基于网络层析的 IPv6 拓扑推断方法研究

金 睿, 李浩天, 魏松杰

南京理工大学网络空间安全学院 南京 中国 210094

摘要 针对 IPv6 网络环境中的拓扑探测和节点推断问题, 提出一种结合 SRv6 协议的网络层析方法 SNTTI。首先改进传统 Back-to-Back 探测报文结构和公共路径度量指标, 设计了一种带有链路状态罚项的公共路径长度度量公式, 通过两次不同负载下的测量结果差异衡量网络状况, 减少了链路带宽和阻塞对于探测结果的影响; 接着利用 SRv6 承载协议的路径可编程性, 通过在探测报文的 SRH 中引入 OAM 扩展位, 根据返回消息的时间间隔对两种基本网络拓扑结构进行区分, 同时推断出目标节点在网络中的拓扑位置; 改进了基于邻居节点加入的动态拓扑构建算法, 并且针对网络中部分节点不支持 SRv6 的情况进行了讨论, 提升了拓扑推断方法的适用性; 最后提出了汇聚多源探测结果的拓扑融合算法, 利用子树节点的分布特征进行拓扑融合, 通过计算特征序列的余弦相似度定位不同拓扑中的同一个中间节点, 然后补充边的分布, 进而拟合得到网络中的一般拓扑结构。实验结果表明, 本文提出的方法在 IPv6 网络环境中具有较好的探测效果, 能够适应严格 SRv6 以及松散 SRv6 条件下的拓扑探测, 在树状拓扑环境下较 MCPM 等方法准确率提高 1.42～1.85 倍, 在复杂配置和较大规模的一般网络拓扑下具有稳定的探测性能, 精度较 MCPM 等方法提高 1.29～1.44 倍, 探测效率提高 1.46～1.82 倍, 优于现有的同类网络层析探测方法。

关键词 IPv6; 网络拓扑测量; 网络层析; SRv6

中图法分类号 TP393 **DOI 号** 10.19363/J.cnki.cn10-1380/tn.2025.05.05

Research on IPv6 Topology Inference Methods with Network Tomography

JIN Rui, LI Haotian, WEI Songjie

School of Cyber Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China

Abstract In view of the issue of network topology detection and node inference in the IPv6 environment, a network tomography method SNTTI combined with SRv6 protocol is proposed. Initially, the traditional Back-to-Back probe message structure and the common path metric indicators are improved. A common path length metric formula with link state penalties is designed to measure the network condition by comparing the measurement results obtained under two different loads. This approach mitigates the influence of link bandwidth and blocking on probe results. Subsequently, leveraging the path programmability offered by the SRv6 protocol, the SNTTI method introduces an OAM extension bit in the SRH of the probe message. By distinguishing between the two fundamental network topologies based on the time intervals of return messages, it infers the target node's topological position and enhances the dynamic topology construction algorithm based on neighbor node addition. Furthermore, the case of partial nodes in the network that do not support SRv6 is discussed to improve the applicability of the topology inference method. Furthermore, a topology fusion algorithm is presented to aggregate probe results from multiple sources. This algorithm utilizes the distribution characteristics of subtree nodes for topology fusion. By calculating the cosine similarity of feature sequences, identifies the same intermediate node in different topologies and complements the distribution of edges, resulting in the approximation of general topology structures in the network. Experimental results demonstrate the effectiveness of the proposed method

通讯作者：魏松杰，博士，副教授，Email: swei@njust.edu.cn

本课题得到工信部工业互联网创新发展工程项目“工业企业网络安全综合防护平台”（No. TC200H01V）资助。

收稿日期：2023-06-03；修改日期：2023-07-20；定稿日期：2025-03-05

in IPv6 network environments. It adapts well to both strict and loose SRv6 conditions for topology probing. In tree-like topology environments, it achieves an accuracy improvement of 1.42 to 1.85 times compared to methods like MCPM. In large-scale and complexly configured general network topologies, it shows stable probing performance with accuracy improvements of 1.29 to 1.44 times and probing efficiency enhancements of 1.46 to 1.82 times compared to MCPM and outperforms existing similar network tomography methods.

Key words IPv6; network topology measurement; network tomography; SRv6

1 引言

随着计算机网络规模的不断扩大, 网络拓扑信息在网络安全防护、网络协议的设计以及网络性能优化等方面具有越来越重要的意义。网络拓扑结构是网络管理的基础信息, 它可以描绘网络设备节点之间的连接关系。掌握准确的网络拓扑结构有助于更有效地运行网络功能、优化网络结构、减少性能瓶颈、防范安全攻击。而随着 IPv6 的日益普及, 对 IPv6 网络管理和安全的迫切需求也对网络拓扑测量技术提出了新的要求和挑战。

网络拓扑测量方案主要分为基于内部节点协作的测量以及基于网络层析的端到端测量。基于内部节点协作[1-4]的方法可以通过从内部路由器反馈的路由信息快速准确地推断拓扑结构, 但随着隐私安全需求的提高, 以及现有网络系统和设备的非协作性特点, 该方案的准确性和可行性日益降低, 而网络层析方法仅通过端到端的测量来推断网络内部拓扑结构, 不需要借助内部节点的协作[5]。

近年来, 许多学者致力于网络层析拓扑推断领域的研究, 尝试将贝叶斯方法、统计分析方法和拓扑学相关理论结合, 以提高该领域的推断精度和可靠性。Kevin 等人[6]提出了一种基于 Möbius 反演和随机线性矩阵理论的 Möbius 拓扑推理算法, 该算法使用多元积累量来描述网络中不同节点之间的关系并使用 Möbius 变换来处理这些信息以推断网络结构, 该方法能够适应多种不同类型的网络结构, 但是依赖于大量的计算并且需要全面监控和记录网络流量, 使得其在实际网络中的应用受到限制。Bowden 等人[7]提出一种基于条件随机场的模型, 将拓扑结构推断建模为一种图形式的条件概率分布, 从而有效地处理了空间依赖关系, 但是该方法需要进行条件随机场的优化计算和数据预处理, 计算复杂度较高。另外该方法期望数据是服从高斯分布的, 但是现实数据往往不符合这个假设, 这可能会导致推断结果的偏差。Fei 等人[8]提出了一种基于端到端测量时延的高阶统计特征来推断网络拓扑结构的方法, 该方法使用分形分析方法对网络中的时延数据进行分析, 无需先验信息即可准确地推断出网络的拓扑结构, 但是该方法仅基于端到端的时延信息, 未考虑其他因素(如带宽和拥塞)的影响, 在网络质量较差的环境下其准确性会降低。

现有网络层析算法中, 常用的加性特征量有基于 Back-to-Back 时延协方差的加性特征量和基于“Sandwich”包时延差的加性特征量。姜守达等人[9]在“Sandwich”探测的基础上进行了改进, 对于多次测量的结果进行排序重组, 同时在拓扑构建时利用 TTL 跳数信息选择匹配路径, 按照公共路径长度匹配搜索新加入节点的插入位置, 减少了测量过程中所需的测量次数。Ye 等人[10]利用 3-Packet 方式来探测三叶子节点结构, 该方法发送三个具有短时间间隔的小探测包来推断包含三个叶节点的子集结构。3-Packet 比传统 Back-to-Back 探测方案发送更少的探测包, 可以准确测量非平稳网络中子集结构的公共路径长度, 然后通过两两比较基于延迟协方差的公共路径长度推断出子网结构。上述方法是对之前方法的融合和改进, 一定程度解决了网络波动对于测量结果的影响, 但是如果直接迁移到 IPv6 网络, 仍然无法区分被探测节点对是二叉树结构还是处于同一条链路上, 因而在真实环境中准确率较低。

上述研究都是基于 IPv4 网络环境, 而对于 IPv6 网络的拓扑测量, 目前的研究大多聚焦在基于内部节点协作的测量方法, HS[11]、6Topo[12]等探测模型以及 Beverly 等人[13]的研究都是针对 IPv6 网络环境下的主动拓扑发现, 该技术会受到路由器 ICMPv6 速率限制、隐私信息保护以及匿名路由等各种因素的影响, 并且在探测前需要做大量的准备工作。

本文针对过去 IPv4 网络层析拓扑推断的问题,

为提高 IPv6 环境下网络拓扑测量的精准度和探测效率, 提出一种基于 IPv6 段路由(Segment Routing IPv6, SRv6)[14]承载协议的拓扑推断算法 SNTTI(SRv6-based Network Topology Tomography Inference)。首先改进了传统 Back-to-Back 公共路径测量方案, 提出一种带有罚项的公共路径长度计算公式; 其次利用 SRv6 协议的路径可编程性提出一种推断节点间基本拓扑关系的方法; 最后, 提出一种汇聚多探测结果的拓扑融合算法。该方法能够在满足隐私保护需求的前提下实现不同网络规模下的拓扑探测, 是后续 IPv6 网络空间测绘的关键步骤和重要基石。

2 相关概念

2.1 网络模型

采用无向图 $G=(V,E)$ 表示网络拓扑模型[15], 其中 $V=\{v_1,v_2,\cdots,v_n\}$ 代表网络节点集合, 包括 IPv6 网络中的活跃路由器和主机, $E=\{e_1,e_2,\cdots,e_m\}$ 代表节点间的逻辑链路集合。V 包含一个根节点 s、内部节点集 W 和目标节点集 D, 在本实验中根节点 s 为拓扑测量的探测源, 内部节点和目标节点的集合用非根节点集合 $U=V/\{s\}$ 表示, 目标节点集 D 包括叶节点和部分内部节点, 节点示例如图 1 所示。每个内部节点 $w\in W$ 至少包含两条逻辑链路而每个逻辑链路 $e\in E$ 可能包含多个连续的物理链路。除了根节点任何一个节点 $v\in U$ 都有一个唯一的父节点 $f(v)\in V$; 除了叶节点, 任何一个节点 $v\in V$ 都有子节点集 $c(v)=\{j\in V:f(j)=v\}$。对于树 $T=(V,E)$, 令 $l(v_i)$ 代表节点 v_i 的子孙节点, 如果 v_i 是一个叶节点, $l(v_i)=\{v_i\}$; 如果不是, $l(v_i)$ 代表 v_i 的所有子孙节点。从 s 到任意目标节点 $i\in U$ 的路径表示为 $p(i)$, 其长度用 $\rho(i)$ 表示, 该路径由许多的逻辑链路 e 组成; 对于一对目标探测节点 $\{i,j\}$, 令 $f(i,j)$ 代表离二者最近的公共父节点, 从根节点 s 到父节点 $f(i,j)$ 的路径被称为目标节点对 $\{i,j\}$ 的公共路径, 用符号 $p(i,j)$ 表示, 其长度用 $\rho(i,j)$ 表示。任意节点 $k\in U$ 与其父节点之间的逻辑链路 $(f(k),k)\in E$ 记为 e_k。本文使用 $G_{\text{sub}}=(V_{\text{sub}},E_{\text{sub}})$ 代表拓扑 G 的一个子图, G_{sub} 是由 G 的节点子集 V_{sub} 和连接 G_{sub} 中节点的边子集 E_{sub} 构成, 且满足 $V_{\text{sub}}\subseteq V$, $E_{\text{sub}}\subseteq E$。

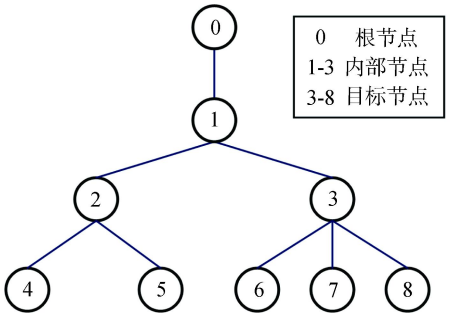


图 1 网络模型示例
Figure 1 Network Model Example

2.2 基于时延协方差的加性特征量

定义 d 为树 $T=(V,E)$ 的加性特征量[16], 当 d 同时满足:

$$0<d(e)<\infty,\forall e\in E \tag{1}$$

$$d(i,j)=\sum_{e\in \mathrm{p}(f(i,j))} d(e),\forall i,j\in V \tag{2}$$

其中 $d(e)$ 为链路 e 的长度, 在不同的加性特征量下有不同的度量单位, $d(i,j)$ 代表节点 i 和 j 公共路径的链路长度。由于基于时延协方差的加性特征量仅需发送少量的探测报文就能获得相对准确的度量值, 因此本文采用基于时延协方差的加性特征量来衡量节点对的公共路径长度。

下图所示为最简二叉树拓扑模型, s 代表探测源, R1 和 R2 代表探测报文的两个接收者, 也即是被探测节点, 各节点之间的逻辑链路如图 2 所示, 每段逻辑链路由许多段物理链路组成:

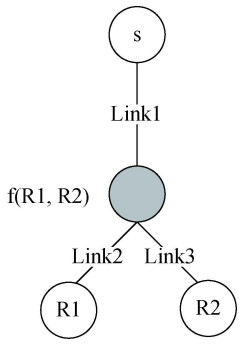


图 2 链路示例
Figure 2 Link Example

令 Y_1 和 Y_2 分别代表探测报文从 s 到 R1 和 R2 的延迟, 让 $X_{1,1}$ 和 $X_{1,2}$ 分别代表发向 R1 的探测报文在链路 Link1 和 Link2 上经历的时间延迟, 让 $X_{2,1}$ 和 $X_{2,3}$ 分别代表发向 R2 的探测报文在链路 Link1 和

Link3 上经历的时间延迟, 根据探测包延迟的加性特征有:

$$Y_1 = X_{1,1} + X_{1,2} \tag{3}$$

$$Y_2 = X_{2,1} + X_{2,3} \tag{4}$$

根据 Back-to-Back 探测包的特点, 除了前一个数据包造成的后一个数据包的排队延迟外, Back-to-Back 的两个探测包在公共路径上经历的传播延迟是一样的, 因此, 可以将(4)式改写为:

$$Y_2 = X_{1,1} + X_{2,3} + Z \tag{5}$$

其中 Z 代表第二个探测包在 Link1 中的排队时延波动。计算 Y_1 和 Y_2 的方差有:

$$\mathrm{Var}(Y_1) = \mathrm{Var}(X_{1,1}) + \mathrm{Var}(X_{1,2}) \tag{6}$$

$$\mathrm{Var}(Y_2) = \mathrm{Var}(X_{1,1}) + \mathrm{Var}(X_{2,3}) \tag{7}$$

通过将(6)(7)相加, 然后计算路径延迟之和的方差可以得到:

$$\mathrm{Var}(S_{1,2}) = 4\,\mathrm{Var}(X_{1,1}) + \mathrm{Var}(X_{1,2}) + \mathrm{Var}(X_{2,3}) \tag{8}$$

其中 $S_{1,2}$ 代表路径延迟观测值 Y_1 和 Y_2 的总和, 从而可得 Y_1 和 Y_2 的协方差为:

$$\mathrm{Cov}(Y_1, Y_2) = \frac{\mathrm{Var}(S_{1,2}) - \mathrm{Var}(Y_1) - \mathrm{Var}(Y_2)}{2} \tag{9}$$

通过结合(6)(7)(8)(9)式, 可以得到探测包在 Link1、Link2 和 Link3 上的时延方差如下:

$$\mathrm{Var}(X_{1,1}) = \mathrm{Cov}(Y_1, Y_2) \tag{10}$$

$$\mathrm{Var}(X_{1,2}) = \mathrm{Var}(Y_1) - \mathrm{Cov}(Y_1, Y_2) \tag{11}$$

$$\mathrm{Var}(X_{2,3}) = \mathrm{Var}(Y_2) - \mathrm{Cov}(Y_1, Y_2) \tag{12}$$

从(10)式可以看出, 在公共路径 Link1 上的时延方差等于探测时延的协方差 $\mathrm{Cov}(Y_1, Y_2)$, 所以可以通过计算端到端延迟的协方差得到公共路径时延的方差。假设网络中无背景流量, 则在公共路径中经历的中间节点越多, 即目标节点对 $\{\mathrm{R1}, \mathrm{R2}\}$ 的公共路径越长, Back-to-Back 探测包时延协方差越大, 公共路径时延的方差越大。

2.3 拓扑推断准则

为了确保通过端到端测量得到的公共路径长度能够准确推断出拓扑结构, 本文做了以下假设[10]:

(1) 在端到端测量过程中, 网络拓扑结构保持不变;

(2) 不同链路的数据包延迟或丢失在统计上是独立的;

(3) 在同一链路上的不同次探测数据包的延迟或丢失在统计上是独立的。

基于上述假设, 可以通过比较公共路径长度的度量值来恢复拓扑。图 3 展示了公共路径长度模型, f_1 表示离节点对 $\{\mathrm{i}, \mathrm{j}\}$ 最近的公共父节点, $\mathrm{p}(\mathrm{i}, \mathrm{j})$ 代表源节点 s 到叶节点对 $\{\mathrm{i}, \mathrm{j}\}$ 的公共路径, f_2 表示离叶节点对 $\{\mathrm{i}, \mathrm{k}\}$ 最近的公共父节点, $\mathrm{p}(\mathrm{i}, \mathrm{k})$ 代表源节点 s 到叶节点对 $\{\mathrm{i}, \mathrm{k}\}$ 的公共路径, 从图 3 中可以看到, 公共路径 $\mathrm{p}(f_1)$ 比 $\mathrm{p}(f_2)$ 包含更多的物理链路, 因此 $\mathrm{p}(f_1)$ 的长度度量值 $\mathrm{p}(f_1)$ 要大于 $\mathrm{p}(f_2)$ 的长度度量值 $\mathrm{p}(f_2)$, 而探测过程就是获取节点对两两的公共路径度量值, 从而推断出拓扑结构, 即被探测节点对的公共路径度量值越大, 其最近公共父节点离源节点就越远, 从而可以根据公共路径的长度度量值大小逐一插入目标节点来恢复树状拓扑结构。

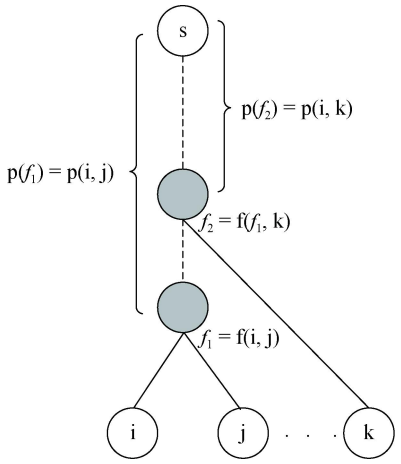


图 3 公共路径长度示例

Figure 3 Common Path Length Example

3 基于 SRv6 的拓扑推断算法

3.1 改进的公共路径度量指标

网络层析拓扑推断的关键是获得目标节点对公共路径长度的准确度量值, 衡量公共路径长度最常见的指标之一是延迟协方差[16], 但由于不同路径的带宽和阻塞不同, 协方差也会受到影响, 因此本文设计了改进的公共路径测量方案, 以减少带宽及链路状态对测量结果的影响。

为了获取更准确的公共路径长度, 本文构造了两组探测包: 第一组为经典的 Back-to-Back 探测包, 其承载的数据为空; 第二组在第一组探测包的基础上, 增大其承载的数据, 同时保证整个报文长度略小于传输路径中的最大传输单元(Path Maximum

Transmission Unit, PMTU)。两次测量结果的差异性衡量了本次测量中带宽及链路状态对于探测结果的影响, 并在公共路径长度计算公式的基础上将其作为罚项, 以减少不同探测间带宽及链路状态的影响, 从而控制影响公共路径度量值的变量仅为链路长度。在 2.2 节中已经得到公共路径 Link1 的长度度量公式:

$$M = \mathrm{Cov}\left(Y_1, Y_2\right) \tag{13}$$

该公式通过 Back-to-Back 数据包的延迟数据来映射公共路径的长度, 但是该公式并没有考虑带宽及链路状态所带来的影响, 因此通过再次发送 Back-to-Back 数据包, 重新设置每个 ICMPv6 探测报文的承载数据量, 使得整个报文长度不超过 PMTU, 再次探测得到:

$$M' = \mathrm{Cov}\left(Y_1', Y_2'\right) \tag{14}$$

重新构造的含有罚项的公共路径度量公式为:

$$\mathrm{Metric} = M - \left(\frac{M' - M}{M'}\right) \cdot M = \frac{M^2}{M'} \tag{15}$$

其中 M 和 M' 分别代表两次探测的公共路径长度度量值, 罚项为 $\left(1 - \frac{M}{M'}\right) \cdot M$, 衡量两次测量结果的差异性。

3.2 基本拓扑推断

本节介绍如何进行最基本的拓扑推断, 也就是对于一组目标节点对的拓扑推断。在本文中假设目标节点同时包含叶节点和内部节点。如图 4 所示, R1 与 R2 是一对目标节点, 二者存在两种可能的拓扑结构, (a)为二叉树拓扑; (b)为线性拓扑, 二者属于父节点和子节点关系。

在 IPv6 环境下, 本文基于网络层析方法, 利用 SRv6 承载协议进行两种拓扑结构的区分。SRv6 是指将 Segment Routing 技术应用于 IPv6 数据平面, 通过在 IPv6 报文中插入一个段路由扩展头(Segment Routing Header, SRH)实现路径可编程性, SRH 中包含了由 IPv6 地址列表表示的 Segment List, 转发过程中报文的目的地址将逐段地被更新, 完成逐段转发。本文依据真实环境中 SRv6 协议配置情况分两种情况讨论, 各节点之间的探测包延迟如图 4 所示。

3.2.1 严格 SRv6

在严格 SRv6 环境下, 所有节点均支持 SRv6 协议, 本文构造了一个带有 IPv6 路由扩展头的 ICMPv6 Request 数据包, 该数据包中有两个需要特殊指定的位置, 一个是 SRH 中的 Flags 字段, 另一个是段列表(Segment List, SL),其中 Flags 字段的配置需要借助 SRv6 OAM(Operations, administration and maintenance)。SRv6 OAM 主要用于监控 SRv6 路径的连通性和快速进行故障检测, 当前 SRv6 的 OAM 扩展主要包括两种方式: 一种是在 SRH 中引入 O-bit(OAM 比特位), 另一种是引入 End.OP SID(OAM endpoint with punt), 本文利用的是前者。其中 O-bit 位于 SRH 中的 Flags 字段, 该字段的格式如图 5 所示。

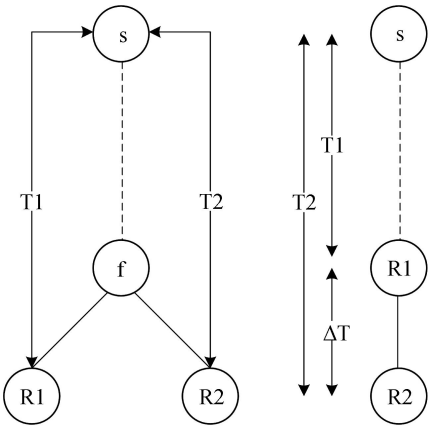


图 4 拓扑及探测时间示例

Figure 4 Topology and Detection Time Example

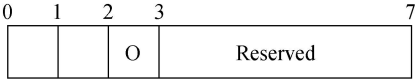


图 5 SRH 的 Flags 字段格式

Figure 5 Flags Field Format of SRH

O-bit 用于指示 OAM 处理, 若 O-bit 置位, 则每一个 SRv6 Endpoint 节点需要复制一份数据并打上时间戳, 然后上送复制的报文和时间戳到控制平面处理。由于每一个 SRv6 Endpoint 节点都需要响应携带 O-bit 的 ICMPv6 报文, 所以基于 O-bit 就可以实现逐段检测。

因此通过将 Flags 字段设置为 0x20, 也就是让其中的 O-bit 等于 1 同时其他位置置 0, 可以让指定路径中的设备响应 ICMPv6 Reply 报文, 而设备只需要配置 SRv6 End SID 或者 End.X SID, 然后在 SL 中指定需要经过的 SID 或者出口 IP, 通过其响应时间便可以得到从发送者到两个接收者之间各段逻辑链路的时间延迟。SRv6 探测报文结构如图 6 所示, 图中 Segment List 中最后一个元素是 R1, 其次是 R2, 而实际中探测报文会先经过 R1, 然后到达 R2。

SR 探测报文的路径如图 7(a)和(c)所示, 使用 T_1 和 T_2 分别代表数据包到达 R1 和 R2 的延迟, 令

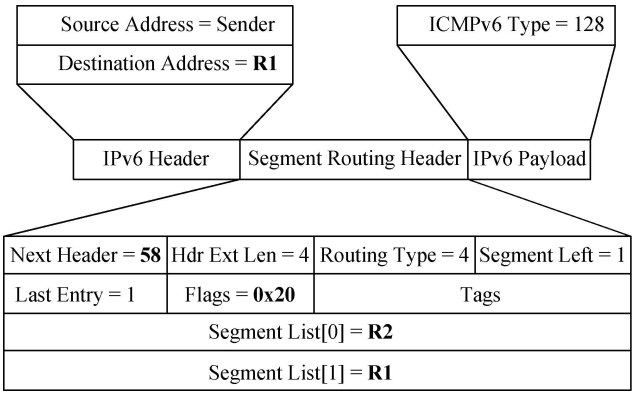


图 6　SRv6 探测报文

Figure 6　SRv6 Probe Message

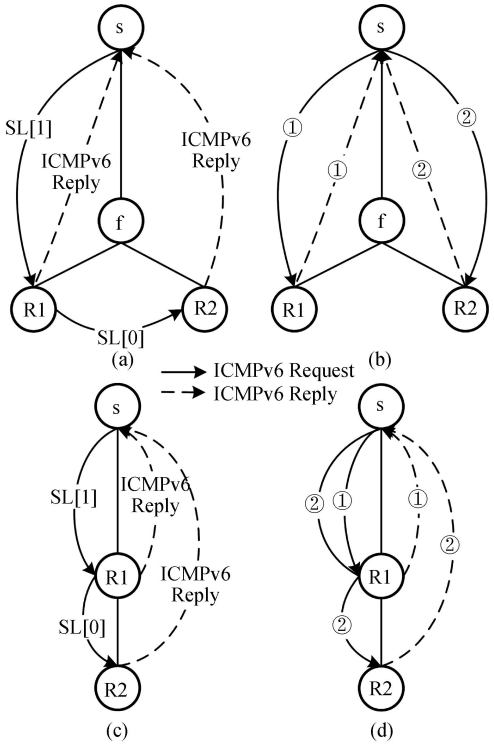


图 7　严格 SRv6 报文走向

Figure 7　Strict SRv6 Message Direction

$\Delta T = \left|T_2 - T_1\right|$。然后再次分别对于两个接收者发送普通的 ICMPv6 Request 报文(图 7(b)和(d)①②过程), 使用 T_1' 和 T_2' 分别代表 R1 以及 R2 的响应时延, 令 $\Delta T' = \left|T_2' - T_1'\right|$, 比较 ΔT 和 $\Delta T'$, 由于(c)和(d)中报文路径一致, 均为依次经过 R1 和 R2, 因此两次探测时间差是一致的, 如果二者相等, 就说明二者属于线性拓扑结构, 也就是图 4(b), 否则为二叉树拓扑结构, 即图 4(a)。

3.2.2　松散 SRv6

在松散 SRv6 环境下, 只有部分 IPv6 节点支持 SRv6, 这种情况在现实中较为常见。本文假设在两个被探测节点中至少有一个支持 SRv6, 在这种情况下拓扑探测过程和严格 SRv6 相似, 只不过需要调整段列表 Segment List 中地址的压栈顺序, 第一跳需要是支持 SRv6 的节点, 目的是让探测报文依次遍历两个节点。判断节点是否支持 SRv6 可以通过构造一个 SRv6 报文观察其是否执行了 End SID 功能。在线性拓扑结构中, SR 探测报文的走向有如下两种情况, 其中 T_1' 和 T_2' 分别代表目的地址为 R1 和 R2 时探测包所用时长。

在图 8 中, 箭头代表了 SR 报文经历的节点顺序, (a)中 SR 报文目的地址依次为 R1 和 R2, (b)中 SR 报文目的地址依次为 R2 和 R1。探测报文的两种走向取决于 R1 和 R2 哪一个支持 SRv6, 如果是 R1 支持则探测报文走向为(a)所示, 如果是 R2 支持则为(b)所示。无论哪一种情况, 都可以通过严格 SRv6 中的探测步骤获取到 $\Delta T = \left|T_2 - T_1\right|$ 以及 $\Delta T' = \left|T_2' - T_1'\right|$, 其中 T_2、T_1 和 T_2'、T_1' 分别对应 R2 和 R1 的 ICMPv6 探测时延以及 SR 探测报文时延。而观察图 8 可知, T_2' 和 T_1' 的差值可能是 T_2 和 T_1 之间差值的一倍, 即图 8(a), 或者为 0, 表示二者相等, 即图 8(b), 因此在线性拓扑的情况下两次探测的时延差 $\Delta T'$ 和 ΔT 可能满足(16)(17)中的任何一个等式, 只要观察到其中的一个等式成立就可推断出 R1 和 R2 的拓扑结构:

$$\Delta T' = \Delta T \tag{16}$$

$$\Delta T' = 0 \tag{17}$$

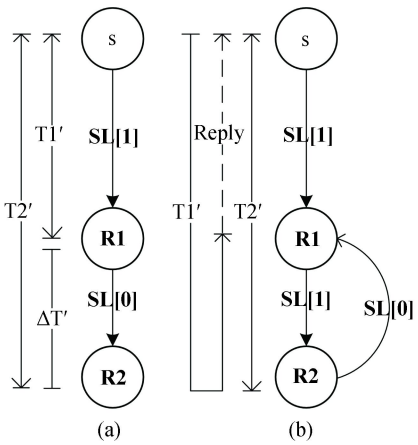


图 8　松散 SRv6 报文走向

Figure 8　Loose SRv6 Message Direction

其中第一个等式对应上图中(a)的探测顺序, 该情况下与严格 SRv6 环境中的探测顺序完全相同, 而第二个等式是在只有 R2 支持 SRv6 的情况下满足的,

对应上图(b)。如果不满足上述两个等式, 则认为二者为树状拓扑结构。基于 SRv6 的基本拓扑推断算法(SRv6-based Basic Topology Inference, SBTI)的详细过程见算法 1。

算法 1 基于 SRv6 的基本拓扑推断算法

输入 根节点 s, 目标节点集 D, 任何两个被探测节点 $\hat{i},\hat{j}\in D,\hat{i}\neq\hat{j}$

输出 两个目标节点的基本拓扑结构

1 分别对两个目标节点发送 ICMPv6 Request 并且记录响应时间, 分别记为 T_1 和 T_2, 令 $\Delta T=\left|T_2-T_1\right|$

2 判断是严格 SRv6 环境还是松散 SRv6 环境

3 **IF** 严格 SRv6 环境

4 Segment List[1]设置为响应时间短的地址

5 **ELSE IF** 松散 SRv6 环境

6 Segment List[1]设置为支持 SRv6 节点的地址

7 **END IF**

8 发送 SRv6 探测报文, 设响应时间为 T_1' 和 T_2', 记 $\Delta T'=\left|T_2'-T_1'\right|$

9 **IF** ΔT 和 $\Delta T'$ 满足式(16)(17)中的任何等式

10 基本拓扑为线性拓扑即图 4(b)

11 **ELSE**

12 基本拓扑为树状拓扑即图 4(a)

13 **END IF**

3.3 邻居节点加入

本节介绍邻居节点在进行插入时如何选择插入位置。在 3.1 节通过探测得到目标节点之间的公共路径长度并进行排序后, 根据排序结果把度量值最大的一对节点作为基节点并将其父节点加入到目标节点集中, 然后重新计算和比较该父节点和其他目标节点之间的共享路径长度, 根据排序结果添加新的节点即父节点的邻居节点来形成整体的拓扑结构。图 9 所示为邻居节点插入的三种位置, 分别是成为父节点的子节点、父节点和兄弟节点, 当目标节点对是二叉树结构时, 邻居节点加入后的拓扑如(a)(b)(c)所示, 当目标节点对为父子关系时, 邻居节点加入后的拓扑如(d)(e)(f)所示。

本文在 RNJ[16]算法的基础上, 针对如图 9 所示的三种插入位置设计了如下的拓扑构建算法。该算法从一个包含所有目标节点的子集开始, 每一步中它选择一个可能是邻居的节点对(即兄弟节点, 在拓扑树中具有相同的父节点), 将它们从目标节点集中删除, 然后创建一个新节点作为其父节点加入到原来的树中, 并添加该父节点到目标节点集中。整个过程不断迭代, 直到目标节点集中只剩下一个节点, 它将成为根节点(探测源)的子节点。详细过程见算法 2, 其中步骤 2 到 11 为判断和更新基本拓扑, 步骤 12 到 19 为判断剩余节点和父节点的关系, 步骤 20 到 21 是计算新生成的父节点和剩余目标节点的共公共路径长度, 并为新生成的父节点寻找最合适的兄弟节点以作为下一个循环的输入。

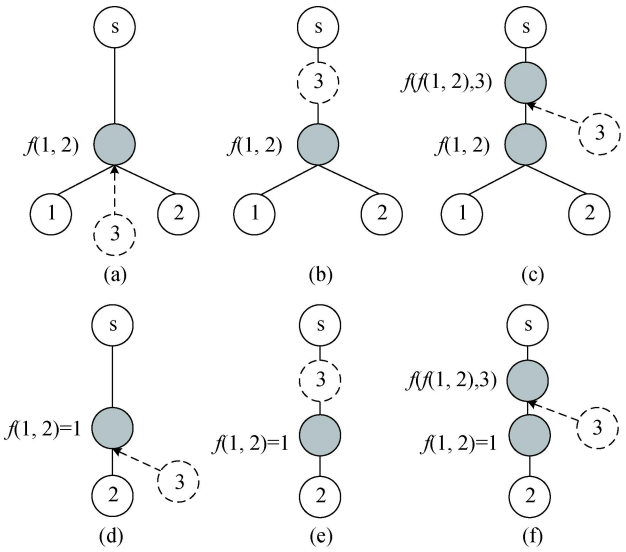


图 9 新节点加入位置

Figure 9 New Node Joining Position

算法 2 基于邻居节点加入的动态拓扑构建算法

输入 根节点 s, 目标节点集 D, 从根节点到任何两个目标节点的公共路径长度度量值集合 $\left\{\hat{\rho}(i,j):\forall i,j\in D,i\neq j\right\}$

输出 构建的树状拓扑 $\hat{T}=\left(\hat{V},\hat{E}\right)$

初始化 $V=\{s\}\bigcup D$

$E=\varnothing$

$\hat{l}=\left\{\hat{l}\left(v_i\right),1\leqslant i\leqslant|V|\right\}$

1 循环

2 找到两个目标节点 i^* 和 j^* 使得

$\left\{i^*,j^*\right\}=\arg\max_{i,j\in D}\hat{\rho}(i,j)$

3 使用算法 1 推断目标节点的拓扑关系

4 **IF** 拓扑为二叉树结构

5 为 i^* 和 j^* 创建一个父节点 f

6 **ELSE**

7 将离源节点 s 更近的一个目标节点设为父节点 f

8 **END IF**

9 $D=D\setminus\{i,j\}$ //从目标节点集中去除

10 $\hat{V}=V\cup f$ //将父节点加入到结果节点集中

11 $\hat{l}(f)=\hat{l}(i^*)\cup\hat{l}(j^*),\ \hat{E}=\hat{E}\cup\{(f,i^*),(f,j^*)\}$

12 对于任何 $k\in D$

13 **IF** $\left|\hat{\rho}(i^*,j^*)-\hat{\rho}(i^*,k)\right|\leqslant\frac{\Delta}{2}$ **AND** $\Delta\leqslant\min_{e\in E}d(e)$ [16]

14 $D=D\setminus k$

15 $\hat{l}(f)=\hat{l}(f)\cup\hat{l}(k)$

16 $\hat{E}=\hat{E}\cup(f,k)$

17 **END IF**

18 $\hat{\rho}(f)=\hat{\rho}(i^*,j^*)$

19 $D=D\cup f$

20 对于任何 $k\in D$，计算:

21 $\hat{\rho}(k,f)=\frac{1}{|c(f)|}\sum_{z\in c(f)}\hat{\rho}(k,z)$ //计算父节点和剩余目标节点的公共路径长度

22 直到算法满足条件 $|D|=1$，结束循环

23 $\forall k\in D$，$\hat{E}=\hat{E}\cup(s,k)$

3.4 拓扑融合

由于新生成的节点地址未知, 因此如何在不同的拓扑中定位同一个节点比较困难, 而拓扑融合的关键就是找到不同探测结果中的同一个节点, 而这些节点往往地址未知。对于每次探测来说, 目标节点集是固定的, 因此可以将这些目标节点在以未知节点为根的子树中的分布情况视为未知节点的一个特征序列, 通过比较不同特征序列的相似性找到不同拓扑中的同一个节点。因此本文提出一种基于子树中目标节点分布特征的拓扑融合方案。

该方案包括以下三个步骤: 首先, 在每一个探测结果中获得以未知节点为根并以目标节点集为子孙节点的子树结构, 根据该子树中包含的目标节点位置对根节点进行编码, 每一个未知节点的特征序列是一个矩阵, 包含其中所有目标节点的 IPv6 地址以及其特征值, 该特征值等于子树中目标节点深度与探测结果中整棵树深度的差值; 其次, 比较不同探测结果的未知节点之间特征序列相似性, 相似性最大的认为是同一个节点, 因为不同探测过程中探测源是改变的, 而被探测节点保持不变, 因此在底层结构中保有更多的相似性。比较相似性的方法是, 调整特征序列以使两个特征序列在相同位置上的地址相同, 为确保两个矩阵的标识顺序一致, 互相补充缺省节点地址并将其特征值为设为 0, 以两个序列的余弦相似度衡量未知节点的相似性; 最后, 对于匹配成功的节点, 互相补充边的分布。

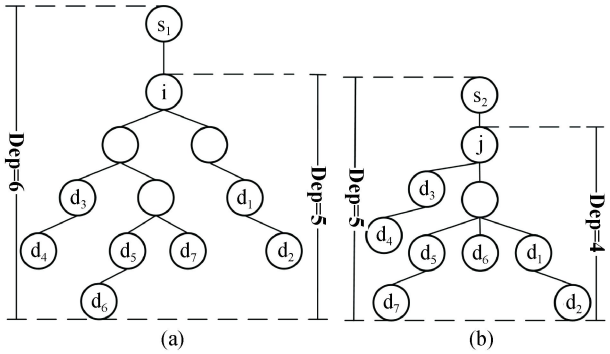


图 10 特征序列示例

Figure 10 Example of Feature Sequence

图 10 所示为同一拓扑的两个探测结果, 其中 s_1 和 s_2 为探测源, d_1 到 d_7 为目标节点, i 和 j 为需定位的未知节点。以(a)为例, 其深度为 6, 目标节点 d_6 在以 i 为根节点的子树中的深度为 5, 因此 d_6 在子树 $T_{\text{sub}}(i)$ 中的特征值为 1。同理可得节点i的特征序列为:

$$\boldsymbol{Fp}(i)=\begin{bmatrix} d_6 & d_2 & d_4 & d_5 & d_7 & d_1 & d_3 \\ 1 & 2 & 2 & 2 & 2 & 3 & 3 \end{bmatrix}_{2\times7}$$

节点 j 的特征序列为:

$$\boldsymbol{Fp}(j)=\begin{bmatrix} d_6 & d_2 & d_4 & d_5 & d_7 & d_1 & d_3 \\ 2 & 1 & 2 & 2 & 1 & 2 & 3 \end{bmatrix}_{2\times7}$$

基于子树中目标节点分布特征的拓扑融合方案详细过程见算法 3。

算法 3 基于子树目标节点分布的拓扑融合方案

输入 m 个探测源的探测结果集合: $R=\{T_1=(V_1,E_1),T_2=(V_2,E_2),\cdots,T_m=(V_m,E_m)\}$ 只含有目标节点的拓扑 G_{final}

输出 拓扑融合结果 G_{final}

初始化 对于每一个 $\hat{T}\in R$ 的中间节点 $\hat{i}\in\hat{W}$，$T_{\text{sub}}(\hat{i})=(\hat{V}_{\text{sub}},\hat{E}_{\text{sub}})$ 为 $\hat{T}$ 中以 $\hat{i}$ 为根的子树, 其节点集 $\hat{V}_{\text{sub}}$ 中包含目标节点集 D 的子集 $D_{\text{sub}}=\hat{V}_{\text{sub}}\cap D=\{k_1,k_2,\cdots,k_n\}$

1 计算 D_{sub} 中每一个目标节点 k 的特征值 $I(k)$:

$$I(k)=dep_{\max}-dep(k)$$

其中 $dep_{\max}$ 代表整颗树 $\hat{T}$ 的深度, $dep(k)$ 代表该目标节点在子树中的深度。

2 表示节点 $\hat{i}$ 的特征序列为:

$$\boldsymbol{Fp}(\hat{i})=\begin{bmatrix} k_1 & k_2 & \cdots & k_n \\ I(k_1) & I(k_2) & \cdots & I(k_n) \end{bmatrix}_{2\times n}$$

3 相似度计算。对于R中任何两个不同的树状拓扑 T' 和 T'', $\forall j\in W', \forall k\in W''$, $j\neq k$, 其中 W' 和 W'' 分别是 T' 和 T'' 中间节点的集合, 分别获得节点 j 和 k 的特征序列。

4 将两个序列中第一行唯一标识(地址)进行互补, 并确保标识顺序一致, 缺省特征值设为 0。

5 将两个矩阵的第二行视为两个 n 维的向量, 分别记为 $\overrightarrow{\boldsymbol{fp}(i)}$ 和 $\overrightarrow{\boldsymbol{fp}(j)}$, 然后计算其余弦相似度:

$$\mathbf{Sim}(j,k)=\frac{\overrightarrow{\boldsymbol{fp}(i)}\cdot\overrightarrow{\boldsymbol{fp}(j)}}{|\overrightarrow{\boldsymbol{fp}(i)}|\times|\overrightarrow{\boldsymbol{fp}(j)}|}$$

6 选择不同拓扑中相似性最大的一对节点作为同一个节点加入到 G_{final}

7 补充边的分布, G_{final} 中边的集合为所有探测结果边的并集

4 实验仿真

为了对本文提出的 SNTTI 算法的性能进行综合评价, 本文在 NS3-3.33 以及 Mininet-2.2.2 仿真软件环境中进行实验, 并与目前主流方法进行对比。仿真设备的处理器为 AMD Ryzen 7 5800H with Radeon Graphics (16 CPUs, ～3.2GHz), 内存为 32GB。首先搭建了两种不同的实验环境, 分别为树状拓扑与一般拓扑, 以便评估算法在公共路径度量、拓扑推断和构建、拓扑融合等环节中的表现。推断效果评价指标采用完整性、准确性以及推断效率。完整性是指拓扑推断结果中正确的节点以及边的总数在真实拓扑中总数的比例, 在本实验中用召回率来表示。准确性是指推断结果中正确的节点以及边的总数占推断结果总数的比例, 在本实验中通过准确率来表示。使用发送探测包数和 F_1 分值的关系来度量算法的推断探测效率。

$$\text{准确率}=\frac{\text{正确的节点}+\text{正确的边}}{\text{推断结果中节点以及边的总数}} \tag{18}$$

$$\text{召回率}=\frac{\text{正确的节点}+\text{正确的边}}{\text{真实拓扑中节点以及边的总数}} \tag{19}$$

4.1 树状网络拓扑实验

为对 SNTTI 算法在树状拓扑中的性能进行评估, 本文采用 NS3 网络仿真工具构造了如图 11 所示的网络, 该网络包含 24 个节点和 23 条链路。

在该拓扑中采用随机早期探测(Random Early Detection, RED)队列算法来控制分组的收发行为, TCP 拥塞控制算法采用 TCP New Reno, 所有节点之间通过 CSMA 信道连接, 默认情况下链路传输速率设置为 10Mbps, 传播延迟设置为 2ms, 为体现真实网络环境中的包丢失情况, 采用 BurstErrorModel 错误模型以 0.01 的突发错误率丢失分组, 均匀随机变量设置为 [Min=1|Max=3] 。背景流量采用 OnOffApplication 应用层协议产生, 其中发送持续时间设置为 2s, 停止时间设置为 3s, 每次发送数据包大小为 500 字节, 分组发送速率为 0.3Mbps, 通过随机选择不同的套接字种类产生 TCP 和 UDP 流量。所有的 UDP 流和 TCP 流的发送节点和接收节点在网络节点中随机选择。

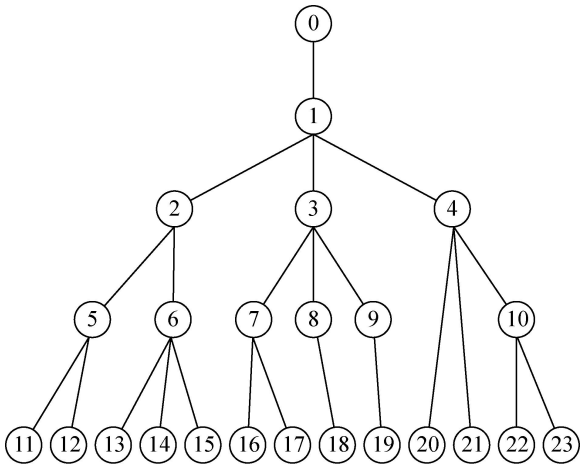


图 11 树状网络拓扑
Figure 11 Tree Network Topology

实验环境中所有节点均为 IPv4/IPv6 双栈节点, 为每个节点手动分配 IPv6 地址, 节点间采用 RIP 路由协议, 其中节点 0 为探测源, 通过在节点 0 中安装 TapBridge 网络设备可以实现 NS3 节点和物理节点之间的通信。由于本探测方案能够对非叶子节点之间的关系做出准确判断, 为体现算法优越性, 目标节点设置如表 1 所示, 剩余节点作为内部节点评估算法性能, 在本实验中, 探测包的 SR 功能通过松散路由实现。

为比较改良后的拓扑推断算法的性能, 本文在不同网络带宽下与目前算法中性能较好的 MCPM[9] 和 SSFTI[10]算法进行比较。本文进行两组仿真实验, 第一组实验中链路带宽较大, 即默认带宽 10Mbps; 第二组实验中链路带宽较小, 设置为 2Mbps。探测包为改进的 Back-to-Back 探测包, 其中第一个分组无负载数据, 第二个分组负载为 1050 字节。对每对目

表 1 被探测节点配置

Table 1 Detected Node Configuration

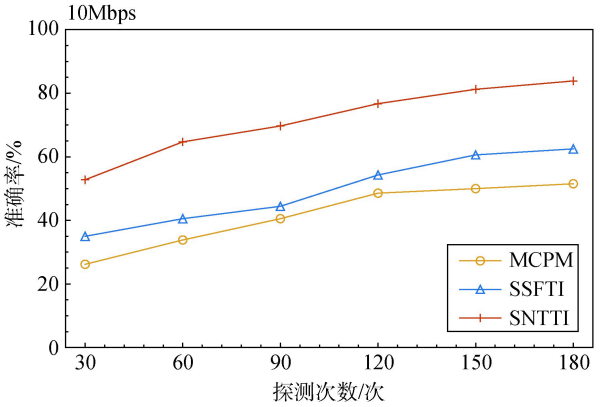
节点编号	地址	是否支持 SRv6
5	2001:3::5/64	是
7	2001:3::7/64	是
9	2001:3::9/64	是
11	2001:4::11/64	否
12	2001:4::12/64	是
13	2001:4::13/64	是
14	2001:4::14/64	否
15	2001:4::15/64	是
16	2001:4::16/64	否
17	2001:4::17/64	否
18	2001:4::18/64	是
19	2001:4::19/64	否
20	2001:4::20/64	是
21	2001:4::21/64	是
22	2001:4::22/64	是
23	2001:4::23/64	否

标节点分别探测 30、60、90、120、150 和 180 次, 采用 SNTTI、MCPM 和 SSFTI 三种算法推断拓扑结构, 比较最终生成的拓扑准确率。

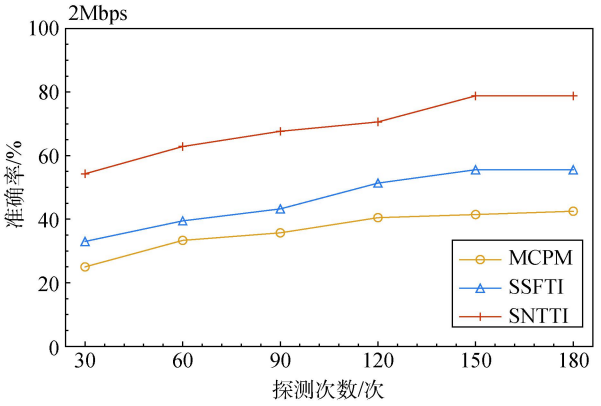
图 12 分别给出了第一、二组实验的结果, 三种算法拓扑推断的准确率整体上都随着探测次数的增加而增加。本文提出的 SNTTI 算法无论在带宽较小还是较大的情况下都具有更好的准确性和完整性, 且在链路带宽较小时, 这种优势更加明显, 这是因为 SNTTI 改进了公共路径度量算法, 减小了带宽或者链路阻塞对于探测结果的影响。由于本文设计的实验环境增加了内部目标节点与 SRv6 的配置, 因此 SNTTI 能够获取到更多的关于内部网络的信息以支撑拓扑的判断, 而其余方法由于不能对两种基本拓扑结构进行区分, 导致遗漏掉更多的边并且会错误的推断出新的父节点, 因此准确率较低。在带宽为 2Mbps 的环境下, 探测次数达到 180 次时, SNTTI 算法拓扑推断准确率为 78.79%, 较 SSFTI 算法提高约 1.42 倍, 较 MCPM 算法提高约 1.85 倍。

4.2 一般网络拓扑实验

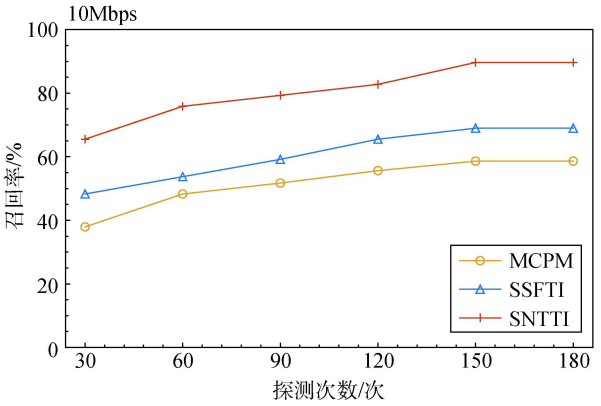
为体现本算法在一般拓扑(即存在逻辑环路的拓扑)环境下拓扑探测的性能, 本文在一般网络拓扑中



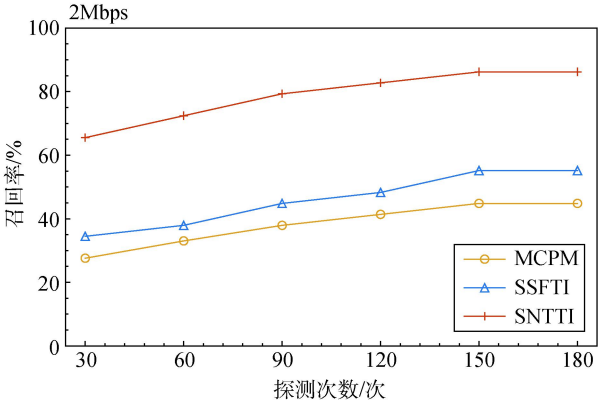
(a) 10Mbps下树状拓扑准确率



(b) 2Mbps下树状拓扑准确率



(c) 10Mbps下树状拓扑召回率



(d) 2Mbps下树状拓扑召回率

图 12 状拓扑实验结果

Figure 12 Tree Topology Experimental Results

将 SNTTI、SSFTI、OLTD[8]以及 MCPM 算法进行对比。本文利用 Brite 生成器[17]根据 Barabasi-Albert 模型生成不同规模的路由器级随机拓扑结构，参数配置如表 2:

表 2　Brite 配置

Table 2　Brite Configuration

参数	含义	规模 1	规模 2	规模 3	规模 4
HS	生成拓扑的层级结构的深度	10	10	10	15
LS	在层级结构中每个层级上的节点数量(个)	5	10	10	10
N	生成拓扑的节点数量(个)	25	50	75	100
Model	拓扑模型	BA	BA	BA	BA
Node Placement	节点的放置方式	Random	Random	Random	Ran-dom
m	度分布模型中的常量值	2	2	2	2
Growth Type	网络的增长模式	Increme ntal	Increme ntal	Increme ntal	Increme ntal
BWdist	带宽分布模型	Constant	Constant	Constant	Constant
MaxBW	生成链路的最大带宽(Mbps)	2	2	2	2
MinBW	生成链路的最小带宽(Mbps)	10	10	10	10

本实验在 SDN 环境中进行，实验利用 Mininet 模拟 IPv6 网络，同时依赖 Linux 内核实现 SRv6 数据平面；在控制平面中，Linux 节点使用 Python 开发的控制器提供 gRPC 南向 API。探测包为改进的 Back-to-Back 探测包，其中第一个分组无负载数据，第二个分组负载为 1050 字节。进行拓扑探测时，在节点中随机选择 50%的节点作为被探测节点，然后在被探测节点中选择 3 个以上度中心性较大的节点作为探测源的直连节点。每个探测源通过 SNTTI、SSFTI、OLTD 以及 MCPM 算法分别获得以当前位置为根的拓扑结构，同时 SNTTI 算法需要进行拓扑融合，汇聚探测结果，其余算法选择节点与边最多的探测结果。通过对于不同规模网络的探测，得到如图 13 所示的推断结果以及图 14 所示的算法探测效率，本实验中利用 F_1 分值与发包数的关系衡量算法效率，其中 F_1 计算公式如下:

$$F_1 = \frac{2 \times precision \times recall}{precision + recall} \tag{20}$$

在图 13 中横坐标代表网络规模，总节点数分别为 25、50、75 和 100，纵坐标代表算法探测结果，包括节点数、边数以及总和。

如图 13 所示，在不同规模的网络下，SNTTI 算法的整体表现始终高于其他算法。由于 OLTD、SSFTI 算法基于传统 Back-to-Back 探测包而 MCPM 基于“Sandwich”探测包，均默认基本拓扑结构为二叉树拓扑，而真实环境中节点间存在图 4(b)中的情况，因

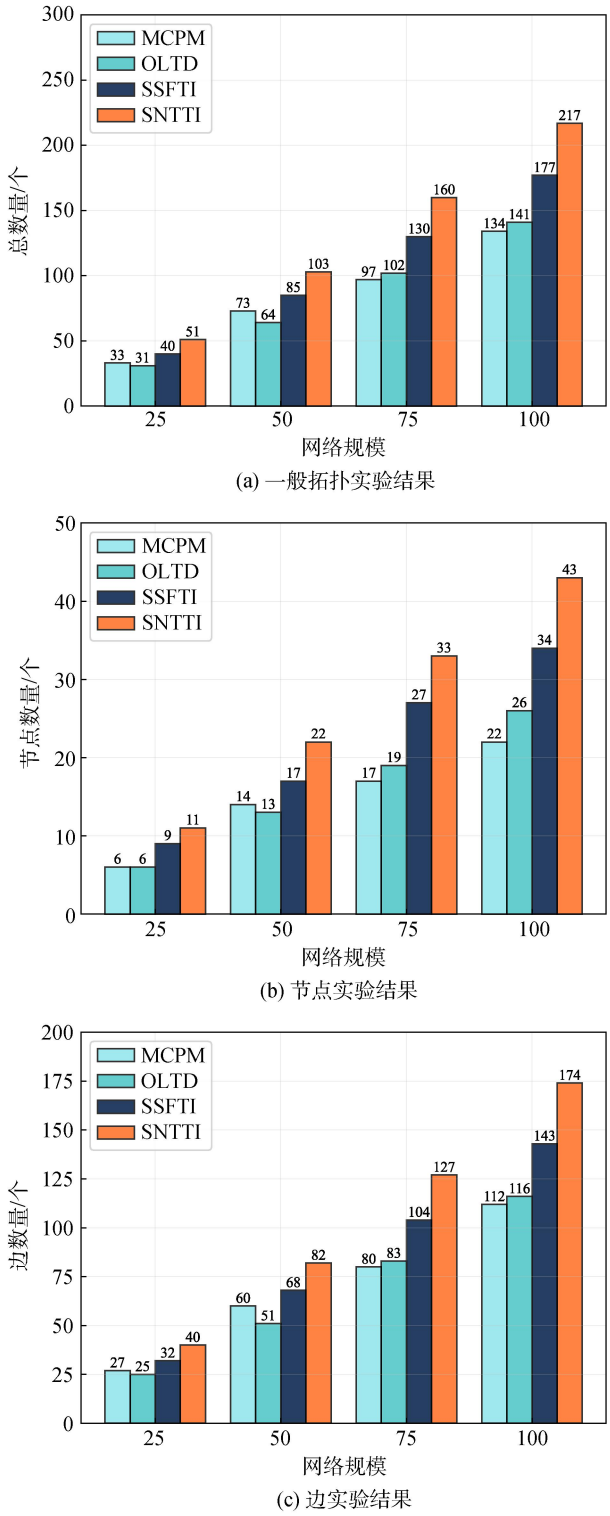


图 13　一般拓扑实验结果

Figure 13　General Topology Experimental Results

此对于拓扑结构的识别不够准确, 随着网络规模的增加, OLTD 等方法无法识别的拓扑数量也随之增加; 而 SNTTI 始终保持稳定的增长幅度, 这可以归结于 SNTTI 拓扑融合的方式能够更适应网络层析的底层逻辑并且对于 IPv6 环境更加具有针对性, 而其他方法本身是针对树状拓扑的识别, 没有识别逻辑环路的算法设计, 因此在存在逻辑环路的网络环境中, 会遗漏掉更多的边。

图 14 展示了在 50 个节点规模下算法 F_1 分值与发包数的关系。

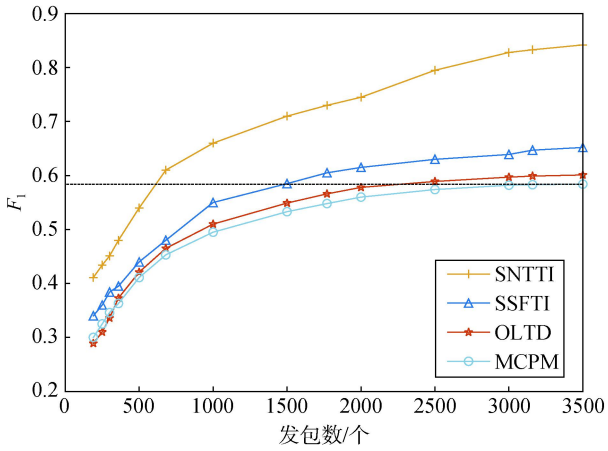


图 14 一般拓扑 F_1 得分

Figure 14 General Topology F_1 Scores

从图中可以看到, 在相同的发包量下 SNTTI 算法具有更高的 F_1 分值, 虽然 SNTTI 算法单次探测需要发送更多的探测包, 但是在相同的发包量下可以获得关于网络内部更多的信息, 同时 SNTTI 借助 SRv6 协议能够识别其他方法无法分辨的拓扑结构, 使其精度具有更高的上限。而 OLTD、MCPM、SSFTI 等方法由没有消减不同链路带宽和阻塞带来的影响以及缺乏对基本拓扑结构的推断等算法设计问题, 随着发包数量的增加其精度增长缓慢。当发包数达到 3500 个时, SNTTI 算法的 F_1 分值可以达到 0.84, 相较于 SSFTI 方法 0.64 提高了 1.29 倍, 相较于 OLTD 方法 0.60 提高 1.4 倍, 相较于 MCPM 方法 0.58 提高了 1.44 倍; 当 F_1 分值接近 MCPM 方法的上限时, SNTTI 算法的发包数较 MCPM 方法减少了约 82%, 较 OLTD 算法较少约 57%, 较 SSFTI 算法减少约 46%, 因此 SNTTI 算法极大的提高了拓扑探测的精度和探测效率。

5 结论

本文利用网络层析方法对 IPv6 网络进行拓扑探测, 首先改进了传统的 Back-to-Back 探测包, 提出了带有链路状态罚项的公共路径度量公式, 减少了带宽对于探测结果的影响; 其次利用 SRv6 协议对于节点的基本拓扑以及插入位置进行了更加精准的判断, 该方法支持严格 SRv6 和松散 SRv6 两种环境; 最后提出一种拓扑融合方法来汇聚不同探测源的探测结果。该方法通过测试在不同的网络规模和拓扑结构中具有较高的准确率、召回率和探测效率, 较传统网络层析拓扑推断方法能够从 IPv6 网络中获取更多的有效信息支撑网络拓扑推断。本文将网络层析方法应用到 IPv6 环境中, 为后续 IPv6 网络的管理和测绘提供有效的信息支撑, 而对于非 SRv6 环境下的 IPv6 网络拓扑测量, 是未来的一个主要研究方向。

参考文献:

[1] Luo Z H, Liu J J, Yang G Z, et al. High-Speed Path Probing Method for Large-Scale Network[J]. *Sensors*, 2022, 22(15): 5650.

[2] Zhu X. Research on Active Detection Techniques for Large Scale Network Topology[D]. Harbin Engineering University, 2017.
(朱新立. 大规模网络拓扑主动探测技术研究[D]. 哈尔滨工程大学, 2017.)

[3] Gobjuka H, Breitbart Y J. Ethernet Topology Discovery for Networks with Incomplete Information[J]. *IEEE/ACM Transactions on Networking*, 2010, 18(4): 1220-1233.

[4] Zhou C J, Xing J G, Liu H B. Research on Network-Layer Topology Discovery Algorithm Based on Multi-Protocol[J]. *Computer Science*, 2017, 44(S1): 361-365.
(周长建, 邢金阁, 刘海波. 融合多协议的网络层拓扑发现算法研究[J]. *计算机科学*, 2017, 44(S1): 361-365.)

[5] Ye J. A study of optimal measurement and identification methods for network topology[D]. University of Electronic Science and Technology of China, 2020.
(叶剑. 网络拓扑结构的优化测量和识别方法研究[D]. 电子科技大学, 2020.)

[6] Smith K D, Jafarpour S, Swami A, et al. Topology Inference with Multivariate Cumulants: The Möbius Inference Algorithm[J]. *IEEE/ACM Transactions on Networking*, 2022, 30(5): 2102-2116.

[7] Bowden R, Veitch D. Finding the Right Tree: Topology Inference Despite Spatial Dependences[J]. *IEEE Transactions on Information Theory*, 2018, 64(6): 4594-4609.

[8] Fei G L, Ye J, Wen S, et al. Network Topology Inference Using Higher-Order Statistical Characteristics of End-to-End Measured Delays[J]. *IEEE Access*, 2020, 8: 59960-59975.

[9] Jiang S D, Yin W T, Yang J L, et al. Topology Inference Based on Maximum Common Path Matching[J]. *Acta Electronica Sinica*, 2016, 44(9): 2189-2196.
(姜守达, 尹文涛, 杨京礼, 等. 基于最大公共路径匹配的拓扑推断算法[J]. *电子学报*, 2016, 44(9): 2189-2196.)

[10] Ye J, Fei G L, Zhai X M, et al. Network Topology Inference Based on Subset Structure Fusion[J]. *IEEE Access*, 2020, 8: 194192-194205.

[11] Chan M N, Zhang Y. Target Selection Technique for IPv6

Network Topology Measurement[J]. *Intelligent Computer and Applications*, 2020, 10(9): 36-42.
(产毛宁, 张宇. IPv6 网络拓扑测量目标选择技术[J]. *智能计算机与应用*, 2020, 10(9): 36-42.)

[12] Zhu Z Y, Chen M, Wang Z F. 6Topo: A Novel Method of Measuring IPv6 Network Topology[J]. *Journal of Chinese Computer Systems*, 2020, 41(6): 1209-1215.
(朱正一, 陈鸣, 王占丰. 6Topo: 一种测量 IPv6 网络拓扑的新方法[J]. *小型微型计算机系统*, 2020, 41(6): 1209-1215.)

[13] Beverly R, Durairajan R, Plonka D, et al. In the IP of the Beholder[C]. *The Internet Measurement Conference 2018*, 2018: 308-321.

[14] Filsfils C, Camarillo P, Leddy J, et al. SRv6 network programming[J]. *Internet-Draft*, 2017.

[15] Rai A, Modiano E. Topology Discovery Using Path Interference[C]. *2019 IFIP Networking Conference* (*IFIP Networking*), 2019: 1-2.

[16] Ni J, Xie H Y, Tatikonda S, et al. Efficient and Dynamic Routing Topology Inference from End-to-End Measurements[J]. *IEEE/ACM Transactions on Networking*, 2010, 18(1): 123-135.

[17] Chang Y C, Lin H T, Chu H M, et al. Efficient Topology Discovery for Software-Defined Networks[J]. *IEEE Transactions on Network and Service Management*, 2020, 18(2): 1375-1388.

金睿 于 2021 年在山西大学电子商务专业获得学士学位, 现在南京理工大学网络与信息安全专业攻读硕士学位。研究领域为网络空间测绘、网络管理与度量等。Email: jinrui@njust.edu.cn

魏松杰 于 2009 在美国特拉华大学获得博士学位, 现任南京理工大学副教授, 研究领域为网络协议与服务、网络安全、区块链技术、无线网络与移动计算等。Email: swei@njust.edu.cn

李浩天 于 2022 年在南京理工大学网络空间安全专业获得学士学位, 现在南京理工大学网络与信息安全专业攻读硕士学位。研究领域为分布式系统和网络、区块链技术与应用等。Email: lht1115766130@njust.edu.cn