PFT: A Congestion Avoidance Method based on Proactive Flow Throttling at Endpoints

Xingyun Qi[1], Mingche Lai[1], Dezun Dong[*], Yi Dai, Junsheng Chang, Jijun Cao
College of Computer, National University of Defense Technology, Changsha, China
{qi_xingyun, laimingche, dong, daiyi, changjunsheng, caojijun}@nudt.edu.cn

***Abstract*—Network congestion will result in severe performance degradation in a large-scale lossless interconnection network if no effective countermeasure is taken. An implementable, effective and cost-efficient design is crucial to practical network congestion control. Most existing reactive and proactive congestion control schemes still suffer from complicated congestion signals and parameter tuning, which significantly increases implementation complexity and the risk of incorrect settings that cause poor performance. In this work, we make a successful attempt to present a novel congestion control that minimizes the complexity and cost of protocol implementation. We explore proactive flow throttling (PFT) techniques at host network interface. PFT merely uses the local credit information at the endpoint node. Our design can quickly sense the network congestion and respond to it by proactive flow throttling with simple parameter configuration. This method has be successfully implemented and verified in one high-performance network interface chip at an extremely low cost. We use the state-of-the-art benchmark dedicated to network congestion testing to evaluate our design. The extensive results show that our method can quickly detect the occurrence of network congestion, and effectively perform congestion control in a real-time manner, significantly improving the application performance.**

***Index Terms*—high performance computing, interconnection, network congestion, credit based flow control, congestion control**

I. Introduction

Congestion management is an essential issue in high-performance computing (HPC) interconnection networks. Congestions in HPC systems can be classified into network congestion and endpoint congestion [7] [8]. Network congestion happens when the load on links exceeds their bandwidth. Congestion decreases network throughput and increases packet latency, and significantly impacts application performance and user experience, especially for latency-sensitive applications [30] [19]. Endpoint congestion occurs when endpoints are oversubscribed. Pure network congestion can be efficiently handled by adaptive routing, however, endpoint congestion has to be addressed by effective congestion control. HPC designers tend to build interconnection networks with full bisection bandwidth to meet the needs of communication-intensive applications [1] [2] [3] [6]. In this type of interconnection network, most persistent congestion occurs in edge routers of the network, forming the endpoint congestion.

The objective of practical congestion control in HPC network is to manage congestion within acceptable costs [38] [39] [26]. Prior congestion control mechanisms can be categorized as either reactive or proactive congestion control [27] [23] [5]. Reactive congestion control detects congestion and take actions after the congestion has emerged. Explicit congestion notification (ECN) is a representative reactive mechanism in the InfiniBand network [21]. However, ECN often needs at least one round trip time (RTT) to sense the congestion, and thus responds slowly to the network congestion. Proactive congestion control mechanisms avoid the congestion through reservation, and they only send data when there is no resource oversubscription in the network. In proactive congestion control schemes, the network is often modeled as a virtual big switch. Existing proactive congestion control mechanisms are effective in relieving conflicts through making reservation in the source node proactively. However, their performance can be limited by the inappropriate granularity of the reservation and the mismatched scheduling granularity between the source node and the destination node, which do not fit well with complicated and time-varying HPC network traffics where the large flows and small flow are mixed [13].

In this work, we aim to design a proactive congestion control scheme while minimizing the complexity and cost of protocol implementation. We exploit the basic idea behind proactive congestion control mechanisms, and view the network as one big virtual switch. We explore two important properties of HPC network, high-speed and lossless, which make that a single point of persistent congestion can quickly spread through the network, and invoke credit backpressure to the source hosts. Based on this observation, we present a proactive flow throttling (PFT) technique at host network interface chip (NIC). PFT in one NIC decides whether to perform congestion control and how to perform congestion control merely according to current credit information in the edging router that the NIC is connected with. We implement our PFT in one high-performance NIC, having a network port bandwidth of 100Gbps. We further utilize the NIC with PFT to build a testbed of interconnection network that consists of 120 nodes and are connected into a fat-tree topology. We take advantage of the state-of-the-art benchmark dedicated to HPC network congestion testing, GPCNet, to evaluate the effective of PFT in our testbed, and reports the extensive experiment

[1]These authors contributed equally to this work and should be considered co-first authors

*Corresponding author

978-3-903176-32-4 © 2021 IFIP

results.

The main contribution of this paper is a flow throttling method based on transmitting credits at source node in network. We can control the transmitting speed through adjust parameters in PFT according current congestion status. On a real implemented interconnection, we demonstrates that the PFT method can reduce the network congestion considerably. Moreover, the hardware overhead of PFT is very low. Only some counters, configuration registers is needed to realize the overall hardware.

The rest content of this article is as follows: Section II introduces related works. Section III presents the PFT congestion avoidance method based on end node credit in this article. Section IV evaluates and verifies the method through experiments. Section V gives relevant conclusions.

II. Related work

Generally speaking, network congestion control can be carried out at multiple levels of the network such as the transport layer, network layer, and link layer [16]. For Internet, as the bottom layer of the network may be implemented in different ways, the reliable transmission of data is generally implemented in the transport layer software such as TCP. Therefore, the congestion control is generally performed at the transport layer. A typical implementation is TCP congestion control mechanism [17] [18] [20], including TCP Vegas, TCP Reno, etc. The characteristics of which are slow start, congestion avoidance, fast retransmission, fast recovery, selective response, etc. These methods can improve the performance of network transmission and reduce the congestion possibility to a certain degree.

In the lossless networks oriented to high-performance computing and data center, link-level flow control is realized through PFC [12], credit flow control [9] and other mechanisms. Some networks even provide reliable transmission mechanism by text checking and retransmission [21] [22], to achieve lossless transmission. In this case, congestion control can be implemented at the network layer or link layer, such as QCN [24] [25], DCQCN [28] and ECN [29] [31] [33] commonly used in data centers. For InfiniBand [21] and Omni-Path [42], congestion control is also performed at the network layer. When a message is routed in the network, if it encounters a congested link, a special mark will be wrote into a predefined field on the message header to indicate the specific path and location of the network congestion. After that, the message continues to be transmitted forward. After reaching a certain network end node, this messages will be absorbed, and the associated congestion information will be processed. According to the current congestion status collected from arriving packets, the location of source nodes that cause network congestion is analyzed, and congestion notification messages will be sent to the congestion source nodes. When receiving the congestion notification messages, the source nodes adjust the sending rate to complete congestion control.

The basic idea of these switch-based congestion control is that the switch collects network congestion information, and recognizes the location of congestion. Then it will inform the source nodes to take corresponding measures [8] [4] [14] [15]. The advantage is that the source of congestion ports be found more accurately, and the congestion can be effectively controlled. However, all these methods need congestion information of various parts in the network to generate congestion notification messages. This will increase the complexity of router and end node design. And the transmission of backward notification messages also consumes network bandwidth, which may cause network congestion. At the same time, the congestion state of the network may spread quickly, so there will be a certain lag in the occurrence and notification of network congestion. Before the congestion control message reaches the source node, the congestion state of the network may have been propagated to the source node. To solve this problem, the congested source node does not need to wait for the congestion control messages generated in the network. It can proactively reduce or suspend the message sending rate according to the current local congestion status, so as to respond to network congestion as soon as possible.

Algorithm 1

```
1: if s = Normal then
2:     if c>t_e then
3:         s = Congested
4:     end if
5: else if s = Congested then
6:     if c<t_s then
7:         s = Normal
8:     end if
9: end if
```

In order to effectively use the information on end nodes for congestion control, researchers have proposed a variety of solutions. [34] [15] [10] [11] [32] proposed a method to analyze the RTT (Round Trip Time), receptions, and losses information on the end node, and designed a Congestion Control Plane (CCP) to perform Congestion management. [35] proposed a congestion control method named TIMELY based on the network RTT time information. It decides the level of network congestion by calculating the RTT time between the source and destination nodes, and takes certain measures at the source to control congestion. Ref. [36] has improved end node congestion control for small messages SMSRP and LHRP based on the analysis of SRP [37]. ExpressPass [40] used end-to-end credit transfer for bandwidth allocation and fine-grained packet scheduling, and ExpressPass++ [41] improved the sequence-based feedback control of ExpressPass.

The methods mentioned above all make good use of the information at the end nodes for congestion control. However, some methods require operating systems and software to participate in the scheduling of congestion control, and some require information exchange between the source and destination nodes that are far away from. As a result, those are unable to reflect the current network congestion in time, and results in ineffective congestion control.

If hardware can be used to perform simple and effective congestion control based on the local information on the end node, it will considerably decrease the design complexity and improve the control efficiency. At the same time, in some existing networks, routers may not support the function of checking or monitoring of congestion status and the routing of congestion control messages. In this case, the above methods cannot be used for congestion control. Therefore, we need a method of congestion control only performed at the network end node.

III. Endpoint credit based congestion control

A. Credit Based Flow Control

In the high speed interconnection network design, in order to avoid the end node data transmission speed being too fast, and the ingress buffer of the next stage router being overflow, credit based flow control is used between two adjacent nodes in the network. This mechanism is only a point-to-point flow control, which is performed at both ends of a transmission link. When initializing, the link receiver reserves a certain capacity of receiving buffer for incoming packets, and at the same time the transmitter allocates the same amount of transmission credits. Each time the transmitter sends a packet, one credit will be deducted from the existing credit. When the packet has entered the receivers buffer, the receiver will inform the transmitter every time a packet is removed from the buffer for processing. Upon receiving the credit release signal, the transmitter will add one credit to the existing credit. Before sending a packet, the transmitter must check whether there is enough credit firstly. If there is enough transmitting credit, it can send a message to the receiver. If there is no transmitting credit, it indicates that all the buffer space at the receiver has been occupied by unprocessed packets, and current packet cannot be sent out at this time.

Under this flow control mechanism, the credit used by the transmitter (c) corresponds to the size of the buffer that has been occupied in the input buffer of the receiver. When the network is not congested, the data in the input buffer at the receiver will be used and removed away quickly, and credit will be returned to the transmitter. In this way, c will remain at a low level. If the network is congested, the data in the input buffer of the receiver will be blocked, so few credits will be returned to the transmitter. This will result in a gradual increase of c. It can be seen that the used credit value c on the transmitter of the network reflects the degree of network congestion to a certain extent. The PFT congestion avoidance method proposed in this paper uses the used credit value on the end node to limit the rate of packets injected into the network.

In order to realize the network congestion control function, we need to solve the following two problems: (1) How to decide whether the network has encountered congestion or the congestion has been eliminated (that is, how to determine when to enter the flow throttling state and when to exit the flow throttling state); (2) How to perform congestion control when the network is congested (that is, how to control the message sending speed).

Algorithm 2

$cycle\_count = 0$
$congested\_count = 0$
if $s = Normal$ **then**
  if $cycle\_count = T_e$ **then**
    $cycle\_count = 0$
  else
    $cycle\_count = cycle\_count + 1$
  end if
  if $cycle\_count = T_e$ **then**
    $congestede\_count = 0$
  else if $c > T_e$ **then**
    $congested\_count = congested\_count + 1$
  end if
end if
if $congested\_count = T_e{}'$ **then**
  $s = Congested$
end if
if $s = Congeste$ **then**
  if $cycle\_count = T_s$ **then**
    $cycle\_count = 0$
  else
    $cycle\_count = cycle\_count + 1$
  end if
  if $cycle\_count = T_s$ **then**
    $congestede\_count = 0$
  else if $c < T_s$ **then**
    $congested\_count = congested\_count + 1$
  end if
end if
if $congested\_count = T_s{}'$ **then**
  $s = Normal$
end if

B. Conditions for Entering/Quitting Flow Throttling

First, we define that the transmitting node has two different congestion control states: $Normal$ state and $Congested$ state. If the node is in the $Normal$ state, it means that the current network is not congested. On the other way, if the node is in the $Congested$ state, it means that the current network has been detected to be congested, and the rate of packets injects into the network needs to be limited. The PFT congestion avoidance method provides two strategies to determine whether the current node should enter the $Normal$ or $Congested$ state: Fixed Threshold(FT) and Average Threshold(AT).

Suppose the current congestion control state of a node is s. Under the FT strategy, two thresholds are set: entry threshold t_e and exit threshold t_s ($t_e \geq t_s$). The node congestion state transition process under the FT strategy is shown in Algorithm 1.

Under the FT strategy, whether a certain port currently needs to perform flow throttling is determined based on the current credit used by that port. When the network is not congested,

the value of c is very small. With the emergence of network congestion, the packet transmission speed in the network slows down, and the value of c gradually increases. If c exceeds a certain threshold (t_e), it enters the $Congested$ state to slow down the inject rate of the source. After a period of time, when network congestion gradually eliminates, the value of c will gradually decreases. If c is less than a certain threshold (t_s), it is considered that the network congestion is eliminated, so the congested state is quitted, and the transmitting rate recovers to the normal rate.

This strategy only decides whether the network has been congested based on the currently used credit. In some circumstance, this method may be too simple. If a number of packets has been sent out without returning credit within a period of time, and this state has been continued, then it can be considered that the network was congested. That is, not only the value of current credit is used, but also the historical information is used. So that the decision will be more accurate. Based on this idea, we proposed the AT strategy.

Under the AT strategy, we give a period of time T_e and $T_e{}'$, T_s and $T_s{}'$, where $T_e > T_e{}'$, $T_s > T_s{}'$. Let $cycle\_count$ and $congested\_count$ be counters.

Under the AT strategy, whether a certain port needs to perform flow throttling is determined according to how long the currently used credit is higher than a preset threshold in a period of time. During this period of time (T_e), if there is a long period of time ($T_e{}'$) that the used credit is higher than the threshold, it is considered that the credit cannot returned in time. So we can consider that there may be network congestion, and flow throttling must be perform. After entering the speed limit state, if in a long period of time (T_s), the used credit is lower than a preset threshold ($T_s{}'$), it is considered that the credit can be returned in time. Then we consider that the network is not congested, and we can exit the rate limit state. The AT strategy not only uses the current credit value, but also counts historical credit information, which will be more accurate when used to identify the network congestion. However, this method increases hardware overhead and increases the complexity of design implementation slightly.

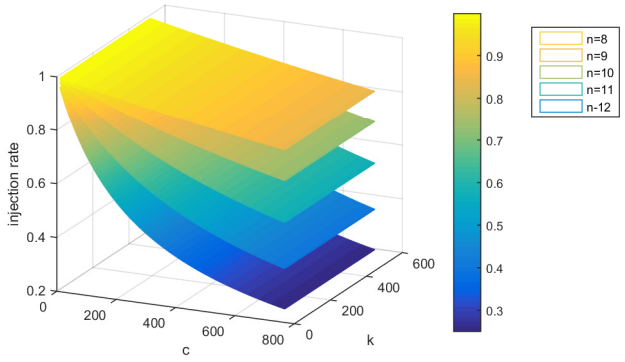
C. Flow Throttling

When the node is in the $Normal$ state, it sends the message at the maximum speed. Only when it is in the $Congested$ state will it control the sending speed and perform flow throttling. Controlling the sending speed of data is actually controlling the sending interval between two adjacent flits. The sending interval between flit should be determined according to the currently used credit value c. The larger the c, the larger the sending interval.

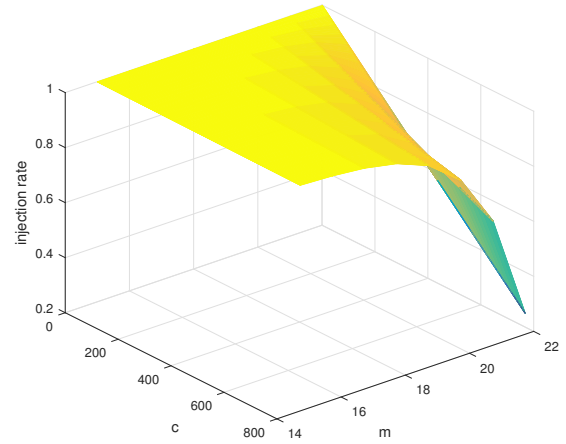
We give three parameter m, n, and k which are all integers. Define a function of c:

$$f(c) = 2^m/2^n c + k = 2^{m-n} c + k \tag{1}$$

In the $Congested$ state, there are two congestion control strategies: Fixed Interval(FI) and Random Interval(RI). Under the FI policy, let $f(c)$ be the sending interval between two adjacent flits. When m, n, and k are given, $f(c)$ increases as c increases. While in the FI strategy, c is shifted left by m bits, then shifted to the right by n bits, and then added k. The final result of $f(c)$ is the interval between two flits. If $m>n$, it is equivalent to finally shifting c to left by $(m-n)$ bits and adding k. If $m<n$, it is equivalent to finally shifting c right $(n-m)$ bits and adding k. Only simple hardware is needed to realize the operation. Then flit transmission interval can be determined.



(a) FI



(b) RI

Fig. 1. Relation between injection rate and m, n, k.

Under the RI strategy, whether data is allowed to be sent in the current clock cycle is random. The probability of a packet been sent is related to c. The smaller the value of c, the greater the probability that the current transmission is allowed. The method is to generate a random number r every clock cycle and compare it with $f(c)$. If $r \geq f(c)$, data can be sent in the current clock cycle; if $r<f(c)$, data can not be sent now. When the network is more congested and the value of c is larger, $f(c)$ is also larger, and the probability of data been sent is small. So transmission rate control can be achieved.

D. Congestion Rate Analysis

Under the PFT congestion avoidance method, the response rate of network congestion control is closely related to the currently used credit value c and the congestion control strategy. Fig. 1(a) and Fig. 1(b) respectively show the relationship between the rate of packet injection into the network and c under FI and RI strategies.

Fig. 1(a) shows the change trend of the injection rate with the change of c and k under the FI strategy when $m = 0$, $n = 8, 9, 10, 11, 12$. It can be seen from the figure that when n is a constant value, with the increase of c, the injection rate gradually decreases. Under the same value of c and k, the larger the n, the smaller the injection rate. And as c increases larger, the faster the injection rate decreases. We can conclude that by setting reasonable n and k (for example, set n between 8-12, $k = 0$), the injection rate can be continuously changed in a larger interval with the change of c. That is, a fairly good congestion control can be achieved.

Fig. 1(b) shows the injection rate changes with c and m under the RI strategy when $n = 0$ and $k = 0$. It can be seen from the figure that when m is between 15-22, the injection rate changes significantly with the difference of c. At this time, the congestion avoidance mechanism can play a good role.

IV. Evaluation

A. Performance Criterion

Before discussing experiment results, we will give a standard for evaluating the congestion control methods, and use it to measure the effect of congestion control under different parameter configurations.

In a common network evaluation system, the delay or throughput of the network is usually used as an important index to measure the performance of a network. In some systems that use small packets for frequent and fast data transmission, the network delay is required to be as low as possible. Whereas in some systems that carry out large data communication, the network throughput is more important. Considering the complexity of real systems, we define a criteria for evaluating a congestion control algorithm.

Suppose that when the network is not under congestion control, its delay and throughput rate are D_0 and B_0, respectively. When congestion control algorithm C is adopted, its delay and throughput rate are D_C and B_C. Then the control factor of the congestion control algorithm C is defined as:

$$F_C = \frac{B_C}{D_C} / \frac{B_0}{D_0} = \frac{B_c \cdot D_0}{B_0 \cdot D_C} \tag{2}$$

In general, F_C>1, and the larger the F_C, the better the effect of congestion control algorithm C. Later we will use the F_C to evaluate the effect of the PFT congestion control algorithm under different parameters.

B. Evaluation Environment

In order to evaluate the effect of the PFT congestion control strategy proposed in this paper, we built an actual interconnected network environment and evaluated the performance using GPCNet, a benchmark dedicated to network congestion testing. This tool is a benchmark proposed by researchers at Argonne National Lab, Lawrence Berkeley National Lab and Cray to evaluate network congestion. The benchmark is desired to be topology-agnostic, MPI implementation-agnostic, and network provider-agnostic, and can characterize network congestion impacts on various system architectures. The tool is portable across a variety of topologies and architectures, and can generate complex communication patterns with simple tuning.

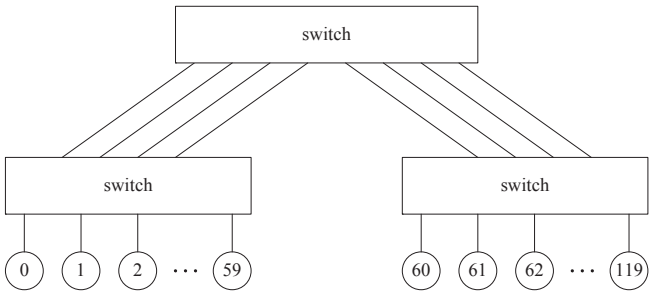


Fig. 2. Topology of experiment.

We designed an ASIC network interface chip (NIC) based on the PFT congestion control method proposed in this article. The NIC chip is implemented using 28nm CMOS process, including a network port bandwidth of 100Gbps, and a PCIE Gen3 x16 interface to connect to the CPU. Its core frequency is 700MHz. Based on the NIC chip, we built a 120-node fat-tree interconnection network to evaluate the PFT congestion avoidance algorithm. The network topology used for testing is shown in Fig. 2.

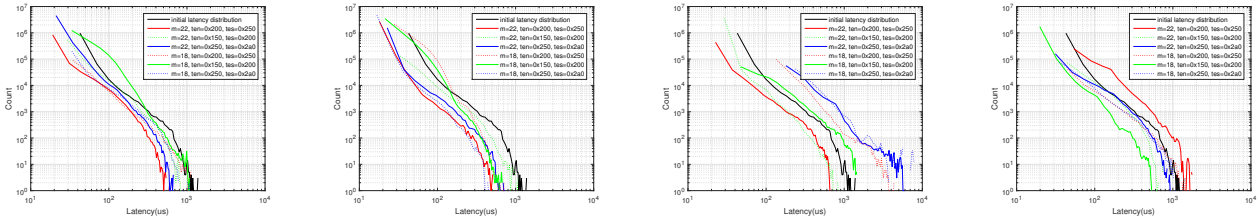
C. Experiment Result

Under the GPCNeT test framework, different PPorts (number of Processes per network Port) are used to test network congestion scenarios. The default PPort values are 1, 8, and 16. In order to analyze the network congestion better, this article uses PPort=16 for experiments.

Fig. 3 and Fig. 4 respectively show the distribution curves of network delay and bandwidth in p2p mode under different strategies. The ordinate is the number of communication processes with different delay or bandwidth in each interval. The black solid line is the original distribution curve when the network is congested, and the other colors are the distribution curve after congestion control obtained according to different PFT strategies and parameters.

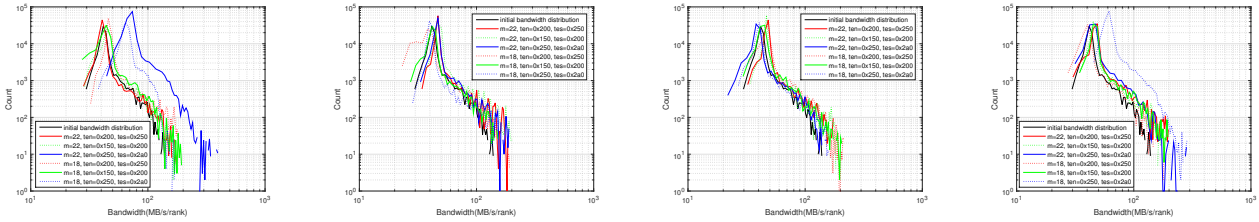
Fig. 3 shows the distribution of the number of communication processes with different delays under different strategies. The more the curve is to the left, the smaller the overall delay in this configuration. And the more the right, the larger the overall delay. It can be seen from the figure that in heavy network congestion, different PFT strategies can be adopted to reduce network delay by adjusting configuration parameters.

Fig. 4 shows the distribution of the number of communication processes with different bandwidths under different



(a) FT-FI. (b) AT-FI. (c) FT-RI. (d) AT-RI.

Fig. 3. Distribution of the processes number with different delays.



(a) FT-FI. (b) AT-FI. (c) FT-RI. (d) AT-RI.

Fig. 4. Distribution of the processes number with different bandwidth.

strategies. The more to the left of the curve, the smaller the bandwidth in this configuration. Whereas the more to the right of the curve, the larger the bandwidth in this configuration. It can be seen from the figure that when the network is in heavy congestion, using different PFT strategies and adjusting configuration parameters can increase the network bandwidth.

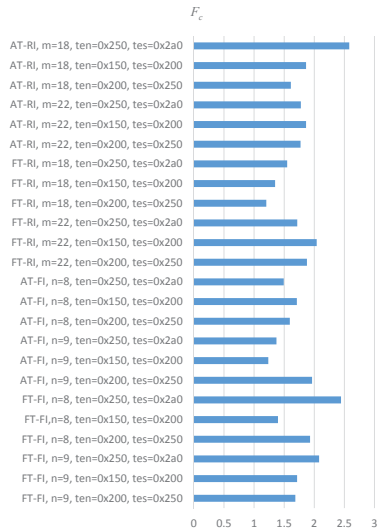


Fig. 5. FC under different strategies and configurations.

Fig. 5 shows the value of the congestion control factor F_C under different strategies and configurations. As mentioned above, F_C characterizes the congestion control effect of the PFT congestion control method. The larger the value is, the better the congestion control effect appears under this configuration. It can be seen from the figure that under most strategies and configurations, the value of F_C is greater than 1, indicating a good congestion control effect. Among them, under the AT-RI strategy, when $m = 18$, $n = 0$, $k = 0$, $threshold\_enter$ =0x250, $threshold\_escape$ =0x2a0, the control factor F_C will be 2.58. In this configuration, the network congestion can be alleviated greatly.

V. Conclusion

Aiming at the network congestion problem in massively parallel interconnection networks, this paper proposed an proactive flow throttling method based on end node creditPFT. With this method, the congestion information in the intermediate nodes of the network is not required during congestion control. Only the sending credit of the end node can be used to adjust the sending flow at the data source.

In the hardware implementation of PFT, it is only necessary to add 4 counters at the sending end, as well as some of parameter configuration registers and related FSM(Finite State Machine) to realize efficient flow control and automatic adjustment process at the network sending end. Moreover, because PFT does not need to collect congestion information on the entire transmitting paths, it would be more sensitive to network congestion. In a real interconnection network system using PFT, we found that by adjusting the control strategy and parameters of the PFT, fairly good congestion avoidance effect can be achieved. Therefore, PFT is more suitable for a network environment where congestion control can only be performed at the network end node.

Acknowledgment

This work was supported by the National Key Research and Development Program of China (No. 2018YFB2202300) and National Postdoctoral Program for Innovative Talents (No.BX20190091).

REFERENCES

[1] D. Chen, N. A. Eisley, P. Heidelberger, R. M. Senger, Y. Sugawara, S. Kumar, V. Salapura, D. Sattereld, B. Steinmacher-Burow, and J. Parker. The ibm bluegene/q interconnection fabric. IEEE Micro, 32(1), 2012.

[2] R. Alverson, D. Roweth, and L. Kaplan. The gemini system interconnect. In Proceedings of the IEEE Symposium on High Performance Interconnects, 2010.

[3] G. Faanes, A. Bataineh, D. Roweth, T. Court, E. Froese, B. Alverson, T. Johnson, J. Kopnick, M. Higgins, and J. Reinhard. Cray cascade: a scalable hpc system based on a dragony network. In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, 2012.

[4] Kang Jin, Cunlu Li, Dezun Dong, Binzhang Fu. HARE: History-Aware Adaptive Routing Algorithm for Endpoint Congestion in Networks-on-Chip. In International Journal of Parallel Programming, 2019.

[5] Cunlu Li, Dezun Dong, Xiangke Liao, John Kim, Changhyun Kim. DeepHiR: Improving High-radix Router Throughput with Deep Hybrid Memory Buffer Microarchitecture. In Proceedings of the 33rd ACM International Conference on Supercomputin, 2019.

[6] Zhengbin Pang, Min Xie, Jun Zhang, Yi Zheng, GuibinWang, Dezun Dong, Guang Suo. The TH Express high performance interconnect networks. Frontiers of Computer Science, 8(3):357-366, 2014.

[7] Saurabh Jha, Archit Patke, Jim Brandt, Ann Gentile, Mike Showerman, Eric Roman, Zbigniew T. Kalbarczyk, William T. Kramer, Ravishankar K. Iyer. A Study of Network Congestion in Two Supercomputing High-Speed Interconnects. In Proceedings of th IEEE Symposium on High-Performance Interconnects, 2019.

[8] Shan Huang, Dezun Dong, Wei Bai. Congestion control in high-speed lossless data center networks: A survey. Future Generation Computer Systems, 2018.

[9] N.T. Kung, R. Morris. Credit-based flow control for ATM networks. IEEE Network, Vol. 9, Issue. 2, 1995.

[10] Shan Huang, Dezun Dong, Wei Bai. Congestion Control in High-Speed Lossless Data Center Networks: A Survey. In Future Generation Computer Systems, 2018.

[11] Tianye Yang, Dezun Dong, Cunlu Li, Liquan Xiao. CRSP: Network Congestion Control through Credit Reservation. In Proceedings of 16th IEEE International Symposium on Parallel and Distributed Processing with Applications, 2018.

[12] IEEE Std 802.1Qbb-2011. IEEE Standard for Local and metropolitan area networks–Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks–Amendment 17: Priority-based Flow Control, 2011.

[13] G. Pfister, V. A. Norton. Hot spot contention and combining in multistage interconnection network. IEEE Trans. Comput. 100 (10) 943948, 1985.

[14] Zihao Wei, Dezun Dong, Shan Huang, Liquan Xiao. EC4: ECN and Credit-Reservation Converged Congestion Control. In ICPADS, 2019.

[15] Cunlu Li, Dezun Dong, Zhonghai Lu, Xiangke Liao. RoB-Router: A Reorder Buffer Enabled Low Latency Network-on-Chip Router. In IEEE Transactions on Parallel and Distributed Systems, 2018.

[16] Olivier Bonaventure. Computer Networking: Principles, Protocols and Practice. 2011.

[17] M. Allman, V. Paxson and W. Stevens. TCP congestion control. 1999.

[18] L. Brakmo, S. O'Malley and L. Peterson. TCP Vegas: New techniques for congestion detection and avoidance. In Proceedings of the ACM Special Interest Group on Data Communication, 1994.

[19] Yang Bai, Dezun Dong, Shan Huang, Zejia Zhou, Xiangke Liao. SSP: Speeding up Small Flows for Proactive Transport in Datacenters. In Proceedings of 22nd IEEE International Conference on Cluster Computing, 2020.

[20] C. Fu, S. Liew. TCP Veno: TCP enhancement for transmission over wireless access networks. IEEE J. Sel. Areas Commun., vol. 21, no. 2, pp. 216-228, 2003.

[21] InfiniBand Architecture Specification: Release 1.3. InfiniBand Trade Association, 2015.

[22] Greg Faanes, Abdulla Bataineh, Duncan Roweth, Tom Court, Edwin Froese, Bob Alverson, Tim Johnson, Joe Kopnick, Mike Higgins, James Reinhard. Cray Cascade: A scalable HPC system based on a Dragonfly network. In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, 2012.

[23] Ke Wu, Dezun Dong, Cunlu Li, Shan Huang, Yi Dai. Network Congestion Avoidance through Packet-chaining Reservation. In Proceedings of 48th IEEE International Conference on Parallel Processing, 2019.

[24] R. Pan, B. Prabhakar, A. Laxmikantha. Qcn: Quantized congestion notification. IEEE802, vol. 1, 2007.

[25] M. Alizadeh, B. Atikoglu, A. Kabbani, A. Lakshmikantha, R. Pan, B. Prabhakar, M. Seaman. Data center transport mechanisms: Congestion control theory and ieee standardization. In Communication, Control, and Computing, 2008 46th Annual Allerton Conference, 12701277, 2008.

[26] Dinghuang Hu, Yang Bai, Dezun Dong, Shan Huang, Xiangke Liao. Converging Credit-based and Reactive Datacenter Transport using ECN and RTT. In Proceedings of 22nd IEEE International Conference on High Performance Computing and Communications, 2020.

[27] Renjie Zhou, Guoyuan Yuan, Dezun Dong, Shan Huang. APCC: Agile and Precise Congestion Control in Datacenters. In Proceedings of 16th IEEE International Symposium on Parallel and Distributed Processing with Application, 2020.

[28] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M.H. Yahia, M. Zhang. Congestion control for large-scale rdma deployments. In Proceedings of the ACM Special Interest Group on Data Communication, 2015.

[29] Aleksandar Kuzmanovic. The Power of Explicit Congestion Notification. In Proceedings of the ACM Special Interest Group on Data Communication, 2005.

[30] Fei Lei, Dezun Dong, Xiangke Liao, Jos Duato. Bundlefly: A Low-Diameter Topology for Multicore Fiber. In Proceedings of 34th ACM International Conference on Supercomputing, 2020.

[31] P. Newman. Backward explicit congestion notification for atm local area networks. In Global Telecommunications Conference, 1993.

[32] Tianye Yang, Dezun Dong, Cunlu Li, Liquan Xiao. BFRP: Endpoint Congestion Avoidance through Bilateral Flow Reservation. In Proceedings of 37th IEEE International Performance Computing and Communications Conference, 2018.

[33] K.-Y. Siu, H.-Y. Tzeng. Intelligent congestion control for abr service in atm networks. In Proceedings of the ACM Special Interest Group on Data Communication, 1994.

[34] Akshay Narayan, Frank Cangialosi, Deepti Raghavan, Prateesh Goyal, Srinivas Narayana, Radhika Mittal, Mohammad Alizadeh, Hari Balakrishnan. Restructuring Endpoint Congestion Control. In Proceedings of the ACM Special Interest Group on Data Communication, 2018.

[35] Radhika Mittal, Vinh The Lam, Nandita Dukkipati, Emily Blem, Hassan Wassel, Monia Ghobadi, Amin Vahdat, Yaogong Wang, David Wetherall, David Zats. TIMELY: RTT-based Congestion Control for the Datacenter. In Proceedings of the ACM Special Interest Group on Data Communication, 2015.

[36] Nan Jiang, Larry Dennison, William J. Dally. Network Endpoint Congestion Control for Fine-Grained Communication. In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, 2015.

[37] N. Jiang, D. U. Becker, G. Michelogiannakis, W. J. Dally. Network congestion avoidance through speculative reservation. In In Proceedings of the 2012 IEEE 18th International Symposium on High-Performance Computer Architecture, 2012.

[38] Dezun Dong, Shan Huang, Zejia Zhou, Wenxiang Yang, Hanyi Shi. FastCredit: Expediting Credit-based Proactive Transports in Datacenters. In Proceedings of 23rd IEEE International Conference on Parallel and Distributed Systems, 2020.

[39] Dezun Dong, Ke Wu. Reducing Tail Latency in Proactive Congestion Control via Moderate Speculation. In Proceedings of 22nd IEEE International Conference on High Performance Computing and Communications, 2020.

[40] Inho Cho, Keon Jang, Dongsu Han. Credit-Scheduled Delay-Bounded Congestion Control for Datacenters. In Proceedings of the ACM Special Interest Group on Data Communication, 2017.

[41] Zejia Zhou, Dezun Dong, Shan Huang, Zihao Wei. ExpressPass++: Credit-Effecient Congestion Control for Data Centers. In 2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking, 2019.

[42] Mark S. Birrittella, Mark Debbage, Ram Huggahalli. Intel Omni-path Architecture: Enabling Scalable, High Performance Fabrics. In IEEE 23rd Annual Symposium on High-Performance Interconnects, 2015.