http://bhxb.buaa.edu.cn jbuaa@buaa.edu.cn
DOI: 10.13700/j.bh.1001-5965.2021.0463

基于TCP流RTT测量的Tethering行为检测架构

戴显龙[1, 2, 3]，程光[1, 2, 3, *]，陆广垠[1, 2, 3]，金斌磊[1, 2, 3]

(1. 东南大学 网络空间安全学院，南京 211189；2. 东南大学 江苏省泛在网络安全工程研究中心，南京 211189；
3. 紫金山实验室，南京 211111)

摘　　要：Tethering行为是一种移动设备通过自身传输介质共享其互联网连接服务的行为，其不仅对移动互联网造成运营压力和收益影响，还对移动互联网隐藏其内部网络结构，造成网络安全隐患。由于Tethering存在诸多混淆和规避方法，现有Tethering行为检测技术难以有效检测。鉴于此，分析了移动互联网通信基站中，Tethering行为终端在数据流量的处理、转发等特征，以及移动互联网用户流量中传输控制协议(TCP)流往返时延(RTT)的相关特性，提出一种基于TCP流RTT测量的Tethering检测架构，构建了所提架构的测试网络环境。实验结果表明：所提架构在检测Tethering行为中具有有效性，实现了利用无监督学习和被动监测网络流量对移动互联网中Tethering行为的有效检测，对Tethering行为检测的准确率达到97.50%。

关 键 词：Tethering检测；往返时延；网络地址转换检测；无监督学习；移动互联网
中图分类号：TP393
文献标志码：A　　**文章编号**：1001-5965(2023)06-1414-10

Tethering行为是专指移动设备通过使用自身的通用串行总线(universal serial bus, USB)、蓝牙或Wi-Fi来实现共享自身互联网连接的行为[1]，其核心实质是基于网络地址转换(network address translation, NAT)技术来构建移动热点共享网络。与园区网边界提供NAT技术的高性能路由设备不同，Tethering设备在进行转发数据的同时，其自身也可进行互联网服务的使用与访问，因此，其转发与吞吐能力有限。随着大量手持终端开始配备高吞吐量无线网卡(如支持802.11ac/ax协议[2])，以及无限流量套餐的普及，在各Tethering行为的媒介中，Wi-Fi Tethering行为已最为常见。用户可通过设备自身的Wi-Fi网卡构建小型无线局域网，将其移动数据网络作为互联网连接出口，实现互联网服务的共享。

然而，Tethering行为作为一种会私自提供互联网服务的潜在行为，对移动网络运营商的运营和收益造成冲击。如今手持终端上网已经成为访问互联网的主要方式，据统计，截止2023年1月[3]，全球蜂窝移动数据连接数已高达84.6亿，超过世界人口总数，仅在中国蜂窝移动数据连接数就高达约16.9亿。而另一方面，Tethering行为可以对外隐藏内网结构，为攻击者提供非法隐匿接入互联网的便利。因此，对Tethering行为进行检测和管控有重要意义。

在现有研究中，检测Tethering行为主要基于NAT设备检测方案，即面向签名的检测方案和面向行为的检测方案。面向签名的检测方案主要是在传输控制协议(transmission control protocol, TCP)、网际协议(internet protocol, IP)、超文本传输协议(hyper text transfer protocol, HTTP)等协议字段中寻找签名或指纹实现检测，但此类方案常受限于字段修改、加密等规避行为。面向流量行为特征的识别方法主要依据NAT转发前后的流量行为特征，结合机器学习(machine learning, ML)方法进行识别，

收稿日期：2021-08-13；**录用日期**：2021-11-14；**网络出版时间**：2021-11-23 09:24
网络出版地址：kns.cnki.net/kcms/detail/11.2625.V.20211122.1609.006.html
基金项目：国家重点研发计划(2018YFB1800602)
*通信作者. E-mail：chengguang@seu.edu.cn

引用格式：戴显龙，程光，陆广垠，等. 基于TCP流RTT测量的Tethering行为检测架构[J]. 北京航空航天大学学报，2023，49(6)：1414-1423. DAI X L，CHENG G，LU G Y，et al. Tethering behavior detection architecture based on RTT measurement of TCP flows[J]. Journal of Beijing University of Aeronautics and Astronautics，2023，49(6)：1414-1423 (in Chinese).

但网络环境等差异性因素, 将直接影响模型训练的准确性。因此, 传统方案难以有效检测 Tethering 行为。

往返时延(round trip time, RTT)[4] 是指从发送方发出数据包开始, 到发送方收到来自接收方的确认(接收端收到数据后便立即发送确认消息)所经历的时间差, 是评估网络性能的一个重要指标, 也是评估网络利用率的依赖机制之一[5]。通常, RTT 不仅包括传播时延, 还包括末端处理时延、中间节点的处理时延、排队时延及转发数据时的发送时延。因此, RTT 是整个待测量网络的一个固有属性, 是无法人为修改的。

针对 Tethering 行为检测所面临的困难, 基于对移动基站中 Tethering 行为终端对数据流量的处理、转发等流量特征及移动互联网用户流量中 TCP 流 RTT 的相关特性, 以及 RTT 在待测量网络中不可修改的特性, 本文提出一种基于 TCP 流 RTT 测量的 Tethering 行为检测架构, 该架构包括预处理、无监督分析与模型训练、检测与识别三部分。其次, 本文构建了所提架构的网络测试实验环境。最后, 通过实验验证了本文架构的有效性, 实现了基于被动监测网络流量, 对移动互联网中的 Tethering 行为进行有效检测, 达到 97.50% 的准确度。

1　相关工作

1.1　传统 Tethering 行为检测方案

在现有的产品中, 思科 ASR5500 系列产品[6] 中利用包括生存时间 (time to live, TTL[7])、用户代理 (user agent, UA) 字符串[8]、操作系统指纹、域名及 TCP 同步数据包的序列号[9] 作为检测所需特征来实现对 Tethering 行为进行检测。这类传统的检测方法主要基于协议字段中类似签名或指纹的固有特殊字段来实现检测, 且这些字段通常不会因为 Tethering 行为的使用而发生变化, 并具有一定的 NAT 转发特征。此外, 面向签名的 Tethering 检测方案中, 有研究者结合应用层协议标识特征来实现识别, Zhang 等[10] 通过对比分析 NAT 转发前后的数据包, 收集 HTTP 数据并提取 cookie 标识、应用标识、用户代理信息等字段来识别不同的主机。此类方法存在一定局限性, 受限于特殊字段可篡改性、应用程序相关使用场景限制及字段加密等诸多因素。

另一方面, 面向流量行为特征的 Tethering 行为检测方法, 主要依据不同场景下产生流量的行为特征, 即区别单 IP 对应多终端(NAT/Tethering)与单 IP 对应单终端(非 NAT/非 Tethering)这 2 种场景下的不同流量行为特征, 并结合 ML 算法来有效识别。例如, Abt 等[11] 选取 NetFlow 数据的 9 个特征, 分别利用支持向量机 (support vector machines, SVM) 和 C4.5 算法实现被动识别。Gokcen 等[12] 利用朴素贝叶斯和 C4.5 算法, 根据 7 个流量特征来识别。Komarek 等[13] 提出利用 HTTP 的 8 个相关特征结合 SVM 识别。Salomonsson[14] 利用各版 Windows 更新与不同厂商杀毒软件的流量特征来构建识别模型。Khatouni 等[15] 通过对流级统计信息被动测量, 使用去除了 IP 地址、端口号和应用层信息相关属性之后的 8 个特征实现对 NAT 行为的检测, 并对 10 个不同的 ML 分类算法在该任务场景中的效果进行评估。

1.2　基于 RTT 的无线网络测量

RTT 是网络传输中固有的属性, 也是评价网络性能的一个重要指标。目前利用 RTT 属性在无线网络中的应用主要有无线网络测量、设备定位、隐私保护、异常检测等不同领域。

Ibrahim 等[16] 基于 IEEE 802.11mc 协议实现了一种基于信号飞行时间的高精度测距方法, 该方法利用时间测量协议测量出 RTT, 可实现 Wi-Fi 设备定位, 允许设备自我测量与其他 Wi-Fi 设备的距离。Han 等[17] 通过集成 RTT 和行人航位推算测量, 来估计终端的位置, 该方法可以扩展到车辆、无人机定位、压缩传感等领域。

由于缺乏集中权限, 通过 Mesh 技术组建的无线网络 (wireless Mesh network, WMN) 容易受到外部和内部攻击, 因此, Roy 和 Khan[18] 提出一种有效的切换认证协议, 在切换过程中保护随机数和转移票的隐私以防止外部攻击, 并提出基于 RTT 的检测协议来抵抗 WMN 内部的攻击。

Hou 等[19] 提出一种基于 RTT 时间序列的异常检测方法, 通过无监督机器学习方法, 首先, 将数据序列分割方面的优化问题形式化来找到大规模 RTT 时间序列中的变化点。然后, 利用其分布统计数据, 在变化点的不同侧将这些段标记为正常或异常, 从而实现异常节点的识别。

文献 [6-10] 基于流量中特殊字段的相关检测方法会受限于修改、加密等规避行为; 而文献 [11-15] 基于流量行为特征和机器学习的检测方法会受限于共享网络中终端数量及其他网络因素的制约。因此, 现有 NAT 检测技术的核心思想难以有效地检测 Tethering 行为。然而, RTT 作为网络传输中固有的属性, 作为评价网络性能的重要指标, 在网络测量方面具有一定的参考价值, 且不会因为协议字段的修改、流量的加密而产生变化。因此, 本文利用 TCP 流 RTT 的相关特性来弥补传统 NAT 检测

方案在 Tethering 行为检测上的不足。

2 Tethering 行为检测架构

通常，智能终端使用 Wi-Fi Tethering 功能的行为场景如图 1 所示，通常发生在具有 4G/5G 数据上网功能的终端设备上。设备 D 通过自身的 Wi-Fi 网卡构建小型无线局域网 V，并充当共享网络 V 的网关，将自身移动数据网络作为互联网连接出口，挂载多个终端，共享互联网服务。因此，A、B、C 的全部流量会通过 D 的打包和转发来使用外部公用网络。

由于 D 自身会产生直接访问外部网络的流量，Tethering 行为会增加设备 D 对流量的处理时间和传输时间。因此，A、B、C 的流量数据包 RTT 相较于 D 存在明显突变。由于设备的转发性能限制，TCP 流 RTT 始终存在，基于这一测度，可以有效避免传统 NAT 检测的规避行为，具有较强的适应性。

基于此，本文提出一种基于 TCP 流 RTT 测量的 Tethering 行为检测架构，如图 2 所示，本文架构包括预处理、无监督分析与模型训练、检测与识别三部分，可部署在定制的测量设备中，并挂载在移动基站上进行测量。其中，预处理阶段是对于给定的待测设备 IP 所对应的 TCP 流进行收集并逐流计算其 RTT 值，通过去噪声处理，生成最终的 RTT 样本集合；在无监督分析与模型训练阶段，对 TCP 流 RTT 样本集合进行无监督分析，进而训练生成检测模型；最后，检测与识别阶段，利用生成的检测模型对待测设备的 TCP 流进行在线检测，并实现对该设备 Tethering 行为的流量监管。

2.1 预处理

本阶段主要对移动基站中待测设备 IP 对应的所有 TCP 流进行处理，并逐流计算、存储和分析

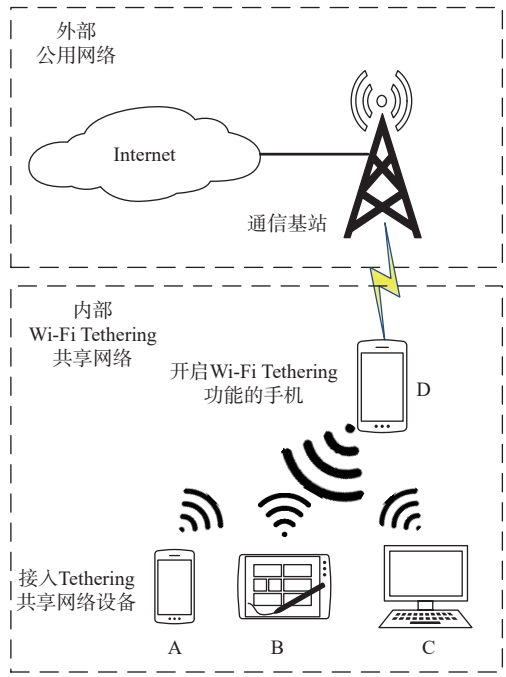

图 1 智能终端 Tethering 行为场景描述

Fig. 1 Tethering behavior scenario description for smart device

RTT 值，最终生成 RTT 样本集合。

预处理阶段的核心算法如下所示：

输入： 全流量 All_Traffic，待检测 IP 地址 IP_Detect，处理系数 k

输出： RTT 样本集合 rtt_list

```
for each tcp_packet v in All_Traffic do
    if v.dst = IP_Detect then
        if v.SYN = 1 or v.FIN = 1 then
            seqToackNUM ← v.Seq + 1
        else
            seqToackNUM ← v.Seq + len(v.payload)
        end if
        TS[seqToackNUM] ← v.ts
```

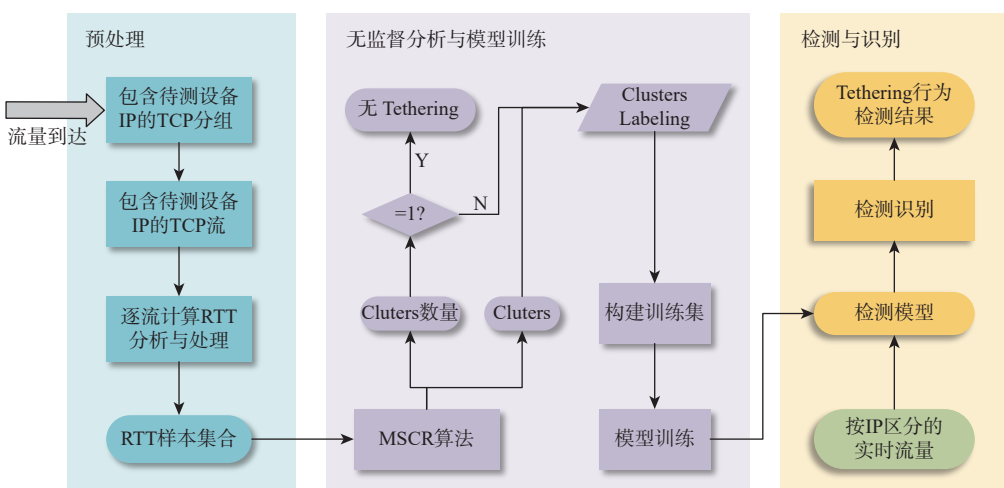

图 2 本文架构

Fig. 2 Framework of this paper

```
9      else if v.src = IP_Detect then
10         flow_rtt.append(TS[v.Ack]-v.ts)
11     end if
12     if TCP 流结束 then
13         pre_rtt.append(min(flow_rtt))
14     end if
15   end for
16   rtt_list = pre_rtt.sort()
17    rtt_list.remove(rtt for rtt in r_list if rtt.count_number ≠ 1)
18    rtt_list.remove(rtt for rtt in r_list if rtt > r_list[len(r_list)*k] )
19   return rtt_list
```

鉴于 Tethering 设备可能存在的系统卡顿、数据包排队、转发、吞吐量等影响因素, 计算不同 TCP 流 RTT 值的平均值或中位数会带入更多的环境因素影响, 因此, 选取更具有代表性的最小值。为排除相关噪声影响, 本阶段将统计所有 TCP 流的 RTT 值生成一个集合, 对集合中个数为 1 的 RTT 值进行删除, 此外, 按照处理系数 k 来保留相应数量的 RTT 值, 生成最终的样本集合。

其中 ts 表示 TCP 包到达检测点的时间戳, TS 字典用来存储 TCP 包的回复包的确认号和 TCP 包的时间戳。对于待检流量, 在给定待检测 IP 地址后, 算法的 2～8 行处理“目的 IP”为待检测 IP 的 TCP 包, 提取 TCP 包的序列号 Seq, 当 SYN 或 FIN 位置为 1 时, Seq 值加上 1, 否则 Seq 值加上 TCP 包的载荷长度, 得到 TCP 包的回复包的确认号 seq-ToackNUM, 与 TCP 包的 ts 值作为键值对存入 TS 字典中。计算方法如下:

$$\mathrm{RTT}=\begin{cases} ts_{ACK=Seq+1}-ts_{Seq} & \text{SYN=1,Ack=1或FIN=1,Ack=1} \\ ts_{ACK=Seq+payloadLen}-ts_{Seq} & \text{SYN=0且FIN=0} \end{cases} \quad (1)$$

9～11 行是对源 IP 为待检测 IP 的 TCP 包进行处理, 提取 TCP 包的确认号 Ack, 在 TS 字典中查找 Ack 值对应的 ts 值, 与 TCP 包的 ts 值的差值存入 TCP 流的 flow_rtt 列表中。

12～14 行获取 TCP 流的 RTT 值, 16～18 行是对 RTT 值进行分析预处理。考虑到 Tethering 设备一般为智能终端设备, 其转发性能远不及专用路由设备。在设备不开启 Tethering 行为时, 其产生的 RTT 值势必存在抖动。然而多数情况下, 抖动会存在于一个较为平稳的区间内, 只有当设备发热、处理数据卡顿等时, 其抖动会有明显激增, 从而导致 RTT 值异常的偏大。这些因抖动较大而产生的元素会由聚类算法单独聚为一类或多类别, 对真实的聚类结果造成一定影响。同理, 若开启 Tethering 行为, 相应挂载的设备也会因为设备自身的因素而产生个别明显的抖动。因此, 本文架构将这些异常的样本视为噪声样本。通过大量实验, 保留从小到大 85%～93% 的样本元素, 可以得到较好的聚类效果。本文架构引入系数 k 来指代这一去噪声比例。此外, 样本中个数为 1 的元素与其他元素的相关性不强, 本文视其为异常情况, 在预处理阶段也将其一起删除。基于上述预处理思路生成最终的 RTT 样本集合。

本阶段的时间复杂度应为 $O(n^2)$, 其中 n 为流数。若计算产生的 RTT 个数为 m, 则预处理去除噪声所产生的时间复杂度应为 $O(m^2)$, 因此, 本阶段的总体复杂度为 $O(n^2+m^2)$。

2.2　无监督分析与模型训练

本阶段核心任务是对预处理阶段生成的 RTT 样本集合进行无监督分析, 并训练出检测模型, 是本文架构的核心阶段。

无监督分析可以通过聚类对 RTT 样本集合作出分析, 用于区分少量挂载设备的流量和噪声流量, 对于样本中类别数量事先不明确的情况能够有效应对。基于这几个因素的考虑, 现有的无监督学习算法中, Mean shift 算法[20] 可以不依赖任何先验条件, 且原理简单、迭代效率高。其基本思想是在数据集中选定一个初始点并以该点为中心划定大小为 L 的搜索区域, 并利用指定参数带宽 b 使算法在迭代搜索过程中对 L 的大小进行确定。带宽 b 也决定了参与迭代的采样点数量及聚类的收敛速度和准确性[21]。

本文架构基于 Mean shift 算法思想, 提出一种面向 RTT 的聚类分析算法 (Mean shift clustering algorithm for RTT, MSCR), 以评估样本集合中 RTT 值的类别数量。MSCR 步骤描述如下:

输入: RTT 样本集合 rtt_list, 带宽参数 b 阈值 d
输出: 类别数量 n_clusters, 聚类中心集合 T

```
1   n ← 0
2   for each r_a in rtt_list do
3      while |shift| < d do
4         M.clear()
5            M.append(u for each u in rtt_list if distance(u_m, r_a) ≤ b)
6         num_k ← len(M)
7         C ← C.extend(M)
8         for each u in M do
9            shift ← calculate_meanshift(u, r_a)
10        end for
```

```
11        r_a.renew(r_a, shift)
12      end while
13      f = 0
14      for i from 0 to n−1 do
15        if |T[i] − r_a| < d then
16          C[i].extend(C); f = 1; break
17        end if
18      end for
19      if f = 0 then
20        T[n] ← r_a; C[n] ← C; n++
21      end if
22    end for
23    n_cluster ← n − 1
24    return n_cluster, T
```

步骤 1 对于有着 n 个 RTT 值的样本集合 rtt_list = $\{r_i\}, i \in [1,n], i \in N^+$中，随机选取一个元素 r_a 作为中心点；

步骤 2 对于给定的带宽 b，获取离中心点元素 r_a 距离不大于 b 的所有元素 (num_k 个)，记做集合 $M = \{u_m\}, m \in [1,k], m \in N^+$，并设集合 M 中的元素属于簇 C，算法第 3～5 行。

步骤 3 计算从中心点 r_a 开始到集合 M 中每个元素 u 的偏移量并相加，计算平均值，得到偏移均值 shift：

$$\text{calculate\_meanshift}(u, r_a) = \frac{1}{k}\sum_{u_m \in L_b}(u_m - r_a) \tag{2}$$

式中：L_b为以带宽 b 划定的搜索区域；u_m为搜索区域范围内的点，其中$L_b \approx M$。

步骤 4 中心点沿着 shift 的方向移动，移动距离是偏移均值的模，移动到偏移均值位置，更新中心点：

$$r_a^{t+1} = M^t + r_a^t \tag{3}$$

式中：M^t为 t 状态下求得的偏移均值；r_a^t为 t 状态下的中心，此步骤对应算法第 11 行中的 renew() 函数，即利用 t 状态下 r_a 的更新 t+1 状态下的 r_a。

步骤 5 重复步骤 2～4，直到偏移量的大小满足设定的阈值要求，记住此时的中心点。

步骤 6 重复步骤 1～5 直到所有的点都被归类。

步骤 7 分类：计算每类对每个点的访问频率，结果最大的，即为当前点集合的所属类。

步骤 8 输出类别数量 n_cluster 与聚类中心集合 T。

本阶段用到的特征样本属于一维，因此，用到的核函数维度较低，由于使用了 Ball Tree 函数对每个聚类内核成员进行了相应的查找，因此，MSCR 的复杂度为 $O(\psi * n * \log(n))$，其中，n 为样本总量，ψ 为迭代次数。

利用 MSCR 得到的类别数量 n_cluster 与聚类中心 T，对于聚类为 1 类，本文架构认为该待测设备具有 Tethering 行为，可暂停检测或重新获取 RTT 样本集合进行聚类分析；对于聚类为多类别的，则认为有 Tethering 行为，基于聚类中心集合 T 生成训练集，进而训练出相应的检测模型，以支持后续的持续检测。模型生成过程如下，其中 3～4 行是生成模型的具体流程。

输入： 聚类中心集合 T，聚类类别数量 n_cluster

输出： 训练模型

```
1  train_dataset = Empty
2  if n_cluster ≠ 1:
3    train_dataset.cluster_labeling(T)
4    model = MSCR.train(train_dataset)
5    return model
6  else:
7    return No_Tethering
8  end if
```

2.3 检测与识别

本阶段任务主要是对已经确定开启 Tethering 行为设备的流量进行持续关注，利用上一阶段生成的检测模型对该设备进行持续检测，并对该设备的 Tethering 行为流量进行统计，以达到监管的目的。具体任务如下，对于确定有 Tethering 行为的设备 IP 对应的 TCP 流进行在线统计，计算每一条 TCP 流的 RTT 值，并通过上一阶段生成的检测模型进行检测，从而实现对 Tethering 行为的持续检测与关注。

本阶段任务具体描述如下：

输入： 检测模型 model，确定 IP 对应的实时流量 R_Traffic

输出： 检测结果 result

```
1  while True:
2    for each tcp_flow in R_Traffic do
3      rtt = calculate_rtt()
4      result = MeanShfit.predict (rtt, model)
5    end for
6    return result
7  end while
```

本阶段检测与关注的对象是持续到达的流量。其中，第 3 行是对每条 TCP 流的 RTT 值进行获取，第 4 行是通过获取的 RTT 值和上一阶段获取的检测模型进行检测。对于实时流量，本阶段功能模块在按照待测 IP 的单条 TCP 流进行 RTT 值计算时所产生的时间复杂度为 $O(n)$，n 为该待测 IP 单条

TCP 流产生的 RTT 值总数。由于预测阶段仅需要根据已经生成的预测模型直接进行预测, 预测阶段的时间复杂度即为 $O(1)$。因此, 本阶段的总体时间复杂度为 $O(n)$。

3　实验验证与分析

3.1　流量数据集构建方法

3.1.1　训练集流量采集、构建方法

本文实验所用的训练集流量即为 Tethering 设备的全流量, 这对应了真实环境中 Tethering 设备在基站中产生的所有流量。由于实验硬件条件限制, 本文使用 PC 充当模拟基站并作为测量点, 进而模拟真实基站, 全流量采集示意图如图 3 所示。

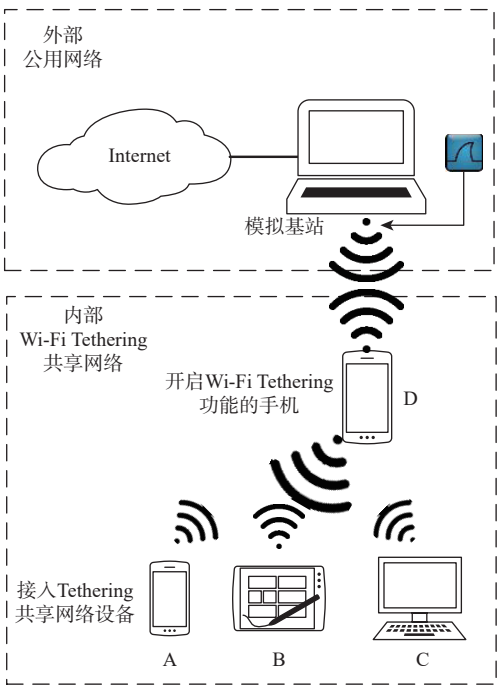

图 3　全流量采集示意图

Fig. 3　The description of the traffic measurement

流量采集具体步骤如下:

1)充当模拟基站的 PC 通过网卡 1 接入互联网, 并通过网卡 2 构建无线网络 W, 模拟手机上网所需网络。

2)设备 D 关闭移动数据上网功能, 接入无线网络 W, 同时开启 Tethering 行为, 构建共享网络 V。

3)设备 A、B、C 接入共享网络 V 中进行上网。

4)在模拟基站中对网卡 2 的全部数据流量通过 wireshark 或 tcpdump 等工具进行获取, 最终通过预处理阶段的算法构建本文实验中的训练集。

3.1.2　测试集流量采集、构建方法

测试集的建立, 旨在验证本文架构训练得到的分类模型对于该 IP 产生的流量在实时检测中的可行性与准确性, 对于测试集的采集与构建, 本文利用如图 4 所示的采集环境。

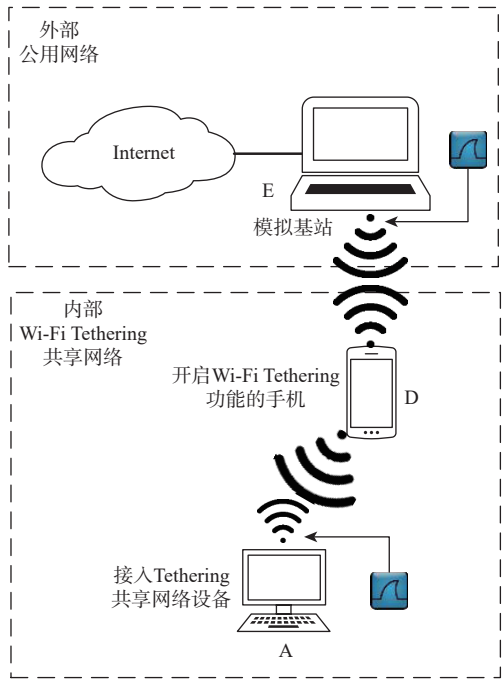

图 4　测试数据集采集与构建

Fig. 4　Collection and establishment of test dataset

具体步骤如下:

1)充当模拟基站的 PC, E 通过网卡 1 接入互联网, 并通过网卡 2 构建无线网络 W; 设备 D 接入无线网络 W, 通过 Tethering 行为构建共享网络 V; A 接入共享网络 V。

2)同时在 E 和 A 上通过 wireshark 或 tcpdump 等工具采集网络流量, E 抓取的 DataSet18 为将要分类的测试集, 包含有 D 转发后的 A 流量; 而设备 A 上抓取的 DataSet19 为其自身产生的流量, 即为 Tethering 设备转发之前的流量。

3)比对 DataSet18 和 DataSet19 中的数据包, 含有相同数据的包, 即为有 Tethering 行为数据; 剩下的则为无 Tethering 行为数据。最终生成验证使用的测试集 S。

3.2　实验环境与数据集

本文实验中, 为了验证 MSCR 的有效性, 以及尽可能接近真实基站的情况, 在硬件环境的选择上, 避免了同一设备的反复使用。数据集的采集工作, 选用性能不同的 2 台笔记本电脑作为模拟基站, 选用不同品牌的手机作为 Tethering 行为设备, 并挂载不同数量的终端。实验的硬件平台主要参数如表 1 所示。

本文实验采用的网络流量, 来源于实验模拟基站, 共计约 16.40 GB。其中包含无 Tethering 行为、

表1 实验硬件平台的主要参数

Table 1 Main parameters of experiment hardware platform

序号	设备名称	型号	主要参数
1	模拟基站(PC)	Dell Vostro 14-5480	CPU: Core i5-5200U; RAM: 4 GB; 512 GB HDD; OS: Win 8 Pro
		Terrans Force X911	CPU: Core i7-4940MX; RAM: 32 GB; 1TB SSD; OS: Win 10 Pro
2	Tethering设备	Huawei P40	CPU: Kirin 990 5G; RAM: 8 GB; OS: Android 10
		Mi10 Pro	CPU: snapdragon 865; RAM: 12 GB; OS: Android 11
		Mi11 Ultra	CPU: snapdragon 888; RAM: 12 GB; OS: Android 11
3	挂载设备(手机)	iPhone 6	CPU: Apple A8; OS: ios 12.5.4
		iPhone 8 Plus	CPU: Apple A11; OS: ios 14.6
		Redmi 7	CPU: snapdragon 632; OS: Android 9
		Mi10 Pro	CPU: snapdragon 865; OS: Android 11
		Mi11 Ultra	CPU: snapdragon 888; OS: Android 11
		Mi9	CPU: snapdragon 855; OS: Android 11
4	挂载设备(PC)	ThinkPad X1C 2016	CPU: Core i7-6600U; RAM: 8 GB; OS: Win 10 Pro

有Tethering行为(挂载不同数量上网终端)。此外,DataSet1～DataSet17为模拟基站采集的全流量,用于验证本文架构在检测Tethering行为的有效性,其中,DataSet1～DataSet11为Dell笔记本模拟基站获取,DataSet12～DataSet17为X911笔记本模拟基站获取。

DataSet18和DataSet19为生成标注行为标签测试集的数据集,为了验证本文架构训练得到的模型在新到流量中识别Tethering行为的有效性,基于本文的流量采集与处理方法,DataSet18和DataSet19构建的测试集中共有7 712个TCP流,其中有Tethering行为流4 312个,无Tethering行为流3 400个。表2为实验数据集的具体信息。

表2 实验数据集信息

Table 2 Information of datasets in the experiment

数据集	Tethering模式	手机设备型号	待测设备IP地址	大小/MB	分组个数
DataSet1	无Tethering	iphone6	192.168.137.109	1 003	1 000 000
DataSet2	无Tethering	P40	192.168.137.157	912	1 000 000
DataSet3	无Tethering	P40	192.168.137.157	412	600 000
DataSet4	TetheringD+A	P40、iphone6	192.168.137.157	973	1 000 000
DataSet5	TetheringD+A	P40、iphone6	192.168.137.157	910	1 000 000
DataSet6	TetheringD+A	P40、iphone6	192.168.137.157	900	1 000 000
DataSet7	TetheringD+A	P40、iphone6	192.168.137.157	943	1 000 000
DataSet8	TetheringD+A	P40、iphone6	192.168.137.157	466	500 000
DataSet9	TetheringD+A	P40、iphone6	192.168.137.157	439	500 000
DataSet10	TetheringD+A+B	P40、iphone6、iphone8 Plus	192.168.137.157	961	1 000 000
DataSet11	TetheringD+A+B+C	P40、iphone6、iphone8 Plus、Redmi7	192.168.137.157	1 935	2 000 000
DataSet12	无Tethering	Mi10 Pro	192.168.137.238	984	1 000 000
DataSet13	无Tethering	Mi11 Ultra	192.168.137.142	1 024	903 103
DataSet14	TetheringD+A	Mi10 Pro、Mi11 Ultra	192.168.137.238	1 136	1 200 000
DataSet15	TetheringD+A	Mi11 Ultra、Mi10 Pro	192.168.137.142	1 218	1 400 000
DataSet16	TetheringD+A+B+C	Mi10 Pro、Mi11 Ultra、Mi9、X1C	192.168.137.238	1 157	1 200 000
DataSet17	TetheringD+A+B+C	Mi11 Ultra、Mi10 Pro、Mi9、X1C	192.168.137.142	1 372	1 500 000
DataSet18	TetheringD+A	P40、X1C	192.168.137.157	859	909 785
DataSet19	无Tethering	P40	192.168.137.157	376	403 776

3.3 MSCR有效性验证

本节实验通过对每个数据集的预处理和MSCR的运行,对MSCR的有效性进行验证。首先,对本文提出的去噪声方法进行对比实验,分别用MSCR对每个数据集去噪声前、去噪声后所生成的训练集进行聚类,以DataSet12为例,图5为对比效果。

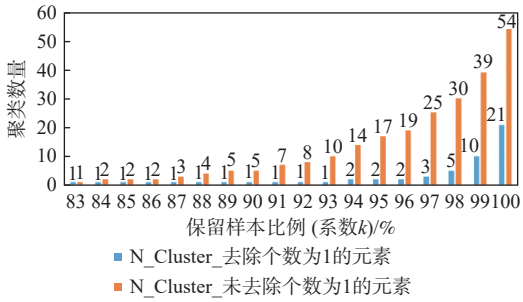

图 5　去噪声有效性举例 (DataSet12)

Fig. 5　Examples of de-noising effectiveness (DataSet12)

通过大量实验对比，本文架构可以很大程度上保证 MSCR 的聚类效果。在图 5 中，若不进行去噪声，无 Tethering 行为的设备流量会因为自身原因而被聚类成 54 类，直接会被误判为有 Tethering 行为。最终，在保留从小到大 85%～93% 的样本元素，可以得到较好的聚类效果。因此，在实验中，本文对于 RTT 样本集合生成算法的处理系数 $k = 0.9$，即保留了从小到大占样本总量 90% 的 RTT 用于生成最终的 RTT 集合，作为 MSCR 的输入。数据集实验结果如表 3 所示。

表 3　数据集实验结果

Table 3　Experimental results of data sets

数据集	Tethering模式	RTT数量	*n*_clusters
DataSet1	无Tethering	1 176	1
DataSet2	无Tethering	6 347	1
DataSet3	无Tethering	7 850	1
DataSet4	TetheringD+A	3 435	2
DataSet5	TetheringD+A	1 979	2
DataSet6	TetheringD+A	2 754	2
DataSet7	TetheringD+A	2 358	9
DataSet8	TetheringD+A	423	2
DataSet9	TetheringD+A	1 820	2
DataSet10	TetheringD+A+B	3 251	2
DataSet11	TetheringD+A+B+C	6 377	2
DataSet12	无Tethering	2 279	1
DataSet13	无Tethering	1 877	1
DataSet14	TetheringD+A	5 168	4
DataSet15	TetheringD+A	4 497	2
DataSet16	TetheringD+A+B+C	4 171	7
DataSet17	TetheringD+A+B+C	2 606	19

数据集 DataSet1～DataSet3、DataSet12 和 DataSet 13 对应了设备 D“无 Tethering 行为”的情况，从实验结果可以看出，聚类结果均仅为 1 类、中心点均为 1 个，即能准确识别出“无 Tethering 行为”。

DataSet4～ DataSet9、DataSet14 和 DataSet15 均为设备 D 挂载 1 个设备的情况。其中，DataSet4～DataSet6、DataSet8 和 DataSet9、DataSet15 的聚类结果均为 2 类，中心点为 2 个，即识别出“有 Tethering 行为”。不同的是，DataSet7 显示聚类结果为 9 类，中心点为 9 个；DataSet14 显示聚类结果为 4 类，中心点为 4 个。DataSet7 与 DataSet14 在获取时，存在设备 D 负载较大的情况。

DataSet10 为设备 D 开启 Tethering 并挂载设备 A、B 的情况，其聚类结果为 2 类，中心点为 2 个。DataSet11、DataSet16、DataSet17 均为设备 D 开启 Tethering 并挂载设备 A、B、C 的情况，其聚类结果分别为：DataSet11 为 2 类，其中心点为 2 个；DataSet16 为 7 类，其中心点为 7 个；DataSet17 为 19 类，其中心点为 19 个。

综合以上分析，对于无 Tethering 行为、挂载不同数量终端的 Tethering 行为，本文提出的 MSCR 均具有一定的有效性，实验验证了 MSCR 在对待测设备的 TCP 流 RTT 值特征进行无监督学习分析，可以有效地得到相应的结果。

3.4　Tethering 行为实时检测有效性验证

本节实验的验证方案如下：采用 DataSet4～DataSet9 作为生成训练集的原始数据集，通过预处理和无监督分析后，训练并得到检测模型，利用该模型对测试集 S 进行分类测试。

本节实验的评价指标采用准确率 Accuracy，旨在验证本文架构在确定检测结果的情况下可以继续对新到流量进行较为准确的检测。

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{4}$$

式中：TP 为正确检测的正样本数目；FN 为被错误检测的正样本数目；FP 为被错误检测为正样本的负样本数目；TN 为正确检测的负样本数目。

表 4 为本文架构所得到的分类模型对测试集的分类结果，并生成了图 6 的准确度混淆矩阵。实验结果如下：测试集中确定为 Tethering 行为的流中，共有 4 204 个 TCP 流被识别为有 Tethering 行为流，108 个被识别为无 Tethering 行为流，准确率为 97.50%；在确定为无 Tethering 行为的流中，共有 2 394 个 TCP 流被识别为无 Tethering 行为，1 006 个则被识别为有 Tethering 行为，准确率为 70.41%；生成的

表 4　DataSet18 和 DataSet19 构建测试集分类结果

Table 4　Classification results of test dataset built by DataSet18 and DataSet19

标签名称	标签总个数	分类结果		准确率/%
		Tethering	无Tethering	
Tethering	4 312	4 204	108	97.50
无Tethering	3 400	1 006	2 394	70.41

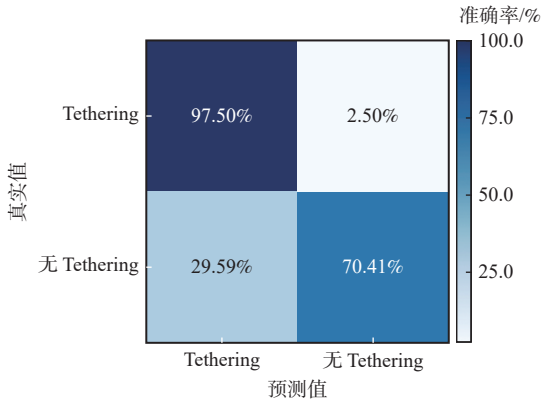

图 6 准确率混淆矩阵

Fig. 6 Accuracy confusion matrix

模型对整个测试集的识别平均准确率为 85.6%。

实验结果验证了本文架构在确定设备有 Tethering 行为之后生成的模型，可以有效对该设备进行实时分析和检测，尤其是对具有 Tethering 行为的流量有较高的检测准确率。

4 结 论

本文通过分析移动互联网通信基站中 Tethering 行为终端在数据流量的处理、转发等流量特征，以及移动互联网用户流量中 TCP 流 RTT 的相关特性，得出以下结论：

1）本文提出一种基于 TCP 流 RTT 的 Tethering 行为检测架构，该架构利用所提 MSCR 可以实现基于无监督学习方法对 Tethering 行为进行识别。

2）通过实验，验证了 MSCR 在 Tethering 行为检测中的可行性，其中，无 Tethering 行为对应的聚类结果为 1 类，有 Tethering 行为对应的聚类结果为多类。

3）本文构建了测试网络环境，实验结果验证了本文架构在检测 Tethering 行为中的有效性，实现了利用无监督学习和被动监测网络流量对移动互联网中的 Tethering 行为进行有效检测。

4）通过实验，实现了对 4 312 个确定为有 Tethering 行为的数据和 3 400 个确定为无 Tethering 行为的测试集进行准确分类，其平均准确率为 85.6%。

5）本文验证了 RTT 这一测度在检测 Tethering 行为上的有效性，也验证了 RTT 测度作为固有属性可以有效防止规避检测的行为。未来的工作中，将通过研究增加其他测度来提高检测精度。

参考文献（References）

[1] WIKI. Tethering[EB/OL]. (2020-03-08)[2021-03-21]. https://en.wikipedia.org/wiki/Tethering.

[2] CHOI J. Detection of misconfigured Wi-Fi tethering in managed networks[J/OL]. Preprints, 2020, (2020-03-08) [2021-03-21]. https://www.preprints.org/manuscript/202002.0189/v1. DOI: 10.20944/PREPRINTS202002.0189.V1.

[3] We Are Social. Digital 2023: China[EB/OL]. (2021-02-09) [2021-03-21].https://datareportal.com/reports/digital-2023-China.

[4] 胡治国, 田春岐, 杜亮, 等. IP网络性能测量研究现状和进展[J]. 软件学报, 2017, 28(1): 105-134.
HU Z G, TIAN C Q, DU L, et al. Current research and future perspective on IP network performance measurement[J]. Journal of Software, 2017, 28(1): 105-134(in Chinese).

[5] DALAL P, SARKAR M, KOTHARI N, et al. Refining TCP's RTT dependent mechanism by utilizing link retransmission delay measurement in wireless LAN[J]. International Journal of Communication Systems, 2017, 30(5): 1-20.

[6] CISCO. Cisco ASR 5000 ECS Administration Guide StarOS Release 21.18[EB/OL]. (2020-10-05)[2021-03-21]. https://www.cisco.com/c/en/us/td/docs/wireless/asr_5000/21-18_6-12/ECS-Admin/21-18-ECS-Admin/21-17-ECS-Admin_chapter_011000.html#id_39377.

[7] STRAKA K, MANES G. Passive detection of nat routers and client counting[C]//Advances in Digital Forensics II: IFIP international Conference on Digital Forensics. Berlin: Springer , 2006: 239-246.

[8] MAIER G, SCHNEIDER F, FELDMANN A. NAT usage in residential broadband networks[C]//International Conference on Passive and Active Network Measurement. Berlin: Springer, 2011: 32-41.

[9] PARK H, SHIN S, ROH B, et al. Identification of hosts behind a NAT device utilizing multiple fields of IP and TCP[C]//2016 International Conference on Information and Communication Technology Convergence. Piscataway: IEEE Press, 2016: 484-486.

[10] ZHANG B, GUAN Y, NIU W, et al. A hybrid packet clustering approach for NAT host analysis[C]//2015 IEEE International Conference on Communication Software and Networks. Piscataway: IEEE Press, 2015: 432-438.

[11] ABT S, DIETZ C, BAIER H, et al. Passive remote source NAT detection using behavior statistics derived from netflow[C]//IFIP International Conference on Autonomous Infrastructure, Management and Security. Berlin: Springer, 2013: 148-159.

[12] GOKCEN Y, FOROUSHANI V A, HEYWOOD A N Z. Can we identify NAT behavior by analyzing traffic flows?[C]//2014 IEEE Security and Privacy Workshops. Piscataway: IEEE Press, 2014: 132-139.

[13] KOMAREK T, GRILL M, PEVNY T. Passive NAT detection using HTTP access logs[C]//2016 IEEE International Workshop on Information Forensics and Security. Piscataway: IEEE Press, 2016: 1-6.

[14] SALOMONSSON S. Exploring NAT host counting using network traffic flows[D]. Karlstad : Sweden Karlstad University, 2017: 59-73.

[15] KHATOUNI A S, ZHANG L, AZIZ K, et al. Exploring NAT detection and host identification using machine learning[C]//2019 15th International Conference on Network and Service Management. Piscataway: IEEE Press, 2019: 1-8.

[16] IBRAHIM M, LIU H, JAWAHAR M, et al. Verification: Accuracy evaluation of Wi-Fi fine time measurements on an open platform [C]//Proceedings of the 24th Annual International Conference on Mobile Computing and Networking. New York: ACM , 2018: 417-427.

[17] HAN K, YU S M, KIM S L, et al. Exploiting user mobility for Wi-

Fi RTT positioning: A geometric approach[J]. IEEE Internet of Things Journal, 2021, 8(19): 14589-14606.

[18] ROY A K, KHAN A K. Privacy preservation with RTT-based detection for wireless mesh networks[J]. IET Information Security, 2020, 14(4): 391-400.

[19] HOU B, HOU C, ZHOU T, et al. Detection and characterization of network anomalies in large-scale RTT time series[J]. IEEE Transactions on Network and Service Management, 2021, 18(1): 793-806.

[20] CHENG Y. Mean shift, mode seeking, and clustering[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1995, 17(8): 790-799.

[21] COMANICIU D. An algorithm for data-driven bandwidth selection[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2003, 25(2): 281-288.

Tethering behavior detection architecture based on RTT measurement of TCP flows

DAI Xianlong[1, 2, 3], CHENG Guang[1, 2, 3, *], LU Guangyin[1, 2, 3], JIN Binlei[1, 2, 3]

(1. School of Cyber Science and Engineering, Southeast University, Nanjing 211189, China;
2. Jiangsu Province Engineering Research Center of Security for Ubiquitous Network, Southeast University, Nanjing 211189, China;
3. Purple Mountain Laboratories, Nanjing 211111, China)

Abstract: Tethering behaviour is the sharing of an Internet connection service with other connected devices by using a mobile smart device as a NAT gateway. It will share the smartphone's data plan, especially the unlimited data plan. So, it can put ISPs under additional pressure to operate mobile Internet and have an impact on their revenue. It can hide the internal network structure from the public network same as Network Address Translation (NAT). It also provides the possibility for illegal devices to access anonymously. Due to many limitations and circumventing methods in tethering detection, the existing NAT detection technology is difficult to detect tethering behavior. In order to process and forward data traffic, we examine the features of tethering behaviors terminal devices in mobile Internet communication base station. We also analyze the relevant characteristics of RTT in TCP flows in mobile Internet traffic. Then, we propose a tethering detection method based on unsupervised analysis of RTT in TCP flows, and construct the test network environment of this method. The experimental results verify the effectiveness of this method in detecting tethering behavior, and realize the effective detection of tethering behavior in mobile Internet by passive network traffic monitoring ,with an accuracy of 97.50%.

Keywords: Tethering detection; round trip time; network address translation detection; unsupervised learning; mobile internet

Received: 2021-08-13; **Accepted**: 2021-11-14; **Published Online**: 2021-11-23 09:24

URL: kns.cnki.net/kcms/detail/11.2625.V.20211122.1609.006.html

Foundation item: National Key R & D Program of China (2018YFB1800602)

*** Corresponding author**. E-mail: chengguang@seu.edu.cn