Software Defined Backpressure Mechanism for Edge Router

Xiangqing Chang[1], Jun Li[1], Guodong Wang[1], Zexin Zhang[1,2], Lingling Li[1],Yalin Niu[1]

[1]Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China

[2]University of Chinese Academy of Sciences, Beijing, China

Email: changxiangqing@cstnet.cn, lijun@cnic.cn,{wangguodong, zhangzexin, lilingling, niuyalin}@cstnet.cn

***Abstract*—Increased network traffic has put great pressure on edge router. It tends to be more expensive and consumes more resources. Buffer is especially the most valuable resource in the router. Given the potential benefits of reducing buffer sizes, a lot of debate on buffer sizing has emerged in the past few years. The small buffer rule, for example, was challenged at edge router. Instead of buffer sizing, the goal of our work is to find out a way to relieve the pressure of edge router and logically enlarge its buffer size without incurring additional costs. In this paper, taking advantage of the global view of SDN, we proposed Software Defined Backpressure Mechanism (SD-BM) to alleviate the pressure of edge router. Particularly, we gave Refill and Software Defined Networking based RED (RS-RED) algorithm, which makes it possible to enlarge the network buffer logically and offloads traffic from busy egress router to free ingress devices. Simulation results show that it has comparable performance, both in time delay and loss rate, with edge router which has large buffer in traditional way. The results can have consequences for the design of edge router and the related network.**

Keywords—Software Defined Networking; backpressure mechanism; refill queue

I. Introduction

Edge router, or WAN(wide area network) router, is usually high-end router which resides at the network edge and aggregates all data coming from various places within an enterprise network. Edge router is a critical device to provide connection with WAN through WAN link. As a greater number of applications are delivered via the cloud, there has been a strong trend that the amount of data carried by the network has put increasing pressure on the edge router. Except for the port density and bandwidth upgrades, large buffer is the most influential factor for edge router. To make the most of valuable WAN link, network operators follow the rule-of-thumb [1], namely the Bandwidth-Delay Product (BDP) rule, and require that router manufacturers provide 250ms of buffering. For a router runs at an aggregate rate of 40Gb/s, the BDP rule mandates a buffer size of 1.25 GigaBytes. Such a large packet buffer complicates the design of high-speed routers, leads to higher power consumption, more board space, and makes them very expensive [2]. Further, one of the primary technological limitations of next generation optical routers is the difficulty in building large optical buffers [3].So, it is extremely difficult to build packet buffers at 40Gb/s and beyond [4]. Appenzeller et al. [4] argued that a small buffer rule could be used with desynchronized flows, but Dhamdhere et al. [5] challenged that much larger buffers are needed at the edge of the network. Furthermore, network operators are likely to buy the router with larger buffers [4] for the bursty nature of traffic. It means that the BDP rule is still widely used by router manufactures and believed by operators.

Rather than sizing router buffers, the goal of our work is to find a proper way to reduce the pressure of edge router, and make it more feasible to still use smaller buffer for edge router without incurring expensive cost by logically enlarging the whole network buffer.

We observed that the local resource allocation model of network devices is an important factor to restrict the pressure reducing for edge router. Each device in the enterprise network, from the access device and distribution device to the core device and border edge router, implements its function independently. Over the last decade, although numerous AQM(Active Queue Management) schemes have been proposed including Random Early Detection (RED) [6], Proportional Integral (PI) [7] etc., they are available to local node. Each device makes resource control decision without knowing others. So, the access or distribution devices(we call them ingress devices later in this paper) possibly still allow the incoming traffic to keep pouring in while the border edge device has traffic congestion, because only the border edge router can detect the congestion of WAN link. From Fig.1,we can see that the queue lengths of ingress devices are always zero whenever the queue length of edge router reaches the maximum value. Actually, network operators usually try to reduce the pressure of edge router by making static rate limit configurations for arrival flows at the ingress devices. But the inbound and outbound traffic is dynamic, and this leads to bad user experience and inefficiency of WAN link.

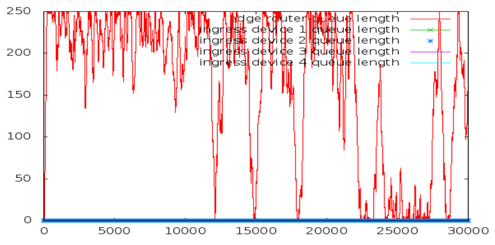

Fig. 1. Queue lengths of ingress devices(mainly shown as blue line) and edge router(shown as red line)

With the paradigm of Software Defined Networking(SDN) [8],we proposed a new Software Defined Backpressure Mechanism(SD-BM) to relieve the pressure of edge router dynamically. It has two significant features: 1) dynamically setting the drop rate of incoming traffic by centrally calculating at SDN controller. 2) globally caching the packets by fully using the cache of ingress devices.

The contributions of this paper are as follows:(1)To the best knowledge of the authors, it is the first work to make use of free cache of ingress devices in traffic control. This makes it more possible to have edge router use relatively small buffer. (2)While traditional AQM methods drop packets probabilistically on edge router at the onset of congestion, we start to drop packets at the ingress devices to reduce the workload of edge router. (3)We evaluated the new approach and found it could show competitive performance compared with traditional strategy.

II. Related Work

There is a rich heritage of work in sizing router buffers and traffic control that informed our design. We describe a subset of these related efforts in this section.

A. Research work in sizing router buffers

Router buffers are sized today based on the classical BDP rule [1].The rule states that routers should have sufficient queueing capacity of Bandwidth-Delay Product to maintain full bottleneck utilization. The BDP rule can be found in architectural guidelines. As the BDP rule implies large and expensive buffers when link speeds scale up to gigabit level and above, Appenzeller et al. [4] have suggested sizing strategies that significantly reduce the amount of required buffering. The publication of this result has since gained considerable interest in the buffer sizing problem, including papers questioning the applicability of this result. Dhamdhere and Dovrolis [5] presented a case where even BDP leads to high packet loss rate in edge networks, and they suggest that increased buffer size is needed. For more work on router buffer sizing, [2] can be referenced. In summary, it turns out that the different rules are under different assumptions. It should be premature to reduce buffer in routers.

B. Research work in traffic control

TCP congestion-avoidance optimization: There are two main types:1) source control algorithms. Van Jacobson introduced congestion control into TCP in 1988 [9]. Then, more TCP control algorithms emerged, for example, Tahoe [10], NewReno [11], Vegas [12] and SACK [13]. 2) Active Queue Management(AQM). AQM is a class of queue management algorithms at intermediate node. It probabilistically drops packets before buffer overflow. Over the last decade, numerous AQM schemes have been proposed including Random Early Detection (RED) [14], Random Exponential Marking (REM) [15],Adaptive Virtual Queue (AVQ) [16],Proportional Integral (PI) [7],Fuzzy and Neural PID [17] etc.. Use packet marking instead of packet dropping to signal network congestion, is called explicit congestion notification (ECN) [18]. But, ECN is generally disabled by default due to heterogeneity of the Internet. These schemes promoted the development of traffic control technology, but they are mainly for TCP flows, and limited to local node.

Resource Scheduling: All kinds of queue management techniques, e.g., FQ(Fair Queuing) [19],WRR(Weighted Round Robin) [20], and traffic shaping/limiting techniques are provided on network routers. These functions are also limited to one node.

Traffic Engineering with SDN: With the principle of Max-Min fair allocation, B4 [21] and SWAN [22] use SDN in the context of inter-DC WAN to arrange the traffic by choosing different routing tunnels. They also use bandwidth enforcer to limit the bandwidth occupation, but it is executed in the condition of knowing the traffic demand, and this function was not described in detail. The focus of B4 and SWAN is on traffic routing.

III. Software Defined Backpressure Mechanism

Our work is motivated by the need to alleviate the pressure of edge router and achieve sufficient buffer without incurring expensive cost. We want to seek a solution that can use the global resource. Thus, we put forward Software Defined Backpressure Mechanism (SD-BM). Further, in order to be convenient to compare with traditional RED under new architecture, we try to preserve key features that RED possesses. So, we specifically propose Refill and SDN-based RED algorithm(RS-RED). In this section, we firstly introduce SDN-based system design. Then we give detail description for RS-RED. Finally, the influence of the new algorithm is analyzed.

A. SDN-based system architecture

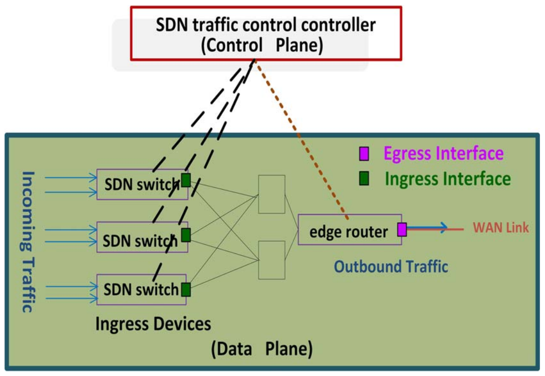

Fig. 2. The architecture of SD-BM

The architecture of SD-BM is shown in Fig. 2. The control plane is decoupled from network hardware to centralized controller, which has a global view of the network and orchestrates all activity. It collects statistical data from each device of the network through SNMP or OpenFlow, makes the decision of control policy, and installs the policy through southern interface protocol to the ingress switches in data plane. Then, all the traffic flowing through data plane will follow the control policy. From the data plane in Fig.2,we can see the network topology we focus on. In the system, the ingress devices will be SDN switches which are fully controlled by controller. The ingress interfaces, shown as little green boxes in Fig.2., are the control points of the data plane traffic. According to the state of network, the execution of dropping and buffering packets is started at the ingress switches instead of edge router. The WAN interface of edge router can be considered as egress interface.

B. *Refill and SDN-based RED algorithm (RS-RED)*

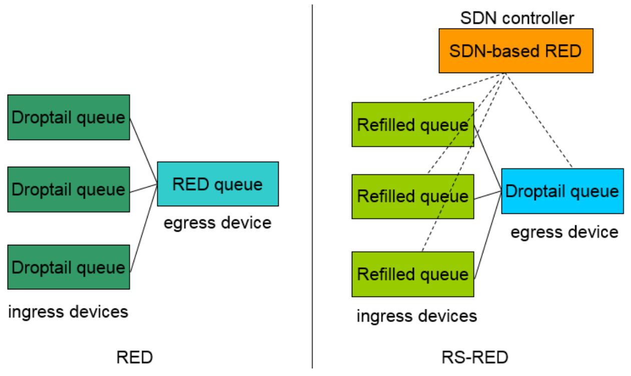

Fig. 3. RED vs RS-RED

As shown in Fig. 3, there are vital differences between RS-RED and the traditional RED: 1) For RED, RED queue is only set at the bottleneck device. The queue type of ingress devices is default queue Drop Tail. However, for RS-RED, the ingress devices adopt a new kind of queue called refill queue, which can store packets together with the Drop Tail queue of egress device. 2) For the traditional method, there is no cooperation among the queues, because each device work independently. However, for RS-RED, the queues at ingress devices and egress device are centrally controlled by SDN controller. Thus, they can work coordinately to achieve the common goal of congestion avoidance for the network.

Just like RED, RS-RED has the algorithm for computing the average queue size and the algorithm for calculating the packet dropping probability. Particularly, RS-RED has the new algorithm for refill queue.

In the algorithm for computing the average queue size, RS-RED collects queue occupancy information and calculates the average queue size of the network using the low-pass filter given by RED. To represent the whole occupancy status of the ingress queue and the egress queue, virtual queue is adopted. RS-RED will calculate the average queue size avg_{ℓ_v} of the virtual queue length ℓ_v at SDN controller.

$$\ell_v = \ell_v + \sum_{t=0}^{k}((\sum_{i=1}^{n} V_i(k)) - V_\mu(k))$$

$$avg_{\ell_v} = (1 - w_q) * avg_{\ell_v} + w_q * \ell_v$$

Where $V_i(k)$ is the incoming traffic rate of ingress interface i, and $V_\lambda(k)$ is the sum of $V_i(k)$, $V_u(k)$ is the outgoing rate of WAN interface over a sampling period, w_q is the queue weight.

Algorithm for calculating the packet dropping probability determines how frequently the device drops packets. Just as RED, the dropping probability p_b is calculated through comparing the average queue length to the minimum threshold $\min_{th}$ and the maximum threshold $\max_{th}$ .However, different from RED, p_b is calculated at SDN controller, and then distributed to the ingress devices. In order to ensure the number of arriving packets between dropping packets is a uniform random variable rather than a geometric random variable, RED calculates the final dropping probability p_a by considering the count n of passed packets since the last dropped packet. For RS-RED, the p_a is calculated at ingress device, since n need to be collected in real time for each ingress device in data plane.

```
At ingress devices:

For each packet is about to leave:
   If ( $p_a$>0)
   {
      Retain the packet in the probability  $p_a$
   }

For each packet arrival:
   If ( $p_a$>0 and $\ell_i$ > $L_I$ )
   {
      Drop packet in the probability  $p_a$
   }
```

Fig. 4. RS-RED: Algorithm for refill queue

Algorithm for refill queue is shown in Fig.4. Refill queue is the critical component to accomplish backpressure mechanism. Without refill queue, the ingress queues are always empty even when the length of egress queue is close to the maximum value, and the total packet loss is not reduced. In order to proactively share the load with egress device, we need to have a way to store packets at ingress device. So, when the ingress device find that p_a is not zero, it will retain the packet at the head of the egress queue in the probability of p_a.This looks like that the packet ,which is about to leave ,is refilled to the ingress queue. So, we call it refill queue.

When the length of the refill queue reaches the threshold of the ingress queue, it will drop packet in the probability of p_a. This ensures that the occupancy of the ingress queue maintains at a proper level. Refill queue can offload traffic pressure for the egress router with peak shaving effects.

C. *Queuing delay analysis of refill queue*

Free buffers of the ingress queues can be utilized by using refill queue. However, it also make an influence on queuing delay. In this section, from mathematical point of view, we try to analyze the queuing delay incurred by refill queue.

Assume that the queue size of refill queue is ℓ_i, the queue length threshold is L_i, the total number of retaining times caused by refilling is m_i, the link capacity between ingress device and egress device is C_I. Thus, the total queuing delay τ incurred by refill queue can be modeled as

$$\tau = \tau_{\ell_i} + \tau_{m_i} = \frac{\ell_i}{C_I} + \frac{m_i}{C_I}$$

For each packet in the ingress refill queue, it may be either retained in the probability of p_a, or sent in the probability of 1-p_a. Assume that p_m is the probability that the retaining times for ℓ_i packets is m_i. According to Binomial theorem, p_m can be mathematically depicted as

$$p_m = C_{\ell_i}^{m} p_a^{m}(1-p_a)^{(\ell_i - m)}$$

Let p_m * be the maximum of p_m. Let τ_m * be the maximum queuing delay when the probability is p_m *.

To get the τ_m *, we represent p_m * as A(p_a)×B(m_i). Where A(p_a) = $p_a^{m}(1-p_a)^{(\ell_i - m)}$, B($m_i$) = $C_{\ell_i}^{m} = \dfrac{\ell_i!}{m!(\ell_i - m)}$.

For A(p_a),we have:

If p_a = 0.5, then A(p_a) = $(0.5)^{\ell_i}$. If p_a > 0.5, then A(p_a) < $(p_a)^{\ell_i}$ .If p_a < 0.5, then A(p_a) < $(1-p_a)^{\ell_i}$.

In summary, $A(p_a) \le p_a^{\ell_i}$ (0.5 < p_a < 1).

Let ℓ_i equal to L_i, i.e. 20,30,40,50, we drew the function curve of A(p_a) and B(m_i) with Matlab as Fig.5. We can see that A(p_a) begins to grow rapidly when p_a is greater than 0.9 (or less than 0.1). The maximum value A* of A(p_a) occurs when p_a is 1. The maximum B(m) occurs when m is equal to the half of ℓ_i .This m is defined as m*. Let B* be the maximum value of B(m), let s be the size of packet . Then we have the maximum value τ * of τ :

$$\tau^* = \tau_{\ell_i} + \tau_m^* = (L_i + \frac{L_i}{2}) \times \frac{s}{C_I}$$

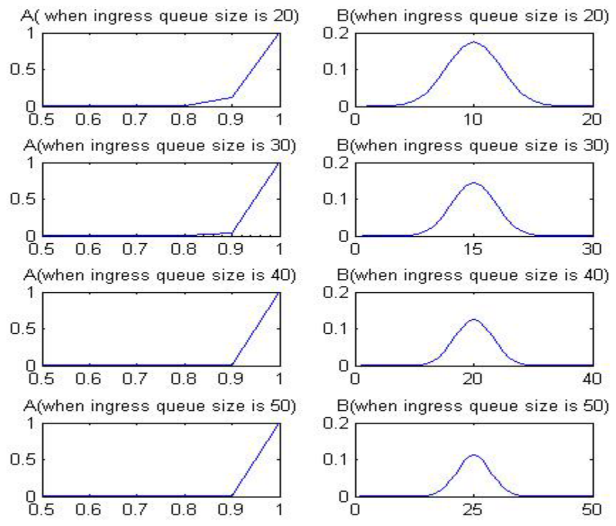

Fig. 5. The function curve of A(p_a) and B(m_i)

Set s as 1000 bytes. Since we evaluate the scheme with ns3, considering the performance of our ns3, we set C_I as 10Mbits/s. In real network, C_I should be gigabit or above, which is much faster than we assumed. This implies τ * of the real network will be much smaller than we gave in Table I.

TABLE I. TIME DELAY OF REFILL QUEUE (IN MAXIMUM PROBABILITY)

L_i	A*	m*	B*	p_m*	τ_m*	τ*
20	1	10	0.176	17.62%	8ms	24ms
30	1	15	0.135	13.5%	12ms	36ms
40	1	20	0.125	12.5%	16ms	48ms
50	1	25	0.108	10.8%	20ms	60ms

IV. EVALUATION

A. Test Network Condition

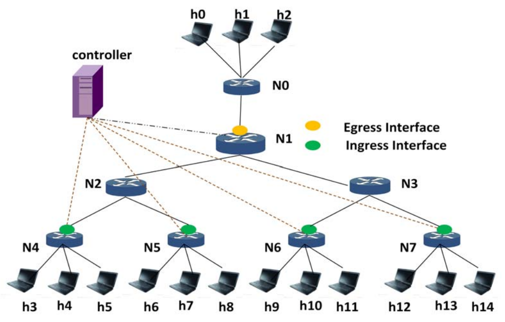

Fig. 6. Test network

We implemented a prototype module in ns3 version 3.19 and conducted extensive simulation using the network shown in Fig. 6. The topology is acquired by emulating an enterprise production network. In order to focus on the method described in this paper, we removed the alternative paths between ingress interfaces and egress interface. There are 8 network nodes and 15 hosts. N4,N5,N6 and N7 are ingress devices, N1 is the edge router. In the experiments, testing traffic will be initiated from h3~h14,and ended with h0~h2. TCP NewReno is used by these source and sink nodes. All the packets of testing traffic have the same size of 1000 bytes. The bandwidth of all the links is 10M. The sampling period is 40ms.

B. *Traffic Sources*

Simulation of Internet traffic itself is a difficult problem. The pragmatic approach taken here is to use mixed TCP and UDP traffic. Since small numbers of simultaneous TCP flows likely to be far more bursty than aggregations of very large numbers of TCP flows [1], we focus on small numbers of flows to simulate bursty behavior. We use ON/OFF sources which meet the Pareto distribution to generate self-similar traffic [23].

C. *Test Results*

Based on the tests, we made the following discoveries:

1) Location of dropping packets is changed .

With traditional RED method, packet loss only occurs at the location of egress interface. When RS-RED is used, packet loss occurs at the location of ingress interfaces or both of ingress and egress interfaces. The percentage of dropped packets can be regulated by adjusting parameters. The variation of dropping location implies that the traffic can be reduced in advance for edge router with RS-RED.

2) RS-RED has comparable performance with router that has large buffer.

In this section, we want to verify if the RS-RED has the potential to make it unnecessary for edge router to use large buffer. For RS-RED, we set the egress queue size is 100 packets, and refill queue size is 50 packets. For RED, we set the egress queue size is 300 packets which is near the product of RTT and C_O. C_O is the egress link capacity. The total buffer size for RS-RED, i.e.,50*4+100 = 300, is equal to the egress queue size of RED.

As shown in Fig.7, we can see that the performance of RS-RED, including throughput, loss rate, delay and jitter, can be nearly as good as RED with large buffer size at edge router. Due to limitations on space, we only show the traffic case where the total sending rates are $1\times C_O$,$1.4\times C_O$,$1.8\times C_O$.At the same time, although RED is a well-known algorithm to ensure low delays, the time delay of RS-RED is even less than RED. It means that RS-RED provides a new way to persuade network operators to use edge router with smaller buffer.

3) Influence of refill queue size

In this section, we want to observe the performance trend with the variation of the refill queue size. In Fig.1, we've seen the ingress queue lengths are zero. However, as shown in Fig.8, corresponding to the refill queue threshold, the ingress queue lengths are almost 20,30, and 40, respectively.

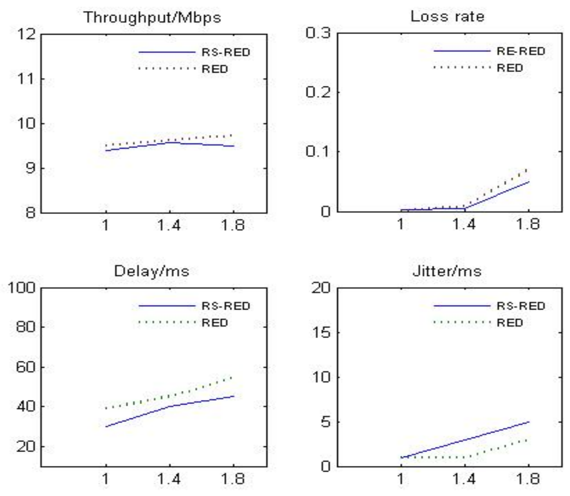

Fig. 7. RS-RED vs. RED with large buffer

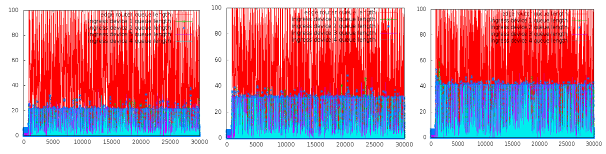
Fig. 8. The ingress queue lengths under corresponding refill queue threshold(20,30,40)

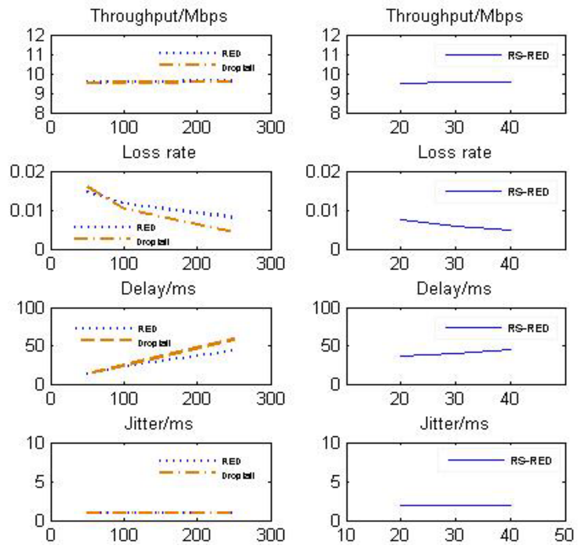

Fig. 9. Comparing RS-RED ,RED and Drop-Tail in different queue size

As shown in Fig.9, RS-RED has similar trend with RED and Drop-Tail. If the queue size increases, the packet loss degrades, the time delay increases, the throughput and jitter have little change. But RS-RED changes in a smooth way.

Then, under the same condition of egress queue, we particularly compared the performance on time delay and loss rate for RED and RS-RED in table II and III. The data is gathered with the traffic generated by 40 flows with a total rate of $2C_O$. In table II, based on the test results, we calculated the average time delay increase caused by refill ingress queue with different threshold. According to table I, we gave the theory value of τ *. We can see that the actual average value is less than τ *. This is unsurprising. From section III, we know that the value of τ * is the maximum value that emerges in a maximum probability. From table III, we can conclude that the time delay introduced by refill queue is acceptable. Similarly, on loss rate, we calculated the percentage reduced by refill ingress queue. It shows that the loss rate can be reduced by 40.4% when the refill queue threshold is 50 packets.

D. Scope of our results and future work

In the test, we assume that there is one bottle-neck link in the network. We also assume that each ingress device receives similar traffic, so the packet dropping probability can be distributed indiscriminately. In order to obtain a better comparability, we choose to get RS-RED implemented in a way of reforming RED in this paper. Actually ,we have also tested other SDN-based control algorithm to cooperate with refill queue. It shows that the result can be better with proper control algorithm and proper parameters. There remains scope for further refined algorithm and evaluation in future.

TABLE II. TIME DELAY

Method		Average time delay	Increasement	Maximum theory value
RED		81ms	NULL	NULL
RS-RED	20	88ms	7ms	24ms
	30	97ms	16ms	36ms
	40	106ms	25ms	48ms
	50	117ms	36ms	60ms

TABLE III. LOSS RATE

Method		Loss Rate	Reduced Percentage
RED		2.266%	NULL
RS-RED	20	2.062%	9%
	30	1.867%	17.6%
	40	1.715%	24.3%
	50	1.350%	40.4%

V. CONCLUSION

In order to alleviate the pressure of edge router, Software Defined Backpressure Mechanism(SD-BM) was introduced. Particularly, we proposed RS-RED algorithm to implement SD-DM. Instead of taking control mainly at edge router, it focuses on dynamically controlling the traffic at ingress devices according to the whole state of network. With test results, it proved that the pressure of edge router can be offloaded to entrance devices by dropping packets at ingress devices. The size of network buffer can be enlarged logically through caching packets at ingress device with refill buffer. This provides a new way to avoid use large buffer for edge router. According to theory analysis and simulation results, the time delay incurred by refill queue is acceptable. In future, refined control policy will be further studied.

ACKNOWLEDGMENT

This work is supported by the Around Five Top Priorities of “One-Three-Five” Strategic Planning, CNIC(Grant No. CNIC_PY_1401), Chinese Academy of Sciences.

REFERENCES

[1] C. Villamizar and C. Song,High performance tcp inansnet. ACM Computer Communications Review,24(5), 1994

[2] A. Vishwanath, V.Sivaraman,M. Thottan, Perspectives on Router Buffer Sizing: Recent Results and Open Problems, ACM Computer Communications Review,39(2), 2009

[3] P. Bernasconi et al., Architecture of an Integrated Router Interconnected Spectrally (IRIS), In IEEEHPSR, Poland, 2006

[4] G. Appenzeller, I. Keslassy, and N. McKeown, Sizing router buffers, In ACM SIGCOMM 2004, Portland, Oregon, USA, August 2004

[5] A.Dhamdhere, C. Dovrolis, Open issues in router buffer sizing,ACM Sigcomm Computer Communication Review,36(1),January 2006

[6] S. Floyd, V. Jacobson, Random early detection gateways for congestion avoidance, IEEE/ACM Transactions on Networking, vol. 1, no. 4, August 1993

[7] C.V. Hollot, V. Misra, D. Towsley, W. Gong, On desinging improvede conrto1lers for AQM routers supporting TCP flows, INFOCOM, 2001

[8] N. McKeown, Software-defined Networking, INFOCOM, 2009

[9] V. Jacobson, M J. Karels, Congestion avoidance and control, Computer Communication Review, 1988, 18(4).

[10] V.Jacobson, Modified TCP Congestion Control and Avoidance Algorithms, Technical Report, Apr 1990.

[11] S.Floyd, T.Henderson, The New-Reno Modification to TCP’s Fast Recovery Algorithm, RFC 2582, Apr 1999.

[12] S. Lawrence, M. Brak, L. Larry, TCP Vegas: end to end congestion avoidance on a global Internet, IEEE Journal on Selected Areas in Communications, 1995, 13(8).

[13] E. Blanton, M. Allman, K. Fall, L. Wang, A Conservative Selective Acknowledgment (SACK)-based Loss Recovery Algorithm for TCP, RFC 3517, April 2003

[14] B. Braden, D. Clark, J. et al., Recommendations on Queue Management and Congestion Avoidance in the Internet, RFC 2309, Apr. 1998.

[15] S. Athuraliya, S.H. Low, V.H. Li, and Q. Yin, REM: Active Queue Management, IEEE Network, Vol. 15, No. 3, May 2001.

[16] S. Kunniyur, R. Srikant, An Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management, IEEE/ACM Transactions on Networking, Vol. 12 ,2004

[17] Ren Fengyuan, Wang Fubao, Ren Yong, Shan Xiuming, PID Controller for Active Queue Management, Journal of Electronics and Information Technology, Vol25,No.1,Jan.2003

[18] K.K. Ramakrishnan, S. Floyd, D. Black,The Addition of Explicit Congestion. Notification (ECN) to IP, RFC 3168, Sep. 2001.

[19] Demers,A., Keshav,S., Shenker,S, Analysis and Simulation of a Fair-queueing Algorithm, ACM SIGCOMM,1989

[20] H.E. Louis, Round robin scheduling for fair flow control in data communication networks, Ph.D. thesis, MIT, Dec. 1986.

[21] S. Jain, A. Kumar, S. Mandal, et al., B4: experience with a globally-deployed software defined WAN, SIGCOMM, 2013.

[22] C. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, R. Wattenhofer, Achieving High Utilization with Software-Driven WAN, SIGCOMM,2013.

[23] P. Pruthi, A. Erramilli, Heavy-Tailed ON/OFF Source Behaviour and Self-Similar Traffic, ICC 95, June 1995.