FlowSitter: Labeling and Tracking Simulated Flows in Cloud-based Network Testbed

Jianing Ding, Peng Zhang*, Rong Yang, Qingyun Liu, and Li Guo
Institute of Information Engineering
Chinese Academy of Science
Beijing, China 100093
{dingjianing,pengzhang,yangrong,liuqingyun,guoli}@iie.ac.cn

***Abstract*—Network testbed is of much value to researchers because of its ability to study applications running on real hosts and "somewhat real" networks. Nowadays, cloud-based network testbed is emerging as a commercial infrastructure that eliminates the need for maintaining expensive computing hardware and provides multi-tenant shared resources. The new style testbed will dynamically schedule multiple tasks on distributed computing resources and simulate "somewhat real" flows, in spite of the correlations among these flows, while the tenants are enjoying the high level of scalability and convenience. Therefore, keeping track of the identification of a certain flow for correlating with the tenant or task it belongs to becomes a crucial factor in evaluating application performance and calculating usage statistics for the correct tenant. To this end, we design FlowSitter, a flow labeling and tracking model that utilizes the active network flow watermarking technique. It labels a flow with its ownership denoted by bit-based watermark with arbitrary length and tracks it at decoder entities. FlowSitter is also designed to minimize sample size and resist various known attacks. Preliminary results show that FlowSitter can achieve a high positive rate and has desirable resilience against packet-drop attack.**

***Index Terms*—cloud computing; network testbed; multi-tenant; flow watermarking; flow tracking**

I. Introduction

Network experimentation environments that emulate some aspects of the environment-network testbeds-offers great assistance to researches of new protocols and services. In contrast to simulated environments, testbeds like Emulab[1] provide more realistic testing grounds for developing and experimenting with software. Cloud computing, the long-held dream of computing as a utility, makes network testbeds even more attractive as a service. If so, a large number of tenants will publish their simulation tasks into cloud-based network testbed, and the testbed will dynamically schedule these tasks on distributed computing resources and simulates "somewhat real" flows in spite of the correlation among these flows. That means the simulated flows in same task could be created from different nodes, which brings a challenge for evaluating simulation tasks and calculating statistics for the correct tenant. Therefore keeping flow identification tracked is of great significance for cloud-based network testbed. Here, identification could be information related with specific task and tenant that the testbed can recognize. Apparently, storing the corresponding relation between each flow and its ownership into cache or database is neither efficient nor scalable. Another method of directly padding identification into packet payload is obviously bandwidth consuming and no privacy protected, thus impracticable.

A light-weight method known as active network flow watermarking (ANFW) technique has been extensively employed in realms of stepping stones and anonymous communication systems [2]. ANFW embeds special watermarks into flows by manipulating the inter-packet delays since the randomization of delays provides the entropy. The flow-dependent nature of ANFM contributes to its strong correlation with watermarked flows. Other good characters of ANFM like invisibility, low-overhead and scalability are also desirable for cloud-based network testbed. Recent researches have employed ANFW in various realms beyond the limitation of stepping stones and anonymous systems.

Unfortunately, even the state-of-the-art ANFW systems are not suitable for cloud-based network testbed. For example, their large sample size is not efficient enough for muti-task processing and less of them combine both bit-based content and security mechanism at the same time. To address this, we proposed a cloud-based network testbed applicable ANFW scheme named FlowSitter for labeling flows with bit-based identification at watermarker entities, and then tracking the identification of flows at decoder entities with a relative small sample size. Additionally, the ANFW roots including invisibility, low-latency, scalability and resilience to attacks are also considered in designing FlowSitter. Our work is also the first that addresses identification tracking problem by leveraging ANFW technique in cloud-based network testbed.

II. overview of FlowSitter scheme

FlowSitter is an interval-based ANFW model, therefore it considers the flow as a collection of intervals of length T, with an initial time o. The i_{th} interval includes packets time sequence $P_i = \{t_1^i, t_2^i, ..., t_{CS_i}^i\}$ what we call **packets pattern** during a time period $I_i = [o + iT, o + (i + 1)T)]$, where CS_i denotes the packets number in I_i. Next we will describe our approach from three perspectives: content, position and generation. They are also three core designing philosophies of most ANFW systems.

*Corresponding author: Peng Zhang(pengzhang@iie.ac.cn)

A. watermark content

The bit information S^{bit} with length L_S is transformed from identification information S through a private character-to-bit mapping function $map()$. We also introduce the binary value of the length ofS^{bit} denoted by H with fixed length L_H and a pseudo-timestamp PTS which is the hash value of watermaker's current system time. Thus the complete bit-based content is $DS^{raw} = PTS||H||S^{bit}$, where $||$ is the string concatenation operator.

B. watermark position

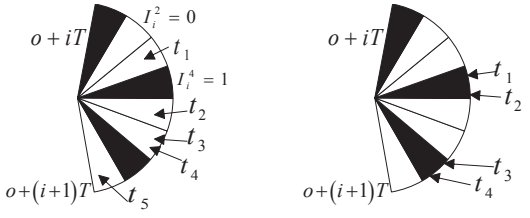
Each time interval is actually a watermark position since FlowSitter takes a time-based sample size T for one bit. This ensures the order of embedding bits into watermark positions can be determined. Sequential order is not desirable for security consideration, thus FlowSitter randomizes the watermark positions according to the following formula:

$$\{\pi_h, \pi_j\} = hash(PTS||\{h,j\}||V_h^*||K_t) mod\ \{L_H, L_S\} \quad (1)$$

where $\{h, j\}$ are indices of $\{H, S^{bit}\}$, V^* denotes the number of bits 1 in $\{H, S^{bit}\}$, and K_t is a secret key reserved in watermarker and decoder entity. One thing to note here is that decoding of H must be done before S's when tracking the watermark because L_S need to be computed from H. At last, the actual watermark position will not keep sequential order of the bit stream but be permuted depending on the watermark itself.

C. watermark generation

When the watermark content and position are determined, we need to decide how to represent binary bit as inter-packet delays. FlowSitter partitions each time period I_i into several slots with the similar length and allocates a "0" or "1" marker to each slot according to a bit map M that records all the markers of slots. Then FlowSitter generates the packets pattern by shifting CS_i packets into slots with right marker. If there is no simulated packet during time intervals I_i, then I_i will be considered no bit watermarked inside at decoder entities. Fig.1



(a) time sequence in I_i for $P_i = 0$(b) time sequence in I_i for $P_i = 1$

Fig. 1. packets pattern representation for bit 0 and 1

shows the corresponding relation between packets pattern and 0-1 bit. The black filled slots have the marker "1" and the white ones have the marker "0". Packets pattern shown in Fig.1(a) with all packets falling within the "0" slots then indicates bit '0' and the case illustrated in Fig.1(b) indicates bit '1' in the same way. Therefore, parameters involving M and slot number R should be shared in watermarker and decoder entities. Fig.2 depicts an overview of our approach from the perspective of labeling and decoding process.

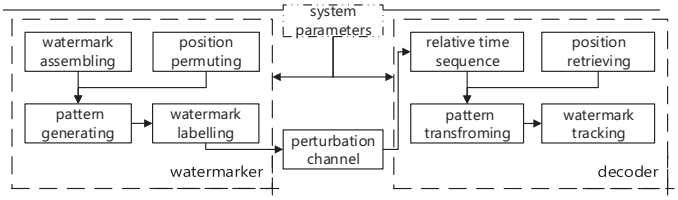
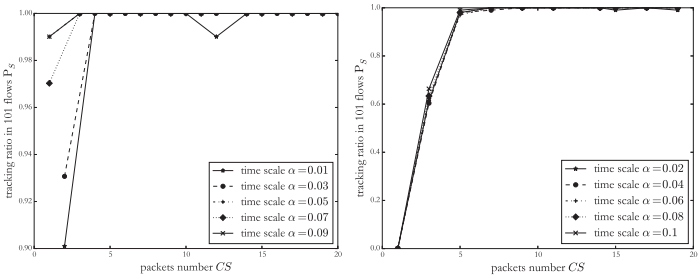


Fig. 2. Stages of labeling and decoding process

III. PRELIMINARY RESULTS

We have implemented a FlowSitter prototype and evaluated a set of combinations of two parameters $\{CS_i, \alpha\}$ by generating 101 labeled flows for each of them in both reliable and packet-drop experimental environment. α is a parameter to scale the real inter-packet delays. Fig.3(a) illustrates the positive tracking rates of 101 flows in reliable environment. As can be seen, these rates are nearly 100% as long as CS_i is greater than 4. Fig.3(b) shows FlowSitter could maintain high positive rate by increasing CS_i for more redundancy under the packet-drop condition.

We believe FlowSitter will show its capability of correlating and tracking flows in cloud-based network testbed. By the symposium, we expect to illustrate a more comprehensive analysis of FlowSitter as more emperiments are performed.



(a) tracking rates in reliable environment(b) tracking rates in 25% packet-drop environment

Fig. 3. positive tracking rates of 101 flows under different settings

ACKNOWLEDGMENT

The research work is supported by Strategic Priority Research Program of the Chinese Academy of Sciences under Grant (No.XDA06030602)and National Natural Science Foundation under Grant (No.61402464, No.61402474).

REFERENCES

[1] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An integrated experimental environment for distributed systems and networks," *Operating Systems Review*, vol. 36, no. SI, pp. 255–270, 2002.

[2] X. Wang and D. S. Reeves, "Robust correlation of encrypted attack traffic through stepping stones by manipulation of interpacket delays," in *Proceedings of the 10th ACM conference on Computer and communications security*. ACM, 2003, pp. 20–29.