An Implementation of TCP Pacing Algorithm based on Multi-core Heterogeneous Systems

Shang Qiuli[1,a], Tu Bo[1,b], Liu Bingshuang[1,c,*], Dai Shuaifu[1,d] and Zhang Jianyu[1,e]

[1] Seven North Road No.42 Building 16, Changping District, Beijing, China

[a]shangqiuli@chanct.com, [b]tubo@chanct.com, [c]liubingshuang@chanct.com, [d]daishuaifu@chanct.com, [e]zhangjianyu@chanct.com

[*]Corresponding Author

Keywords: TCP pacing, Multi-core network platform, Implementation

Abstract. By the features of parallel processing and hardware acceleration, multi-core network processor (NP) are becoming the mainstream platform of next generation network edge devices. In order to take advantage of multi-core NP platform and improve the performance of TCP pacing algorithm, this paper proposes an implementation of TCP pacing algorithm based on multi-core heterogeneous systems. The implementation provides a timing method that is more accurate than the clock interrupt-based timing in Linux kernel. And it is an optional alternative to the traditional TCP implementations. The experimental test based on prototype platform shows that, the smoothness performance of the proposed implementation is improved by 27.78%, and the resource utilization is also better than the traditional scheme.

Introduction

Since streaming media services are sensitive to traffic smoothness, control congestion algorithms based on rate pacing, namely RBP (Rate-based Pacing) algorithms come into being, including TFRC [1], TEAR [2], RAAR [3], etc. As a unilateral TCP pacing algorithm, TCP-SRP [4] achieves congestion control by pacing the packet transmission rate in order to avoid the sawtooth waving of TCP traffic and improve the smoothness of streaming media service.

Without loss of generality, as a typical TCP pacing algorithm, this paper chooses TCP-SRP [4] to discuss the implementation of these algorithms. In today's Internet, TCP NewReno is recognized as the most widely used TCP implementation [6]. Therefore, TCP-SRP refers to the traffic model of TCP NewReno as the rate pacing formula [7].

$$R(p)=\frac{\frac{1}{p}+\frac{1}{E[n]}\left(\frac{1}{1-p}+2^{1+\log_2\frac{W}{4}}-1\right)}{\left(\frac{W}{2}+2+(W-1)p\right)RTT+\frac{1}{E[n]}\left(\frac{f(p)}{1-p}RTO+\left(1+\log_2\frac{W}{4}\right)RTT\right)} \tag{1}$$

By the features of parallel processing and hardware acceleration, multi-core network processor (NP) can support data processing for network storage protocols and TCP/IP in high-speed network environment. Thus, multi-core NPs are becoming the mainstream platform of next generation network storage systems, high-performance network servers, and network edge devices [4, 5].

For the demands on both protocol acceleration and transparent development, a transparent accelerating software architecture for network storage based on multi-core heterogeneous systems has been presented in previous works [5]. The design of heterogeneous system divides the processing cores into two sets: one carries a general-purpose operating system which is Linux operating system (OS) in this paper; the other carries the network processing operating system which is the simple execution environment (SE) provided by Cavium OCTEON.

In order to take advantage of multi-core NP platform and improve the performance of TCP pacing algorithm, this paper proposes an implementation of TCP pacing algorithm based on multi-core heterogeneous systems. By integrating the mechanisms of the prototype platform including inter-core message and Timer Ring, it provides a timing method that is more accurate than the clock interrupt-based timing in Linux kernel, which further helps to smooth the TCP traffic bursts. Besides, the implementation is transparent to the upper layer, and it is an optional alternative to the traditional TCP implementations.

Methodology

On basis of the mentioned transparent accelerating software architecture [5], the implementation of TCP pacing algorithm (e.g. TCP-SRP [4]) based on multi-core heterogeneous systems is shown in Fig.1. The implementation shows that, the sending TCP packets are encapsulated by IP and MAC layer, and further physically transmitted by the hardware sending unit PKO (Packet-Output Processing Unit) on multi-core network processor. It uses the modules of original TCP stack including TCP state machine, TCP session, TCP timer, and reliable transmission, etc. It further bypasses the TCP congestion control and sliding window parts of traditional TCP, and achieves flow control and congestion control by implement sending rate control in network adaption layer.

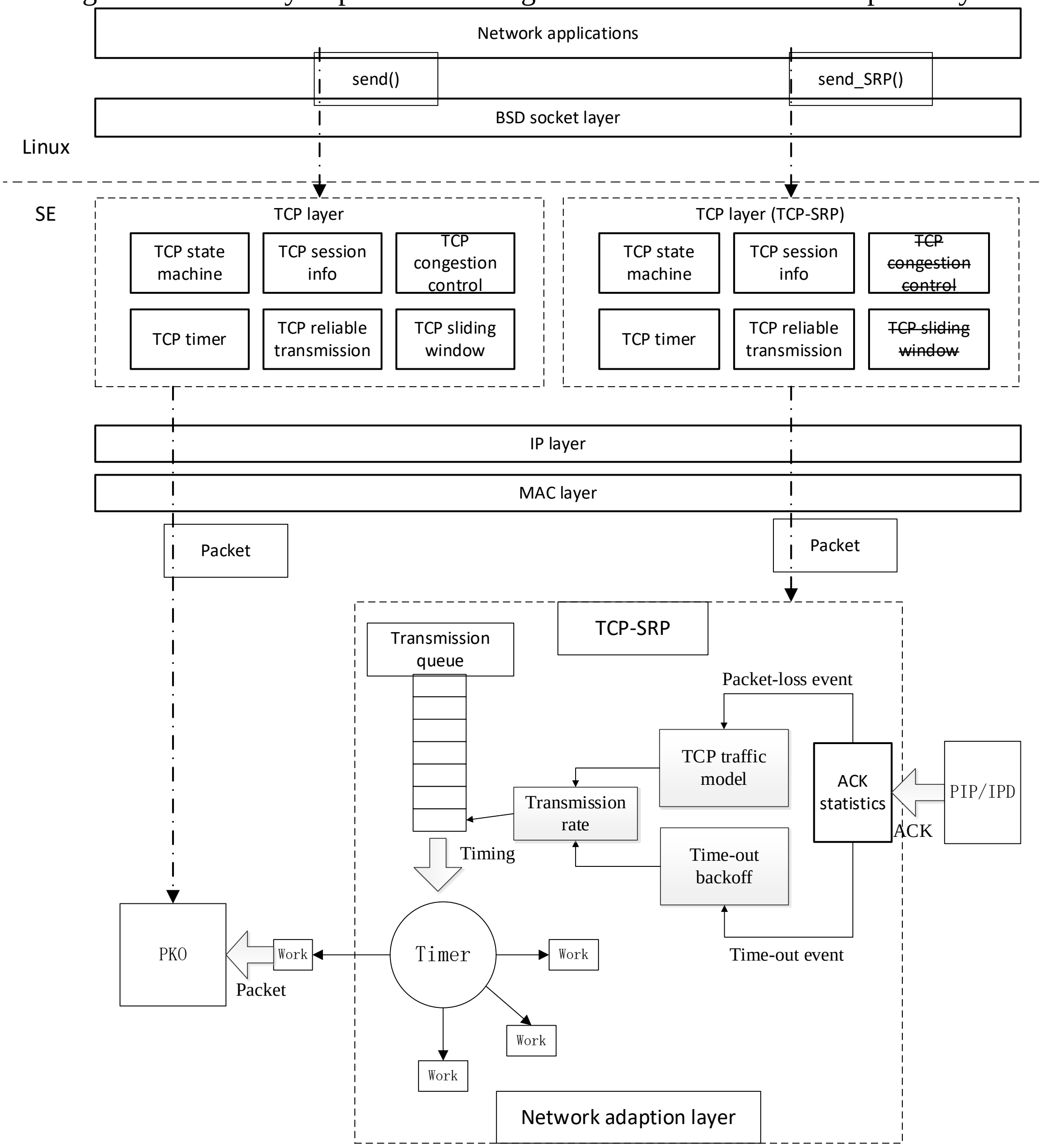

Fig. 1. The implementation of TCP-SRP based on multi-core heterogeneous systems.

In network adaption layer, TCP-SRP counts ACK packets submitted by the hardware receiving unit PIP/IPD (Packet Input Processing/Input Packet Data Unit), and identifies packet-loss event and time-out event. For packet-loss event, TCP-SRP calculates the packet transmission rate, namely the transmission period intervals, according to the packet-loss event probability and the TCP traffic model. For time-out event, TCP-SRP obtains backoff timer according to the time-out backoff strategy. Further, based on the sending timers, TCP-SRP encapsulates the packets in the transmission queue into Timer Work, namely an inter-core message mechanism, and then submits them to Timer Ring.

Timer Ring is the hardware timing unit provided by Cavium OCTEON processors. Timer Ring rotates a "gear tooth" in each "tick" duration (timing step) and releases/submits the time-up works on the “gear tooth”. In the CN58XX processors, it defines 16 independent Timer, the granularity of each "tick" is 1s / 800MHz = 0.00125us. Therefore, Timer can time the packet transmission intervals according to the timing messages, and submits the time-up packet to PKO.

Based on the P-SPL multi-core topology proposed by the previous work [5], it implements the TCP/IP stack of TCP-SRP and the original heterogeneous TCP/IP stack on separate group of cores in parallel. In this way, TCP-SRP can be an optional alternative to the traditional TCP algorithms.

Evaluation

This paper uses “Hili” server [8] based on Cavium OCTEON CN5860 as the experimental environment. It tests the traffic smoothness of TCP-SRP by the proposed implementation based on multi-core heterogeneous systems (“heterogeneous scheme” for short). And this paper takes the traditional Linux kernel-based architecture (“Linux scheme” for short) as the comparison.

As the main performance indicator of TCP pacing algorithms, traffic smoothness is represented by the Coefficient of Variation of protocol flow throughput. COV is defined in Eq. 2. COV indicates traffic fluctuation, the lower the COV, the lower the traffic burstiness, and the better the smoothness.

$$COV_p = \frac{\sqrt{\frac{1}{(t_t - t_s)/\delta} \sum_{i=1}^{(t_t - t_s)/\delta} (R_p(t_s + \delta \cdot i) - \overline{R_p})^2}}{\overline{R_p}} \tag{2}$$

The traffic smoothness is an important index to streaming media transport protocol. Fig. 2 illustrates the smoothness of TCP-SRP. The COV of heterogeneous scheme is less than Linux scheme by 27.78% on average, which indicates the traffic smoothness of heterogeneous scheme is much better. As the concurrent flow number increasing, the congestion goes heavy, the clock interrupt overhead of the traditional Linux scheme affects the smoothing performance of TCP pacing, then the advantage of heterogeneous scheme becomes obvious. Fig. 2 shows that, when concurrent flow number > 50, the COV of heterogeneous scheme is less than Linux scheme by 31.09%.

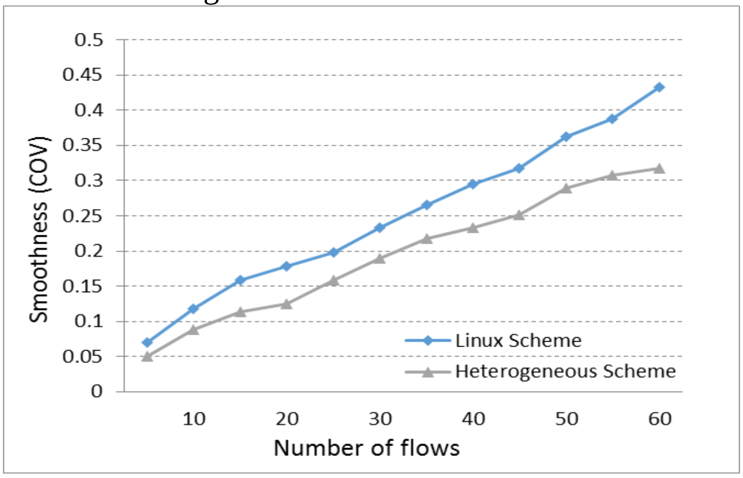

Fig. 2. Traffic smoothness.

On the other hand, the resource utilization of heterogeneous scheme is also better than the Linux scheme. Table 2 illustrates the CPU and memory overheads of the schemes on a prototype platform with 16 processing cores (10 for SE system, 6 for Linux system) and 8GB memory.

Table 2. Resource utilization.

	Heterogeneous scheme	Linux scheme
CPU utilization	Linux: 4%~11% = 7.5% SE: 90%	78%~85% = 80%
CPU overhead	6*800MHz*7.5%+(16-6)*800 MHz *90% = 7.56GHz	16*800MHz*80% = 10.24GHz
Memory utilization	Linux: 17%-33%=25% SE: 100%	100%
Memory overhead	2GB*25%+6GB*100%=6.5GB	8GB*100%=8GB

Conclusion

In order to improve the performance of TCP pacing algorithms by using multi-core NP platform, this paper proposes an implementation of TCP pacing algorithm based on multi-core heterogeneous systems. By integrating the mechanisms of prototype platform including inter-core message and Timer Ring, it provides a more accurate timing than the clock interrupt-based timing of Linux kernel, which further helps to smooth the TCP traffic bursts. Besides, the implementation is transparent to the upper layer, and it is an optional alternative to traditional TCP implementations. The experimental test based on prototype platform shows that, the smoothness performance of typical TCP pacing algorithm on the proposed implementation is improved by 27.78%, and the resource utilization is also better than the traditional scheme.

Acknowledgements

This work was financially supported in part by the National Natural Science Foundation of China (61402125).

References

[1] M. Handley, S. Floyd, J. Padhye, "TCP friendly rate control (TFRC): protocol specification," Sonstiges, vol.1, pp. 146–159 (2003)

[2] O. Volkan, R. Injong, Y. Yung, "TEAR: TCP emulation at receivers -- flow control for multimedia streaming," Dept Computer Science, pp. 1-24 (2000)

[3] Y. H. Liu, Y. Hu, G. Z. Zhang, "Study on TCP-friendly protocol in self-similar traffic network," Chinese Journal of Computers, vol. 27, no. 1, pp. 42-51 (2004)

[4] Shang Q, Wang J, Chen X. A TCP Traffic Smoothing Algorithm Based on Rate Pacing [M]// Computational Science and Its Applications – ICCSA 2016. Springer International Publishing, 2016.

[5] Shang Q, Wang J, Chen X. A Transparent Accelerating Software Architecture for Network Storage Based on Multi-core Heterogeneous Systems [M]// Computational Science and Its Applications – ICCSA 2016. Springer International Publishing, 2016.

[6] J. Zhang, F. Ren, L. Tang, "Modeling and solving TCP Incast problem in data center networks," IEEE Transactions on Parallel & Dis-tributed Systems, vol. 26, no. 2, pp. 478-491 (2015)

[7] W. Sun, T. Wen, Z. Feng, "Steady state throughput modeling of TCP NewReno," Journal of Computer Research & Development, vol. 47, no. 3, pp. 398-406 (2010)

[8] Zha Q, Zhang W, Zeng X, et al. A High Performance Multi Core Network Processing System[C]// International Conference on Computer Technology and Development, 3rd (ICCTD 2011), 2011.