SDN 网络中面向服务的网络节点重要性排序方法

张 笛 李兴华 刘 海 马建峰
(西安电子科技大学网络与信息安全学院 西安 710071)

摘 要 在 SDN 中,作为网络大脑的 Controller,不仅为下层转发节点 Switch 提供数据转发流表,还为上层应用程序提供各类网络基本信息,负担繁重.特别是当网络规模较大时,Controller 会成为网络性能的瓶颈.实验表明,当一个 Controller 管理 300 个 Switch 节点时,平均首 ping 时延超过 300 ms,而访问百度、新浪等网站仅需 20 ms 左右,可见其对 Controller 负载造成相当大的影响,因此,有必要对其进行优化.已有工作表明,通过选取少量关键节点可实现对网络状态的可观性与网络行为的可控性.因此,减轻 Controller 负载的一个有效途径是对其所管辖的网络节点进行重要性排序,选取出重要节点,通过收集重要节点的网络服务信息去评估全网节点的网络状态,Controller 再根据全网状态向这些重要节点下发相关控制策略,通过重要节点把控制策略扩散到全网节点从而实现全网控制.无论是信息采集还是下发控制流的过程,Controller 仅仅操纵这些重要节点,既减少了信息采集工作又减少了控制流的下发工作,从而达到降低 Controller 负担的目的,因此,该文提出了一种面向服务的网络节点重要性排序方法.对于每一种网络服务,结合 SDN 集中控制以及获取全局参数的特性,不但考虑了 Switch 的网络信息,如通信量、网络拓扑等,而且结合了具体的网络服务选取对应的服务参数,对 Switch 进行重要性排序.Controller 仅仅需要对排名靠前的 Switch 节点进行信息的收集与控制,从而达到降低自身负担的目的.该文以 SDN 网络中检测 TCP 洪泛攻击这一网络服务为例,进行了大量的实验,实验证实了参数选取的有效性,并且将该方案与已有方案进行了实验对比,该方案优于已有的排序方案.当选取排名在前 40% 的 Switch 节点进行信息收集时,不仅能够减少控制器 60% 的负担,还可以保证攻击检测程序成功率高达 94%,同时,所提方案本身给 Controller 带来的负担仅占 2%.

关键词 SDN;节点排序;Controller 负担;面向服务;TCP 洪泛攻击
中图法分类号 TP309 **DOI 号** 10.11897/SP.J.1016.2018.02624

Service-Oriented Node Importance Ranking Method in SDN

ZHANG Di LI Xing-Hua LIU Hai MA Jian-Feng
(*School of Cyber Engineering, Xidian Universality, Xi'an* 710071)

Abstract In SDN, Controller is the brain of the software defined network. It not only provides the flow tables of data forwarding for Switch nodes in the lower layer, but also provides all kinds of information for applications in the upper layer. The experiment shows that when a controller manages 300 switches, the delay of setting up flow tables for the ping command is up to 300 ms. However, the delay of large domestic sites is only 20 ms. With the expansion of network, Controller will become the bottleneck of the network, which will directly affect the performance of the whole network. Therefore, how to reduce the burden of Controller is a key to ensure the efficiency of network. The existing work shows that it can realize observability and controllability of the network by selecting a few important nodes, so an effective way of reducing the burden of controller is to rank all the network nodes and pick out some important nodes, then, we can collect network services information of these important nodes to evaluate the status of the entire

收稿日期:2017-06-12;在线出版日期:2017-12-26.本课题得到国家自然科学基金(U1708262,U1736203,61772173,61672413)资助.
张 笛,女,1993 年生,硕士研究生,主要研究方向为网络与信息安全.E-mail:2529932426@qq.com.**李兴华**(通信作者),男,1978 年生,博士,教授,博士生导师,主要研究领域为网络与信息安全、隐私保护、云计算、安全协议形式化方法.E-mail:xhli1@mail.xidian.edu.cn.**刘 海**,男,1984 年生,博士研究生,主要研究方向为密码算法设计、隐私保护.**马建峰**,男,1963 年生,博士,教授,主要研究领域为信息安全、编码理论、密码学.

network. According to the status of the entire network, Controller distributes control strategy for these important nodes, these important nodes spread control strategy to the entire network in order to control the entire network. Controller just manipulates these important nodes both for collecting information and distributing strategy. It not only reduces the work of collecting information, but also reduces the work of distributing strategy. As a result, it reduces the burden of Controller. Aiming at the above problems, a service-oriented network node importance ranking method is proposed. The software defined network can acquire global parameters, so this paper can easily choose network parameters, like network topology, traffic and so on. Packet loss and delay are different in different network service, so big variations in the importance of the same node have opened up in different network service. Thus, the paper also picks out some parameters related to the specific network service, packet loss and delay are as fixed parameters, this paper also supports adding new parameters related to the specific network service. After selecting parameters, normalization processing for all the parameters is made, and then the paper determines the weight of all the parameters by using principal component analysis method. The importance indexes of Switch nodes are obtained according to making weighted summation. Finally, the paper sorts Switch nodes by importance index. Controller just needs to collect information from top-ranked Switch nodes in order to reduce its burden. This paper takes the service of detecting TCP flooding attack in SDN as a case for experiments. Extensive experiment results indicate the effectiveness of parameter selection, especially for those parameters related to network service, and this paper is better than existing schemes. When collecting information from the top 40% of the Switch nodes, it can not only reduce 60% of the Controller's burden, but also ensure the success rate of attack detection as high as 94%. However, the burden brought to the controller by the scheme itself only takes 2%.

Keywords　SDN; node ranking; the burden of Controller; service-oriented; TCP flooding attack

1　引　言

软件定义网络(Software Defined Network,SDN)是将控制层面与数据转发层面解耦而提出的一种新型的网络架构.与传统网络架构相比,由于 SDN 具有集中控制,能让网络管理员便捷地进行网络管理和部署各类应用服务的特点,因此它已得到各商业公司或组织的广泛应用.例如,Google 公司就基于 SDN 架构搭建其全球骨干网络;亚马逊 AWS 也利用 SDN 架构设计面向全球用户的大规模弹性计算云服务网络.

在 SDN 网络中,Controller 主要负责处理转发节点 Switch 的各类请求、与网络中其他 Controller 进行信息交互以及为网络应用服务提供信息.随着网络规模的增大,Controller 的工作量也增大.实验表明,当一个 Controller 管理的 Switch 数量达到 300 个时,网络首 ping 某一个 IP 地址,Controller 为该 ping 命令制定到达该 IP 地址的流表所花的时延就高达 300ms,而访问百度、新浪等网站仅需 20ms 左右,可见网络规模的增大会加重 Controller 的负担,进而影响到整个网络的性能.因此,如何降低 Controller 资源消耗、提高工作效率,是保证网络高效运行的关键.

已有文献[1-2]表明,通过选取少量关键节点可实现对网络状态的可观性与网络行为的可控性.因此,减轻 Controller 负载的一个有效途径是对其所管辖的网络节点进行重要性排序,选取出重要节点,通过收集重要节点的网络服务信息去评估全网节点的网络状态,Controller 再根据全网状态向这些重要节点下发相关控制策略,通过重要节点把控制策略扩散到全网节点实现全网控制.无论是信息采集还是下发控制流的过程,Controller 仅仅操纵这些重要节点,既减少了信息采集工作又减少了控制流的下发工作,从而达到降低 Controller 负担的目的.

由于传统网络采用分布式的架构,主要存在以下问题:分布式架构的传统网络难以实时获取到各个节点的流量、数据包转发频率等信息,所以这些方法主要依据拓扑结构等静态信息对网络节点进行重要性的排序,信息收集的不够全面;当网络拓扑结构发生变化时,由于传统网络的分布式特征,难以及时地更新全网信息,导致信息更新的时延较长;传统网络不能直接从全局角度收集网络节点的信息,而是通过分布式的方式,分别获取局部信息,通过迭代或递归的方式得到网络节点的全局信息,这会降低信息收集的速度和准确度.而在 SDN 网络中,由于控制器的集中控制,所以能够容易获取到更为丰富的网络信息(如:通信量、转发节点活跃度等),而它们对判断节点重要性有着重要的影响.当网络拓扑发生变化时,由于 Controller 具有全局网络拓扑视图,可以及时感应到拓扑的变化,从全局的角度快速及时地更新全网信息,而传统网络很难做到及时地更新全网信息,所以传统网络一般忽略网络拓扑更新问题,这样的排序结果必定会随着网络拓扑的不断变化而失去其准确性.

另外,传统的重要性排序方法不是面向具体的网络应用服务,没有将影响网络服务的参数考虑进来,使得排序结果难以满足不同网络服务的需求,这是因为 SDN 网络中并没有事先定义好各个节点的网络功能,而是通过网络应用服务来定义的,某些不重要的网络节点在某些应用服务下也许会是很关键的节点.不同的网络应用服务对丢包率和时延的要求不同,比如电子邮件服务对时延的要求较低,视频语音服务则对时延的要求较高,在线支付服务则对丢包率要求相对较高等.由于网络应用服务要求的不同,同一个节点可能对于不同的网络应用服务的重要性差异很大,所以选取丢包率和时延作为固定的网络服务参数,当然网络服务参数也可以添加新的参数,可以由有相关经验的网络管理员指定,如网络入侵检测程序可将各项异常报警的次数添加到网络服务参数中,使得网络服务参数的信息更加丰富,排序结果也会更加精确.

目前也有一些降低 SDN 网络中 Controller 的负担的工作,文献[3]提出了一种新的数据包分类算法,通过结合现有的数据包头部识别技术对数据包进行更加快速地分类,提高 Controller 识别数据包的速度,从而降低 Controller 的工作负担.文献[4]提出了一种新的细粒度流量监测算法,通过把交换机的 TCAM 流表项分成两个部分,每个部分应用不同的优化技术来提高细粒度流量监测性能,从而对 Controller 流量监测工作进行了优化.上述两种方案都集中在 Controller 的基本工作上,而不是面向具体的应用服务,没有提出针对 Controller 上具体的应用服务来减轻其负担的方法.而随着网络应用服务的增加,它们会对 Controller 性能产生重要的影响.

本文利用 SDN 集中控制的优势,不但考虑了通信量、网络拓扑、Switch 与 Controller 交互信息这些网络参数,而且由于同一个网络节点在不同的网络服务中的重要性可能不同,所以本方案与所要提供的具体网络应用服务相结合,将网络服务参数考虑进来,提出一种 SDN 网络中面向服务的 Switch 节点重要性排序方法,为各种网络服务的信息采集和控制策略的下发提供了支撑.本文贡献如下:

(1)提出了 SDN 网络中面向服务的网络节点重要性排序方法.该方法不但考虑了 Switch 的静态和动态网络信息,而且结合具体的网络服务,选取对应的网络参数和服务参数,并赋予参数权重,经过拟合得到量化节点重要性的指标和排序结果.据我们所知,这是第一个 SDN 网络中面向服务的网络节点重要性排序方法.

(2)我们以 SDN 中检测 TCP 洪泛攻击服务为例进行了大量的实验.结果表明,当对整个网络排名在前 40%的节点进行信息收集时,能够减少控制器 60%的负担,还可以保证攻击检测程序成功率高达 94%,并且所提方案本身给 Controller 增加的负担非常有限.

2 相关工作

2.1 传统网络中的网络节点重要性排序方法

2.1.1 基于节点近邻的排序

该方法主要通过分析节点的邻居信息来评价节点的重要性,文献[5]提出了一种基于半局部信息的节点重要性排序算法,半局部信息指的是节点的四阶邻居信息,总邻居数目越多,节点越重要.该算法认为四阶邻居节点数目相同则重要性相同,忽略了网络位置对节点重要性的影响.文献[6]针对文献[5]中忽略节点位置这一问题提出用 K-壳分解法评价节点重要性,首先循环地把邻居数为 1 的节点及其连接的边全部去掉,直至所剩节点中没有度为 1 的节点为止,去掉的所有节点组成 1-壳,接下来去除节点度为 2 的节点及其邻边,得到 2-壳,以此类

推，得到 k-壳，此算法认为处于内层的节点拥有较高的影响力，该算法由于划分出的层数过少，所以对大规模网络节点的区分度不大. 文献[7-8]均针对文献[6]中划分层数过少的问题进行了改进，使得划分层数得到大大提升，节点排序结果也会更加精确，但是都忽略了网络中往往存在着度(即邻居节点的个数)很小但很重要的桥节点这一事实，文献[9]中对复杂网络中的传播动力学展开研究，发现很多复杂网络中存在着桥节点，并且能够通过桥节点提高疾病传播的可预测性. 文献[10]中提及到了桥节点，由该桥节点引发了环形网络的单点故障问题，如果某一个节点发生故障，整个网络都会瘫痪.

2.1.2　基于路径的排序

该方法主要通过分析节点间的最短路径来确定节点重要性指标，文献[11]中提及的接近中心性算法通过计算目标节点与网络中其他所有节点的最短路径长度的均值来衡量目标节点的重要性，均值越小，节点接近中心性越大，节点就越重要，但该方法仅仅考虑了最短路径，没有考虑其他路径，并且该方法的复杂度太高. 文献[12]中提及的 Katz 中心性方法在接近中心性方法[11]的基础上还考虑了网络中的非最短路径，并对不同长度的路径加权来评价节点重要性，路径长度越短，权值越大，虽然评价结果更加精确，但只适用于网络规模较小、无环路的网络中，具有一定的局限性，并且计算复杂度较高. 文献[13]中提及的介数中心性方法认为经过一个节点的最短路径数越多，这个节点就越重要，由于时间复杂度较高使其应用受到限制.

2.1.3　基于特征向量的排序

该方法主要通过分析节点的近邻数量以及近邻的重要性来确定节点的重要程度，文献[14]中提及的方法是一种基于特征向量的方法，该方法不仅要考虑节点的邻居数量，也要考虑节点所处的周围环境，即邻居节点的重要性，在邻居节点数相同的情况下，邻居节点越重要，那么该节点也相对越重要，通过求解特征向量的方法得到节点的重要性指标，但该方法的收敛时间较长. 文献[15]中提及的 PageRank 算法是 Google 搜索引擎的核心算法，它认为一个页面的重要性取决于指向它的其他页面的数量以及那些页面的重要性，虽然该方法收敛性有了很大提升，但该方法仅适用于各节点之间有相互引用关系的网络中.

2.2　SDN 网络中 Controller 负担减轻的方法

目前已有降低 SDN 网络中 Controller 负担的相关研究，文献[3]提出了一种新的数据包分类算法，通过结合现有的数据包头部识别技术对数据包进行更加快速地分类，提高 Controller 识别数据包的速度，进而降低 Controller 的工作负担. 文献[4]提出了一种新的细粒度流量监测算法，通过把交换机的 TCAM 流表项分成两个部分，每个部分应用不同的优化技术来优化 Controller 的流量监测工作. 上述两种方案虽然都能够一定程度的减轻 Controller 负担，但是它们都集中在减轻 Controller 基本工作的负担上，没有针对 Controller 上多种应用服务来减轻其负担. 而在 SDN 网络中，Controller 将要承载大量的应用服务，这会导致网络性能的严重下降[4]，降低其服务质量，乃至出现网络崩溃的情形[16]，所以有必要提出一个面向多种服务的减轻 Controller 负担的方案.

3　本文方案

本文提出一种 SDN 网络中面向服务的 Switch 节点重要性排序方法. 对于每一类网络服务，该方法运用主成分分析法对网络参数和服务参数进行降维处理，并且赋予权重，最终得到量化节点重要性的指标，通过指标比较得到节点重要性排序结果. 根据排序结果，选取出适当比例的网络节点，Controller 仅对这些重要节点进行本类网络服务信息的采集，通过收集到的网络服务信息去评估全网节点的网络状态，也为控制策略的下发提供了一定的参考，从而达到降低 Controller 负担的目的，由于本方法与具体网络服务相结合，不同应用服务得到的排序结果可能会不同，从而解决了某些不重要节点在其它的应用服务中可能会是很关键的节点这一问题.

3.1　参数选取

本算法根据 SDN 网络集中控制以及能够获取全局参数的特性，结合具体的网络服务信息，选取出可以有效反映转发节点重要性的网络参数和网络应用服务参数. 网络参数根据 SDN 网络中的标准通信协议 OpenFlow[17] 所提供的接口来获取，而服务参数依赖于具体的网络应用服务.

3.1.1　通信量

通信量是 Switch 节点在单位时间内发送的数据量. 由于转发节点的最根本的工作就是完成数据包的转发，因此该参数具有衡量转发结点重要性的能力，同时 OpenFlow 协议的 Counter 字段有该项信息的统计，使得该项信息收集十分简便，对网络造

成的影响非常小，其数学表示如下：

$$CN_i = cn_i(t) - cn_i(t-1),\ i=1,2,\cdots,m \quad (1)$$

其中，CN_i 表示 v_i 节点的通信量参数，$cn_i(t)$ 表示 v_i 节点到 t 时刻累计发送的数据量，m 代表网络中的节点数目.

3.1.2 网络拓扑

由于网络节点的性能与节点的物理布局息息相关，因此网络拓参数可以用来衡量一个网路节点在网络内的重要程度. 本文所讨论的拓扑图是无向无权连通图，记为 $G=(V,E)$，其中 $V=\{v_1,v_2,\cdots,v_m\}$ 为图 G 的节点集合，$E=\{e_1,e_2,\cdots,e_n\}$ 为图 G 的边集合，m 和 n 分别是图 G 的节点数和边数. 图 G 的邻接矩阵记为

$$\boldsymbol{A}_{m\times m} = \begin{bmatrix} a_{11},a_{12},\cdots,a_{1m} \\ a_{21},a_{22},\cdots,a_{2m} \\ \cdots \\ a_{m1},a_{m2},\cdots,a_{mm} \end{bmatrix},$$

当 $a_{ij}=1(1\leq i,j\leq m)$ 当且仅当节点 v_i 与 v_j 之间有连边，否则 $a_{ij}=0$.

由于网络拓扑的复杂性，运用单一参数无法全面衡量节点重要性，因此本方案通过下述三种参数对其进行衡量. 同时，Controller 拥有全局网络实时拓扑信息，所以网络拓扑获取不会额外增加 Controller 的工作量. 另外，即使网络的拓扑结构发生了变化，Controller 也能够及时感知，并进行重新排序. 选取参数具体如下：

(1) 离心中心性

该方法考察节点到网络中其他节点的最大距离，由于 SDN 网络中 Controller 具有全局网络拓扑信息，该参数很容易能够获取到. 即

$$ECC_i = \max_j(d_{ij}),\ i,j=1,2,\cdots,m \quad (2)$$

其中，ECC_i 表示 v_i 节点的离心中心性参数，d_{ij} 表示节点 v_i 与 v_j 之间最短路径的长度，$\max_j(d_{ij})$ 则表示节点 v_i 到其他所有节点的距离的最大值，m 代表网络中的节点数目.

(2) 接近中心性

该方法是基于路径的排序方法，考察节点到网络中其他节点的平均距离，由于 SDN 网络中 Controller 具有全局网络拓扑信息，该参数很容易能够获取到，并且当网络拓扑发生变化时，Controller 可以及时感应并更新该参数数据. 其数学表达式如下：

$$CC_i = \frac{m-1}{\sum_{j\neq i} d_{ij}},\ i,j=1,2,\cdots,m \quad (3)$$

其中，CC_i 表示节点 v_i 的接近中心性参数，d_{ij} 表示节点 v_i 与 v_j 之间最短路径的长度，m 代表网络中的节点数目.

(3) 度中心性

该方法是基于节点近邻的排序方法，考察节点的邻居数目，即邻居数目越多，节点相应越重要，其数学表达如下：

$$DC_i = deg_i,\ i=1,2,\cdots,m \quad (4)$$

其中，deg_i 表示节点 v_i 的度数，DC_i 表示节点 v_i 的度中心性参数，m 代表网络中的节点数目.

3.1.3 转发节点活跃度

该参数表示节点参与网络活动的活跃程度，结合 SDN 的具体场景，选取了下述三个参数来衡量节点的活跃度.

(1) 转发频率

数据包转发频率是单位时间内转发节点发送的数据包的数量. 由于网络中存在一些只携带少量数据的数据包，这些数据包通常被用来实现各种协议的目标功能，具有重要意义，然而对这部分数据包转发的工作意义无法通过数据量来衡量. 因此，选取该参数来衡量这部分工作的意义，即：

$$EF_i = \frac{ef_i(t)-ef_i(t-1)}{\Delta t},\ i=1,2,\cdots,m \quad (5)$$

其中，EF_i 表示节点 v_i 的转发频率，$ef_i(t)$ 表示节点 v_i 到 t 时刻总共发送的数据包数量，m 代表网络中的节点数目.

(2) Asynchronous 消息频率

在数据转发平面与控制平面的跨平面通信中，OpenFlow 协议定义了 Asynchronous 消息，该消息主要包括 Switch 向 Controller 发送的请求消息，Asynchronous 消息频率指单位时间内转发节点 Switch 向 Controller 发送的 Asynchronous 消息的次数，刻画了 Switch 对 Controller 的依赖程度，从一定程度上反映了 Switch 节点参与网络活动的活跃度. 另外，Asynchronous 消息最终都会传送到网络控制设备 Controller 上，所以直接从 Controller 上便可获取该参数信息，信息获取几乎不会消耗 Controller 的资源. 其数学表示如下：

$$QF_i = \frac{qf_i(t)-qf_i(t-1)}{\Delta t},\ i=1,2,\cdots,m \quad (6)$$

其中，QF_i 表示节点 v_i 的节点请求频率，$qf_i(t)$ 表示 v_i 节点到 t 时刻累计发送的 Asynchronous 消息的次数，m 代表网络中的节点数目.

(3) Controller-to-Switch 消息频率

在数据转发平面与控制平面的跨平面通信中，

OpenFlow 协议还定义了从 Controller 端发起的 Controller-to-Switch 消息，Controller 通过该消息对 Switch 进行日常管理与维护. Controller-to-Switch 消息频率指单位时间内 Switch 收到的来自 Controller 的 Controller-to-Switch 消息的次数. 在 SDN 网络中，一个转发节点 Switch 接收到的 Controller-to-Switch 消息次数越多，说明该节点参与的网路活动越多，因此选取该参数来衡量节点的活跃度. 因为该类消息都由 Controller 端发出，所以相关信息可以在 Controller 端获取，并且不会给 Controller 带来太大的负担. 其数学公式如下：

$$CF_i=\frac{cf_i(t)-cf_i(t-1)}{\Delta t},\ i=1,2,\cdots,m \quad (7)$$

其中，CF_i 表示节点 v_i 的流表修改频率，$cf_i(t)$ 表示 v_i 节点到 t 时刻累计修改的流表次数，m 代表网络中的节点数目.

3.1.4　网络服务信息

随着网络需求的快速增长，Controller 设备上部署了越来越多的网络服务，这些服务在带来便利的同时，也消耗着大量的 Controller 资源. 我们除了可以通过简化程序、提高算法效率等方法降低 Controller 负担，还可以通过对网络节点进行重要性排序，仅对重要的网络节点进行选择性的重点部署，通过收集重要节点的信息来达到评估全网节点状态的目的，从而降低了 Controller 的负担.

网络应用服务参数是与具体的网络服务有关的参数，不同的网络应用服务对丢包率和时延的要求不同，比如电子邮件服务对时延的要求较低，视频语音服务则对时延的要求较高，在线支付服务则对丢包率要求相对较高等. 由于网络应用服务要求的不同，同一个节点可能对于不同的网络应用服务的重要性差异很大，所以选取丢包率和时延作为固定的网络服务参数，当然该类参数也可以添加新的参数，可以由有相关经验的网络管理员指定，如网络入侵检测程序可将各项异常报警的次数添加到网络服务参数中，使得网络服务参数的信息更加丰富，排序结果也会更加精确.

本文用 $TS_i=\{S_{i1},S_{i2},\cdots,S_{in}\}$ 表示 v_i 节点的网络应用服务信息，其中 $S_{i1},S_{i2},\cdots,S_{in}$ 表示所选取的各项网络服务参数，这些参数还要经过参数归一化以及历史参数的后续处理，才能作为方案的参数输入.

3.2　方案工作流程

本方案利用 SDN 的集中控制特性方便的获取到上述介绍的各种参数信息，并且在 Controller 上建立数据处理模块，最终将计算结果通知给网络服务程序，使其可以在数据转发平面进行重点部署，最终实现优化网络资源，特别是降低 Controller 负担的目的.

本文方案通过对获取到的转发节点的多元组数据的量化分析，得出网络节点重要性的量化数据. 首先进行参数归一化处理，从而消除各个参数数值上的差异，对于一些具有历史性特征的参数通过使用层次分析法进行参数处理，得到处理后的参数信息，接下来通过主成份分析法确定各个参数所占的权重，最后通过加权求和将参数进行最终拟合，得到每个节点的重要性指标，通过比较指标大小得到节点的重要性排序结果，指标值越大，则节点越重要.

3.2.1　参数归一化

为避免各参数数值上的差异对最终分析结果造成影响，本算法先将原始参数数值进行归一化处理，本文采用最大-最小值归一化方法，其数学表示如下：

$$x_i=\frac{a_i-\min\limits_{j=1,\cdots,m}a_j}{\max\limits_{j=1,\cdots,m}a_j-\min\limits_{j=1,\cdots,m}a_j} \quad (8)$$

其中 x_i 为该参数归一化后的结果，a_i 为节点 v_i 的原始参数值，$\max\limits_{j=1,\cdots,m}a_j$ 为所有节点的该参数的最大取值，$\min\limits_{j=1,\cdots,m}a_j$ 为所有节点该参数的最小取值.

3.2.2　历史参数处理

网络运行状态一般都是连续变化的，当前所处的状态与历史状态（以前的状态）有一定的关联性，也就是具有历史性特征. 因此在评价具有历史性特征的参数时，为了使评价结果更加精确，同时消除偶然性，不仅要考虑这些参数的当前数值，还要考虑其历史数值. 本文提及的网络服务参数、通信量、转发节点活跃度这三类参数都具有历史性特征，因此利用历史数据和当前数据对这三类参数进行一次处理，即为每个历史数据和当前数据赋予权重，再通过加权求和的方式求出代表该参数的数值，计算公式如下：

$$P=\sum_{j=1}^{n}\alpha_j T_j,\ j=1,2,\cdots,n \quad (9)$$

其中，P 表示经过处理后的参数数值，如通信量、Asynchronous 消息频率等，T_j 表示该项参数的第 $j-1$ 次历史数据，α_j 表示 T_j 的权重.

由于 $T_j(j=1,2,\cdots,n)$ 两两之间的相对关系依据时序的先后顺序很容易确定，距离当前时间越近，那么该项历史数据就越重要，对应的权重也会越大，而层次分析法（Analytic Hierarchy Process，AHP）

恰恰可以通过参数间的两两相对关系将所有参数拟合成一个量化指标，任何两项历史性参数之间都会有明确的时序先后关系，也就是说任何两项历史性参数都会有明显的重要性差别. 因此本算法选用AHP确定权重 α_j 和 P[18]. 其算法如下：

算法 1. 历史参数处理算法.

输入：$T=\{T_1,T_2,\cdots,T_j,\cdots,T_n\}$，某时刻转发节点某指标的当前值及历史值

输出：$\boldsymbol{P}$ 综合了历史信息的拟合值

1. 由时序信息构造判断矩阵 $\boldsymbol{A}$；

2. 将判断矩阵 $\boldsymbol{A}$ 的每一列归一化：$\bar{b}_{ij}=b_{ij}\Big/\sum_{k=1}^{n}b_{kj}$ $(i,j=1,2,\cdots,n)$；

3. 将归一化后的矩阵按行求和：$\bar{\omega}_i=\sum_{j=1}^{n}\bar{b}_{ij}$ $(i,j=1,2,\cdots,n)$；

4. 对向量 $\bar{\boldsymbol{W}}=[\bar{\omega}_1,\bar{\omega}_2,\cdots,\bar{\omega}_n]^{\mathrm{T}}$ 进行归一化：$\omega_i=\bar{\omega}_i/\sum_{k=1}^{n}\bar{\omega}_k(i=1,2,\cdots,n)$，得到特征向量 $\boldsymbol{W}=[\omega_1,\omega_2,\cdots,\omega_n]^{\mathrm{T}}$；

5. 求得特征根值：$t_{\max}=\sum_{i=1}^{n}(AW)_i/n\omega_i$，计算一致性指标 $CI=(t_{\max}-n)/n+1$，对照平均随机一致性指标进行一致性检验，如果不通过则调整判断矩阵 $\boldsymbol{A}$，跳转到步1；

6. 对权值向量归一化：$\alpha_i=\omega_i\Big/\sum_{k=1}^{n}\omega_k(i=1,2,\cdots,n)$，得到最终权值：$\boldsymbol{\alpha}=[\alpha_1,\alpha_2,\cdots,\alpha_n]^{\mathrm{T}}$；

7. 通过加权求和得到 $\boldsymbol{P}$：$\boldsymbol{P}=\sum_{j=1}^{n}\alpha_j T_j$.

在算法1中，首先通过 T_{ij} 的时序信息初始化判断矩阵 $\boldsymbol{A}$，其初始化过程如下：

$$\boldsymbol{A}=\begin{bmatrix}b_{11}&\cdots&b_{1n}\\ \vdots&\ddots&\vdots\\ b_{n1}&\cdots&b_{nn}\end{bmatrix},$$

其中，b_{ij} 代表 T_i 与 T_j 相对关系，其数学表达式如下：

$$b_{ij}=\begin{cases}1, & T_i \text{ 与 } T_j \text{ 同样重要}\\ 2\sim3, & T_i \text{ 比 } T_j \text{ 稍微重要}\\ 4\sim5, & T_i \text{ 比 } T_j \text{ 重要，且 } b_{ij}=1/b_{ji}\\ 6\sim7, & T_i \text{ 比 } T_j \text{ 明显重要}\\ 8\sim9, & T_i \text{ 比 } T_j \text{ 非常重要}\end{cases}.$$

之后通过矩阵运算得出判断矩阵 $\boldsymbol{A}$ 的特征向量 $\boldsymbol{W}$ 和特征根 $\omega_1,\omega_2,\cdots,\omega_n$，再通过特征根进一步验证判断矩阵 $\boldsymbol{A}$ 是否具有一致性，若检验未通过，则重新构造判断矩阵，若检验通过，则可应用本次计算的特征根求得权重 α_j，最终获得 P.

在算法1中，Controller首先通过时序关系构造判断矩阵，此时需要的时间复杂度为 $O(n^2)$，随后再对矩阵每一列进行归一化处理，该步骤需要的时间复杂度是 $O(n^2)$，再对归一化后的矩阵按行求和，其时间复杂度为 $O(n^2)$，对向量进行归一化处理得到特征向量这一步骤需要的时间复杂度是 $O(n)$，求特征根的时间复杂度是 $O(n^2)$，得到最终权值需要的时间复杂度是 $O(n)$，由于历史性参数之间有明显的时序先后关系，故一次性就能通过一致性检验，故一致性检验阶段的时间复杂度为 $O(1)$，加权求和的时间复杂度为 $O(n)$，故算法1的时间复杂度为 $O(n^2)$.

3.2.3 参数最终拟合

经过参数归一化和历史性参数处理两个步骤后，参数完成了所有的前期处理，得到了各项参数用于最终拟合的数据，本文通过主成份分析法对所有的拟合参数赋予权值，最后再对所有的拟合参数进行加权求和. 本文通过以下公式将所有拟合参数进行加权求和，得到节点的重要性指标.

$$s_i=l_1CN_i+l_2TP_i+l_3VD_i+l_4TS_i \tag{10}$$

其中，s_i 表示节点 v_i 的重要性指标，l 表示节点 v_i 的各参数的权重. CN_i 代表节点 v_i 的通信量参数，TP_i 代表节点 v_i 的网络拓扑参数，VD_i 代表节点 v_i 的节点活跃度参数，TS_i 代表节点 v_i 的网络服务参数.

另外，网络拓扑参数、节点活跃度参数和网络服务参数包含多个子参数，其计算公式如下：

$$TP_i=l_1ECC_i+l_2CC_i+l_3DC_i,\quad i=1,2,\cdots,m \tag{11}$$

$$VD_i=l_1EF_i+l_2QF_i+l_3CF_i,\quad i=1,2,\cdots,m \tag{12}$$

$$TS_i=\sum_{k=1}^{n}l_{ik}S_{ik},\ i=1,2,\cdots,m \tag{13}$$

由于各个参数意义各不相同，因此无法主观地判断各参数强弱关系. 而主成分分析法(Principal Components Analysis，PCA)根据各参数的离散程度来确定权重，并且计算速度快，复杂度低. 因此本算法选用PCA确定各拟合公式中的权重 λ_j 和最终拟合结果[19]. 其算法如算法2.

算法 2. 参数拟合算法.

输入：n，参数个数

m，全网节点数目

$v_i=\{p_{i1},p_{i2},\cdots,p_{in}\}$，转发节点 v_i 的各项参数

输出：拟合结果 $\boldsymbol{P}$

1. 初始化输入变量，并生成记录所有节点输入参数的参数矩阵 $\boldsymbol{A}(m\times n)$；

2. 将参数矩阵 $\boldsymbol{A}$ 的每一列归一化：$\bar{b}_{ij}=\sum_{k=1}^{m}b_{kj}$ $(i=1,2,\cdots,m)$；

3. 将归一化后的矩阵按行求和：$\bar{\omega}_i=\sum_{j=1}^{n}\bar{b}_{ij}$ $(i=1,2,\cdots,m)$；

4. 对向量$\overline{\boldsymbol{W}}=[\bar{\omega}_1,\bar{\omega}_2,\cdots,\bar{\omega}_n]$进行归一化：$\omega_i=\bar{\omega}_i\Big/\sum_{k=1}^{m}\bar{\omega}_i$ $(i=1,2,\cdots,m)$，并计算特征向量，记作$\boldsymbol{W}=[\omega_1,\omega_2,\cdots,\omega_n]^{\mathrm{T}}$；

5. 求得各项参数权重：$\lambda_j=\omega_j\Big/\sum_{i=1}^{n}\omega_i$ $(j=1,2,\cdots,n)$；

6. 求得节点参数拟合结果：$P_i=\sum_{j=1}^{n}\lambda_j p_{ij}$ $(i=1,2,\cdots,m)$；

7. 求得全网拟合矩阵：$\boldsymbol{P}=[P_1,P_2,\cdots,P_m]^{\mathrm{T}}$.

在上述算法中，首先计算矩阵 $\boldsymbol{A}$ 的特征矩阵 $\boldsymbol{W}$，再根据特征矩阵 $\boldsymbol{W}$ 得出各项参数的权重 λ_j，最后对各项参数加权求和得到拟合结果 P_i，从而得到记录全网节点拟合结果的矩阵 $\boldsymbol{P}$. 以计算 TP_i 为例，首先建立全网信息矩阵 $\boldsymbol{A}(m\times 3)$，矩阵包括全网节点的 ECC_i、CC_i 和 DC_i 信息，通过各列（每列代表一个参数）数据的离散程度得出矩阵 $\boldsymbol{A}$ 各列的权重 λ_1、λ_2 和 λ_3，再根据每列的权重对矩阵 $\boldsymbol{A}$ 的每行（每行代表一个节点）做加权拟合得到每行的 TP_i 信息，以此类推，最终得到每个节点的参数拟合结果 P_i，通过比较 P_i 的大小，得到节点的重要性排序结果，P_i 越大，节点就越重要.

在算法 2 中，Controller 首先生成参数矩阵，此时需要的时间复杂度为 $O(n^2)$，随后再对矩阵每一列进行归一化处理，该步骤需要的时间复杂度是 $O(n^2)$，再对归一化后的矩阵按行求和，其时间复杂度为 $O(n^2)$，对向量进行归一化处理得到特征向量这一步骤需要的时间复杂度是 $O(n)$，求特征根的时间复杂度是 $O(n^2)$，得到最终权值需要的时间复杂度是 $O(n)$，加权求和的时间复杂度为 $O(n)$，故算法 2 的时间复杂度为 $O(n^2)$.

4　实验及数据分析

由于本文提出的排序方法是面向服务的，除了网络服务中固有的时延和丢包率两个参数之外，其它的服务参数会应网络应用服务的不同而变化. 我们以 SDN 中检测 TCP 洪泛攻击服务为例进行实验，在所提方法的指导下选取出对该网络服务重要的节点，仅通过对这些节点进行信息采集来实现攻击检测，我们通过在不同规模网络中的仿真实验来验证方法的有效性.

4.1　实验环境

实验的物理机配置为 3.30 GHz CPU 和 4 GB 内存. 使用 mininet 和 Pox 仿真程序搭建了三种不同规模的网络，这三种规模的网络分别拥有 100、200 和 300 个 Switch 节点和与 Switch 节点数目相同主机，其中 Switch 节点与一个主机直接相连. Switch 间的随机网络由程序随机生成的连通图决定，利用 Scapy 模拟了 TCP 洪泛攻击，TCP 洪泛攻击检测程序选自网络开源程序①，假设实验过程中服务内容需要长期保持不变.

4.2　实验数据分析

4.2.1　参数选取必要性分析

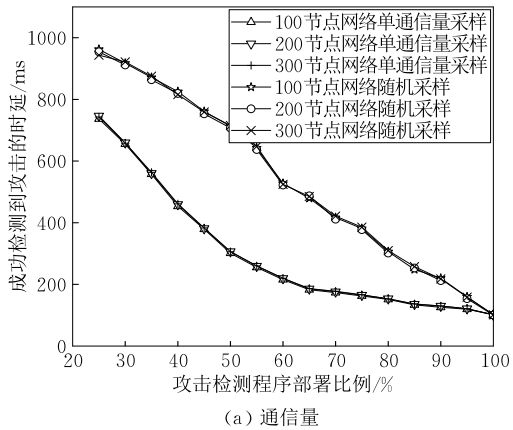
通过实验，我们对每个参数选取的必要性进行了验证. 由于网络服务参数由网络管理人员根据所要优化的网络服务具体制定，不同的网络服务对应不同的服务参数，所以本文并没有对该参数进行必要性验证. 对每个参数进行验证时，我们直接把其它参数从本算法中删除，仅依据单一参数对节点进行排序，从而得出单一参数状态下 TCP 洪泛攻击检测程序采样比例与攻击检测程序成功检测到攻击的时延的关系. 通过与随机选取结果进行对比，验证了选取参数的必要性. 实验结果如图 1 所示.

由图 1 可知，在三种网络规模下，无论是随机采样还是单参数采样，随着 TCP 洪泛攻击检测程序节点部署比例的增长，攻击检测程序成功检测到攻击的时延都会随之减少. 同时，在攻击检测程序部署比例相同的情况下，每个单参数对应的成功检测到攻击的时延都远远低于随机采样对应的成功检测到攻击的时延，说明了这些参数对攻击检测程序有着重要的影响，验证了参数选取的有效性.

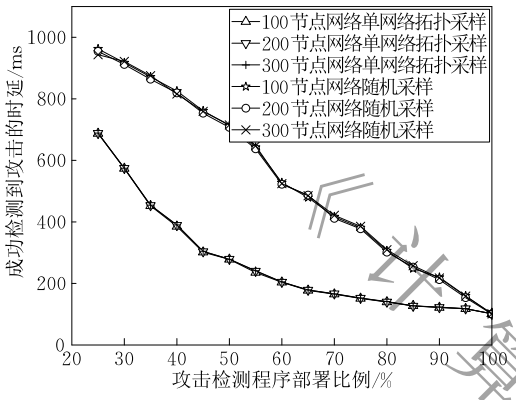
4.2.2　TCP 洪泛攻击检测程序采样比例与 Controller 响应速率的关系

由于 mininet 仿真程序无法实际显示控制器的工作参数，如：CPU 使用率、存储器使用率等，但可以显示 Controller 和每个 switch 通信的信息. 考虑到 Controller 的一个主要工作就是为新生通信制定和下发相应流表项，当网络首次 ping 一个 IP 地址的时候，主要时延是由 Controller 制定和下发该 ping 命令到达目的 IP 地址的流表项产生的（而之后对于相同目的地址的 ping 命令，Switch 按照 Controller 在首 ping 时下发的流表项直接进行转发，而不需要请求 Controller 再次为该命令制定流表项）. 因此在仿真实验中，本实验选用网络首 ping 时延来衡量 Controller 响应速率，网络首 ping 时延越长，表明 Controller 响应速率越慢，其工作负担越重. TCP 洪泛攻击检测程序采样比例与 Controller 响应速率的关系如图 2 所示.

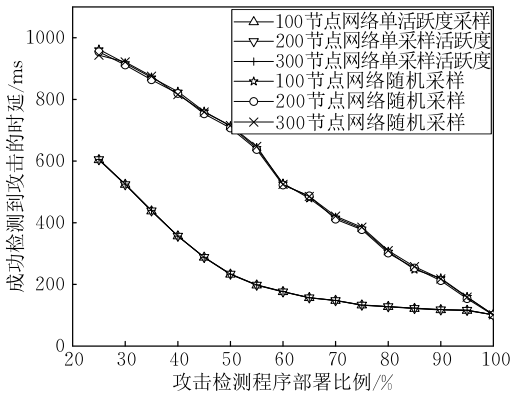
① https://mailman.stanford.edu



(a) 通信量



(b) 网络拓扑



(c) 活跃度

图 1 单一参数对成功检测到攻击时延的影响

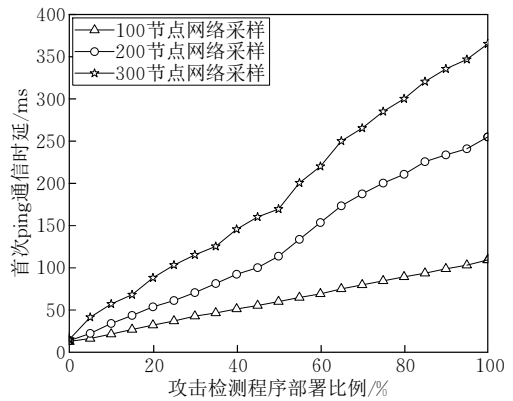


图 2 攻击检测程序节点部署比例对 Controller 响应速率的影响

以图 2 为例,在三种规模的网络中,随着采样比例的增长,首 ping 时延也逐渐增大,代表了 Controller 工作负担增加导致其响应速率的降低. 可见攻击检测程序的大规模采样活动对 Controller 的负担有着严重的影响. 当采样比例达到 100%时,拥有 300 节点网络的首 ping 时延高达 300 ms,而访问百度、新浪等网站仅需 20 ms 左右,可见其对 Controller 负载造成相当大的影响. 因此,有必要对其进行优化.

4.2.3 攻击检测程序采样比例对其服务效果的影响

实验中,网络服务选取的是 TCP 洪泛攻击检测程序,因此网络服务质量可用攻击检测程序成功检测到攻击的时延和攻击检测成功率来衡量. 实验把本方案选取结果与随机选取结果对应的成功检测到攻击的时延进行了对比,其效果如图 3 所示. 攻击检测程序节点部署比例与攻击检测成功率的关系如图 4 所示.

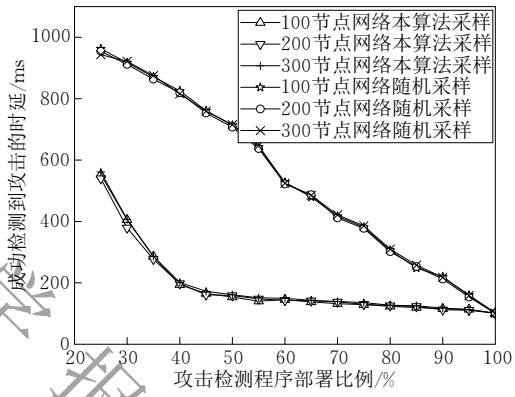


图 3 攻击比例与成功检测到攻击的时延检测程序部署的关系

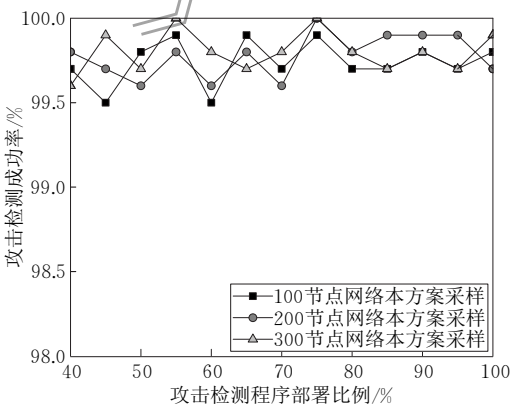


图 4 攻击检测程序部署比例与攻击检测成功率的关系

在图 3 中,随着攻击检测程序节点采样比例的增大,成功检测到攻击所需的时延随之减少,检测程序对攻击的响应效率也随之提高. 需要注意的是:一方面,虽然部署比例的增加会加重 Controller 的负担,但收集的信息却更加丰富全面,因此成功检测到

攻击的时延仍然会减少；另一方面，当部署比例相同时，不同网络规模下成功检测到攻击所需的时延差异不大，网络规模的增大虽然会加重 Controller 的负担，但是部署比例相同的情况下，网络规模越大对应的节点部署个数越多，当然收集的信息也会更多，所以在节点部署比例相同的情况下不同规模的网络造成的差异并不明显. 与随机方式相比，本方案对于减少攻击检测的时延有着较显著的作用，说明了方案的有效性. 在本方案中，当采样比例达到一定比例(40%～50%)后，随着采样比例增长，检测到攻击所需的时延逐渐趋于平缓，可见采样的比例应该控制在 40%以上.

如图 4 所示，当采样比例在一定范围(40%～100%)变化时，攻击检测成功率都比较高，采样比例在该区间的变化对攻击检测成功率的影响不大. 结合图 2，当节点采样比例为 40%时，Controller 的首 ping 通信时延比节点全网采样时缩短了约 60%的时间，也就是减少了控制器 60%的负担. 因此，可通过本方案对网络中重要节点进行选择性信息采集来检测全网的受攻击的态势，减轻 Controller 负载. 在保持较好网络应用服务质量的前提下，尽可能地减少对 Controller 负载造成的影响.

4.2.4　节点选择的稳定性

实验中通过对本方案和随机方案节点选择相似度(本次选取结果与上次选取结果中相同节点所占的比例)进行对比，来衡量方案节点选择的稳定性. 结果如图 5 所示.

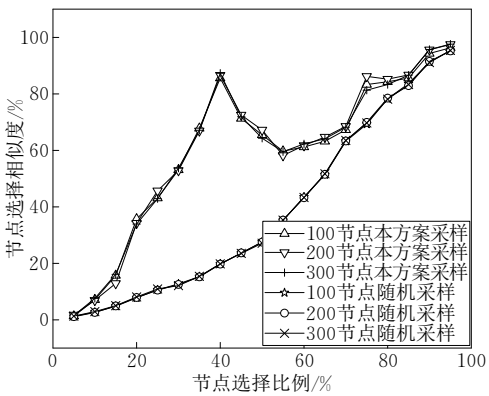


图 5　节点选择比例与选择相似度的关系

在图 5 中，在三种规模的网络下，本方案的选择相似度始终高于随机采样的选择相似度，说明其具有更高的稳定性，且网络规模的变化对节点稳定性几乎不会产生影响. 本文的节点选择比例在 40%左右时出现一个峰值，说明对于 TCP 洪泛攻击检测程序来说，网络中存在 40%左右的重要节点.

4.2.5　算法有效性分析

我们按照本方案排序结果依次破坏排名靠前的 Switch 节点[20]，通过首 ping 通信时延的变化来反映被破坏节点对全网造成的影响，影响越大说明所破坏的节点越重要，从而说明方案的有效性. 因为如果重要的 Switch 节点被删除了，Controller 将以很大概率需要重新为 Ping 命令选择路由和制定流表，以到达目的 IP 地址，这会增加首 ping 时延；如果删除的 Switch 节点不是很重要，可能重新为 Ping 命令选择路由的概率比较低，时延增加不明显. 为了说明方案的有效性，我们随机地破坏相同数量的节点，观察其对通信时延的影响，并与本算法对应的通信时延作比较. 由于时延数量级比较小(Ping 通全网节点的平均时延是 0.103ms)，所以我们采取增长比例的方法. 实验结果如图 6 所示.

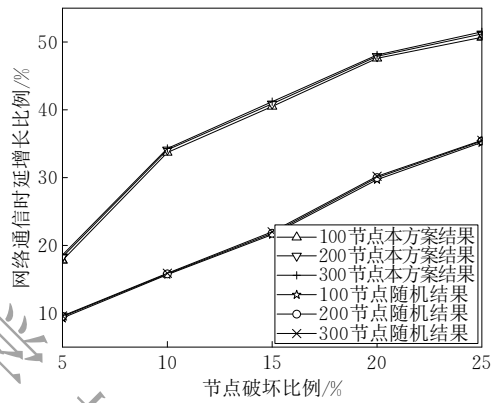


图 6　节点破坏比例与网络通信时延的关系

如图 6 所示，当删除节点达到一定比例时，ping 不通的情况比较普遍，Controller 将会重新为 Ping 命令选择路由和制定流表，这会对网络通信造成较大的影响. 由图 6 可以看出，随着节点破坏比例的增加，网络通信时延也随之增加，并且破坏本方案产生的排名靠前的 Switch 节点对应的通信时延明显高于随机破坏节点对应的通信时延，说明选取的节点对于网络来说确实重要.

4.2.6　本方案与现有方案的对比

由于传统网络中的网络节点重要性排序方法主要有三类，本文从三类排序方法中分别选取出一个最新的方案与本文方案进行对比，在基于节点近邻的排序方法中，本文选取了文献[8]，基于路径的排序方法中，本文选取了文献[11]，基于特征向量的排序方法中，本文选取了文献[14].

由前面的实验结果可得，无论网络规模是 100 节点、200 节点还是 300 节点，当网络节点部署比例相同时，成功检测到攻击的时延差异非常小，即成功

检测到攻击的时延与网络规模无关.当节点破坏比例相同时,网络通信时延增长比例差异也非常小,即网络通信时延增长比例与网络规模无关.由于网络规模对实验结果几乎没有影响,所以与现有方案进行对比时,本文仅采用 300 节点的网络规模.

图 7 给出了各种方案成功检测到攻击的时延与攻击检测程序部署比例的关系,由图 7 可得,随着攻击检测程序部署比例的增长,各种方案对应的成功检测到攻击的时延都在降低,部署的节点越多,收集的信息也就越丰富,那么成功检测到攻击的时延就会越低.当网络节点部署比例相同时,现有排序方案对应的检测到攻击的时延全部高于现有方案的检测时延,说明本方案比现有排序方案更优.图 8 给出了本方案和现有方案的网络通信时延增长比例与节点破坏比例的关系,当节点破坏比例逐渐增大时,各方案对应的网络通信时延增长比例都在不断增长,当节点破坏比例相同时,现有排序方案对应的网络通信时延都低于本文方案的,说明本文方案破坏的节点更加重要,也就是说本文方案排序结果更加高效,从而说明了本文方案的有效性.

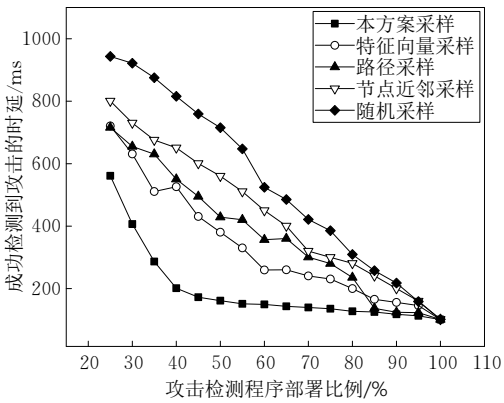


图 7 本方案与现有排序方案的成功检测到攻击的时延与攻击检测程序部署比例关系

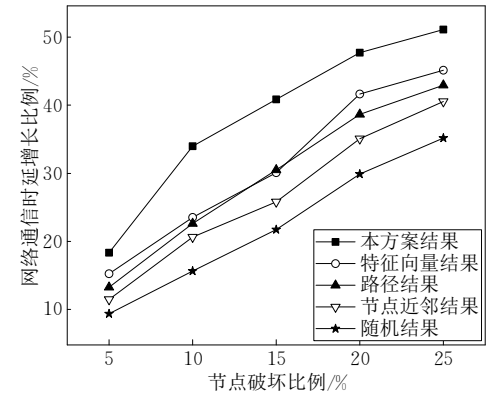


图 8 本方案与现有排序方案的网络通信时延增长比例与节点破坏比例的关系

4.2.7 网络服务参数的有效性

由于本文是 SDN 网络中面向服务的网络节点重要性排序方法,所以有必要对网络服务参数的选取进行验证,本文分别依据所有参数和去除掉网络服务参数的其他参数对网络节点进行排序,得到本方案和无服务参数采样对应的实验结果图,如图 9、图 10 所示.

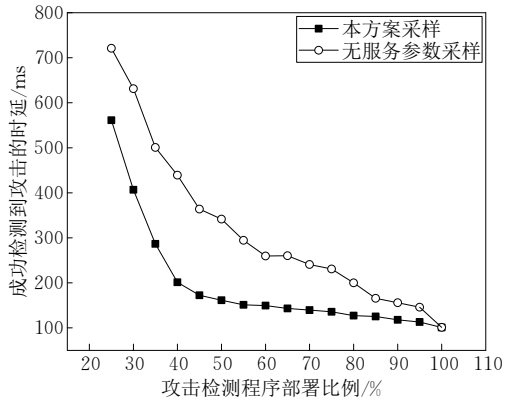


图 9 无服务参数和本方案对应的成功检测到攻击的时延与攻击检测程序部署比例的关系

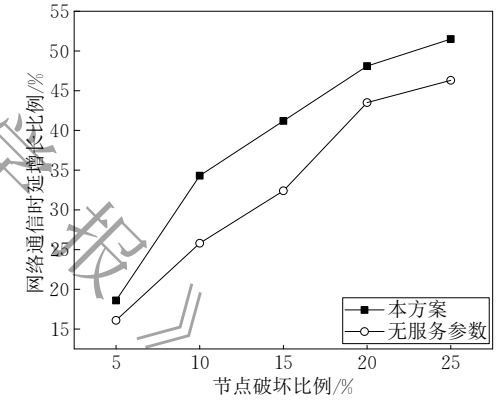


图 10 无服务参数与本方案的网络通信时延增长比例与节点破坏比例的关系

图 9 中,当网络节点部署比例相同时,有服务参数的方案比无服务参数的方案的成功检测到攻击的时延低,说明有服务参数的方案收集的信息越丰富,也就是说选取的节点会更重要,节点排序结果更精确.图 10 中,当节点破坏比例相同时,有参数的方案比无参数的方法对应的网络通信时延会更大,也就是说删除的节点更重要,综合图 9 和 10 的实验结果,说明了网络服务参数能够使得本文方案的排序结果更加精确.

4.2.8 方案本身带给 Controller 的负担

本节对方案本身给 Controller 带来的负担进行了分析.通过分析本方案运行前后的 Controller 的

平均首 ping 时延，得到方案本身对 Controller 响应速率的影响，计算公式如下：

$$PD_{add} = (PD_{after} - PD_{before})/PD_{before} \quad (14)$$

其中，PD_{before} 表示运行本方案之前 Controller 的平均首 ping 时延，PD_{after} 表示运行本方案之后 Controller 的平均首 ping 时延，PD_{add} 表示本方案的运行使得 Controller 的平均首 ping 时延增加的百分比，即本方案给 Controller 带来的负担.

通过 100 轮仿真实验，在没有部署攻击检测程序的情况下，即没有部署网络服务的情况下，我们得出：当网络规模为 100 个 Switch 节点时，Controller 在运行本方案前后的平均首 ping 时延分别是 12.51 ms 和 12.76 ms；当网络规模为 200 个 Switch 节点时，Controller 在运行本方案前后的平均首 ping 时延分别是 13.46 ms 和 13.73 ms；当网络规模为 300 个 Switch 节点时，Controller 在运行本方案前后的平均首 ping 时延分别是 14.84 ms 和 15.17 ms. 结合式(14)可得 Controller 的平均首 ping 时延仅增加了 2%，说明所提方案本身给 Controller 带来的负担非常有限.

5　结　论

针对大规模的 SDN 网络中 Controller 负载很重的问题，利用 Controller 集中控制能够实时获取更为全面丰富的网络信息的优势，我们将网络参数及其应用服务参数进行综合考虑，提出一种面向服务的网络节点 Switch 重要性排序方案，选取排名靠前的少量关键节点进行该类服务的信息的收集. 并以检测 TCP 洪泛攻击该网络应用服务为例，通过实验验证了该选取结果在保持较高的服务质量的同时，又能够有效减少网络服务大规模部署给 Controller 带来的负担. 并且所提方案本身给 Controller 带来的负担非常有限.

参　考　文　献

[1] Liu Y Y, Slotine J J, Barabási A L. Observability of complex systems. Proceedings of the National Academy of Sciences, 2013, 110(7): 2460-2465

[2] Liu Y Y, Slotine J J, Barabási A L. Controllability of complex networks. Nature, 2011, 473(7346): 167-173

[3] Perez K G, Yang X, Scott-Hayward S, et al. Optimized packet classification for software-defined networking// Proceedings of the 2014 IEEE International Conference on Communications. Sydney, Australia, 2014: 859-864

[4] Malboubi M, Wang L, Chuah C N, et al. Intelligent SDN based traffic (de) aggregation and measurement paradigm (iSTAMP)//Proceedings of the IEEE Conference on Computer Communications (IEEE INFOCOM). Toronto, Canada, 2014: 934-942

[5] Chen D, Lü L, Shang M S, et al. Identifying influential nodes in complex networks. Physica A: Statistical Mechanics and Its Applications, 2012, 391(4): 1777-1787

[6] Kitsak M, Gallos L K, Havlin S, et al. Identification of influential spreaders in complex networks. Nature Physics, 2010, 6(11): 888-893

[7] Shu Pan-Pan. The Research of Predictive Spread on Complex Networks [Ph. D. dissertation]. University of Electronic Science and Technology of China, Chengdu, 2015 (in Chinese) (舒盼盼. 复杂网络上的传播可预测性研究[博士学位论文]. 电子科技大学，成都，2015)

[8] Wu Y, Xu G N, Gou W H, et al. A design of mine electrical and mechanical equipment of EPON ring network//Proceedings of the 2015 International Conference on Electrical, Automation and Mechanical Engineering. Wuhan, China, 2015: 495-497

[9] Zeng A, Zhang C J. Ranking spreaders by decomposing complex networks. Physics Letters A, 2013, 377(14): 1031-1035

[10] Hu Q, Gao Y, Ma P, et al. A new approach to identify influential spreaders in complex networks//Proceedings of the International Conference on Web-Age Information Management. Beidaihe, China, 2013: 99-104

[11] Saryüce A E, Saule E, Kaya K, et al. Incremental closeness centrality in distributed memory. Parallel Computing, 2015, 47: 3-18

[12] Lü L, Zhou T. Link prediction in complex networks: A survey. Physica A: Statistical Mechanics and Its Applications, 2011, 390(6): 1150-1170

[13] Estrada E, Higham D J, Hatano N. Communicability betweenness in complex networks. Physica A: Statistical Mechanics and Its Applications, 2009, 388(5): 764-774

[14] Martin T, Zhang X, Newman M E J. Localization and centrality in networks. Physical Review E, 2014, 90(5): 052808_1-052808_7

[15] Brin S, Page L. The anatomy of a large-scale hypertextual web search engine. Computer Networks, 2012, 56(18): 3825-3833

[16] Duffield N. Fair sampling across network flow measurements. ACM SIGMETRICS Performance Evaluation Review, 2012, 40(1): 367-378

[17] Mckeown N, Anderson T, Balakrishnan H, et al. OpenFlow: Enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69-74

[18] Yagmur L. Multi-criteria evaluation and priority analysis for localization equipment in a thermal power plant using the AHP (Analytic Hierarchy Process). Energy, 2016, 94: 476-482

[19] Vaswani N, Guo H. Correlated-PCA: Principal components' analysis when data and noise are correlated//Proceedings of the Neural Information Processing Systems Conference. Barcelona, Spain, 2016: 1768-1776

[20] Restrepo J G, Ott E, Hunt B R. Characterizing the dynamical importance of network nodes and links. Physical Review Letters, 2006, 97(9): 094102_1

ZHANG Di, born in 1993, M. S. candidate. Her main research interests include network and information security.

LI Xing-Hua, born in 1978, Ph. D., professor, Ph. D. supervisor. His research interests include network and information security, privacy protection, cloud computing and security protocol formal methodology.

LIU Hai, born in 1984, Ph. D. candidate. His main research interests include cryptographic algorithm design and privacy protection.

MA Jian-Feng, born in 1963, Ph. D., professor. His research interests include information security, coding theory, and cryptograph.

Background

SDN is a new network architecture that decouples the data forwarding layer and the control layer. Controller is the brain of SDN, whose work mainly includes handling all kinds of requests of switches, communicating with other controllers and providing information for App. Our experiment shows that when a controller manages 300 switches, the delay of setting up flow tables for the ping command is up to 300 ms. However, the delays of large domestic sites are around 20 ms. With the network scale enlarging, the burden of controller is heavier and heavier, which will directly affect the whole performance of the network. Therefore, how to reduce the burden of controller is a key to ensure the efficiency of network.

The existing work shows that it can realize observability and controllability of the network by selecting a few key nodes, so an effective way of reducing burden of controller is to rank the network nodes and pick out important nodes, we can collect network service information of important nodes to evaluate the status of the entire network. The traditional network with distributed architecture is hard to get traffic and the forwarding frequency of packets, so these methods rank network nodes in terms of static information. However, SDN with centralized control can obtain much more network information, which is important for judging the importance of network nodes. The traditional methods are not service-oriented, which are difficult to satisfy requirements of different services. Because SDN does not define the function of network in advance, it is defined by the APP. Some nodes are not significant in a specific App, but which are important in other APPs. Some papers focus on basic work of the controller to reduce its burden, but with the increase of network service, they will have significant influence on controller performance.

This paper proposes a method of service-oriented network nodes importance ranking. For each service, we consider static information and dynamic information of switches and combine with specific network service to rank the importance of switches. Controller can collect information just from the top nodes, so as to reduce the burden of controller. We take the service of detecting TCP flooding attack in SDN as an example to do a lot of experiments. The results indicate that when we collect the information from the top 40% of the switches, which can reduce 60% burden of the controller. The success rate of detecting attack can reach 94% and the burden brought to the controller by the scheme itself is very limited.