互联网心跳行为测量与分析

丁 伟[1]，江淮杰[2]，李盼辉[3]

[1,2,3](东南大学 网络空间安全学院，南京 211189)

Email:hjjiang@njnet.edu.cn

摘 要： 心跳机制是保障互联网服务高可用性的常用技术之一。论文在分析讨论相关背景和研究现状的基础上，首先对心跳行为进行了分类，接着论文提出了一组针对应用层心跳流的检测规则，并设计了一个基于用户数据报协议的应用层心跳流检测算法。为验证算法的有效性，论文基于一个通用的报文分析平台，将算法部署在了 40G 带宽的中国教育和科研计算机网南京主节点网络边界，进行了连续 21 天的面向实际流量的测试，观测到了大量的心跳流，实验结果证明了算法的可用性。最后，论文讨论了基于用户数据报协议的应用层心跳流检测算法的应用范围和领域。论文的研究结果表明心跳测量可以为观测互联网提供了一个新的角度，有助于管理者更加全面、有效地管理网络。

关键词： 网络测量；应用层心跳检测；网络管理

中图法分类号：TP393 **文献标识码：**A

Measurement and Analysis of Internet Heartbeat Behavior

Ding Wei[1]，Jiang Huaijie[2]，Li Panhui[3]

[1,2,3](School of Cyber Science and Egineering, Southeast University, Nanjing 211189)

Abstract: The heartbeat mechanism is one of the commonly used techniques to ensure high availability of Internet services. Based on the analysis of the relevant background and research status, the paper first classifies the heartbeat behavior. Then the paper proposes a set of detection rules for the application layer heartbeat flow, and designs a UDP-based application layer heartbeat flow detection algorithm. In order to verify the effectiveness of the algorithm, the algorithm is deployed on the boundary of Cernet Nanjing master node network with 40G bandwidth, based on a general IP packet analysis platform. The 21-day actual traffic-oriented test is performed, and a large number of heartbeats are observed. The experimental results prove the availability of the algorithm. Finally, the paper discusses the application scope and field of UDP-based application layer heartbeat flow detection algorithm. The research results show that the heartbeat measurement can provide a new perspective for observing the Internet and help managers manage the network more comprehensively and effectively.

Key words: Network Measurement;Application Layer’s Heartbeat Detection;Network Management

1 绪论

随着越来越多的应用程序在互联网上使用，提供高可用性服务变得越来越重要。高可用性是一种技术，可以自动检测服务器节点或守护进程的故障，然后使用剩余的冗余节点自适应地重新配置系统以接管故障服务，以便保持不变的服务。为了保障应用程序的高可用性，研究者们提出了许多模型和算法[1]，心跳机制是该领域最常用技术之一。

最初设计心跳机制是为了满足一个应用的基础需求，即一旦某个进程关闭或出现故障，其他进程能及时发现并在一定的时间范围内进行相应的处理。心跳机制的基本思想是，一旦参与进程或通信信道崩溃，其他进程就会意识到这一事实并在某

丁伟，女，1962年生，博士，教授，主要研究领域为网络测量与行为学、网络管理及网络安全.E-mail: wding@njnet.edu.cn. 江淮杰（通信作者），男，1996年生，硕士研究生，主要研究领域为网络安全及网络管理.E-mail: hjjiang@njnet.edu.cn. 李盼辉，男，1994年生，硕士研究生，主要研究领域为网络安全及网络管理.E-mail: phli@njnet.edu.cn.

个时间间隔内变为非活动状态。 为此，进程定期交换称为心跳（包）的简单消息，以通知对方它们的活跃性。如果在特定时间之后没有接收到预期的心跳，则假设相应的进程已经失败或通信介质已关闭。在多个周期没有任何响应之后，则认定该进程已“死亡”[2][3]。在互联网中可以将心跳行为看成实现了心跳机制的一种网络行为。

心跳机制使用简单的规则和极小的代价，就能够长期维持一条通路的正常运转。这个机制在系统诊断、网络协议、移动计算和计算机网络中的故障检测等领域普遍存在[4]。同样的，互联网中的一些恶意行为，如 Karagiannis T 等人提出一些僵尸网络中使用了心跳机制[5]，孟磊等人提出一种基于心跳行为分析进行木马检测的方法[6]。对互联网中心跳行为的测量与分析，有助于挖掘主机间的关联关系，定位服务，进而更好地掌控网络。

2 心跳行为分类

心跳机制最早是使用在在分布式程序中，用于确保如果程序中的进程终止或失败，则程序中的其余进程也会终止。M.G. Gouda [7] 等人提出了四种心跳协议。

1）第一个协议称为二进制心跳协议。该协议涉及两个进程，没有进程可以加入或退出此协议。

2）第二种协议称为静态心跳协议。该协议涉及 n + 1 个进程，没有进程可以加入或退出此协议。

3）第三个协议称为扩展心跳协议。该协议仅以一个进程开始，其他每个进程都可以在以后加入该协议，但没有进程可以离开此协议。

4）第四个协议称为动态心跳协议。该协议仅以一个进程开始，其他每个进程都可以在以后加入该协议，加入协议的每个进程都可以在之后离开。

文献[2]对上述四种不同的心跳协议进行了严格规范的分析。这四种协议涵盖了绝大部分的心跳机制的应用场景。但在具体的网络环境中，存在最广泛的是基于二进制心跳协议的变种心跳行为。这种心跳行为广泛存在于使用长连接方式的网络应用程序的交互过程中。在不同的网络应用中，心跳行为的具体实现也不同。本文根据心跳机制实现的层次不同，将心跳分为传输层心跳和应用层心跳。

2.1 传输层心跳

传输层协议主要有 TCP 和 UDP 两种。其中 UDP 是无连接的、面向 packet 的，而 TCP 协议是有连接、面向流的协议。只有 TCP 机制中包含了心跳机制，即 KeepAlive 机制。TCP 的 KeepAlive 机制默认是关闭的，可以由上层应用打开。其基本原理是当应用设置 TCP 连接时，可以关联一组计时器，其中包含 keepAlive 定时器。当 keepAlive 定时器到达零时，会向对等方发送一个 keepAlive 探针，其中没有数据并且 ACK 标志已打开。如果收到对 keepAlive 探测的回复，则表示连接仍然正常运行；如果对等方未回复 keepAlive 探针，则表示连接可能断开了。进而当连续若干个探针都被无视了时，则认为连接已经断开。

2.2 应用层心跳

许多应用程序自身实现心跳。通常的做法是为心跳定义一种消息类型。在这种情况下，客户端以固定的周期向服务器发送一个数据包，通知服务器它仍在线并传输一些可能需要的数据。使用心跳包的典型协议是 IM，比如 QQ/MSN/飞信等协议。使用心跳机制的目的之一在于确定两个对等方之间的连通性。这里的连通性不仅仅指对等方是否存在连接，而重点关注对等方是否互相具有可用性。结合文献 RFC2498 中给出了一系列关于连通性的定义和参考文献[9]中归纳的几种基础的应用层心跳，本文将应用层心跳分为单向确认心跳和双向确认心跳。

1）单向确认心跳。此种心跳产生的原因是通信一方想要确认另一方的可用性。这种心跳根据确认方向不同又分为两种：一种是发送方发送心跳包，接收方根据收到与否来判断发送方是否可用；另一种是发送方发出心跳包，再根据接收方是否回复来判断接收方是否可用。常见的客户端-服务器模式多使用这种心跳机制。

2）双向确认心跳。此种方式下，通信双方都既是心跳发送方又是心跳接收方，心跳发送方发包，心跳接收方不回包，双方都关注对方的可用性。

3 心跳行为测量算法

3.1 应用层心跳行为测量原理

上文提到，由于心跳行为隐藏在大量的网络流量行为当中，并且没有固定的标识能够识别那些流量属于心跳流量，因此测量心跳只能基于心跳活动体现出的行为特征。

尽管不同程序的应用层心跳的实现不尽相同，但它们的原理和目的总是相似的，基本特征也应该较为相似。M.G. Gouda[7]等人结合实际工作环境，提出了心跳机制的三个主要原则：发送心跳包的速

率应该很小，以减少协议开销；检测延迟应该很小以便增加协议响应性；具有抗丢包性，即由于丢失心跳包而导致所有进程终止的概率应该很小，以便提高协议可靠性。

一个良好的心跳机制必定满足上述三条原则。本文针对这三条原则，总结出心跳行为的特征：

1）“小报文”。心跳的目的是为了维持连接，心跳包一般不带有负载或仅带有少量负载，所以心跳报文应为长度较短的小报文。

2）“低延时”。当连接的对端出现异常后，另一端需要及时的获得这一信息，因此连续两个心跳报文的发送间隔不会过长。

3）“低负载”。心跳机制不宜对正常网络连接造成过高负载，因此发送心跳包在满足低延时的同时，发送频率不会超过一定上限。

4）“连续性”。心跳行为是一个长期的存在于连接对端之间的通讯行为，因此心跳行为的特征会长期存在。

3.2 UDP心跳流检测算法

由于心跳的特点之一是对丢包的高容忍，面向无连接的 UDP 更适合于对应用层心跳的支持。基于这个原因，同时因为 IP 报文使用的协议可以在报文头部的固定字段准确识别，因此我们首先选择基于 UDP 协议的应用层心跳作为检测目标。

在网络应用中，心跳报文一般由客户端向服务端的固定端口发送。UDP 协议是没有连接的，基于 UDP 的网络应用只能保证服务端的接收端口固定，而客户端的发送端口可能在应用运行过程中发生改变。因此本文将一个 UDP 连接，定义为一客户端 IP、一个服务端的 IP 与端口，用一个<客户端 IP，服务端 IP，服务端端口>三元组来表示。而对于一条满足上述条件的 UDP 流，用<客户端 IP，服务端 IP，服务端端口，起始时间，终止时间>五元组来表示。

基于上述特征，本文提出基于 UDP 协议的应用层心跳流的检测思路：

首先，对所有的报文按照<客户端 IP，服务端 IP，服务端端口>进行组流，在组流过程淘汰掉不满足下面规则的 UDP 流：

条件 1：参与组流的报文，长度小于 M 字节。

条件 2：组流时，时间上相邻 2 个报文的时间间隔在 t s 内。

然后，针对通过条件 1 和条件 2 的每个 UDP 流，按如下条件进行筛选：

条件 3：UDP 流的持续时间至少为 T s。

条件 4：报文的流速，不超过 P pps。

最后，将满足条件 3 和条件 4 的 UDP 流存储并分析。

下面对算法的输入，输出，数据结构以及核心算法实现进行介绍。

算法 UDP 心跳流检测算法

算法目标：从 UDP 流量中找出存在心跳行为的流。

算法参数：

M—心跳流中 UDP 心跳流报文的最大长度

t—心跳流中相邻两个报文的最大时间间隔

T—心跳流的最短持续时间

P—心跳流的最大报文速率

UDP_Packets —检测窗口内的 UDP 流量

算法流程：组流，检测。

组流是 UDP 心跳流检测算法的第一步。对 UDP 报文进行长时间粒度的组流，需要有长期的报文记录，而对报文进行长期记录，势必会对存储器产生极大的存储压力。减小组流的时间粒度可以缓解这一问题，但是使用较小时间粒度的组流，又会丢失许多流量特征。因此本章结合 UDP 心跳流检测规则，使用实时组流的方式来减小存储压力。

本文算法构建了一个流记录结构，用来表示一个 UDP 流。结构体如下：

UDP 流 {

开始时间，

结束时间，

报文总数

}

算法组流程序为：检测算法读取到一条 UDP 报文后，判断其长度是否小于 M。若是，则检索流记录集合，查看该报文对应的流记录是否已经建立。如果没有建立，则建立对应的初始流记录，初始的流记录中开始时间和结束时间都设置为第一条报文的时间戳，报文总数设置为 1。如果已经建立对应的流记录，首先比较结束时间与当前报文的时间戳，若差值超过 t，则认为原 UDP 网络流已结束，并创建新的流记录加入集合，否则更新原流记录的结束时间和报文总数。使用这种方法，可以大大降低数据存储的压力。

算法的第二步程序是检测程序，即对当前组好的所有流记录按照检测规则进行检测，检测规则

为：

结束时间 – 开始时间 > T && 报文总数 /（结束时间-开始时间）< P

当满足上述规则后，则判定该流记录对应的三元组对端组之间存在心跳行为。

算法输出：心跳流 {

心跳报文发送方 IP，

心跳报文接收方 IP，

心跳报文接收方端口，

结束时间时间戳

}

算法的总体流程如图 1 所示。

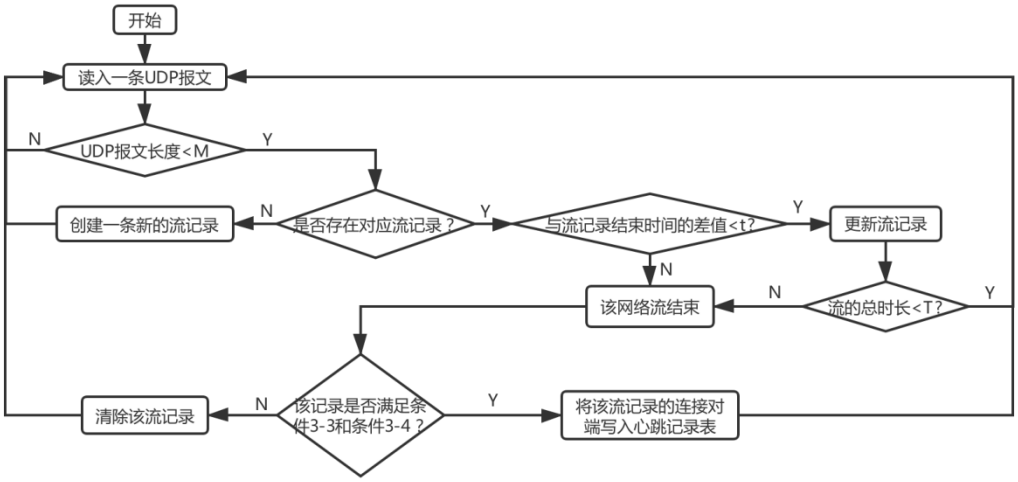


图 1 UDP 心跳流检测流程

Fig. 1 Process of UDP-based heartbeat flow detection

3.3 组合心跳流

论文使将上述算法检测出的心跳流称为原始心跳流。具体有：

定义 1. 原始心跳流. 原始心跳流是一个五元组<srcIP，dstIP，dstPort，startTime，endTime>，表示发送方地址与接收方地址的接收方端口间在开始时间和结束时间内存在心跳。

UDP 心跳流检测算法的最终结果就是一个包含若干原始心跳流的集合。

对于一个时间段 T_duration，使用 [T_begin, T_end] 区间表示。将存在时间与 T_duration 存在重合的所有原始心跳流，即满足 startTime < T_end && endTime > T_begin 的原始心跳流，按照< srcIP，dstIP，dstPort >进行合并，即可得组合心跳流。由此可得组合心跳流的定义 2。

定义 2. 组合心跳流. 组合心跳流是一个三元组<srcIP，dstIP，dstPort>，表示发送方地址与接收方地址的接收方端口间在时间段 T_duration 内存在心跳。

4 算法实现和实验结果

4.1 实验环境

为了测试算法的有效性，我们实现了上述算法，并基于 Violet 报文分析平台，将其部署于 CERNET 南京主节点网络边界，对流经 CERNET 主干网与 CERNET 南京主节点网络边界的 UDP 流量中的心跳行为进行测量，并对结果进行了分析。测量过程中将以南京主节点网内的终端作为客户端，以网外的终端为服务端的 UDP 心跳流进行检测。实验时间从 2019 年 3 月 1 日开始，持续时间为 21 天。

4.2 算法参数选择

本论文的 UDP 心跳流检测主要用于体现主机间的关联性，而短期心跳行为无法主机间关联，因此将心跳流的最短持续时间 T 设置为 2400 秒。根据相关研究[9-14]和工程实现[15-16]显示，大部分的心跳报文间隔设置在 1 秒到 90 秒之间，因此将相邻两个心跳报文的最大时间间隔 t 设置为 120 秒，心跳流的最大报文速率 P 设置为 1 pps。

对于 UDP 心跳流报文长度的阈值，根据前述分析可知心跳报文的长度一般不会很长。本论文基

于 Violet 平台，对流经南京主节点网络边界的 UDP 报文按报文长度进行了统计。统计结果显示，在 60 分钟的统计期中，总计读入 2301298163 条 UDP 报文。按长度对报文分组统计，每一百字节为一组，统计结果如图 2 所示。可以看出 UDP 短报文的长度大多不超过 200 字节，因此将 UDP 心跳流报文长度的阈值 M 设置为 200 字节，这是一个相对保守的设置。

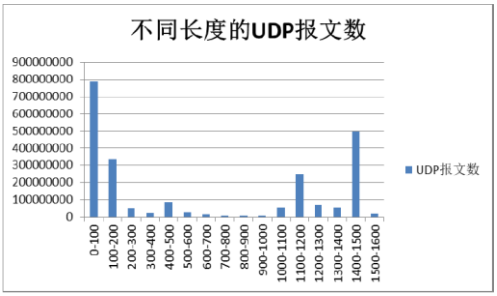


图 2　60 分钟内不同长度的 UDP 报文数

Fig. 2 Count of UDP packets of different lengths in 60 minutes

此外为了提高检测效率，检测时去除了常见 UDP 应用端口的服务器流量，包括（源端口为）53 端口（DNS 协议），123 端口（NTP 协议）以及 161 端口（SNMP 协议）。综上所述所有的参数设置，如下表 1 所示。

表 1　UDP 心跳流检测算法参数

Table 1 Parameters of UDP heartbeat flow detection algorithm

参数	参数含义	取值
M	UDP 心跳流报文的最大长度	200 bytes
t	相邻两个心跳报文的最大时间间隔	120 s
T	心跳流的最短持续时间	2400 s
P	心跳流的最大报文速率	1 pps

4.3 测量结果

图 3 给出了试验期间内以日为单位的检出结果的统计。从该图可以看出，在这 21 天的检测期内，每日检出 UDP 原始心跳流数在 100 万到 150 万之间。合并后得到组合心跳流流数在 50 万到 60 万之间。这表明 UDP 心跳流普遍存在于网络之中。

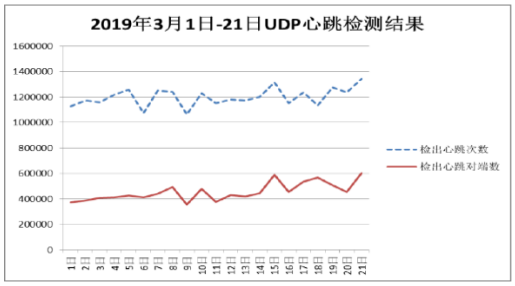


图 3　2019 年 3 月 1 日-21 日 UDP 原始心跳流检测结果

Fig. 3 Results of UDP original heartbeat flow detection from March 1st to 21st, 2019

4.4 检测结果分析

表 2 是 2019 年 3 月 1 日至 3 月 10 日检出的心跳数据，按照心跳接收方的接收端口统计每日检出次数前 6 名的端口号。可以看出在检测期内，每天检出的排名靠前的接收端口基本保持不变。对其中部分数据利用 Filter 子系统进行抓包分析，同时查询相关资料，发现排名靠前的端口，大都是常用的 UDP 网络应用所用端口，比如 17788 端口为 PPS 网络电视，8000 端口为 QQ，3478 端口为流媒体平台 Freecast 等，而其他端口 25607、3478 和 9617 等也是某些著名云服务器常用端口。相同的网络应用，其心跳接收端的端口号一般也相同，如果某个接收端口检出次数较多，则表明该应用在网络中较为活跃。由此，可以基于心跳接收端口，来对 UDP 心跳流进行区分，使用相同接收端口的心跳一般是相同的应用。

表 2　3 月 1 日-10 日检出 UDP 原始心跳流数 TOP6 的端口

Table 2 The ports of TOP6 number of UDP original heartbeat flow detected on March 1st to 10th

日期/排名	1	2	3	4	5	6
3 月 1 日	17788	8000	3478	25607	30013	9617
3 月 2 日	17788	8000	30013	8999	3478	25607
3 月 3 日	17788	8000	30013	3478	25607	3544
3 月 4 日	17788	8000	25607	9617	3478	30013
3 月 5 日	17788	8000	3478	25607	30013	9617
3 月 6 日	17788	8000	3478	25607	30013	9617
3 月 7 日	17788	8000	25607	30013	9617	3478

3 月 8 日	17788	8000	25607	30013	9617	3478
3 月 9 日	17788	8000	30013	8999	9617	12345
3 月 10 日	17788	8000	9617	8999	3478	12345

4.5 案例分析

经过观察，在测试期间 8053 端口的检出的 UDP 心跳流行为数量适中且稳定，因此选择对接收端端口为 8053 的 UDP 心跳流数据进行分析。表 3 是对 2019 年 3 月 1 日至 3 月 7 日间接收端端口为 8053 的 UDP 心跳流检测结果的统计结果。按原始心跳流和组合心跳流分别给出。可以看出在 3 月 1 日到 3 月 7 日之间，对以 8053 端口为接收端口的 UDP 心跳流的检出数据相对稳定。大部分的心跳发送方（客户端）会向多个接收方（服务器）发送心跳包。使用 8053 端口作为 UDP 心跳流接收端口的网络应用的拓扑结构呈现出网状结构，这是一种常见的多服务器网络应用。

表 3　3 月 1 日-7 日检出 8053 端口 UDP 心跳流数据

Table 3 UDP heartbeat data detected on 8053 port on March 1st to 7th

日期	1日	2日	3日	4日	5日	6日	7日
UDP原始心跳流数	6158	6857	6492	6303	5924	5979	5635
UDP组合心跳流数	587	551	587	691	654	621	605
客户端地址数	299	302	316	314	326	321	293
服务器地址数	47	53	50	50	50	56	50

再基于心跳接收方 IP 地址进行分组统计，检出次数前十的接收方 IP 地址如表 4。

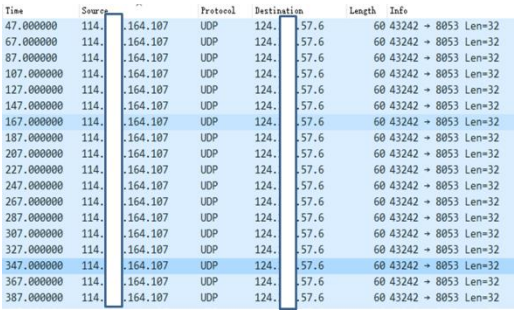
表 3　8053 端口 UDP 心跳流接收端地址

Table 4 Addresses of receivers of UDP heartbeat flow on port 8053

接收方 IP 地址	检出次数
120.**.44.88	4140
120.**.142.94	4038
120.**.138.214	4033
120.**.158.129	3629
124.**.57.6	1903
124.**.57.15	1679
124.**.57.12	1607
124.**.57.11	1548
124.**.57.14	1548
124.**.57.10	1362

可看出 8053 端口 UDP 心跳流的接收端 IP 基本处在 120.**.0.0 和 124.**.57.0 网段。从两个网段各选出检出次数最高的 IP 地址 120.**.44.88，和 124.**.57.6。进一步利用 Violet 平台采集流向 120.**.44.88 和 124.**.57.6 的 8053 端口的 UDP 报文。

图 4 是基于 Violet 的 Filter 子系统对流向 124.**.57.6 的 8053 端口的 UDP 报文的抓包记录。可以看到网内地址为 114.**.164.107 地址主机，向 124.**.57.6 的 8053 端口以 20 秒为周期，发送长度为 32 字节的 UDP 报文。图 5 是这些 UDP 报文的负载部分，可以明显看出这些报文负载的前 14 个字节相同，而第 15 个字节呈现后一个报文比前一个报文递增 20 的特征。这与其报文的发送时间间隔相同，因此该组报文为一组非常典型的由应用程序自定义的心跳报文。



Time	Source	Protocol	Destination	Length	Info
47.000000	114.**.164.107	UDP	124.**.57.6	60	43242 → 8053 Len=32
67.000000	114.**.164.107	UDP	124.**.57.6	60	43242 → 8053 Len=32
87.000000	114.**.164.107	UDP	124.**.57.6	60	43242 → 8053 Len=32
107.000000	114.**.164.107	UDP	124.**.57.6	60	43242 → 8053 Len=32
127.000000	114.**.164.107	UDP	124.**.57.6	60	43242 → 8053 Len=32
147.000000	114.**.164.107	UDP	124.**.57.6	60	43242 → 8053 Len=32
167.000000	114.**.164.107	UDP	124.**.57.6	60	43242 → 8053 Len=32
187.000000	114.**.164.107	UDP	124.**.57.6	60	43242 → 8053 Len=32
207.000000	114.**.164.107	UDP	124.**.57.6	60	43242 → 8053 Len=32
227.000000	114.**.164.107	UDP	124.**.57.6	60	43242 → 8053 Len=32
247.000000	114.**.164.107	UDP	124.**.57.6	60	43242 → 8053 Len=32
267.000000	114.**.164.107	UDP	124.**.57.6	60	43242 → 8053 Len=32
287.000000	114.**.164.107	UDP	124.**.57.6	60	43242 → 8053 Len=32
307.000000	114.**.164.107	UDP	124.**.57.6	60	43242 → 8053 Len=32
327.000000	114.**.164.107	UDP	124.**.57.6	60	43242 → 8053 Len=32
347.000000	114.**.164.107	UDP	124.**.57.6	60	43242 → 8053 Len=32
367.000000	114.**.164.107	UDP	124.**.57.6	60	43242 → 8053 Len=32
387.000000	114.**.164.107	UDP	124.**.57.6	60	43242 → 8053 Len=32

图 4　流向 124.**.57.6 的 8053 端口的 UDP 报文

Fig. 4 UDP packets flowing to the 8053 port of 124.**.57.6

```
21:31:00:20:00:00:00:00:04:22:df:f1:5c:c4:22:6a:ef:51:d2:40:9e:1a:c4:e4:e1:5c:0a:d2:f2:60:7c:8c
21:31:00:20:00:00:00:00:04:22:df:f1:5c:c4:22:7e:8a:3b:9c:ab:0d:76:e9:34:e2:af:b2:6f:da:e6:97:fb
21:31:00:20:00:00:00:00:04:22:df:f1:5c:c4:22:92:62:25:b6:0f:51:e4:ac:e4:5e:ab:77:04:3a:9b:be:1e
21:31:00:20:00:00:00:00:04:22:df:f1:5c:c4:22:a6:bb:3e:09:d6:62:71:d8:88:f3:5f:b2:fc:bb:ec:b1:12
21:31:00:20:00:00:00:00:04:22:df:f1:5c:c4:22:ba:ea:62:13:01:e7:ee:65:81:b4:20:f8:5a:2c:fc:48:20
21:31:00:20:00:00:00:00:04:22:df:f1:5c:c4:22:ce:7a:8d:e2:d6:96:f5:6a:2a:b1:55:5f:d5:8c:cb:23:77
21:31:00:20:00:00:00:00:04:22:df:f1:5c:c4:23:0a:d2:86:11:fd:d3:d3:72:e3:c8:13:13:85:13:dc:e1:9a
21:31:00:20:00:00:00:00:04:22:df:f1:5c:c4:23:1e:51:81:fc:bf:a3:5a:00:27:b3:ab:db:dd:92:78:d7:c1
```

图 5　UDP 报文负载部分

Fig. 5 Payload of UDP packet

之后通过搜索接收端地址信息，发现这些地址均为某品牌 IOT 设备的服务器地址，且其服务器所使用的端口即为 8053 端口。可以看出，若干网内客户端向该品牌 IOT 设备的服务的 8053 端口发送过大量的心跳流，这些客户端可能都安装了与该 IOT 设备服务器通讯的同一应用程序。这个结论适用于所有心跳流。

4.6 基于实验结果的思考

测试的结果一方面说明基于 UDP 的应用层心跳现象在目前的网络中普遍存在，另一方面又表明心跳测量的结果可以用于发现主机间的关联关系、定位服务，可以为管理者掌控网络提供有力的支持。例如，基于心跳测量的结果可以获得安装某款特定应用软件的 IP 的集合，一旦该款软件出现安全漏洞，网络管理者可以根据已掌握的安装地址信息，进行有针对性的响应。

5 应用范围

心跳行为广泛存在于互联网中，在保障网络应用可靠性领域发挥着巨大作用，对心跳流的测量有着重要意义。心跳流测量算法可用于服务定位、恶意网络行为检测、行为一致主机分类等领域，为网络管理和监控提供便利。

5.1 服务定位

心跳流测量算法可以发掘客户端与服务器关系，动态地定位服务 并了解服务状态。正常情况下，客户端向服务器发送心跳包，以便保持稳定的服务，心跳流测量算法检测出客户端的请求心跳包，以此分析服务端的网络地址和服务信息。当服务器退出服务时，将通知客户端不再向它请求服务，心跳流测量算法检测不到相应的心跳包，即表明该服务器上此服务已撤销。

5.2 恶意网络行为检测

某些恶意网络行为如僵尸网络、木马等会使用心跳机制，其目的一般是通知攻击者自己的存活性。心跳流测量算法可用于检测心跳流，从中分析出可能的恶意行为，并定位恶意源。

5.3 行为一致主机分类

正如前述，大部分的心跳发送方（客户端）会向多个接收方（服务器）发送心跳包，呈现出网状拓扑结构。运行同一个网络应用的一组主机的行为也往往存在关联性，心跳流测量算法可以发现这些关联的主机集合，为行为一致主机分类提供支持。

6 结论与展望

本文对互联网中常见的心跳行为进行了分类，并总结了应用层心跳行为的四个特征。在此基础上，本文设计了一个 UDP 心跳流检测算法，用于检测出 CERNET 南京主节点网络边界上经过的 UDP 心跳流，在实际环境中验证了该算法的有效性，并讨论分析了该算法的应用范围和领域。心跳机制是保障网络应用程序高可用性的常用手段，心跳行为在互联网中普遍存在。心跳机制检测算法在系统诊断、分布式计算、故障检测等领域的应用广泛，尤其是可以通过心跳行为检测来发现关联主机间的心跳行为，从而挖掘出主机间的隐秘联系，为管理者管理网络提供更强有力的支持。

References:

[1] Wang Z J , Chen W , Wang H L . Improvement on Real-Time Capability of Heartbeat Mechanism[J]. Advanced Materials Research,2010, 108-111:938-942.

[2] Atif M, Mousavi M R. Formal specification and analysis of accelerated heartbeat protocols[C]// Summer Computer Simulation Conference. 2010.

[3] Feng Y , Wei-Hua G , Zhi-Kun H U , et al. A Heartbeat Model for Communication and Control between Network Nodes[J]. Information & Control, 2008, 37(5):524-528.

[4] Aguilera M K, Chen W, Kawazoe M, et al. Heartbeat: A Timeout-Free Failure Detector for Quiescent Reliable Communication[C]// International Workshop on Distributed Algorithms. Springer-Verlag, 1997.

[5] Karagiannis T, Broido A, Faloutsos M. Transport layer identification of P2P traffic[C]//Proceedings of the 4th ACM SIGCOMM conference on Internet measurement. ACM, 2004: 121 134.

[6] Meng Lei, Liu Sheng-li, Liu Long, et al. Trojan Rapid Detection Method Based on Heartbeat Behavior Analysis [J]. Computer Engineering, 2012, 38(14):13-16.

[7] Gouda M G, Mcguire T M. Accelerated heartbeat protocols[C]// International Conference on

Distributed Computing Systems. 1998.

[8] Summary of Several Common Heartbeat Mechanisms.https://www.rainhome.org/%E6%80%BB%E7%BB%E5%87%A0%E7%A7%8D%E5%B8%B8%E8%A7%81%E7%9A%84%E5%BF%83%E8%B7%B3%E6%9C%BA%E5%88%B6/. 2019.March.

[9] Xie Bin, Gao Yang. Research on Heartbeat Mechanism of Linux High-Availability Cluster[J]. Computer Engineering and Applications, 2004(1):65 67.

[10] Hu Zhi-kun,He Duo-chang,Gui Wei-hua, et al. Remote monitoring system of rectifier based on improved heartbeat mechanism [J]. Journal of Computer Applications , 2008, 28(2):363 366.

[11] Du Bao-li . The Design and Implementation of a Cloud Service System Based on Docker [D]. 2015.

[12] Cano, Dolores M . Improving path failure detection in SCTP using adaptive heartbeat time intervals[J]. ACM SIGMETRICS Performance Evaluation Review, 2011, 39(2):50.

[13] Liu Z L Z, Sha J S J, Yang X Y X, et al. A Novel Dynamic Heartbeat Detection for Grid Membership Management[C]// International Conference on Computer Sciences & Convergence Information Technology. IEEE, 2009.

[14] Yang Guang, Zhou Jing Li, Liu Gang. Research of an Adaptive Failure Detector on iSCSI [J]. Computer Science, 2008, 35(6):90-94.

[15] MongoDB Principle: Replication Set State Synchronization Mechanism .http://www.mongoing.com/. 2019. March

[16] Analysis of Source Code of Hadoop Heartbeat Mechanism. https://www.cnblogs.com/sh425/p/6893528.html

附中文参考文献：

[6] 孟磊, 刘胜利, 刘龙, et al. 基于心跳行为分析的木马快速检测方法[J]. 计算机工程, 2012, 38(14):13-16.

[8] 总结几种常见的心跳机制. https://www.rainhome.org/%E6%80%BB%E7%BB%E5%87%A0%E7%A7%8D%E5%B8%B8%E8%A7%81%E7%9A%84%E5%BF%83%E8%B7%B3%E6%9C%BA%E5%88%B6/

[9] 谢斌,高扬. Linux 高可用集群心跳机制研究[J]. 计算机工程与应用 , 2004(1):65 67.

[10] 胡志坤,何多昌,桂卫华,et al. 基于改进心跳包机制的整流远程监控系统[J]. 计算机应用 , 2008, 28(2):363 366.

[11] 杜宝丽 . 基于 Docker 的云数据库服务系统的设计与实现 [D]. 2015.

[14] 杨光, 周敬利, 刘钢. 一种基于 iSCSI 的自适应故障检测器的研究[J]. 计算机科学, 2008, 35(6):90-94.

[15] MongoDB 原理：复制集状态同步机制 .http://www.mongoing.com/

[16] Hadoop 心跳机制源码分析 . https://www.cnblogs.com/sh425/p/6893528.html