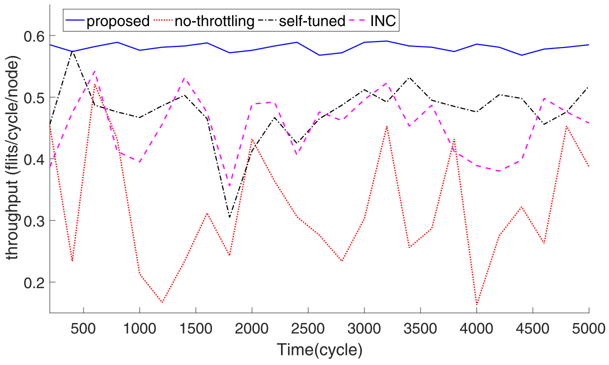

FIGURE 5. Throughput vary with simulation time under random traffic pattern.

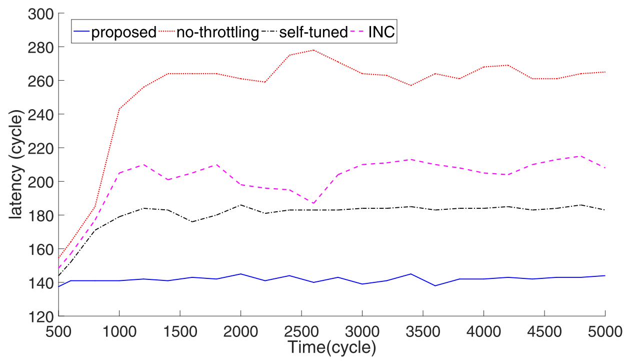

FIGURE 6. Latency vary with simulation time under random traffic pattern.

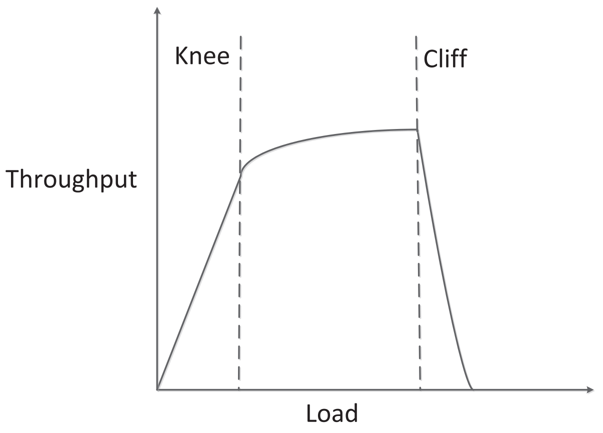

FIGURE 7. The relationship between network load and throughput [12].

fluctuation of throughput and latency than previous work as shown in Fig. 5 and Fig. 6.

2) HIGHER THROUGHPUT

As shown in Fig. 7 that is presented by Jain etc. in 1988, a well-worked network should keep appropriate load in order to provide high throughput [12]. When the load is less than the knee point, network throughput is very small. When the load is more than the cliff point, network throughput decreases quickly. Thus, higher throughput can be achieved only when the load is near to or among the range from the knee point to the cliff point.

Our proposed method prefers to figure out the current injection rate accurately. And it throttles the injection rate not according to the overall network congestion status but according to anchor routers' congestion status. An anchor router is the vane of the impact of source's injection action on RSD network's congestion status. Eq. (4) adjusts throttling rate and makes congestion status of an anchor router be exactly close to the threshold. Thus, over-throttling and under-throttling problems are simultaneously avoided. Its procedure can be intuitively expressed by Fig. 7, i.e., the load range from the knee point to cliff point can be approached by Eq. (4). That is the reason why our proposed method can achieve higher throughput than other two algorithms as shown in Fig. 4.

3) LESS LATENCY

Eq. (4) can avoid under-throttling, so less congestion will appear. Consequently, less latency, as shown in Fig. 3 and Fig. 6, can be achieved by our proposed method. On the other hand, throughput can be defined as the total number of packets received by all destination nodes per unit time. In other words, the more throughput is, the less average end-to-end latency for a packet is. Because our proposed method has higher throughput than other two algorithms, lower latency can be achieved by it, too.

4) TIME COMPLEXITY AND SCALABILITY

Our proposed method consists of 9 steps. Time complexities of Step 3, 4 and 5 are all linearly related to the scale of a RSD. Time complexities of step 1, 6, 7, 8 and 9 are all constants. It can be deduced that time complexity of our proposed method is linearly related to the scale of a RSD. It is easily implemented by software on the source PE.

Our proposed method is a partially centralized algorithm. A main argument against centralized routing mechanisms is their potential scalability problems [25]. Such measures as using clustered blocks of nodes(2*2, 3*3, 4*4 etc) instead of individual nodes [25] can be used to solve this problem. On the other hand, its scalability is better than global-knowledge based throttling algorithms such as self-tuned algorithm.

V. CONCLUSION

In this paper, we propose a fine-grained source-throttling method. Compared with INC algorithm and self-tuned source-throttling algorithm that use empirical values to throttle a source's injection rate, it can be of fine-grained calculating the throttling ratio to adjust the source's injection rate. So over-throttling and under-throttling problems can be avoided more efficiently and intelligently. Our two main contributions: 1, it uses a concept "CCR" to quantify the impact of throttling action on routers; 2, only anchor routers, i.e., the RSD routers that are most affected by the source's throttling action, are used to evaluate congestion status of the RSD network. Those RSD routers that are not as much affected are not taken into account at all. Thus, throttling parameters can be not empirically set but accurately figured out. It can solve over-throttling and under-throttling problems in a fine-grained manner. Simulation results also show that it has higher throughput, lower fluctuation and lower latency than INC and self-tuned source-throttling algorithms under different network scenarios. Additionally, time complexity of our proposed method is only $O(M * N)$ (M*N is the scale of 2D mesh network). In the future work, we will use our proposed method in a manner of AI technology about designing NoC architecture.

REFERENCES

[1] L. Benini and G. De Micheli, "Networks on chip: A new paradigm for systems on chip design," in *Proc. Design, Autom. Test Europe Conf. Exhib.*, Jun. 2003, pp. 418–419.

[2] E. Baydal, P. Lopez, and J. Duato, "A family of mechanisms for congestion control in wormhole networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, no. 9, pp. 772–784, Sep. 2005.

[3] M. Thottethodi, A. R. Lebeck, and S. S. Mukherjee, "Self-tuned congestion control for multiprocessor networks," in *Proc. HPCA 7th Int. Symp. High-Perform. Comput. Archit.*, Jan. 2001, pp. 107–118.

[4] J. Escudero-Sahuquillo, P. J. García, F. J. Quiles, J. Flich, and J. Duato, "FBICM: Efficient congestion management for high-performance networks using distributed deterministic routing," in *High Performance Computing-HiPC 2008*. Springer, 2008, pp. 503–517.

[5] P. Lotfi-Kamran, M. Daneshtalab, C. Lucas, and Z. Navabi, "BARP-a dynamic routing protocol for balanced distribution of traffic in NoCs," in *Proc. Design, Autom. Test Europe*, Mar. 2008, pp. 1408–1413.

[6] S. Pei, J. Yuan, and T. S. Y. Ji, "DATRA: A power-aware dynamic adaptive threshold routing algorithm for dragonfly network-on-chip topology," in *Proc. IEEE Int. Conf Parallel Distrib. Process. Appl.*, Oct. 2019, pp. 288–295.

[7] G. F. Pfister and V. A. Norton, "'Hot spot' contention and combining in multistage interconnection networks," *IEEE Trans. Comput.*, vol. C-34, no. 10, pp. 943–948, Oct. 1985.

[8] L. Tassiulas, "Adaptive back-pressure congestion control based on local information," *IEEE Trans. Autom. Control*, vol. 40, no. 2, pp. 236–250, Feb. 1995.

[9] E. Baydal and P. López, "A robust mechanism for congestion control: Inc," in *Euro-Par 2003 Parallel Process.* 2003, pp. 958–968.

[10] J. Duato, I. Johnson, J. Flich, F. Naven, P. Garcia, and T. Nachiondo, "A new scalable and cost-effective congestion management strategy for lossless multistage interconnection networks," in *Proc. 11th Int. Symp. High-Perform. Comput. Archit.*, Apr. 2005, pp. 108–119.

[11] E. G. Gran, M. Eimot, S.-A. Reinemo, T. Skeie, O. Lysne, L. P. Huse, and G. Shainer, "First experiences with congestion control in InfiniBand hardware," in *Proc. IEEE Int. Symp. Parallel Distrib. Process. (IPDPS)*, Apr. 2010, pp. 1–12.

[12] R. Jain and K. K. Ramakrishnan, "Congestion avoidance in computer networks with a connectionless network layer: Concepts, goals and methodology," Tech. Rep., 1988.

[13] M. Thottethodi, A. Lebeck, and S. Mukherjee, "Exploiting global knowledge to achieve self-tuned congestion control for k-ary n-cube networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 15, no. 3, pp. 257–272, Mar. 2004.

[14] P. Lopez, J. Martinez, and J. Duato, "DRIL: Dynamically reduced message injection limitation mechanism for wormhole networks," in *Proc. Int. Conf. Parallel Process.*, Nov. 2002, pp. 535–542.

[15] E. Baydal, P. Lopez, and J. Duato, "A simple and efficient mechanism to prevent saturation in wormhole networks," in *Proc. 14th Int. Parallel Distrib. Process. Symp. (IPDPS)*, Nov. 2002, pp. 617–622.

[16] A.-H. Smai and L.-E. Thorelli, "Global reactive congestion control in multicomputer networks," in *Proc. 5th Int. Conf. High Perform. Comput.*, Nov. 2002, pp. 179–186.

[17] G. P. Nychis, C. Fallin, T. Moscibroda, O. Mutlu, and S. Seshan, "On-chip networks from a networking perspective: Congestion and scalability in many-core interconnects," in *Proc. SIGCOMM Comput. Commun. Rev.*, Aug. 2012, vol. 42, no. 4, pp. 407–418. [Online]. Available: http://doi.acm.org/10.1145/2377677.2377757

[18] K. K.-W. Chang, R. Ausavarungnirun, C. Fallin, and O. Mutlu, "HAT: Heterogeneous adaptive throttling for on-chip networks," in *Proc. IEEE 24th Int. Symp. Comput. Archit. High Perform. Comput.*, Oct. 2012, pp. 9–18.

[19] H. Zhao and Y. Xue, "RSD fault block model for highly efficient fault-tolerant manhattan routing algorithms in 2D mesh," *Comput. J.*, vol. 59, no. 10, pp. 1511–1526, Oct. 2016.

[20] E. Kakoulli, V. Soteriou, and T. Theocharides, "Intelligent hotspot prediction for network-on-chip-based multicore systems," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 31, no. 3, pp. 418–431, Mar. 2012.

[21] D. Wang, "A rectilinear-monotone polygonal fault block model for fault-tolerant minimal routing in mesh," *IEEE Trans. Comput.*, vol. 52, no. 3, pp. 310–320, Mar. 2003.

[22] Z. Jiang, J. Wu, and D. Wang, "A new fault-information model for adaptive minimal routing in 3-D meshes," *IEEE Trans. Rel.*, vol. 57, no. 1, pp. 149–162, Mar. 2008.

[23] H. Zhao, N. Bagherzadeh, and J. Wu, "A general fault-tolerant minimal routing for mesh architectures," *IEEE Trans. Comput.*, vol. 66, no. 7, pp. 1240–1246, Jul. 2017.

[24] H. Zhao, Q. Wang, K. Xiong, and S. Pei, "A path-counter method for fault-tolerant minimal routing algorithms in 2D mesh," *J. Circuit Syst. Comput.*, vol. 27, no. 4, Apr. 2018, Art. no. 1850054.

[25] R. Manevich, I. Cidon, A. Kolodny, and I. Walter, "Centralized adaptive routing for NoCs," *IEEE Comput. Archit. Lett.*, vol. 9, no. 2, pp. 57–60, Feb. 2010.

HONGZHI ZHAO (Member, IEEE) received the B.S. and Ph.D. degrees from the Information Engineering School, University of Science and Technology, Beijing, in 2001 and 2007, respectively. He is currently an Associate Professor of computer engineering with the School of Computer and Information Technology, Beijing Jiaotong University. His research interests include computer architecture, computer networks, and VLSI design.

NADER BAGHERZADEH (Fellow, IEEE) received the Ph.D. degree from the University of Texas at Austin, in 1987.

He is currently a Professor of computer engineering with the Department of Electrical Engineering and Computer Science, University of California–Irvine, Irvine, where he served as the Chair, from 1998 to 2003. He has been involved in research and development in the areas of: computer architecture, reconfigurable computing, VLSI chip design, network-on-chip, 3D chips, sensor networks, computer graphics, memory, and embedded systems. He has published more than 250 articles in peer-reviewed journals and conferences. His former students have assumed key positions in software and computer systems design companies in the past twenty five years. He has been a PI or Co-PI on more than $10 million worth of research grants for developing next generation computer systems for applications in general purpose computing and digital signal processing and other related areas.

QIANG WANG received master's degree in 2017. His research interests include computer architecture, computer networks, and VLSI design.

YONGCHANG WANG is currently pursuing the Ph.D. degree with the School of Computer and Information Technology, Beijing Jiaotong University. His research interests include computer architecture, computer networks, and VLSI design.