doi:10.3772/j.issn.1002-0470.2016.12.006

基于虚拟化的轻量级背景流量节点生成及部署算法研究①

解维崇②* 李正民** 董开坤③* 沈英宏*

(*哈尔滨工业大学计算机科学与技术学院 哈尔滨 150001)

(**国家计算机网络应急技术处理协调中心 北京 100029)

摘 要 进行了网络攻防分析,针对传统的背景流量模拟技术在生成及部署方面存在背景流量大小受限、背景流量真实性不足、背景流量生成工具部署的数量和位置受限的问题,提出了一种基于虚拟化的轻量级背景流量节点生成及部署算法,该算法是一种基于资源量的目标流量映射、基于最短路由的背景流量应用添加、基于最少通信代价的节点映射的三阶段算法,解决了背景流量节点在网络攻防实验中大规模快速生成及灵活部署的问题。实验结果表明,使用轻量级背景流量节点,生成的背景流量满足网络自相似性,资源消耗小、启动速度快,并且可根据实际网络攻防实验的需求进行动态、灵活、快速的大规模部署。

关键词 背景流量, Docker, 自相似, 动态部署, 虚拟映射

0 引 言

目前,通过攻防实验平台模拟大规模网络安全事件是一种高效方便的方法。网络攻击行为的研究是对网络安全事件本身产生的攻击流量[1]和网络实际中存在的背景流量[2]叠加后的流量进行分析,在缺少背景流量的前提下进行网络行为模拟的研究是难以取得实际效果的。因此,如何在攻防实验平台中快速生成及灵活部署背景流量是一项重要课题。

在生成及部署背景流量方面的研究工作主要包括使用网络离散事件模拟器 NS2[3]、使用流量生成工具以及采用多线程技术。来自哈尔滨工业大学的邹天际[4]使用 NS2 网络离散事件模拟器对自相似性[5]网络 ON/OFF 流量模型进行了深入研究,同时哈尔滨工业大学的李广荣[6]等指出网络模拟器生成的流量接入真实网络环境存在虚拟网卡流量大小受限的问题。Sommers 和 Barford 提出了一种自配置网络流量生成工具 Harpoon[7], Abdo[8]等人基于 FPGA 实现了随机快速分布网络数据的模拟,这两种工具生成的流量真实性强,但是部署的数量和位置受限。吉林大学的刘曜[9]使用多线程技术模拟了不同协议的网络背景流量,并应用于入侵检测系统,不过使用多线程模拟多客户端无法保证网络空间隔离性和 IP 地址的真实性,同时在部署方面需要借助资源消耗较高的虚拟机[10]。

综上,采用如 NS2 网络模拟器的方式节点部署灵活、真实性较强,但是流量大小受限;采用流量生成工具的方式,流量真实性较强,但是部署流量生成工具的数量和位置受限;采用多线程的方式,资源消耗小,但是流量真实性不足且部署不方便;采用虚拟机的方法,流量真实性较强,但因其资源消耗大、资源粒度粗,部署节点的数量和位置受限。而许多网

① 国家科技支撑计划(2012BAH45B01),国家自然科学基金(61100189,61370215,61370211)和国家信息安全计划(2014A085,2015A072)资助项目。

② 男,1991 年生,硕士生;研究方向:网络安全;E-mail: xc123hit@gmail.com

③ 通讯作者,E-mail: kaikun.dong@gmail.com

(收稿日期:2016-04-20)

络攻防实验需要构建真实的网络环境,其中背景流量节点需要满足大规模快速生成及灵活部署的特性。为解决上述问题,本文提出了一种基于虚拟化的轻量级背景流量节点生成及部署算法,其中部署算法采用一种基于资源量的目标流量映射、基于最短路由的背景流量应用添加、基于最少通信代价的节点映射的三阶段算法,根据实际网络攻防实验的需求动态灵活快速地部署大规模背景流量节点。

1 轻量级背景流量节点生成

背景流量是指在真实网络环境中由网络应用程序生成的流量,在网络攻击行为中添加背景流量,可以提高网络行为模拟结果的真实性和准确性。轻量级背景流量节点是指将采用自相似流量模型编写的网络应用加载到轻量级虚拟化容器而形成的节点,其生成的背景流量满足网络的自相似性。本文采用轻量级背景流量节点模拟普通用户,使用轻量级虚拟化容器技术可以满足背景流量节点大规模快速生成、灵活部署、资源消耗小等特性。

1.1 典型流量模型

真实的网络流量是具有自相似的,流量建模的目标是能够更好地反映实际流量的特征并且用于更好地指导实际的应用。当前,网络流量公认的、最重要的统计特征是大时间尺度下的自相似性和小时间尺度下的多分形性。本文采用 Pareto 分布与逆高斯(Inverse Gaussian)分布相混合的 ON/OFF 过程模型来体现自相似性的特点。

基于 Pareto 分布的概率密度函数为

$$f(x) = \begin{cases} \dfrac{\alpha}{\beta}\left(\dfrac{\beta}{x}\right)^{\alpha+1} & x \leqslant \beta \\ 0 & x > \beta,\ \alpha > 0 \end{cases} \quad (1)$$

其中,α 为形状参数,表示 Pareto 分布的重尾程度,α 取值越小,重尾的程度越强;β 为最小截止参数,表示该随机变量能够取到的最小值。

基于 Inverse Gaussian 分布的概率密度函数为

$$f(x;\mu,\lambda) = \sqrt{\frac{\lambda}{2\pi x^3}}\exp\left\{\frac{-\lambda(x-\mu)^2}{2\mu^2 x}\right\} \quad x > 0, \mu > 0, \lambda > 0 \quad (2)$$

其中,Wald 分布是 $\mu = \lambda = 1$ 时逆高斯分布的特例。

生成的背景流量来自于网络应用,在应用层中,会综合考虑多种流量特性参数。在 OFF 阶段,模拟客户处于空闲状态,在 ON 阶段,模拟客户对 WEB 文件进行访问,ON 和 OFF 的时间分布服从 Pareto 分布。在 ON 阶段又进一步划分为若干文件传输阶段和文件暂停阶段。暂停阶段是模拟用户浏览一页完整的 WEB 页面时,传输组成该页面的各个对象之间的间隔时间。文件请求的大小服从 Pareto 分布,每次请求页面的数量服从 Inverse Gaussian 分布,浏览页面的时间服从重尾分布。

将上述混合分布的流量模型应用于容器节点中,产生的网络流量服从网络的自相似性,从而保证流量的真实性。

1.2 背景流量节点生成方案

针对背景流量节点大规模快速部署的特性,本文使用轻量级虚拟化技术。背景流量节点需要保证网络资源的隔离,保证资源限制和进程控制,因此选取 Docker 作为背景流量节点,使用 Docker 的 Namespace 和 Cgroups 技术。

Namespace 是网络名字空间,可以保证网络资源的隔离。通过将网络应用集成到容器中,可以保证背景流量节点拥有真实的内核协议栈,IP 地址等网络资源,可以作为真实的客户端和服务端来大规模的生成。

Cgroups 保证容器的资源限制和进程控制。通过限制、隔离进程组所使用的 CPU、内存等物理资源,可以保证背景流量节点资源的高效利用。通过对网络应用进行控制,可以保证背景流量的产生、挂起以及恢复操作,同时使用进程管理器可以管理多个应用的同时运行。

流量节点客户端和服务端使用 OpenVSwitch 来进行底层通信,使用 Docker 的 None 网络模式与 Docker 默认网络模式相比,这样可以实现更加灵活复杂的网络。Docker 网络通信的原理如图 1 所示。

背景流量节点生成的流程如图 2 所示。

在图 2 中,首先从 Docker 仓库拉取核心精简镜像,预装编译环境,然后判断节点类型,如果是客户端节点则将应用程序加载到容器中,如果是服务端节点则配置特定协议的服务器,通过 Dockerfile 技

术,对镜像不断提交,使用进程管理器 Supervisor 来对应用进行管理,最后生成特定的背景流量节点。

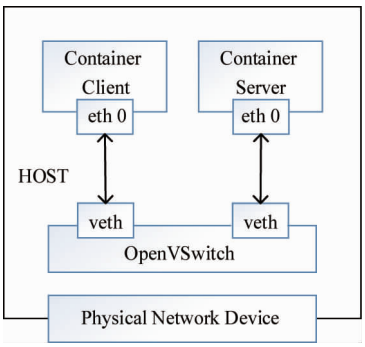


图 1　Docker 网络通信原理图

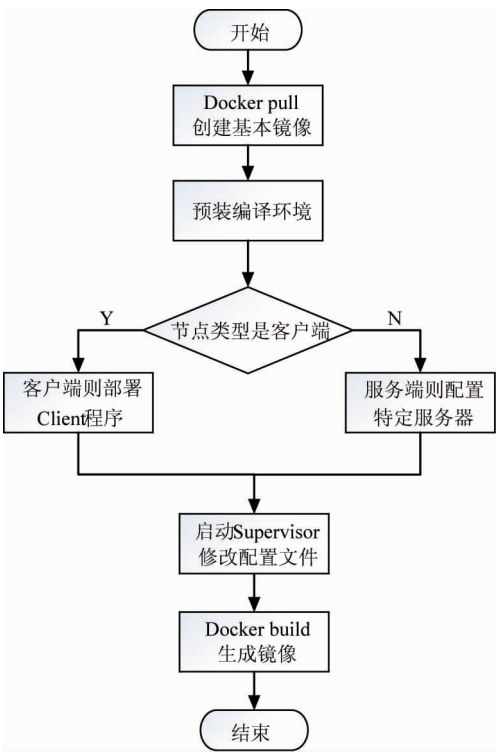


图 2　背景流量节点生成流程图

2　背景流量节点动态部署算法

传统的背景流量节点部署主要是通过将流量产生器接入到目标路径中,或者将使用多线程实现的网络应用程序拷贝到虚拟机中,然后将虚拟机部署到虚拟网络拓扑中。采用上述方法,节点部署的数量和位置受限,无法满足大规模动态灵活部署的要求。因此本文提出了一种三阶段的背景流量节点动态部署算法,能满足大规模动态灵活部署背景流量节点的需求。

为了便于说明背景流量节点动态部署算法,给出以下定义和描述:

定义 1　V:$V(N^v, L^v, B^v)$ 表示虚拟网络拓扑,其中 N^v 表示虚拟网络节点集合,L^v 表示虚拟网络链路集合,B^v 表示目标流量路径集合。

定义 2　P:$P(N^p, L^p)$ 表示底层物理网络拓扑,其中 N^p 表示物理服务器节点集合,L^p 表示物理网络链路集合。

定义 3　M:$M(S^m, R^m)$ 表示虚拟网络到物理网络映射关系,其中 S^m 表示虚拟节点映射到物理服务器的编号列表,R^m 表示物理服务器剩余资源列表。

定义 4　D:$D(C^d, F^d, Q^d)$ 表示流量节点信息,其中 C^d 表示流量节点消耗的资源,F^d 表示流量节点生成的流量大小,Q^d 表示服务端节点承载流量的能力。

三阶段的背景流量节点部署算法原理如图 3 所示。

图 3 表示分别在目标路径 1 到 4 以及 4 到 6 上部署背景流量的结果。目标路径定义为终端路由器之间的路径,V 表示虚拟拓扑,M 表示虚拟映射关系,P 表示物理拓扑。图 3 左半部分表示未部署背景流量节点的虚拟网络拓扑及底层映射方案,图 3 右半部分表示部署背景流量节点后的虚拟网络拓扑以及底层映射方案。

本文提出的背景流量节点部署算法分为三阶段,按照先后顺序执行。算法输入是四要素(V、P、M、D),输出是背景流量节点的部署方案。

三阶段背景流量节点部署算法原理描述如下:

步骤 1 根据流量节点信息 D,对用户需要的虚拟网络 V 的目标流量 B^v 进行映射,计算出所需的流量节点数量以及消耗的资源。

步骤 2 根据最短路由策略确定背景流量应用在虚拟网络 V 中添加的位置,同时保证应用添加数量最少。

步骤 3 根据 M 将流量节点映射到物理拓扑 P 中,如果映射到不属于 M 对应的服务器,则需要计算到原来服务器的隧道连接,同时保证通信代价最少。

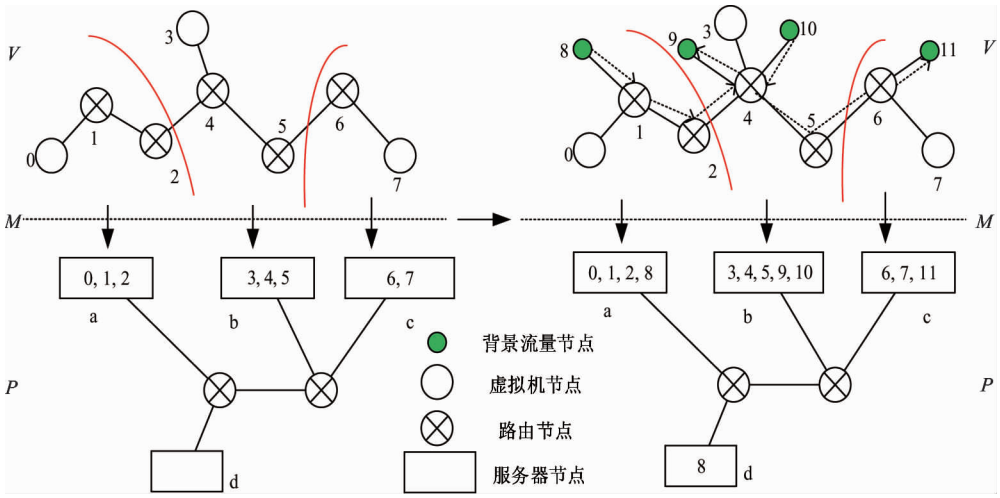


图 3　背景流量节点部署算法原理图

2.1　基于资源量的目标流量映射算法

为满足流量的多样性，本文实现了 HTTP、FTP、TFTP、SMTP、DNS、SSH、TELNET、NTP 八种协议的客户端和服务器节点，并且提出了一种基于资源量的目标流量映射算法（resource mapping algorithm，RMA）。

资源量是指流量节点消耗的 CPU 和内存。首先统计每种协议流量节点的资源量，取结果中资源量最小的节点作为标准流量节点，记为标准资源量 $\{CPU^0, MEM^0\}$。

物理服务器与流量节点的资源量采用一种 max-min 资源分配机制，其中物理服务器总的资源量表示为 $\min\{CPU^p, MEM^p\}$，而流量节点的资源量取决于可支配资源，表示为 $\max\left\{\frac{CPU^i}{CPU^0}, \frac{MEM^i}{MEM^0}\right\}\times\{CPU^0, MEM^0\}$。

流量节点分为客户端和服务端两种类型，需要统计每种协议流量节点产生的流量大小以及每种协议的服务端所能承载的最大流量，目标流量若超过最大流量，则需要对目标流量进行拆分，保证服务端能正常运行。

基于资源量的目标流量映射算法首先判断目标流量是否超过服务端最大流量，如果超过则需要对流量进行拆分；然后通过目标流量计算出需要的流量节点数量，根据节点数量计算出节点消耗的资源量。基于资源量的目标流量映射算法 RMA 描述如下：

Input

虚拟拓扑 V、流量节点信息 D、目标流量路径集合 B^v

Output

流量节点的数量、流量节点消耗的资源

BEGIN

background←NULL //构建目标流量的路径列表

for link in B^v **do**

if link 对应的流量大小 > Q^d **do**

根据 Q^d 对目标流量进行切分，添加到 background 中

else 直接添加到 background 中

end if

end for

for link in background **do**

Client _ amount = link 对应的流量大小/ F^d

Server _ amount = 1

Client _ resource = Client _ amount × C^d 中客户端资源量

Server _ resource = Server _ amount × C^d 中服务端资源量

end for

END

2.2　基于最短路由的背景流量应用添加算法

在虚拟路径上添加背景流量最简单的方法是将每条路径两边的节点都绑定一对流量节点，一个作为流量的发送端，一个作为流量的接受端。在网络行为模拟中，每一对流量节点都需要消耗一定的系统资源。对于小规模网络拓扑，此方法产生的资源

开销可以接受,但对于大规模的网络,此开销是难以接受的。所以必须保证在目标链路上存在一定流量的基础上,减少添加应用程序的个数。

本文使用基于最短路由的应用添加算法(shortest routing application adding algorithm, SR-AAA)。数据包从发送端到接收端沿着最短路径传输。如果在发送端和接收端添加流量节点,则数据包沿着最短路径到达接收端,途中经过的每条链路都会达到目标流量,这样与每条链路两端都添加流量节点相比,不仅仅可以减少应用程序的个数还会减少流量节点产生的系统开销。

虚拟拓扑采用基于最短路径的静态路由。传统的路由计算方法 Dijkstra 算法需要遍历每一对节点,算法的时间复杂度高。本文使用一种基于终端路由器洪范传播的路由算法。该算法通过遍历虚拟网络拓扑的终端路由器,采用广度优先搜索,将终端路由器本身路由表以洪范方式发送到网络中的其他路由器,每个路由器在收到洪范消息时将得到的路由信息加入到自身路由表中,最终使每个路由器都正确的保存了路由信息。

基于最短路由的背景流量应用添加算法 SR-AAA 的描述如下:

Input

虚拟拓扑 V、目标流量路径集合 B^v

Output

流量节点的挂载位置、流量节点间的路由信息

BEGIN

v ←根据 V 生成虚拟拓扑图
RT← v 中的终端路由器
RI←NULL //路由信息
for r in RT **do**
把与路由器 r 直连的主机路由加入到 $r.RI$ 中
end for
for r in RT **do**
BFS (r,v) //对 r 进行 BFS 并将本身 RI 发送到遇到的新路由器 r_n
$r\_n.RI$ += $r.RI$
end for
for Link in B^v **do**
Client _ position = Link. src
Server _ position = Link. dst
Client 路由信息 = Client _ position. RI
Server 路由信息 = Server _ position. RI
end for

END

2.3 基于最少通信代价的节点映射算法

通信代价定义为物理服务器的使用数量以及真实的路由跳数。最小通信代价即要保证背景流量节点映射所需的服务器数量最少且底层路由跳数最少。

通过目标流量映射,确定了所需的流量节点数量以及消耗的资源。通过背景流量应用添加算法,确定了流量节点在虚拟拓扑中的挂载位置。部署算法的第三阶段是将流量节点映射到底层的物理网络。

虚拟路由器挂载的流量节点应当尽量映射到此虚拟路由器对应的物理服务器,因此本文使用基于贪心算法的背包模型来实现背景流量节点映射。

如果虚拟路由器挂载的流量节点不能够完全映射到此虚拟路由器对应的物理服务器,则需要进行二次映射。二次映射需要保证路由跳数最少,因此本文采取对第一次未映射成功的流量节点分别使用广度优先搜索策略,将其映射到离原来的服务器路由跳数最少的目标服务器。

基于最少通信代价的节点映射算法(least communication cost node mapping algorithm, LLC-NMA)描述如下:

Input

虚拟拓扑 V、物理拓扑 P、虚拟网络映射关系 M、背景流量节点挂载位置、背景流量节点消耗资源、流量节点间的路由信息

Output

背景流量节点的底层物理映射结果

BEGIN

v ←根据 V 生成虚拟拓扑图
p ←根据 P 生成物理拓扑图
need _ dict←NULL
for link in background **do**
client = M. link. src
server = M. link. dst
need _ dict. client + = (resource, amount)
need _ dict. server + = (resource, amount)

```
end for
need _ to _ map = need _ dict
for p _ node in need _ dict do
    Sort (need _ dict. p _ node)
    maped. p _ node = NULL
    for v _ node in need _ dict. p _ node do
        if R^m. p _ node > v _ node do
            maped. p _ node + = v _ node
            need _ to _ map. p _ node- = v _ node
        end if
    end for
end for
for node in need _ to _ map do
    bfs(p, node)
    p _ node = 将背景流量节点放入到离 node 最近的
    服务器中
    Vxlan + = (node, p _ node) //添加远程隧道并对
    背景流量节点添加路由信息
    maped. p _ node + = node
    need _ to _ map. p _ node - = node
end for
```

3 实验及结果说明

为验证背景流量节点的真实性和背景流量节点动态部署算法的有效性,总共进行两组测试。背景流量节点真实性测试,用来表明使用本文的背景流量模型生成的网络流量符合自相似性;背景流量节点动态部署算法测试,用来表明使用轻量级虚拟化容器生成的流量节点资源消耗小,生成速度快,可以满足大规模部署,以及验证算法运行结果的正确性与可行性。同时使用单个轻量级背景流量节点来模拟单个普通用户。实验环境如表 1 所示。

表 1 实验环境配置表

配置项目	服务器	KVM 虚拟机	Docker 容器
CPU	16 核	1 核	1 核
内存	64GB	512MB	512MB
操作系统	Ubuntu 14.04	Ubuntu 14.04	Ubuntu 14.04

3.1 背景流量节点真实性测试

本文使用单个轻量级背景流量节点来模拟单个普通用户。因此采用单个 HTTP 协议背景流量客户端节点向服务端节点发送 WEB 请求,在服务端节点捕获生成的网络流量,结果如图 4 中柱形条所示。

采用单个普通用户访问天猫 www. tmall. com 的官方主页,捕获生成的网络流量,结果如图 4 中的曲线所示。

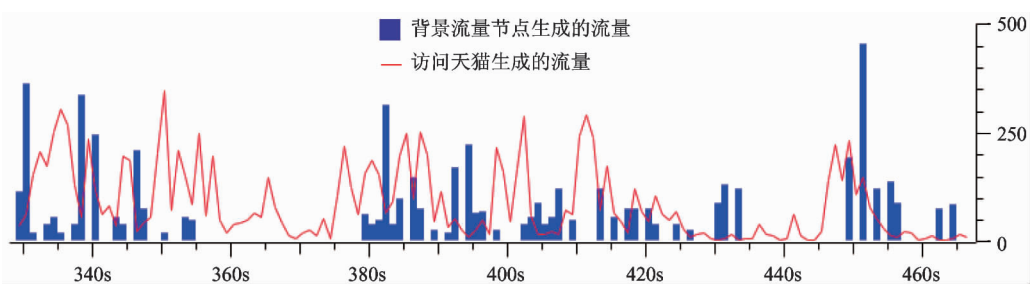


图 4 流量结果对比图

从图 4 中可以看出,单个普通用户访问天猫主页生产的流量满足自相似性,使用单个背景流量节点生成的流量在网络特性上也满足网络的自相似性,从而保证流量的真实性。

3.2 背景流量节点动态部署算法测试

实验使用的 KVM 虚拟机节点和 Docker 容器节点配置相同,如表 1 所示。两种客户端节点生成的 HTTP、SMTP、FTP 协议流量大小分别为 400、300、400,单位kB/s。服务器端节点最大承载流量分别为 20、15、30,单位 MB/s。

通过 docker stats 命令统计,本文三种协议的背景流量客户端节点的资源量分别为(0. 02MB, 40MB)、(0.02MB, 30MB)、(0.04MB, 40MB),服务端节点的资源量分别为(0.8MB,500MB)、(0.6MB,

300MB)、(1.0MB, 500MB)。标准客户端资源量为(0.01MB, 20MB)。根据2.1节的基于资源量的目标流量映射算法,得知三种协议客户端节点分别消耗2、2、4个标准资源量,服务端节点分别消耗40、30、50个标准资源量。通过top命令进行统计,三种协议虚拟机客户端节点分别消耗20、20、20个标准资源量,服务端节点分别消耗50、40、50个标准资源量。

背景流量节点动态部署算法实验拓扑如图5所示。

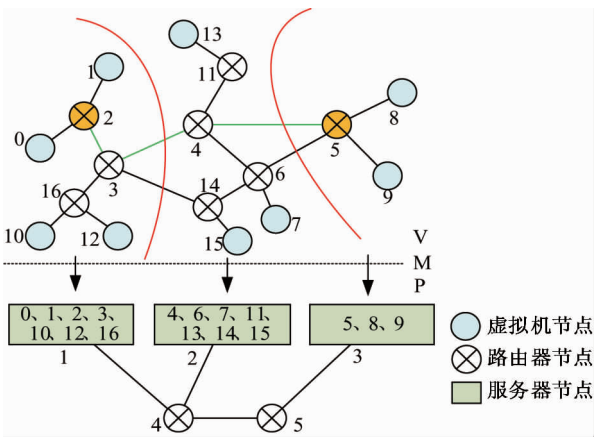


图5　背景流量节点动态部署算法实验拓扑图

在图5的实验拓扑中,需要在目标路径2到5上分别生成40、70、100(单位MB/s)的混合协议背景流量,HTTP、SMTP、FTP流量大小比例关系为2:1:2,根据2.1节的基于资源量的目标流量映射算法,生成40、70、100(单位MB/s)的混合协议背景流量,需要的HTTP、SMTP、FTP协议客户端节点数量分别为(40,70,100)、(27,47,66)、(40,70,100),服务端节点数量分别为(1,2,2)、(1,1,2)、(1,1,2)。实验所使用的三台服务器分别剩余600、2400、2800个标准资源量。

常见的三种背景流量节点部署方案包括随机部署、贪心部署以及虚拟机部署。随机部署指的是背景流量节点随机的挑选服务器进行动态部署;贪心部署指的是背景流量节点每次选择剩余资源最大的服务器进行动态部署;虚拟机部署指的是使用虚拟机作为背景流量节点来进行三阶段动态部署算法。

下面将使用三阶段背景流量节点动态部署算法与常见的三种部署方案从部署结果进行比较,如表2所示。表中的数字表示三种协议节点的数量。

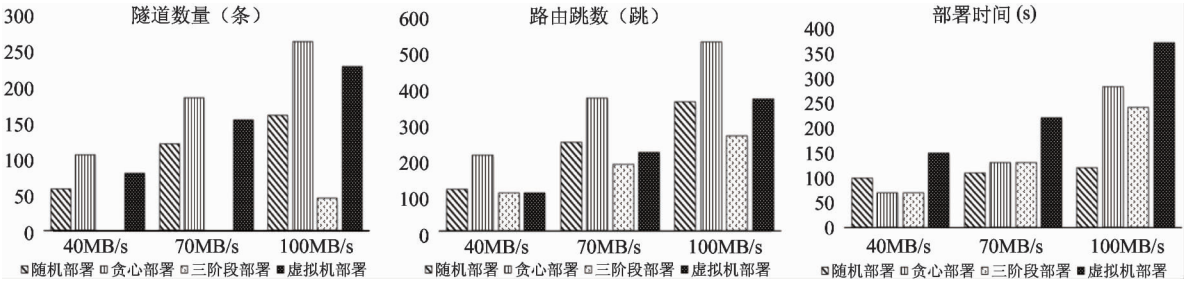
表2　节点部署方案

流量大小	节点分类	1号服务器	2号服务器	3号服务器
40MB/s	随机部署	(20、6、12)	(11、16、12)	(9、5、6)、(1、1、1)
	贪心部署			(40、27、40)、(1、1、1)
	三阶段部署	(40、27、40)		(1、1、1)
	虚拟机部署	(12、6、12)	(32、16、32)	(1、1、1)
70MB/s	随机部署	(34、17、14)	(10、14、35)	(26、16、21)、(2、1、2)
	贪心部署			(70、47、70)、(2、1、2)
	三阶段部署	(70、47、70)		(2、1、2)
	虚拟机部署	(12、6、12)	(48、24、48)	(14、7、14)、(2、1、2)
100MB/s	随机部署	(45、17、32)	(20、36、18)	(35、13、50)、(2、2、2)
	贪心部署			(100、66、100)、(2、2、2)
	三阶段部署	(100、40、80)	(0、26、20)	(2、2、2)
	虚拟机部署	(12、6、12)	(48、24、48)	(44、22、44)、(2、2、2)

根据2.1节的基于资源量的目标流量映射算法以及表2结果可以看出,在目标流量大小为100MB/s时,将目标流量切割为5份。其中,(100、40、80)表示HTTP、SMTP、FTP协议客户端节点数量分别为100、40、80个。

下面将三阶段背景流量节点动态部署算法与常见的三种部署方案从隧道条数、路由跳数、部署时间三个方面进行比较,结果如图6所示。

根据图6中的结果可知,随机部署方案:优点是算法速度快,每台服务器随机部署一定数量的背景流量节点,但是使用的服务器数量多,没有考虑路由跳数,且隧道条数多,网络通信代价高。



（a）隧道数量比较图　　（b）路由跳数比较图　　（c）部署时间比较图

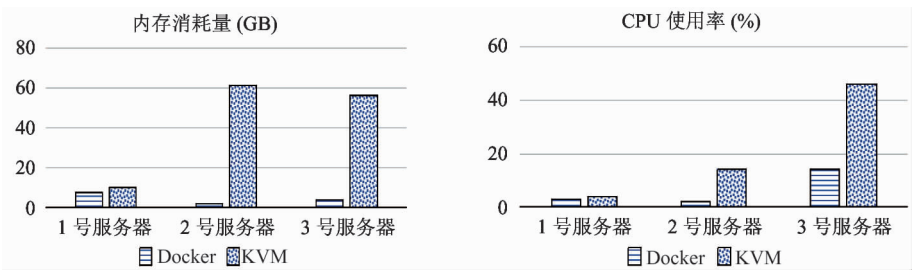
图 6　四种部署方案比较图

贪心部署方案：优点是使用的服务器数量最少，但是没有考虑底层的路由跳数，且隧道条数多，网络通信代价高。

三阶段背景流量节点动态部署方案：优点为充分考虑到物理拓扑的连接关系，优先将节点映射到其对应的服务器中，如果对应的服务器资源不够，则会映射到离原来服务器跳数最少的服务器，网络通信代价低、部署灵活且执行速度快。

虚拟机节点部署方案：优点为虚拟机真实，可以保证完全隔离的网络环境且安全性高，但是不能满足大规模灵活快速部署的需求，并且在部署的数量和位置上都会受限。

通过在目标路径 2 到 5 上部署 100MB/s 的混合协议背景流量，分别从内存消耗量、CPU 使用率两方面对背景流量节点和 KVM 虚拟机节点的部署方案进行比较，结果如图 7 所示。



（a）内存消耗量比较图　　（b）CPU 使用率比较图

图 7　Docker 与 KVM 部署方案比较图

从上述结果中可以看出，使用轻量级虚拟机容器作为背景流量部署方案，在系统资源消耗方面，明显低于 KVM 虚拟机节点。虽然 KVM 虚拟机采用外置磁盘快照生成，但在启动时间上，轻量级虚拟化容器明显快于 KVM 虚拟机，因此使用轻量级虚拟化容器作为背景流量节点的部署是一种很好的方案。

图 5 中的节点 2 到节点 5 的目标路径上部署 40MB/s 的 HTTP 协议流量。根据背景流量节点动态部署算法的结果，测试算法的可行性。

启动背景流量节点同时使用 nload 命令统计目标路径流量大小，实验结果如表 3 所示。

表 3　HTTP 协议流量大小统计表

虚拟节点位置	HTTP 流量大小	数据包源	数据包目的
路由器 2	40MB/s	0 号节点所在子网	8 号节点所在子网
路由器 3	40MB/s	0 号节点所在子网	8 号节点所在子网
路由器 4	40MB/s	0 号节点所在子网	8 号节点所在子网
路由器 5	40MB/s	0 号节点所在子网	8 号节点所在子网

从表3可以看出目标链路生成的流量大小符合预期目标流量大小,验证了部署算法的可行性。

4 结 论

本文通过对轻量级背景流量节点生成以及部署算法的研究,提出了一种使用轻量级虚拟化技术生成背景流量节点的方案及一种流量节点动态部署算法。实验结果证明,使用混合分布的背景流量模型生成的背景流量满足自相似性,使用轻量级虚拟化容器生成流量节点与采用NS2网络模拟器的方式相比,流量大小不会受限;与采用将流量生成工具或者多线程应用程序整合到虚拟机的部署方式相比,资源消耗更小,启动时间更快,节点部署的规模更大更灵活。最后通过算法运行结果以及统计目标链路流量大小证明了动态部署算法的正确性与可行性。

参考文献

[1] Pietro A D, Vasseur J P, Cruz Mota J. Verifying network attack detector effectiveness. US patent: 20160028753, 2016

[2] Qian Y X, Guan X, Jiang M, et al. Modeling and generating realistic background traffic by hybrid approach. *Wireless Communication Over Zigbee for Automotive Inclination Measurement China Communications*, 2015, 12(10): 147-157

[3] 李娄久, 李秀坤. NS2平台的TCP/IP网络拥塞控制算法仿真. 实验室研究与探索, 2015, 34(2):81-83

[4] 邹天际. 网络模拟中背景流量模型的研究:[硕士论文]. 哈尔滨: 哈尔滨工业大学计算机科学与技术学院, 2010. 16-31

[5] 张宾, 杨家海, 吴建平. Internet流量模型分析与评述. 软件学报, 2011, 22(1):115-131

[6] 李广荣, 王琨, 张兆心. 基于NS-3构建网络环境的虚实结合技术的研究. 高技术通讯, 2015, 25(7):685-693

[7] Sommers J, Barford P. Self-configuring network traffic generation. In: Proceedings of the ACM SIGCOMM Conference on Internet Measurement 2004, Taormina, Sicily, Italy, 2004. 68-81

[8] Abdo A, Hall T. FPGA-based testbed for packet switch performance measurement. In: Proceedings of the Conference Record-IEEE Instrumentation and Measurement Technology Conference, San Francisco, USA, 2006. 347-352

[9] 刘曜. 用于测试入侵检测系统的网络背景流量模拟:[硕士论文]. 长春:吉林大学计算机科学与技术学院, 2009. 22-34

[10] 康辰, 朱志祥. 基于云计算技术的网络攻防实验平台. 西安邮电大学学报, 2013, 18(3):87-91

Study of an algorithm for generation and deployment of background traffic nodes based on virtualization

Xie Weichong*, Li Zhengmin**, Dong Kaikun*, Shen Yinghong*

(*School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001)

(**The National Computer Network Emergency Response Technical Team Coordination Center of China, Beijing 100029)

Abstract

The analysis of network attack and defense was conducted, and aiming at the traditional background traffic simulation technology's problems in generation and deployment of traffic nodes such as size limitation of background traffic, authenticity limitation of background traffic, and number and position limitations of the tools for background traffic generation and deployment, an algorithm for generation and deployment of lightweight background traffic nodes based on virtualization was proposed. This is a three-phase algorithm that consists of the target traffic mapping based on resource amount, the adding of background traffic application based on the shortest route, and the node mapping based on the least communication cost, aiming to solve the problems of massive, rapid generation of background traffic nodes and flexible deployment of them in network attack and defense experiments. The experimental results show that using the proposed lightweight background traffic nodes to generate the background traffic can satisfy the self-similarity of networks, the small consumption of resource, the fast starting speed, and the massive, flexible and quick deployment according to the actual network attack and defense experiment.

Key words: background traffic, Docker, self-similarity, dynamic deployment, virtual mapping