

基于时间压缩的流量加速回放方法

王硕, 柏军, 王佰玲, 张旭, 刘红日

(哈尔滨工业大学(威海)计算机科学与技术学院, 山东 威海 264209)

摘要: 随着网络流量的激增, 在虚拟环境下对流量进行回放面临着许多问题。为了满足回放过程中流量完整性和准确性的需求, 提出了一种基于时间压缩的流量加速回放方法, 实现了在虚拟环境下对真实网络流量的加速回放。该方法优先将时间间隔集合中最大的时间间隔进行压缩, 对间隔较小的时间减少压缩, 从而减少时间压缩造成的单位时间流量激增现象, 进而降低流量的丢包率。从丢包率、时间间隔误差、相似度3方面对其效果进行衡量。实验表明, 与等比压缩方法相比, 该方法在相似度上表现较差, 但有着较低的丢包率和时间间隔误差。

关键词: 网络安全; 流量回放; 最大优先; 时间间隔

中图分类号: TP393

文献标识码: A

DOI: 10.11959/j.issn.2096-109x.2021082

Accelerated traffic replay method based on time compression

WANG Shuo, BAI Jun, WANG Bailing, ZHANG Xu, LIU Hongri

School of Computer Science and Technology, Harbin Institute of Technology (Weihai), Weihai 264209, China

Abstract: With the proliferation of network traffic, replaying traffic in a virtual environment faces many problems. In order to meet the requirements of traffic integrity and accuracy in the replay process, an accelerated traffic replay method based on time-compressed was proposed to achieve accelerated replay of real network traffic in a virtual environment. The method prioritized the compression of the largest time interval in the set of time intervals, and reduced the compression of the smaller intervals, thus reducing the traffic spike per unit time caused by time compression, and reducing the packet loss rate of traffic. The effect was measured in terms of packet loss rate, time interval error and similarity. The experiments show that this method has a lower packet loss rate and lower time interval error compared with the isometric compression method, although it performs worse in similarity.

Keywords: cyber security, traffic replay, maximum priority, time interval

收稿日期: 2021-03-15; 修回日期: 2021-08-31

通信作者: 刘红日, liuhr@hit.edu.cn

基金项目: 国家重点研发计划(2018YFB2004200)

Foundation Item: The National Key R&D Program of China(2018YFB2004200)

论文引用格式: 王硕, 柏军, 王佰玲, 等. 基于时间压缩的流量加速回放方法[J]. 网络与信息安全学报, 2021, 7(5): 178-188.

WAGN S, BAI J, WANG B L, et al. Accelerated traffic replay method based on time compression[J]. Chinese Journal of Network and Information Security, 2021, 7(5): 178-188.

1 引言

随着网络技术的不断发展,网络空间变得更加复杂,网络上的攻击类型不断演化,安全问题随之变得更加严峻。网络安全在所有类型的网络中扮演着至关重要的角色。即使使用多种类型的网络安全系统,攻击仍时有发生。当前,网络安全缺乏一种可以提供可控环境的实验平台,网络靶场(CR, cyber range)应运而生。

网络靶场是一类网络系统的仿真平台,服务于网络攻防、训练、网络攻防产品评测和网络新技术验证,用以提升和加强网络系统的安全性、稳定性和其他性能。网络靶场可以检验新型网络技术、进行网络攻防演练,并可以提供安全且被隔离的虚拟网络环境,可以制造出各种可能遭遇的网络攻击场景及模拟基本的用户行为来还原真实的网络环境。通过这种环境,实验人员可以验证某系统的安全性、新的攻击类型、入侵检测系统的防御能力等。各国高度重视网络靶场建设,将其作为支撑网络空间安全技术验证、网络武器试验、攻防对抗演练和网络风险评估的重要手段^[1-3]。

林肯自适应实时信息保证测试平台(LARIAT^[4])是一个开发完善的网络靶场,它被设计为可部署的测试平台,该平台通过生成后台流量,产生真实的攻击并验证成功或失败来确保信息安全。英国的 Breaking Point^[5]商用设备在网络靶场中的主要用途是模拟实际网络的流量,其能够对 OSI 模型中的 7 层都可进行配置来生成流量。该设备支持 150 多种流行的 7 层 Web 应用程序(如 Facebook 和 Google 地图)的流量生成。日本的 starBed^[6]系统主要用于评估真实场景下的新技术,该系统现已发展到了 starBed3, starBed3 将研究领域扩展到了安全性和服务质量。在网络靶场中,一项重要的技术是网络流量、服务与用户行为模拟,该技术是从用户流量行为的角度模拟真实网络环境中的流量行为。

据统计,近 5 年,全球流量增长了近三倍,从 2005 年到 2021 年将增长 127 倍。而且每天流量最大时段的增长幅度更大,在 2016 年至 2021 年,繁忙时段的互联网流量将增长 4.6 倍。到 2021

年,全球固定宽带速度(53.0 Mbit/s)将几乎达到 2016 年(27.5 Mbit/s)的两倍^[7]。流量的剧增给真实的网络环境 and 应用系统带来了很大的挑战。对于繁忙时的网络空间性能测试,可以对流量进行加速回放以应对未来忙时剧增的网络流量。如何在靶场中产生可控的流量成为一个值得研究的问题。流量的加速回放可用于软硬件产品的测压,发现系统瓶颈。因此,流量的加速回放具有很高的研究价值和广阔的应用前景。

一些成熟回放工具的加速回放功能主要通过控制带宽、发包数量和时间等。通过控制时间进行加速的方式主要是将数据包之间的时间间隔进行等比压缩。这种方法会导致原本密集(发包量大)的地方变得更密集,从而导致回放过程中产生丢包,损失回放的完整性。为此,本文提出了一种基于时间压缩的流量加速方法。通过优先将时间间隔较大的进行压缩,来实现对整体时间的压缩,该方法可以有效降低加速过程中的丢包率和时间间隔误差。

2 相关工作

网络流量生成器是网络和安全领域中的重要工具^[8-9],可以在硬件和软件平台上实现。在硬件平台更准确,可以实现更好的性能,但是,源代码封闭并且在专用的高性能硬件上实现网络流量生成的成本高^[10]。软件平台上生成流量常用的方法可分为模拟生成流量和回放已有流量。

2.1 流量生成研究

目前比较成熟的流量生成工具主要有 D-ITG、IPerf 和 Netperf。D-ITG 能够在数据包级别上生成流量,控制发送时间和数据包大小,该工具支持 IPv4 和 IPv6 流量生成,并且能够在网络层、传输层和应用程序层生成流量。D-ITG 简单地产生网络流量,通常用来测试网络设备的压力。IPerf 和 Netperf 主要通过生成数据包来衡量不同类型的网络性能,可以用于评估网络的最大吞吐量。IPerf 主要用于测量最大 TCP 和 UDP 带宽性能,它允许用户调整各种参数和 UDP 特性,并报告带宽、时延抖动、数据包丢失。Netperf 主要关注单向数据传输和 TCP 或 UDP 请求或响应性能。

还有一些学者研究基于模型的流量生成方法,他们通过分析已有流量来提出符合一定特征的流量的方法。例如, Ring^[11]等提出了一种基于流的流量生成方法,该方法首先使用 3 种不同的基于流的预处理方法,以便将流量转换成连续性数据,然后使用 GAN 来生成流量,实验表明 3 种方法中有两种方法能够产生高质量的数据。Hammer^[12]是一个能够模拟复杂和动态的网络、用户和服务器的端到端的流量模拟器,该工具主要促进产品稳定性研究。

2.2 流量回放研究

流量回放方法使用已有的真实流量,通过捕获、再生成的方式在目标网络中实现流量的复现。Parry 等^[13]设计了一个捕获并回放流量的网络取证工具,该工具能够保证回放的数据包时序与原始数据包一致。Lin 等^[14]提出了一种多端口流量回放机制,该机制支持设备自动切换在线或离线状态。设备捕获流量时处于在线状态,当出现异常流量时切换为离线状态,为了有效进行回放,文献[14]提出了一种二分搜索算法来解决由有效载荷异常和过载引起的问题。Hussain 等^[15]提出了一种将网络攻击通过可控的方式回放测试床的方法,该方法分析、检测并捕获旁路流量中的攻击流量,然后将该流量回放一台或多台机器中,用于模拟攻击者行为。Li 等^[16]设计了一种面向虚拟网络的流量回放方法,该方法首先计算虚拟网络拓扑与真实网络拓扑的相似性,从而寻找真实网络流量与虚拟网络的最佳映射方法,确保拓扑相似性最大,实验表明,该方法能够在虚拟网络中较好地完成流量回放任务。Chu 等^[17]设计了一个面向交互式网络场景再现的流速控制系统,实验测试了真实网络流量在不同网络环境中流速控制方法的效果,并从平均误差以及误差方差等方面对系统的控制性能进行了分析。

2.3 流量的加速回放研究

对于加速流量回放问题,现有的研究较少,主要的工具有 TCPCopy、GoReplay 和 Tcpreplay。上述工具除了能够对流量进行加速回放外,还可以将流量按照原始速度回放。TCPCopy 是一种请求复制工具,可以把在线流量中所有基于 TCP 的数据包引入测试系统中。该工具可以对在线服务

器应用的流量按指定倍数复制,然后放入测试服务器中,放大引流主要用于线上压力较小时,通过成倍引流达到对测试服务器进行压力测试的目的。GoReplay 通过控制请求发送的速度或指定每秒的请求数来进行加速回放。这两种加速回放与其他流媒体的方式一样,都是通过丢弃一部分信息来达到加速的效果,这种方式作用在流量上会导致一些数据包的丢失,而丢弃的数据包中可能含有重要的信息,因此存在一定的弊端。GoReplay 回放的只是 HTTP 的流量并不支持对 UDP 流量的加速。Tcpreplay 可用于编辑和重放以前捕获的网络流量,支持将重放速度修改为指定倍数,该方法通过等比控制时间间隔来实现回放速度的改变,在密集流量段实现加速时,时间间隔被压缩,密集流量段会更密集,单位时间发送的数据包过多,从而造成丢包或时间间隔增大等问题。

综上,可以发现流量生成技术和流量回放技术的研究已经取得了一定的成果。现有不少优秀的流量生成工具,如 Iperf、Netperf 和 D-ITG;也有一些关于流量回放的工具,如 TCPCopy、GoReplay 和 Tcpreplay,它们不仅有回放的功能也有加速回放的功能。

3 方法介绍

本节首先介绍流量文件预处理的方法,预处理的目的是为了去除一些与用户行为无关的数据包。然后,详细介绍流量加速回放方法的实现过程,该方法优先将最大的时间间隔进行压缩,避免了小时间间隔(密集段)的压缩,从而降低了丢包率,该方法将大根堆作为基础数据结构,提高了计算效率。

3.1 流量文件预处理

流量文件预处理主要是对流量的过滤。在流量捕获的过程中,文件中包含了一些与回放时间无关的流量。例如,路由器等网络设备之间通信产生的流量(ARP, RARP)、主机与路由器之间通信所产生的流量(ICMP, IGMP)。本文重点回放那些由用户执行网络访问操作而产生符合网络协议语义的流量,因此需要对捕获后的流量进行过滤操作,保留 IP 层的 IP 协议、传输层的 TCP 和 UDP 协议。图 1 为预处理的程序流程。

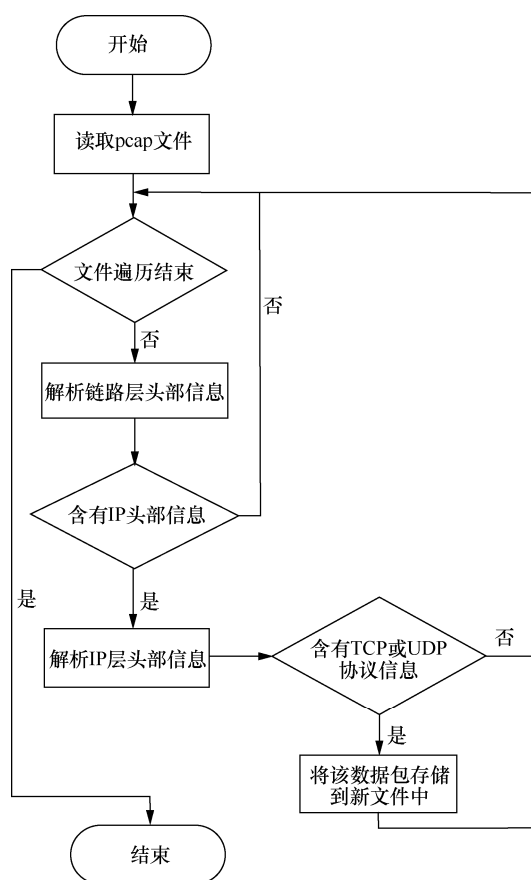


图1 预处理的程序流程
Figure 1 Flow chart of preprocessing

程序首先读取 pcap 文件，然后遍历该文件，依次获得各层的头部信息，判断链路层是否含有 IP 层的数据包，如果没有则读取下一个数据包，如果有，则继续判断 IP 层是否含有 TCP 或 UDP 协议。如果含有 TCP 或 UDP 协议，则将该数据包存储到新的文件，然后读取下一个数据包。最终程序将得到用户产生的 TCP 和 UDP 协议的流量文件。本节最终得到的文件将作为下一节的输入。

3.2 流量加速回放方法

本文的目标是实现流量的加速回放，因此需要考虑如下两个问题：如何保证流量的高准确度；如何保证数据包的较低丢包率。针对以上两个问题，本文提出了一种基于时间压缩的流量加速回放方法，通过优先压缩最大的时间间隔实现对整体的加速。该方法是通过压缩数据包之间的时间间隔来达到加速效果，而不是通过丢弃一些数据包的方式来加速，因此极大地降低了丢包率。当加速比例达到一定数值时，不可避免地会出现

某单位时间内需要发送大量的数据包的情况，这种情况便可能导致数据包丢失。而本文提出的最大优先压缩算法（MPCA，maximum priority compression algorithm）可以降低上述情况的发生概率。

最大优先压缩算法优先将最大的时间间隔进行压缩，压缩后再次寻找最大的时间间隔进行压缩，直到整体时间达到加速幅度。可以建立如下抽象模型：设原始数据包时间戳为 $T_1, T_2, T_3, T_4, \dots, T_n$ ，则所有数据包的时间间隔 t_n 可表示为

$$\begin{cases} t_n = T_n - T_{n-1}, & n \geq 2, n \in \mathbb{Z} \\ t_1 = 0, & n = 1 \end{cases} \quad (1)$$

T_1 为流量的开始时间， T_n 为流量的结束时间，原始流量所用时间则为 $T_n - T_1$ ，集合 $A = \{t_1, t_2, t_3, \dots, t_n\}$ 表示原始流量的时间间隔集合。若加速幅度为 q ($q > 1$)，则被压缩 q 倍后流量的所用时间应为 $(T_n - T_1) / q$ 。最大优先压缩算法每轮迭代将取出集合 A 中最大值 t_j ($1 \leq j \leq n$)，然后压缩 q 倍并将压缩后的值 (t_j / q) 放回到集合 A 的原来位置，该算法的目标函数如式(2)所示。经过 m ($m \leq n$) 轮迭代后即可得到压缩 q 倍后流量的时间间隔集合 A （第 m 轮迭代取出最大值后可能被压缩 q' ($q' \leq q$)，才能得目标函数成立）。

$$(T_n - T_1) - \sum_{j=1}^m \frac{t_j(q-1)}{q} - \frac{T_n - T_1}{q} = 0 \quad (2)$$

而传统的等比压缩算法（GCA，geometric compression algorithm）是通过对时间间隔等比例的压缩来实现加速效果，则压缩后的时间间隔 t'_n 可表示为

$$t'_n = \frac{t_n}{q} \quad n \geq 1, n \in \mathbb{Z} \quad (3)$$

本文提出的方法需要不断寻找集合 A 中的最大值，如果使用遍历搜索，则时间复杂度为 $O(n^2)$ ，耗时很大。为了减少计算时间，本文使用大根堆作为基础的数据结构来提高计算效率。大根堆通常是可以被看作一棵完全二叉树的数组。大根堆满足下列性质：

(1) 大根堆中某个节点的值总小于其父节点的值;

(2) 大根堆是一棵完全二叉树。

如果大根堆的根节点从数组的 1 开始编号, 则数组满足如下关系:

$$\begin{cases} k_i \geq k_{2i}, 1 \leq i \leq \lfloor \frac{n}{2} \rfloor \\ k_i \geq k_{2i+1} \end{cases} \quad (4)$$

向大根堆中插入一个元素后需要对堆结构进行调整以保持其性质, 因此插入一个数据的时间复杂度为 $O(\log n)$ 。堆顶元素为最大值, 因此寻找最大值的时间复杂度为 $O(1)$ 。删除一个数据时需要调整以保持其性质, 因此删除元素的时间复杂度为 $O(\log n)$ 。本文方法需要将堆顶元素压缩后进行堆调整, 这相当于删除了一个元素后添加一个新元素。实际操作中只需要修改堆顶元素, 然后自顶向下调整即可, 因此其时间复杂度为 $O(\log n)$ 。该程序的伪代码如算法 1 所示。

算法 1 最大优先压缩算法

输入 pcap 文件

输出 new pcap file

```

1) read pcap file
2) while IP packet is not empty do
3)   add ts, num, buf to time_intervals[count];
4) index[count], buffer[count];
5)end while
6) build Max Heap with time_intervals and index;
7) time ← time_intervals[count-1]-time_intervals[0];
8) after_time ← time;
9) while after_time > time/q do
10)   get top element from Max Heap;
11)   the element ← (top element)/q;
12)   while the element is not the leaf node
do
13)     new element ← max(element->left, element->right);
14)     if the element < new element do
15)       exchange the element ↔ new element;
```

```

16)   else
17)     break;
18)   end if
19) end while
20)   after_time ← after_time-(top element)*(q-1);
21) end while
22) generate the new file based on buffer and Max Heap
23) replay traffic according to the new file
最大优先压缩算法的流程如下:
Step 1 读取 pcap 文件, 然后获取文件中每一个数据包的时间戳、包序列和内容;
Step 2 利用 Step1 获取的信息构建大根堆;
Step 3 取出大根堆中的堆顶元素;
Step 4 将堆顶元素压缩后调整大根堆;
Step 5 计算压缩后的时间;
Step 6 判断总时间是否满足压缩后的条件, 如果不满足, 重复 Step 3~Step 5, 如果满足, 执行 Step 7;
Step 7 依照包序和对应的时间间隔以及内容生成新的 pcap 文件;
Step 8 利用 libnet 对数据包重构并按照时间间隔进行回放。
```

4 实验

本节主要验证本文提出的基于时间压缩的流量加速方法的性能以及效果。首先介绍实验条件, 然后介绍 4 个对比实验。实验 1 比较 MPCA 与 GCA 在不同倍数下的带宽变化, 实验 2 比较 MPCA 与 GCA 在不同倍数下的丢包情况, 实验 3 分析加速前后的时间间隔误差, 实验 4 分析加速后流量与原始流量的相似度。

实验表明, 在丢包率方面, MPCA 能够有效降低丢包率, 在加速幅度为 20 时, 比 GCA 减少 3.55% 的丢包率。在时间间隔误差方面, MPCA 的误差均值优于 GCA, 前者整体小于后者, 且呈现下降趋势, 后者反之。两者的标准方差比较平稳且基本一致。但在流量相似度方面, GCA 回放的流量与原始流量基本一致, 达到了 99.99% 的相似度, 而 MPCA 的相似度较低。

4.1 实验条件

本文实验的回放场景是虚拟环境,因此首先使用 OpenStack^[18]云计算平台生成虚拟拓扑和虚拟主机;然后构建虚拟机客户端(接收端)和虚拟机服务端(发送端);最后构建一个虚拟控制端用于存储数据包、处理数据以及控制流量加速回放过程。

虚拟机的配置信息如下:CPU 型号为 Intel Xeon E312xx (Sandy Bridge),内存大小为 1.8 GB,磁盘大小为 20 GB,网卡类型为 Virtio network device,操作系统为 CentOS Linux release 7.5.1804 (Core)。其中 Virtio network device 是虚拟以太网卡,它支持 TX/RX 的多队列。

本文实验回放的流量是通过 tcpdump^[24]在实验室的出口采集的。共捕获了大约 10 min 的流量。实际流量每秒发送数据包变化曲线如图 2 所示。

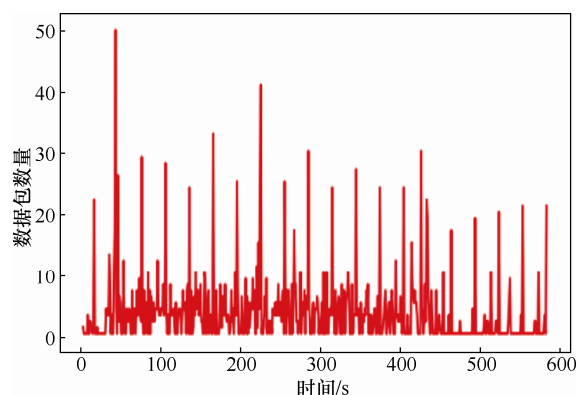


图2 10 min 流量每秒发送数量包数量
Figure 2 Number of packets sent per second for 10 minutes of traffic

在测试环境中,本文实验的流量是在实际生活中抓取的,且其网络传输带宽为 100 Mbit/s,为了使测试环境与真实环境一致,将虚拟网卡的带宽限制为 100 Mbit/s。本文实验中的加速幅度分别使用 2、4、8、10、20。

4.2 实验结果

(1) 实验 1: 带宽情况对比

实验 1 的目的是验证 MPCA 在压缩过程中对带宽带来的影响,图 3 为使用 MPCA 和 GCA 在不同加速幅度下流量的带宽。

从图 3 可以看出,在小幅度加速时,大部分较大的时间间隔被压缩,MPCA 的整体趋势与 GCA 一致,当加速倍数达到 8 倍时,GCA 出现了 3 个高峰点,而 MPCA 基本处于平稳状态,但也有几次小高

峰的出现。加速幅度达到 8 倍以后,在回放的后期,两种方法出现了不同的趋势,造成该现象的原因是原始流量的后期时间间隔较大,发送的数据包较少,MPCA 进行加速回放时候,可能进行了多次压缩,从而导致了带宽的上涨。而 GCA 按照原始比例只进行了一次压缩,故在这段时间发送的数据包总量增幅不大。当加速幅度达到 20 倍时,两种算法都出现 3 个高峰,整体上趋势相同。不同的是 GCA 的波动幅度大于 MPCA,高峰值与低峰值之差远大于 MPCA。

(2) 实验 2: 丢包情况对比

实验 2 的目的是验证加速策略和加速幅度对丢包带来的影响,图 4 为使用 MPCA 和 GCA 在不同加速幅度下回放流量的丢包率。实验表明,丢包率与繁忙程度、加速比成正相关。

从图 4 中可知,不管哪种算法,随着加速幅度的增加,丢包率都会增加,这是因为加速幅度增加需要削减两个数据包之间的时间间隔,这就导致了单位时间内数据包数量增加,而虚拟机在单位时间内可发包的数量受限,因此丢包率上升。在加速幅度相同的情况下,MPCA 在丢包率上明显优于 GCA。在小幅度加速(8 倍以下)中,MPCA 丢包率较小,相对 GCA 具有明显优势。这是因为原始流量中存在一些时间间隔大的数据包,大间隔的时间可能不止一次被压缩,经过几次压缩后便可以达到整体的加速幅度,因而对该时段中间隔短的数据包影响较小,因此丢包率较小。随着加速幅度的增加,两种算法的丢包率的差值逐渐增大,在加速幅度为 20 时,MPCA 比 GCA 减少了 3.55% 的丢包率。

(3) 实验 3: 时间间隔误差分析

为了衡量加速前后的时间间隔误差,将该实验回放的流量使用 MPCA 压缩后的时间间隔表示为 $t'_1, t'_2, t'_3, \dots, t'_n$ 。加速后回放后捕获的流量的时间间隔表示为 $t''_1, t''_2, t''_3, \dots, t''_n$ 。其中, n 表示数据包个数,利用式(5)来计算时间间隔的平均误差,利用式(5)和式(6)来计算时间间隔误差的标准方差。

$$\bar{x} = \frac{\sum_{i=1}^n |t'_i - t''_i|}{n} \quad (5)$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n}} \quad (6)$$

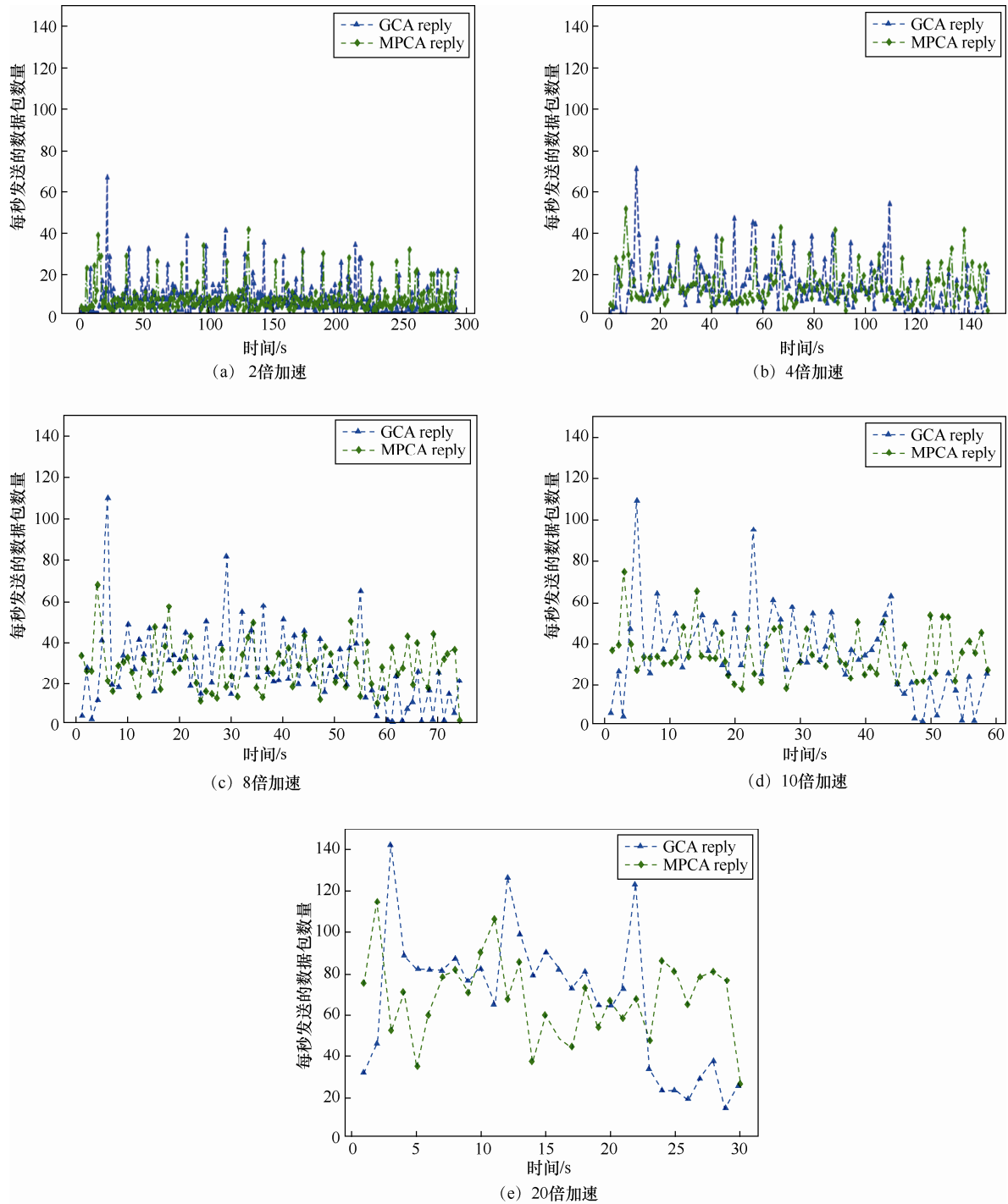


图3 使用MPCA和GCA在不同加速幅度下流量的带宽
Figure 3 Bandwidth under different acceleration amplitudes using MPC and GCA

根据式(5)得到的不同加速幅度下的误差均值如图5所示。

从图5中可知, GCA的误差均值随着加速幅度的增加缓慢增加, 而MPCA的误差均值逐渐减

小, 主要原因是当时间间隔被压缩到一定程度后, 系统将难以控制其准确性。

根据式(5)和式(6)得到不同加速幅度下两种算法的标准方差如表1所示。

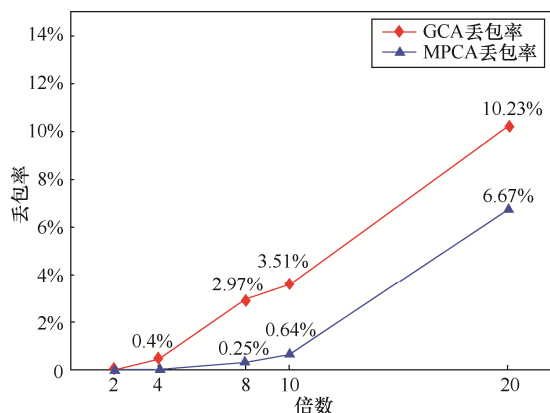


图 4 不同加速幅度下两种方法的丢包率
Figure 4 The packet loss rate of the two methods under different speedup ratios

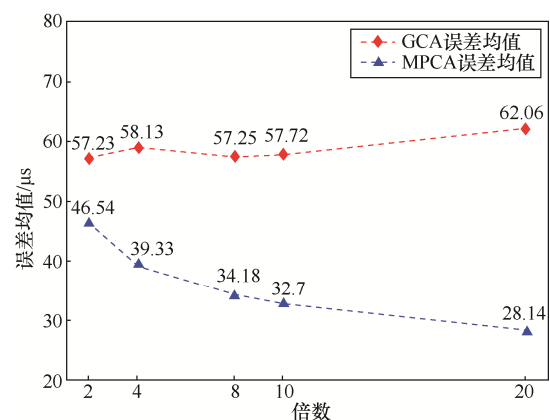


图 5 不同加速幅度下两种方法的误差均值
Figure 5 Mean error values of the two methods under different acceleration amplitudes

表 1 不同加速幅度下两种算法的标准方差
Table 1 Standard deviation of the two algorithms under different acceleration amplitudes

加速幅度	GCA/μs	MPCA/μs
2	75.51	56.91
4	51.47	52.11
8	49.74	53.91
10	48.10	50.37
20	50.41	47.00

从表 1 中可以看出 GCA 在加速幅度为 2 倍时标准方差较大, 之后稳定在 50 μs 左右, 而 MPCA 的方差基本稳定在 50 μs 左右。原因是使用 GCA 后的时间间隔波动幅度较大, 而 MPCA 的时间间隔的波动幅度较小。

(4) 实验 4: 流量回放相似性分析

实验 4 的目的是分析加速回放前后流量的相似性, 流量相似性分析主要通过内容、变化幅度

等特性来验证, 本文提出的方法是基于原始流量的加速方式, 因此在内容上是一致的。本节通过衡量变化幅度来验证回放的相似性。

不同倍数下时间间隔变化曲线如图 6 (比例尺不同) 所示。

从图 6 中可以发现, 对于 GCA, 无论加速倍数是多少, 时间间隔整体的变化趋势保持不变。对于 MPCA, 时间间隔的变化范围逐渐减小, 在小倍数下有许多间隔大于 GCA 的间隔, 这是由于该算法只压缩了少数大的时间间隔便使整体时间达到相应倍数, 因此许多小的时间间隔并没有被压缩, 故保持原样。在 1750 个数据包以后两种算法的得到的时间间隔差距最大。这是由于这些时间间隔较大, MPCA 对其压缩了不止一次。该现象充分显示了 MPCA 的正确性。

本节利用余弦相似度来衡量时间间隔的相似性。余弦相似度通过测量两个向量的夹角的余弦值来度量向量之间的相似性, 已知压缩后的时间间隔为集合 $A = \{t_1, t_2, t_3, \dots, t_n\}$, 设回放后的时间间隔为 $t'_1, t'_2, t'_3, \dots, t'_n$, 其中 n 为数据包个数。其计算方法如公式 7 所示。

$$s(t_i, t'_i) = \frac{\sum_{i=1}^n (t_i \cdot t'_i)}{\sqrt{\sum_{i=1}^n t_i^2} \cdot \sqrt{\sum_{i=1}^n t'^2_i}} \quad (7)$$

利用式 (7) 计算两种加速算法在时间间隔维度上的相似度, 结果如表 2 所示。

表 2 两种加速算法在时间间隔上的相似度
Table 2 The similarity of the two acceleration methods in the time interval

加速幅度	GCA	MPCA
2	99.9%	64.80%
4	99.9%	52.73%
8	99.9%	39.26%
10	99.9%	40.43%
20	99.9%	37.00%

从表 2 中可以得出使用 GCA 的流量与原始流量有着较高的相似性, 且不会因加速幅度的增加而下降, 基本保持在 99.9%。而 MPCA 的相似度随着加速幅度的增加, 相似度呈下降趋势。这是因为 GCA 在削减时间的过程中对每个

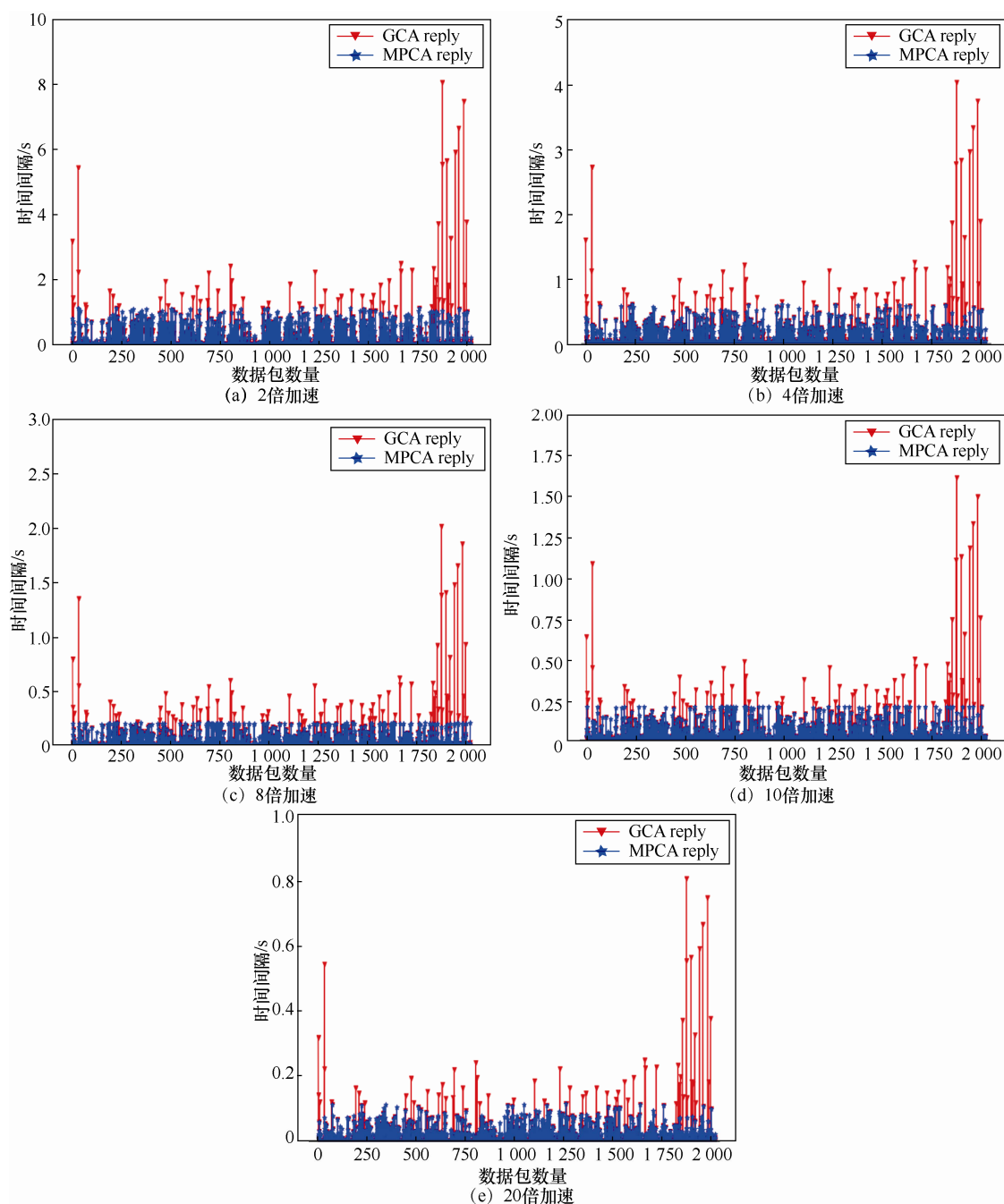


图6 不同加速幅度下的时间间隔
Figure 6 Time interval under different acceleration amplitudes

间隔都进行了等比例的压缩,从整体上保留了原始的特性,故相似性较高,而MPCA优先压缩间隔最大的时间间隔,有的时间间隔可能被压缩不止一次,有些时间间隔可能不会被压缩,故相似性低一些。因此,在相似性方面,GCA要优于MPCA。

5 结束语

现有等比压缩时间间隔加速回放流量方法随着加速幅度的增加,会造成较高的丢包率,从而损失了回放的完整性。为了解决上述问题,本文提出了一种新的基于时间压缩的流量加速

方法。通过优先将时间间隔较大的进行压缩来实现对整体时间的压缩,方法采用大跟堆作为基础数据结构,提高了计算效率,不断寻找并更新最大值和调整时间间隔集合以实现整体时间的压缩。

实验结果表明,在丢包率方面,与 GCA 相比,MPCA 可以有效减少丢包率。加速幅度越大,优势越明显。MPCA 可以避免等比压缩造成某些时间段发包量巨大而造成的丢包现象。在时间间隔方面,MPCA 的误差均值要优于 GCA,前者整体小于后者,且呈现下降趋势,后者反之。两者的标准方差都表现平稳且基本一致,但在相似度上 MPCA 表现较差。

综上,本文提出的基于时间压缩的最大优先流量加速方法能够在低丢包率和低时间间隔误差的情况下实现单机与单机之间的流量加速回放,但相似度较低。下一步,将研究在多台机器之间进行加速回放并研究如何将原始流量合理划分到多台主机进行联动回放,更逼真地还原真实网络的流量交互场景。

参考文献:

- [1] 方滨兴,贾焰,李爱平,等. 网络空间靶场技术研究[J]. 信息安全学报, 2016, 1(3): 1-9.
FANG B X, JIA Y, LI A P, et al. Cyber Ranges: state-of-the-art and research challenges[J]. Journal of Cyber Security, 2016, 1(3): 1-9.
- [2] FERGUSON B, TALL A, OLSEN D. National cyber range overview[C]//Proceedings of 2014 IEEE Military Communications Conference. 2014: 123-128
- [3] WABISZEWSKI JR M G, ANDEL T R, MULLINS B E, et al. Enhancing realistic hands-on network training in a virtual environment[C]//Proceedings of the 2009 Spring Simulation Multiconference. 2009: 1-8.
- [4] ROSSEY L M, CUNNINGHAM R K, FRIED D J, et al. LARIAT: Lincoln adaptable real-time information assurance testbed[C]//Proceedings of, IEEE Aerospace Conference. 2002: 6.
- [5] AN B. Network Testing with Simulated Traffic. Does Realism Matter[R]. 2014.
- [6] MIYACHI T, MIWA S, HASEGAWA S, et al. Hands-on environments for network technologies on StarBED[J]. Educational Technology Research, 2011, 34(1/2): 107-118.
- [7] INDEX C V N. Global mobile data traffic forecast update, 2016-2021[R]. 2017.
- [8] ANGRISANI L, BOTTA A, MIELE G, et al. Experiment-driven modeling of open-source Internet traffic generators[J]. IEEE Transactions on Instrumentation and Measurement, 2014, 63(11): 2529-2538.
- [9] MOLNÁR S, MEGYESI P, SZABÓ G. How to validate traffic generators [C]//Proceedings of 2013 IEEE International Conference on Communications Workshops (ICC). 2013: 1340-1344.
- [10] EMMERICH P, GALLENMÜLLER S, ANTICHI G, et al. Mind the gap - A comparison of software packet generators[C]//Proceedings of 2017 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS). 2017: 191-203.
- [11] RING M, SCHLÖR D, LANDES D, et al. Flow-based network traffic generation using Generative Adversarial Networks[J]. Computers & Security, 2019, 82: 156-172.
- [12] PREVEZANOS I, ANGELOU A, TSELIOS C, et al. Hammer: a real-world end-to-end network traffic simulator[C]//Proceedings of 2017 IEEE 22nd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD). 2017: 1-6.
- [13] PARRY J, HUNTER D, RADKE K, et al. A network forensics tool for precise data packet capture and replay in cyber-physical systems[C]//Proceedings of the Australasian Computer Science Week Multiconference. 2016: 1-10.
- [14] LIN Y D, LIN P C, LIN Y, et al. On-the-fly capture and replay mechanisms for multi-port network devices in operational networks[J]. IEEE Transactions on Network and Service Management, 2014, 11(2): 158-171.
- [15] HUSSAIN A, PRADKIN Y, HEIDEMANN J. Replay of malicious traffic in network testbeds[C]//Proceedings of 2013 IEEE International Conference on Technologies for Homeland Security (HST). 2013: 322-327.
- [16] LI L, HAO Z Y, ZHANG Y Z, et al. Traffic replay in virtual network based on IP-mapping[M]//Algorithms and Architectures for Parallel Processing. Cham: Springer, 2015: 697-713.
- [17] CHU W B, GUAN X H, CAI Z M, et al. Real-time volume control for interactive network traffic replay[J]. Computer Networks, 2013, 57(7): 1611-1629.
- [18] CALLEGATI F, CERRONI W, CONTOLI C, et al. Performance of Network Virtualization in cloud computing infrastructures: the OpenStack case[C]//Proceedings of 2014 IEEE 3rd International Conference on Cloud Networking (CloudNet). 2014: 132-137.

[作者简介]



王硕(1996-),男,山东东营人,哈尔滨工业大学(威海)硕士生,主要研究方向为网络安全、流量回放、用户行为模拟。



柏军（1969- ），男，辽宁鞍山人，哈尔滨工业大学（威海）副教授，主要研究方向为物联网应用技术、计算机体系结构、嵌入式系统等。



张旭（1995- ），男，安徽合肥人，哈尔滨工业大学（威海）硕士生，主要研究方向为网络安全、用户行为模拟。



王佰玲（1978- ），男，黑龙江哈尔滨人，哈尔滨工业大学（威海）教授、博士生导师，主要研究方向为工业互联网安全、信息对抗、信息安全、信息搜索、金融安全。



刘红日（1982- ），男，山东烟台人，博士，哈尔滨工业大学（威海）助理研究员，主要研究方向为网络行为模拟、工业控制系统的流量审计和网络安全互联。