eBPF 를 활용한 네트워크 지연시간 분석

Shaikhitdin Nezametdinov, 임영빈
울산과학기술원

shaikhitdin@unist.ac.kr, ybim@unist.ac.kr

Network latency analysis using eBPF

Shaikhitdin Nezametdinov, Youngbin Im
Ulsan National Institute of Science and Technology

shaikhitdin@unist.ac.kr, ybim@unist.ac.kr

Abstract

This paper introduces a new tool for endpoint delay analysis based on eBPF that provides a detailed analysis of the packet inside Linux Kernel with low overhead and expandable functionality. We verified its accuracy by comparing it with a state-of-the-art delay measurement tool, ELEMENT, and confirmed its low CPU overhead in a cloud environment. It can be easily used both in data centers and user devices for system diagnostic purposes.

Ⅰ. Introduction

Minimizing the delay of the connection is a topic that is crucial to both data center servers and users' devices. One of the significant points that cause a delay is the hosts themselves. TCP is a basis for 85% of internet traffic [3], and it has a major downside of bufferbloat[4], which may lead to the increase of latency up to seconds[2].

One of the technologies that are helpful to the measurement of delay is the extended Berkeley Packet Filter (eBPF) and BPF Compiler Collection (BCC) [7] framework that allows JIT compiling and executing own code in the kernel without any modification or recompilation of the Linux kernel. Due to some restrictions [1], such as executable code size limit and restriction of unbounded loops, it may have limiting factors; however, it does open new opportunities for thorough delay analysis.

The previous study [2] was able to estimate the delay at the host successfully, but, using the technologies mentioned above, it is possible to create a new tool to obtain precise delay values in different network layers.

Ⅱ. Endpoint latency measurement tool using eBPF

Our implemented tool can be split into two parts: the Kernel module and the Analyzer module. The kernel module is responsible for the code injected into the Linux Kernel and gets information about the sent and received messages. The information about packets is accessed using the Linux kernel's sk_buff and sock structs that store packet data and auxiliary data about the packet for upper and lower layers of network layers. The analyzer module uses the information gathered from the Kernel module and tries to find when the packet entered each layer.

The proposed tool can track the state of the connection by creating an event for each packet entry at each computer network layer. The entry points at each layer are carefully chosen based on the operation of packets in the Linux network stack [6]. The packet information is extracted by accessing IP/TCP headers or socket information directly, followed by a sanity and retransmission check by comparing with the information about previous events of the connection. The created event stores all meaningful information about the packet and is sent using BPF ringbuf [5] to the Analyzer module.

After receiving the events from the Kernel module, they are stored in a buffer to mitigate the problem of possible out-of-order events. Afterward, the events are periodically sent to the queues based on their timestamps. Each network layer is represented as two queues for sending and receiving parts for each connection. The stored events are matched by generated per-connection data sequence numbers. After the match, the gathered information is logged and is available to the user of the tool

Ⅲ. Evaluation

For the testing environment, the Linux tc tool was used as an emulator of the network. The emulator sets one side delay to 50 msec and bandwidth to 10Mbps. The resulting delay graphs are shown in Figure 1. The tool was also compared to ELEMENT [2] by running both tools concurrently for the same connection. The delay graphs are represented in Figure 2. We can observe the measured delays are close to each other.

When using Google Cloud Platform's N2 type Compute Engine VMs for the evaluation of system overhead, the CPU usage for 100Mbps connection averaged at 3.5% for both sender and receiver.

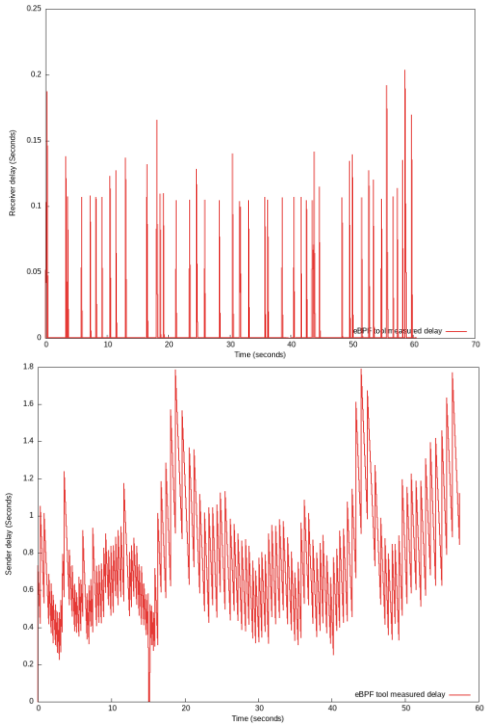

Figure 1. Delay of a connection at the receiver (upper graph) and sender (lower graph) with RTT of 100 msec and bandwidth of 10Mbps. We measure the delay between read/write syscalls and device driver entry.

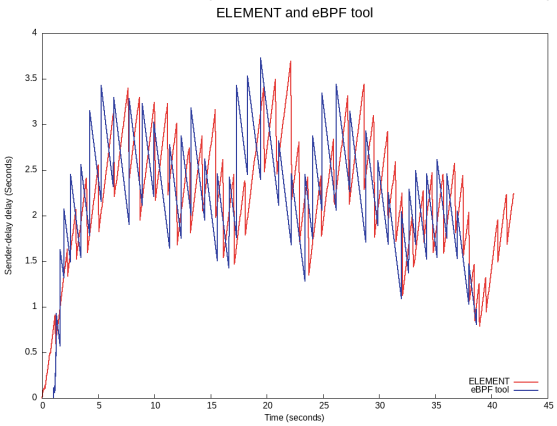

Figure 2. Measured delay comparison of ELEMENT and proposed eBPF-based measurement tool.

Ⅳ. Conclusion

The proposed tool allows measuring the endpoint delay of a TCP connection using BCC and Linux Kernel. The tool allows for tracking the delays at the whole network kernel paths in the Linux kernel by getting timestamps at each entry of network layers. By comparing the information of the different packets, the tool can detect retransmissions and useful packet metadata that help in analyzing endpoint delay. Since the developed tool is implemented in Linux Kernel, it can be easily used in data centers and user devices for system diagnostic purposes.

ACKNOWLEDGMENT

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2021-2017-0-01633) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation). This work was also supported by the Settlement Research Fund (1.190149.01) of UNIST (Ulsan National Institute of Science & Technology).

References

[1] Miano, S., Bertrone, M., Risso, F., Tumolo, M., & Bernal, M. V. (2018, June). Creating complex network services with ebpf: Experience and lessons learned. In 2018 IEEE 19th International Conference on High Performance Switching and Routing (HPSR) (pp. 1-8). IEEE.

[2] Im, Y., Rahimzadeh, P., Shouse, B., Park, S., Joe-Wong, C., Lee, K., & Ha, S. (2019, March). I sent it: Where does slow data go to wait? In Proceedings of the Fourteenth EuroSys Conference 2019 (pp. 1-15).

[3] Qian, Lei & Carpenter, Brian. (2012). A flow-based performance analysis of TCP and TCP applications. 41-45. 10.1109/ICON.2012.6506531.

[4] Gettys, J. (2011). Bufferbloat: Dark buffers in the internet. IEEE Internet Computing, 15(3), 96-96.

[5] BPF ring buffer — The Linux Kernel documentation. [Online]. Available: https://www.kernel.org/doc/html/latest/bpf/ringbuf.html

[6] Networking in the Linux Kernel. OpenWrt Wiki. [Online]. Available: https://openwrt.org/docs/guide-developer/networking/praxis

[7] BCC authors. BPF Compiler Collection (BCC). [Online]. Available: https://www.iovisor.org/technology/bcc