MASTER THESIS XIDIAN UNIVERSITY

西安电子科技大学

硕士学位论文

基于被动测量的网络拓扑推断及可视化研究

作者姓名 练鑫鑫

指导教师姓名、职称 姜奇 副教授

申请学位类别 工学硕士

学校代码 10701　　学　号 1503121488
分 类 号 TP39　　密　级 公开

西安电子科技大学

硕士学位论文

基于被动测量的网络拓扑推断及可视化研究

作者姓名：练鑫鑫

一级学科：计算机科学与技术

二级学科：计算机系统结构

学位类别：工学硕士

指导教师姓名、职称：姜 奇 副教授

学　　院：计算机学院

提交日期：2018 年 6 月

Passive Measurement Based Network Topology Inference and Visualization

A thesis submitted to

XIDIAN UNIVERSITY

in partial fulfillment of the requirements

for the degree of Master

in Computer System Architecture

By

Lian Xinxin

Supervisor: Jiang Qi Title: Associate Professor

June 2018

摘要

随着网络规模的迅速增长，网络结构愈加复杂，网络管理的难度越来越大。通过网络拓扑推断技术获得网络拓扑结构并利用可视化技术将网络拓扑结构以清晰准确的方式进行展现，可以为网络管理员分析网络流量、定位网络故障和优化网络配置提供极大的帮助。

基于被动测量的网络拓扑推断分为基于特定协议的网络拓扑推断和基于生存时间的网络拓扑推断。前者需要网络设备支持特定协议，例如 OSPF（Open Shortest Path First，开放式最短路径优先协议），该方法存在的问题是如果网络设备不支持特定协议，则无法推断网络拓扑结构。后者需要结合少量主动探测手段获取的网络路径推断网络拓扑结构，该方法面临着两个问题，一是推断的网络拓扑结构完整度较低，二是无法推断禁止主动探测的网络的拓扑结构。面对复杂的网络拓扑结构，如何以清晰直观的方式对其进行展示也是研究热点。现有的拓扑可视化工具只能使用特定格式的数据，不具备通用性。为了降低对特定协议的依赖，在减少探测数据包数量或不进行主动探测的情况下推断网络拓扑结构，并将其以清晰的方式进行展示，本文对基于生存时间的网络拓扑推断算法与可视化进行了研究。

首先，介绍了网络拓扑推断与可视化的相关背景和发展现状，对网络拓扑推断算法涉及的相关协议进行说明，研究了几种常见的网络拓扑结构类型，阐述了可视化的概念，对几种可视化布局算法进行描述并比较了它们的优缺点。

其次，提出了基于生存时间的网络拓扑推断算法 PFTTL，该算法在不进行主动探测的情况下，通过路由跳数和 IP（Internet Protocol，网际之间互联协议）地址前缀匹配长度推断网络拓扑结构。在此基础上，对已有的 NDTTL 算法进行分析，指出该算法存在完整度较低的问题，针对该问题，引入连接数预测机制并结合 PFTTL 算法，提出 EPREDICT 算法，在 CAIDA（The Center for Applied Internet Data Analysis，应用互联网数据分析中心）的真实数据集上进行实验，说明 EPREDICT 算法在减少了探测数据包数量的同时，提高了网络拓扑推断结果的完整度。

最后，设计并实现了适用于本课题的拓扑可视化系统，在分析了用户需求的基础上，将系统划分为数据文件解析模块、数据查询模块、推断算法模块、拓扑结构布局模块和图表显示模块。在此基础上，系统采用浏览器/服务器结构，使用 MySQL 数据库存储数据，通过 HTML5、CSS 和 JQuery 开发用户界面，ECharts 展示统计数据，通过对拓扑可视化系统的测试，说明系统可以较好的满足用户需求。

关 键 词：被动测量， 网络拓扑， 拓扑推断， 可视化

ABSTRACT

As the network structure becomes more complex, it's increasingly difficult for network administrators to manage network. The best way for them to analyze network traffic, locate the fault device and optimize network configuration is to obtain the network topology through network topology inference and use visualization technology to display it.

Passive measurement based network topology inference can be divided into specific protocol based network topology inference and Time-to-Life based network topology inference. The former requires the network equipment to support some specific protocols when infering network topology such as OSPF, and the weakness of this method is that it can't infer the network topology if the network equipment does not support specific protocol. The latter needs to use a small number of active measurements when infering the network topology, which faces two problems, the one is that it has low integrity of network topology and the other one is that this method cannot obtain network topology if the active measurement is prohibited. In addition, how to display a complex network topology in an intuitive way is also a big problem. In order to obtain network topology in the case of reducing the number of probe packets or without active measurement and display it in a clear way, we focus on Time-to-Life based network topology inference and visualization.

First, we briefly introduce the background and research status of network topology inference and visualization. Some important protocols are also described. Then we study several network topology types. We describe the concept of visualization and compare several visual layout algorithms.

Second, we propose a Time-to-Life based network topology inference algorithm which is named PFTTL and we can use it to infer the network topology by hop-count and the prefix matching length of two IP address without sending any probe packets. On the basis of PFTTL, we analyze NDTTL which is also a Time-to-Life based network topology inference algorithm and point out that NDTTL has the problem of low integrity. Therefore, we propose EPREDICT to solve the problem by predicting the number of edges. EPREDICT was tested on data and the result show that EPREDICT has higher integrity than NDTTL.

Finally, on the analysis of user's requirements, we design and develop a topology visualization system. The system has six modules, including data reading module, query statistics module, inference algorithm module, network layout module and network analysis module. We adopt B/S structure to develop topology visualization system and use MySQL to store data. In addition, we use HTML5, CSS and JQuery to develop the user interface of the system and use ECharts to display figures. Topology visualization system was tested and the result show that the system fulfill user's requirements.

Keywords: passive measurement, Network topology, topology inference, visualization

插图索引

图 2.1 IPv4 数据包格式 5
图 2.2 IP 地址与子网 6
图 2.3 最长前缀匹配示意图 7
图 2.4 ICMP 报文格式 9
图 2.5 SNMP 管理模型示意图 10
图 2.6 Router LSA 结构示意图 11
图 2.7 Network LSA 结构示意图 12
图 2.8 科学计算可视化的过程 13
图 2.9 树型布局算法结果 14
图 2.10 射线型布局算法结果 15
图 2.11 网格型布局算法结果 15
图 3.1 PFTTL 算法的流程图 18
图 3.2 数据包信息数据结构 18
图 3.3 计算路由跳数的伪代码 19
图 3.4 网络信息数据结构 20
图 3.5 层级节点集合示意图 21
图 3.6 第一个不相等的二进制位示意图 21
图 3.7 确定节点关系的伪代码 22
图 3.8 PFTTL 算法的完整度结果图 24
图 3.9 PFTTL 算法的准确度结果图 24
图 3.10 网络拓扑结构示例 25
图 3.11 预测源节点簇中边的总数的伪代码 32
图 3.12 确定源节点簇中边的伪代码 33
图 3.13 目标节点比例对算法完整度的影响 34
图 3.14 目标节点比例对算法准确度影响 35
图 4.1 拓扑可视化系统的总体架构 38
图 4.2 资源数据库表的关系 40
图 4.3 数据文件解析模块流程 41
图 4.4 OpenLayers 展示背景地图流程 44
图 4.5 ECharts 设置图表参数的格式 45
图 4.6 ECharts 绘制图形流程 46

图 4.7 拓扑可视化系统用户界面.....48
图 4.8 节点地理位置布局图.....49
图 4.9 节点连接图.....49
图 4.10 节点信息图.....50
图 4.11 节点归属地统计图.....50
图 4.12 节点类型图.....51

表格索引

表 4.1 拓扑结构表 39
表 4.2 网络节点表 39
表 4.3 连接关系表 40
表 4.4 网络数据表 40
表 4.5 拓扑结构布局模块的功能测试表 47
表 4.6 性能测试结果表 47

符号对照表

符号	符号名称
∑	求和
U	求并集
*	乘积

缩略语对照表

缩略语	英文全称	中文对照
ARP	Address Resolution Protocol	地址解析协议
BGP	Border Gateway Protocol	边界网关协议
CAIDA	The Center for Applied Internet Data Analysis	应用互联网数据分析中心
ICMP	Internet Control Message Protocol	网络控制报文协议
IP	Internet Protocol	网际之间互联协议
JDBC	Java Data Base Connectivity	Java 数据库连接
LSA	Link State Advertisement	链路状态广播
MMDB	Main Memory Data Base	主存数据库
OSPF	Open Shortest Path First	开放式最短路径优先
SNMP	Simple Network Management Protocol	简单网络管理协议
TTL	Time to Life	生存时间

目录

摘要 I
ABSTRACT III
插图索引 V
表格索引 VII
符号对照表 IX
缩略语对照表 XI
第一章 绪论 1
1.1 研究背景与意义 1
1.2 国内外研究现状 2
1.2.1 网络拓扑推断 2
1.2.2 可视化 3
1.3 论文主要工作 4
1.4 论文章节安排 4
第二章 相关技术介绍 5
2.1 网络基础知识 5
2.1.1 IP 协议 5
2.1.2 子网和子网掩码 6
2.1.3 最长前缀匹配原则 7
2.2 网络拓扑结构类型 7
2.3 常见的网络拓扑推断算法 9
2.3.1 基于 ICMP 协议的网络拓扑推断算法 9
2.3.2 基于 SNMP 协议的网络拓扑推断算法 10
2.3.3 基于 OSPF 协议的网络拓扑推断算法 11
2.4 可视化简介 12
2.5 网络拓扑可视化的图布局算法 13
2.6 本章小结 16
第三章 基于被动测量的网络拓扑推断算法研究 17
3.1 引言 17
3.2 PFTTL:基于生存时间的网络拓扑推断算法 17
3.2.1 算法概述 17
3.2.2 算法流程描述 17

3.2.3 算法验证与分析.....22
3.3 NDTTL 算法的改进.....25
3.3.1 NDTTL 算法的理论模型.....25
3.3.2 NDTTL 算法描述.....26
3.3.3 NDTTL 算法存在的问题.....29
3.3.4 EPREDICT:基于连接数预测机制的网络拓扑推断算法.....29
3.3.5 算法验证与分析.....33
3.4 本章小结.....35
第四章 拓扑可视化系统的设计与实现.....37
4.1 拓扑可视化系统需求分析.....37
4.1.1 系统功能性需求分析.....37
4.1.2 系统非功能性需求分析.....37
4.2 系统总体架构设计.....38
4.3 系统数据库设计.....39
4.4 系统功能模块详细设计与实现.....41
4.4.1 数据文件解析模块.....41
4.4.2 数据查询模块.....42
4.4.3 推断算法模块.....43
4.4.4 拓扑结构布局模块.....43
4.4.5 图表显示模块.....45
4.5 系统开发、运行与测试.....46
4.5.1 开发环境.....46
4.5.2 系统测试.....46
4.5.3 系统运行截图.....48
4.6 本章小结.....51
第五章 总结与展望.....53
5.1 本文总结.....53
5.2 展望.....53
参考文献.....55
致谢.....59
作者简介.....61

第一章 绪论

1.1 研究背景与意义

在“互联网+”的时代，网络出现在社会的各行各业中。庞大的网络用户和层出不穷的网络应用使得人们对网络的要求不断提高，特别是视频会议，在线教育和网络直播等多媒体服务产生的网络流量极大的增加了网络的压力。同时，无处不在的网络应用场景使得网络规模迅速扩大，网络结构越发复杂。这些都为网络管理、网络故障定位和网络流量分析带来了全新的挑战和更高的要求。

网络拓扑推断是发现网络中的网络设备以及各个网络设备之间连接关系的技术，其中，网络设备主要包括路由器和用户终端[1]。网络拓扑推断是了解网络拓扑结构的重要手段。通过对网络拓扑结构进行分析，网络管理员可以快速定位出现故障的网络设备，保证网络能够稳定地为用户提供服务。同时，网络管理员可以在真实的网络拓扑结构上进行仿真模拟，为优化网络结构提供参考经验[2]。

根据测量方式的不同，网络拓扑推断分为基于主动测量的网络拓扑推断和基于被动测量的网络拓扑推断[3]。前者通过向目标网络发送大量探测数据包的方式推断网络拓扑结构，这种方法会极大的增加网络流量。后者通过在目标网络中设置监测节点，监听和记录网络数据包，然后根据数据包中的相关信息推断网络拓扑结构，这种方法对网络的影响较小，因此适用于禁止大规模主动探测的网络。由于各种原因，在实际生活中某些网络的拓扑结构并不是公开可见的，同时，这些网络可能设置了较高等级的网络策略，例如限制用户的网络流量上限或禁止进行主动探测以维护网络的稳定。为了获取这些网络的拓扑结构，本文主要研究基于被动测量的网络拓扑推断。目前，基于被动测量的网络拓扑推断主要分为基于特定协议的网络拓扑推断和基于生存时间的网络拓扑推断。

基于特定协议的网络拓扑推断充分利用网络协议的工作原理，通过设置监测节点监听网络协议数据包，这些网络协议数据包携带了网络拓扑信息，例如 OSPF（Open Shortest Path First，开放式最短路径优先协议）数据包中的链路信息和 BGP（Border Gateway Protocol，边界网关协议）数据包中的路由节点集合等，然后通过分析组合这些信息得到网络拓扑结构。这种方法最大的优点是不需要向网络中发送探测数据包就能获得网络拓扑结构，但该方法的缺点也是显而易见的，如果网络设备不支持 OSPF 协议或 BGP 协议，则无法通过该方法推断网络拓扑结构。

与此不同，基于生存时间的网络拓扑推断通过设置监测节点监听 IP（Internet Protocol，网际之间互联协议）数据包，从 IP 数据包首部提取源 IP 地址和 TTL（Time

to Life，生存时间）信息，然后根据 TTL 和少量主动探测手段获取的网络路径推断网络拓扑结构。由于绝大部分的网络设备都支持 IP 协议，因此该方法的适用范围非常广。但是该方法面临着两个难题，一是在推断网络拓扑结构时，如果主动探测次数过少，该方法推断的网络拓扑结构完整度较低，二是如果目标网络禁止进行主动探测，则无法通过该方法获得网络拓扑结构。

在大规模的网络中，各个网络设备之间相互连接，网络路径数不胜数，纷繁复扰的网络拓扑结构难以被网络管理员理解与分析。面对庞大复杂的网络拓扑结构，如何通过一种清晰直观的方式观察和分析网络拓扑结构也是研究热点之一。虽然已有许多的网络拓扑可视化工具可供选择，但这些工具都存在着一些问题，例如只能使用特定格式的网络数据和功能单一等。因此，利用可视化技术设计并实现一个功能丰富的拓扑可视化工具可以为网络管理者分析网络流量、研究网络结构模型提供有效帮助。

本文旨在降低对特定协议的依赖，在减少探测数据包数量或不进行主动探测的情况下推断网络拓扑结构，并将其以准确直观的方式进行展现。因此，本文主要研究基于生存时间的网络拓扑推断算法与可视化。

1.2 国内外研究现状

1.2.1 网络拓扑推断

网络拓扑结构是了解和分析网络的基础，因此，网络拓扑推断在网络中具有十分重要的作用，备受各国研究学者的关注。目前，基于主动测量的网络拓扑推断通常将 Traceroute 作为测量工具。例如，CAIDA（The Center for Applied Internet Data Analysis，应用互联网数据分析中心）开展了 Skitter、Ark 等多个网络拓扑探测与推断项目，这些项目利用 Traceroute 工具获取网络路径，从而构建网络拓扑结构[4]。但美中不足的是，这种方法需要向网络中发送大量的探测数据包，从而增大网络压力，甚至导致网络出现阻塞。

基于被动测量的网络拓扑推断通过在网络中设置分布式的监测节点采集网络数据，然后根据网络数据推断网络拓扑结构。经过各国学者的多方努力，基于被动测量的网络拓扑推断已经取得一些成果。Pandey S 等人提出了基于 SNMP（Simple Network Management Protocol，简单网络管理协议）的网络拓扑推断算法[5]。该算法首先在监测节点上运行 SNMP 服务，通过 SNMP 服务访问管理信息库，其次从管理信息库中读取网络设备信息和路由表，最后将获取的所有网络设备信息和路由表分析整理，得到网络拓扑结构。该算法容易实现简单并且准确率高，但不足之处在于，如果其他网络设备没有提供 SNMP 服务，则无法从管理信息库中获得网络设备信息和路由表，从而导致网络拓扑推断失败。潘楠等人提出了基于 OSPF 的网络拓扑推断算法[6]。该

算法通过监测节点监听网络中的 OSPF 报文，分析报文中 Router LSA（Link State Advertisement，链路状态广播）和 Network LSA 信息，从而推断网络拓扑结构，但这种方法只能发现单一区域的网络拓扑结构，如果网络中不支持 OSPF 协议，则该算法无法推断网络拓扑结构。Jin C 等人提出根据 IP 数据包中的 TTL 计算网络节点到监测节点的网络距离的方法[7]，这种方法为被动测量方式的网络拓扑推断提供了新的思路。Eriksson B 等人提出了一种基于生存时间的网络拓扑推断算法[8]。该算法首先在待测网络中设置多个监测节点，每个监测节点监听经过其自身的 IP 数据包，然后根据 IP 数据包中的 TTL 字段为每个节点构造路由向量，根据路由向量对节点进行聚类，每个聚类就表示一个子网。但该算法的不足之处在于，如果想要获取整个网络拓扑结构，该算法需要进行少量的主动探测，例如通过 Traceroute 工具获取监测节点与聚类中部分节点的网络路径，此外，该算法只能获得每个聚类中部分节点与监测节点之间的网络路径，而无法确定聚类中各节点之间的连接关系，从而导致网络拓扑结构的完整度较低。蒋锦林提出了一种基于路由跳数的局域网被动发现方法[9]，该方法首先根据 TTL 计算路由跳数，其次根据路由跳数对局域网进行子网划分，然后解析监听的 ARP（Address Resolution Protocol，地址解析协议）报文获得 IP 地址与物理地址的对应关系，以此推断各子网之间的联系，但该方法只适用于局域网的拓扑推断。

1.2.2 可视化

可视化技术的发展为网络管理员理解和分析网络拓扑结构提供了强有力的技术支持。目前，已经有非常多的拓扑可视化工具，各个拓扑可视化工具都各有所长。例如，CAIDA为Skitter和Ark项目研发了基于地理位置信息的拓扑可视化工具Mapnet、GeoPlot和 3D可交互的拓扑可视化工具Walrus[10]，这些工具可以清晰、直观地向网络管理员展现网络拓扑结构。NLANR更加关注网络拓扑结构的数据分析，他们根据网络数据的特点开发了用于网络流量测量和网络性能分析的拓扑可视化工具Cichild[11]，该工具不仅可以展示网络拓扑结构，也具有强大的网络拓扑结构信息分析与处理能力。Bell实验室针对纷繁复扰的网络路径难以清晰显示的问题[12]，根据网络路径信息的特点改良了网络拓扑可视化的图布局算法，将拓扑可视化布局过程分层处理，不仅减少了单次处理网络数据的规模，同时也体现了网络拓扑结构增量布局的过程。DELIS小组根据BGP数据包推断网络拓扑结构并开发了基于BGP的拓扑可视化工具BGPlay[13]，该工具采用改进的图布局算法展示网络拓扑结构，展示效果更好。Au S C等人开发了VLNT可视化工具，该工具采用混合布局算法，提高了网络拓扑结构的布局效率[14]。程远等人提出的基于斥力和张力模型的网络拓扑结构布局算法，这种方法可以较好的对节点进行布局[15]。除此之外，还有非常多的研究人员将可视化技术应用于网络拓扑结构中，这些研究学者为网络拓扑可视化的图布局算法，拓扑可视化系统的研发作出

了非常大的贡献，同时也对可视化相关技术领域的发展与进步起到了积极的作用。

然而，这些拓扑可视化工具都是根据研究学者的自身需求所设计，在使用这些拓扑可视化工具时需要特定格式的网络数据，并不具有通用性。

1.3 论文主要工作

本文旨在设计并实现被动测量下基于生存时间的网络拓扑推断算法及拓扑可视化系统。本论文的主要工作包括以下 3 个方面：

1）简要说明了网络拓扑推断与可视化的研究背景与意义，对网络拓扑推断算法涉及的基础知识进行简单介绍，分析了不同网络拓扑结构类型和常见的网络拓扑推断算法，阐述了可视化技术的内涵，并对多种可视化布局算法进行说明和比较。

2）提出了一种无需进行主动探测的基于生存时间的网络拓扑推断算法，描述算法的流程，并对算法进行实验验证。然后在网络允许进行主动探测的基础上，分析了已有的基于生存时间的网络拓扑推断算法，说明算法存在的问题，提出算法改进方案，对改进后的算法进行实验与验证。

3）设计并实现了一个适用于本课题的拓扑可视化系统。首先分析系统的功能性需求和非功能性需求，根据系统的需求分析完成系统总体架构设计。其次对系统的数据库设计进行说明。然后详细阐述系统各个功能模块的实现方式。最后完成系统的开发、运行与测试工作，给出系统运行截图和测试结果，并对系统测试过程中发现的问题进行讨论。

1.4 论文章节安排

本论文的章节安排如下：

第一章 绪论。简述网络拓扑推断和可视化的研究背景与研究现状，介绍本论文的主要工作，并简单说明本论文的章节安排。

第二章 相关技术介绍。根据本课题的技术背景，讨论现有的网络拓扑结构类型、常见的网络拓扑推断算法、可视化技术和可视化布局算法。

第三章 基于被动测量的网络拓扑推断算法研究。分别对两种不同的被动测量下基于生存时间的网络拓扑推断算法进行说明，并对每种算法进行实验验证。

第四章 拓扑可视化系统的设计与实现。首先根据用户需求完成系统总体架构设计和数据库设计，然后阐述系统各个功能模块的实现方式，最后完成系统的测试工作。

第五章 总结与展望。总结被动测量下基于生存时间的网络拓扑推断算法和拓扑可视化系统的研究成果，对网络拓扑推断算法和拓扑可视化系统的下一步研究方向进行说明。

第二章 相关技术介绍

本章主要对网络拓扑推断算法和网络拓扑可视化相关的理论知识和技术原理进行介绍。首先介绍与网络拓扑推断算法相关的基础知识，然后对网络拓扑结构类型和常见的网络拓扑推断算法进行说明，最后对网络拓扑可视化的图布局算法进行概述。

2.1 网络基础知识

2.1.1 IP 协议

IP 协议是网络中应用最广泛的协议之一，它规定了网络设备之间进行通信时应该遵守的规则。在网络中利用 IP 协议进行数据通信时，IP 协议会完成数据传输等功能，图 2.1 是 IPv4 版本协议的数据包格式。

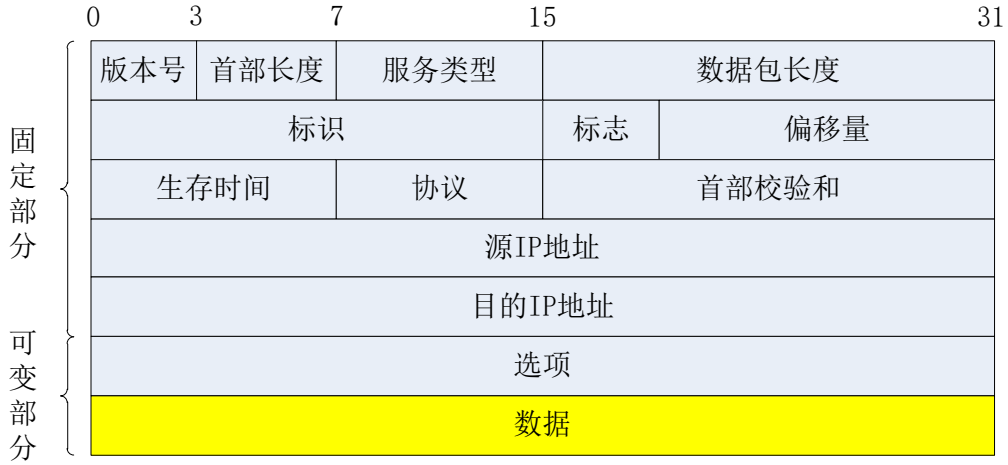


图 2.1 IPv4 数据包格式

IPv4 数据包中各字段含义如下：

- 版本号：占用 4 个比特位，网络设备通过查看版本号判断如何解析 IP 数据包，目前有 IPv4 和 IPv6 版本。
- 首部长度：占用 4 个比特位，通过查看首部长度，网络设备可以知道 IP 数据包中数据部分具体的起始位置。
- 服务类型：占用 8 个比特位，网络设备通过设置服务类型的不同比特位区分不同类型的 IP 数据包。
- 数据包长度：占用 16 个比特位，该字段表示 IP 数据包的总长度。
- 标识：占用 16 个比特位，该字段通过计数的方式产生 IP 数据包的序号。

- 标志：占用 3 个比特位，如果 IP 数据包过大，会对其进行分片并设置该字段的值，网络设备根据该字段判断 IP 数据包的分片情况。
- 偏移量：占用 13 个比特位，该字段表示分片后的 IP 数据包在分片前 IP 数据包中的相对偏移量。
- 生存时间：占用 8 个比特位，该字段保证 IP 数据包不会永远在网络中循环传输。
- 协议：占用 8 个比特位，根据该字段将 IP 数据包的数据交给对应的协议处理。例如，字段值为 1 表示数据交给网络控制报文协议处理。
- 首部校验和：占用 16 个比特位，网络设备通过查看该字段检测 IP 数据包是否存在比特错误，例如，IP 数据包被破坏、篡改等。
- 源 IP 地址：占用 32 个比特位，该字段表示发出 IP 数据包的网络设备的 IP 地址。
- 目的 IP 地址：占用 32 个比特位，该字段表示 IP 数据包目的网络设备的 IP 地址。
- 选项：该字段表示是否允许扩展 IP 数据包的首部。

2.1.2 子网和子网掩码

在实际网络中，每个用户终端拥有一个 IP 地址，路由器拥有多个路由接口，每个路由接口拥有一个 IP 地址。在确定网络区域时，通常将路由器的每个路由接口相互分离，每个路由接口与用户终端通过点对点链路连接在一起，从而产生多个独立的小型网络，这些独立的小型网络称为子网[16]。

每个子网拥有一个唯一的子网掩码。子网掩码是一个 32 比特的二进制数，它将 IP 地址分割为网络地址和主机地址两部分[17]。通过子网掩码，很容易知道 IP 地址所归属的子网与其他子网的关系，从而方便对 IP 地址的寻址操作。

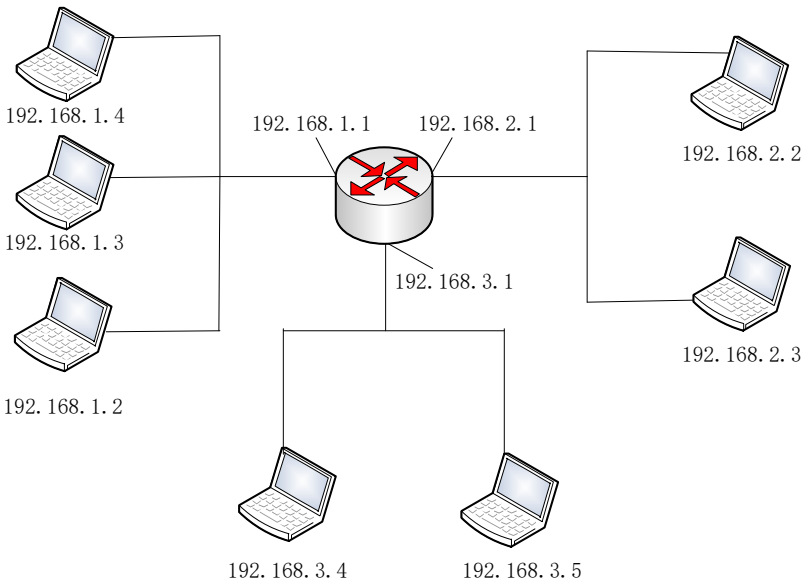


图 2.2 IP 地址与子网

在图 2.2 中，路由器有 3 个路由接口，同时存在 3 个子网，每个子网拥有唯一的子网掩码。其中，IP 地址为 192.168.1.1 的路由接口与 IP 地址为 192.168.1.2、192.168.1.3 和 192.168.1.4 的 3 个用户终端属于同一个子网，子网掩码为 192.168.1.0/24。同理，IP 地址为 192.168.3.1 的路由接口与 IP 地址为 192.168.3.4 和 192.168.3.5 的 2 个用户终端属于同一个子网，子网掩码为 192.168.3.0/24。IP 地址为 192.168.2.1 的路由接口与 IP 地址为 192.168.2.2 和 192.168.2.3 的 2 个用户终端属于同一个子网，子网掩码为 192.168.2.0/24。

2.1.3 最长前缀匹配原则

最长前缀匹配是大部分路由器的默认寻址方式。它是指路由器在传输 IP 数据包时，首先将 IP 数据包中目的 IP 地址与本地路由表中的表项逐一比对，直到找到前缀匹配长度最大的表项，然后将该表项设置为 IP 数据包的下一跳地址[18]。图 2.3 是最长前缀匹配的示意图。

图 2.3 最长前缀匹配示意图

在图 2.3 中，路由器在传输目的 IP 地址为 192.168.1.3 的 IP 数据包时，将目的 IP 地址与本地路由表中的 3 个路由表项进行比对，由于路由表项 1 的前缀匹配长度最大，路由器会将该 IP 数据包的下一跳地址设置为 192.168.1.1。

2.2 网络拓扑结构类型

网络拓扑结构是指网络设备之间的物理连接方式。一般来说，网络中的网络设备可以分为两种：一种是用于交换和传递网络数据的传输节点，包括交换机和路由器，另一种是访问节点，例如用户终端。根据不同的用户需求，目前网络拓扑结构主要分为总线型拓扑结构、星型拓扑结构、环型拓扑结构、网状拓扑结构、树型拓扑结构和混合型拓扑结构[19]。

（1）总线性拓扑结构

总线型拓扑结构是最简单的网络拓扑结构类型，它通过一条公用的数据总线作为传输媒介，多个用户终端直接连接总线并且网络设备之间没有主从之分。任意网络设备都可以直接发送一条网络消息，该消息沿总线传输至其他所有的网络设备。总线型拓扑结构最大的优势是传输媒介的利用效率高，网络线路布置简单且易于扩展，但最大的缺点是一旦网络设备出现故障，网络管理员很难快速定位故障设备。

（2）星型拓扑结构

星型拓扑结构将网络设备划分为中央设备和非中央设备，中央设备负责集中式的数据通信管理，而其他非中央设备与中央设备直接连接，两个不同的非中央设备之间进行数据通信必须经过中央设备。星型结构的优点是方便对设备之间的数据通信进行管理，传输效率较高，但由于每一个非中央设备都需要与中央设备直接相连，通信链路的利用率非常低，同时，该结构需要中心设备具有非常高的可靠性。

（3）环型拓扑结构

环型拓扑结构是将多个网络设备通过点对点的连接方式首尾相接而成的拓扑结构。在环型拓扑结构中，环路上任何一个网络设备在接收到网络数据时，都会将该数据按照单向传输的方式传递给下一个网络设备。这种拓扑结构的好处是不存在链路的竞争，但缺点是一旦环路中某个网络设备出现故障，会导致整个网络故障。

（4）网状拓扑结构

网状拓扑结构与环型拓扑结构十分相似，都是基于点对点的形式将多个网络设备相互连接。但网状拓扑结构要求任意两个网络设备之间都需要进行连接，这种方式虽然可以避免由网络设备的单点故障导致的网络瘫痪，但其拓扑结构过于复杂且经济效益低下，在实际网络中一般不会采用网状结构。

（5）树型拓扑结构

树型拓扑结构将网络设备按照层次进行划分，最顶端的网络设备作为根节点，其余网络设备作为根节点的子节点，根据层次关系，子节点以分支的形式与根节点相连。树型拓扑结构可以视为分层的集中式数据交换管理结构，这种结构的优点是各分支相互独立，某一个分支内的网络设备故障不会导致其他分支网络出现故障，同时，其他的网络设备可以作为分支加入到网络中，扩展性非常高。然而，该结构对根节点的依赖性太大，如果根节点的可靠性较低，则容易导致网络出现故障。

（6）混合型拓扑结构

由于现实情况十分复杂，单一的网络拓扑结构无法满足用户的需求。因此，在设计实际网络的过程中，网络管理员通常会综合考虑多种应用场景的业务需求，在不同的应用场景采用不同的网络拓扑结构，从而使得整个网络性能最优，经济支出最小。例如，综合使用环型拓扑结构和树型拓扑结构等。

2.3 常见的网络拓扑推断算法

2.3.1 基于 ICMP 协议的网络拓扑推断算法

ICMP（Internet Control Message Protocol，网络控制报文协议）主要用于网络设备之间相互交换网络层信息，例如，传递“目标网络不可达”、“目标端口不可达”等差错报文。ICMP 报文包含 ICMP 报文首部和数据部分[20]。图 2.4 为 ICMP 报文格式。

图 2.4 ICMP 报文格式

ICMP 报文中各字段含义如下：

- 类型：占用 8 个比特位，该字段表示 ICMP 报文类型，包括“回显回答”、“目标网络不可达”、“目标主机不可达”、“回显请求”等总计 14 种 ICMP 报文类型。
- 代码：占用 8 个比特位，该字段表示 ICMP 报文的代码，它与类型字段一起共同标识 ICMP 报文的具体类型。
- 校验和：占用 16 个比特位，该字段用于检测 ICMP 报文在网络传输的过程中是否出现错误。
- 选项：该字段根据不同类型的 ICMP 报文具有不同的内容。

目前，Traceroute 工具是基于 ICMP 协议的网络拓扑推断中操作方式最简单，使用范围最广泛的网络探测工具。通过 Traceroute 工具可以确定一个网络设备到另一个网络设备之间的有效网络路径。从 2.1.1 节的 IP 协议中可知，IP 数据包每经过一个中间路由器，其首部的 TTL 值会减 1，若路由器接收到 TTL 值为 0 的 IP 数据包，则会直接丢弃 IP 数据包，并以 IP 数据包首部的源 IP 地址为目的地址发送 ICMP 超时报文[21]。Traceroute 的工作原理是通过发送递增的 TTL 的 IP 数据包确定网络设备之间的有效网络路径，Traceroute 详细的工作步骤如下：

1）源网络设备向目标网络设备发送一个 TTL 值为 1 的 IP 数据包；

2）接收到 IP 数据包的路由器将 IP 数据包首部的 TTL 值减 1，此时 IP 数据包首部的 TTL 值为 0，路由器丢弃该 IP 数据包，并向源网络设备发送 ICMP 超时报文；

3）源网络设备接收到 ICMP 超时报文，获得网络路径上第一个路由器的入口地址，然后源网络设备递增地设置 IP 数据包首部的 TTL 值，将 IP 数据包发向目标网络设备，直到某个 IP 数据包成功抵达目标网络设备。在此过程中，每次的 IP 数据包发送过程会获得网络路径上的一个路由器入口地址，根据 IP 数据包的发送顺序将各地

址组合在一起即可获得一条有效网络路径。

因此，基于 ICMP 协议的网络拓扑推断算法通过 Traceroute 工具向多个目标网络设备进行探测，然后将探测获取的所有网络路径进行合并得到网络拓扑结构。通过这种方法获取的网络拓扑结构准确度较高，但该方法需要向网络中发送大量的 ICMP 报文，从而增大网络的压力。

2.3.2 基于 SNMP 协议的网络拓扑推断算法

SNMP 协议是 TCP/IP 协议簇中的一个应用层网络管理协议，网络管理员可以通过该协议管理网络中支持 SNMP 协议的网络设备，例如修改网络设备配置信息等[9]。目前，SNMP 协议采用 Manager/Agent 模型管理网络设备。图 2.5 为 SNMP 的管理模型示意图。

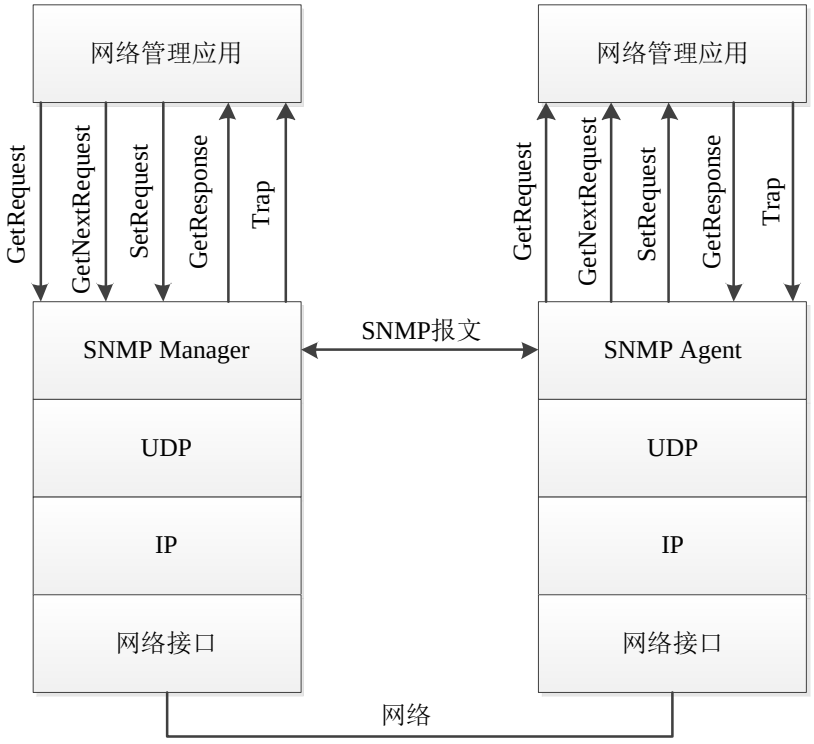


图 2.5 SNMP 管理模型示意图

SNMP 管理模型中的两个关键部分是 SNMP Manager 和 SNMP Agent。SNMP Manager 是网络设备上运行的网络管理进程，它可以向 SNMP Agent 发送请求或接收来自 SNMP Agent 的消息。SNMP Agent 是另一个网络设备上运行的网络代理进程，它不仅负责执行来自 SNMP Manager 的请求、向其发送请求结果和异常报告，还维护着一个保存了 ipRouteTable（路由表）的管理信息库，ipRouteTable 中包含 ipRouteDest（路由目的网络地址）、ipRouteMask（目的网络的掩码）和 ipRouteNextHop（下一跳

路由地址）等路由信息，这些信息都有助于发现网络拓扑结构。

基于SNMP协议的网络拓扑推断算法首先通过SNMP Agent访问网络设备的管理信息库，然后从管理信息库的路由表中读取 ipRouteNextHop 信息，接着以广度优先搜索或深度优先搜索的方式访问其他所有支持 SNMP 协议的网络设备，同时获取网络设备中的路由表信息，最后根据所有的路由表信息获得各网络设备之间的连接情况并绘制网络拓扑结构[9]。该方法具有较高的效率和准确率，但如果网络设备不支持 SNMP 协议，则无法通该方法获得网络拓扑结构。

2.3.3 基于 OSPF 协议的网络拓扑推断算法

OSPF 协议是一种链路状态协议。在一个自治域中，所有运行了该协议的路由器会主动探测与其相邻的路由器之间的链路状态，并将这些链路信息传递给与其相邻的其他路由器，而其他路由器会广播这些链路信息，每个路由器都可以根据接收到的链路信息构建路由表[9]。

现有的算法主要使用 OSPF 报文中的 Router LSA 信息和 Network LSA 信息推断网络拓扑结构。每台支持 OSPF 协议的路由器都可以对外发送携带 Router LSA 信息的 OSPF 报文，这些报文中携带着该路由器所维持的链路信息，主要包括生成该 LSA 的路由器 ID 和路由器网络连接的类型等。图 2.6 为 Router LSA 结构示意图。

图 2.6 Router LSA 结构示意图

与 Router LSA 不同，只有部分指定的路由器才能发送包含 Network LSA 信息的

OSPF 报文。Network LSA 中携带了所有接入了该网络的路由器信息，主要包括生成该 LSA 的路由器 ID、路由器所在网络的掩码和该网络中所有其他的路由器 ID。图 2.7 为 Network LSA 结构示意图。

图 2.7 Network LSA 结构示意图

基于 OSPF 协议的网络拓扑推断算法通常在网络中设置监测节点，然后监听包含 Router LSA 信息和 Network LSA 信息的 OSPF 报文，接着从中获取网络中所有的路由器 ID 和各路由器之间的链路状态信息，最后根据这些信息构建网络拓扑结构。这种方法推断的网络拓扑结构准确度较高，但所涉及的路由信息较为复杂。此外，如果网络设备不支持 OSPF 协议，则无法通过该方法获得网络拓扑结构。

2.4 可视化简介

将事物流程或数据以图形图像的形式进行展示的技术称为可视化。计算机视觉技术的发展使得可视化的含义越发丰富。目前，可视化的研究方向分为科学计算可视化、数据可视化和信息可视化[22]。

（1）科学计算可视化

科学计算可视化是指通过图形图像处理和计算机视觉等技术将数学研究模型的工作流程和大规模的科学工程数据以图形图像的方式进行展现并交互的技术[23]。通过科学计算可视化技术，研究人员可以更加清晰地了解数学模型的建模流程和工程数据的计算结果。

科学计算可视化主要包括模拟、映射、绘制和反馈四部分[24]。模拟部分负责对现实世界中的研究对象进行数学建模，然后按照某种规则将研究对象的原始数据转换为数学上的多维度数据。映射部分通过事先定义的公式对多维度数据进行处理和映射，然后通过空间几何的方式表示经过处理后的结果。绘制部分负责将空间几何形式的结果以清晰、直观的图形图表方式进行展现。反馈部分则由研究人员根据科学方法论和

实际科学工程经验，对所获得的工程数据和结果进行分析，然后根据分析结果不断调整数学模型，从而使数学模型逐渐拟合现实世界的研究对象。图 2.8 是科学计算可视化的过程。

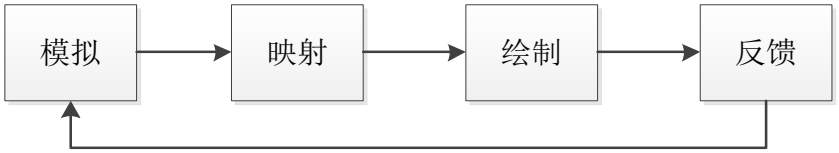


图 2.8 科学计算可视化的过程

（2）数据可视化

在“互联网+”的时代，数据无处不在。面对规模庞大的数据，数据可视化技术变得尤为重要。数据可视化是指对各种各样的数据进行可视化[25]。在应用数据可视化时，首先通过图形图像的形式表示单个数据，其次通过事先约定的数据组合规则将多个数据进行组合整理，然后利用图形图像处理的技术，以多维数据的方式对组合后的数据进行可视化。

数据可视化是一门侧重于数据表达、数据分析和挖掘的综合性研究学科[26]。大数据时代的到来，数据可视化已经成为理解和分析数据的有效手段。目前，数据可视化的技术方向涉及几何、像素和数据层次等多个方面。其中，基于几何的数据可视化是应用最广泛的数据可视化技术[27]。它通过数学几何图形的方式展现不同数据之间的关系，例如条形图、线形图和柱形图等。基于像素的数据可视化从显示设备角度出发，显示设备的屏幕上可以创建多个窗口，每个窗口表示一个维度的数据，每一个维度对应着一个像素，然后通过像素色彩的不同来实现数据可视化[28]。基于数据层次的数据可视化适用于具有层次结构特性的数据[29]，例如，企业的组织架构、计算机文件系统等，它可以帮助人们更容易地从层次型数据中发现数据之间的潜在联系。

（3）信息可视化

信息可视化建立在科学计算可视化和数据可视化的基础上，更加关注大规模的非数值型的抽象数据资源的展现[30]。信息可视化通过对多种多样的信息资源，如图表、文字等，进行过滤、存储和组织整理，借助计算机视觉设计技术为人们提供一个了解具有内在结构的抽象数据的手段。信息可视化技术主要应用在软件系统，如软件项目管理工具中各个代码文件之间的关联性展示，网站和数字图书馆中。

2.5 网络拓扑可视化的图布局算法

网络拓扑可视化的图布局算法是指网络拓扑结构中节点的布局方式，它是网络拓

扑可视化的重要内容之一。一个良好的布局算法可以清晰展示网络拓扑结构中各节点之间的连接关系。目前，布局算法主要分为物理位置布局和逻辑结构布局两种方式。其中，物理位置布局采用地理位置布局算法，该算法根据节点的地理位置信息，如经纬度，来放置节点。逻辑结构布局主要分为树型布局算法、射线型布局算法和网格型布局算法[10]。逻辑结构布局更关注网络拓扑结构的特性，按需分配节点的空间，确保节点之间不存在压盖情况。

（1）地理位置布局算法

地理位置布局算法是应用最广泛的布局算法之一。在使用地理位置布局算法对节点进行布局时，需要知道节点的经度和维度信息。如果是在 3D 立体地球地图中放置节点，则可以直接按照节点的经度和维度设置节点的坐标。如果是在 2D 平面地图中放置节点，则需要根据不同坐标系的转换规则，将节点的经度和纬度转换为 2D 平面地图的坐标。

（2）树型布局算法

树型布局算法将网络拓扑结构中的节点按照树的形式进行放置。树型布局算法主要有两种实现方式。第一种实现方式，首先利用图算法获得网络拓扑结构的最小生成树，其次按照最小生成树的层次关系将树中的节点进行布局，然后以最小生成树作为主干树，将未布局的节点依次加入主干树中，最后根据节点的布局情况对树进行修改以满足布局要求。第二种实现方式，首先从网络拓扑结构中随机选择一个节点作为树的根节点，设置根节点的初始坐标。其次从根节点出发，采用广度优先遍历算法，寻找根节点的每一层的子节点。然后设置节点的横向和纵向距离，计算每层子节点的初始坐标。最后根据初始坐标放置节点，并根据节点的布局情况调整各节点的坐标以满足布局要求[31]。树型布局算法结果的示例如图 2.9 所示。

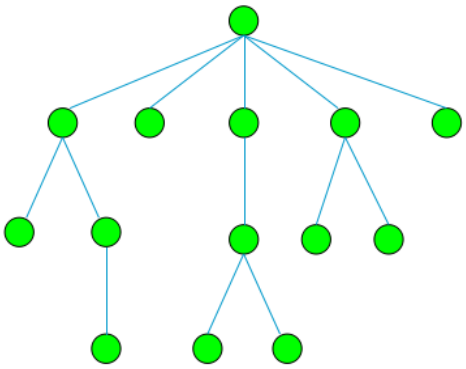


图 2.9 树型布局算法结果

（3）射线型布局算法

射线型布局算法借鉴了树型布局算法的思想。射线型布局算法首先利用图算法获

得网络拓扑结构的最小生成树，其次从最小生成树中随机选择一个节点作为初始节点并设置初始节点的坐标为中心位置，接着采用广度优先遍历算法，从初始节点出发遍历最小生成树，获取初始节点的每一层子节点，然后根据最小生成树的层次关系，逐一将子节点放置在以初始节点为圆心的同心圆上[32]。射线型布局算法结果的示例如图 2.10 所示。

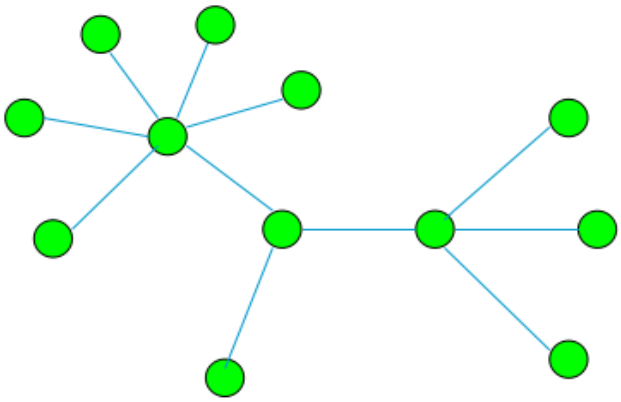


图 2.10 射线型布局算法结果

（4）网格型布局算法

与其他类型的布局算法不同，网格型布局算法在放置节点时，首先用直线将界面分割成多个各自独立的的矩形网格，直线相交的位置用于放置节点，节点之间的连线与横轴或纵轴平行，若不能平行，则可以采用直角弯的形式进行连接[11]。采用网格型布局算法对网络拓扑结构中的节点进行布局时，更像是将棋子放置在围棋盘上。网格型布局算法结果的示例如图 2.11 所示。

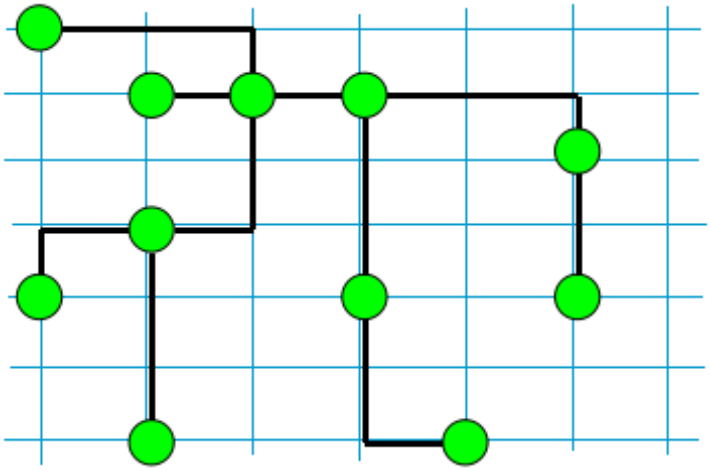


图 2.11 网格型布局算法结果

（5）混合型布局算法

混合型布局算法与网络中的混合型拓扑结构相对应。在对节点进行布局时，根据

节点的聚集情况对各区域的节点采用不同的布局算法。例如，如果某区域的节点数目较少，空间较大，则可以采用树型布局算法对该区域的节点进行布局，而对于节点数目较多，空间较小的区域，则采用射线型布局算法对节点进行布局。

（6）布局算法的比较

根据实际需求采用合适的布局算法对节点进行布局是非常必要的，下面对地理位置布局算法、树型布局算法、射线型布局算法和网格型布局算法进行比较和分析。

地理位置布局算法的实现难度较低，只需要知道节点的归属地信息，然后根据经纬度进行布局，这种方法的优点是节点具有地理位置信息，不足之处在于如果某个归属地的节点过多，会出现节点相互压盖的情况。

树型布局算法的实现难度较低，只需事先设置根节点的坐标，然后按照层次计算子节点的坐标，这种方法的优点是节点之间具有层级关系，符合树型拓扑结构的特性，不足之处在于该方法的空间利用率较低。

射线型布局算法的实现难度较低，在设置完初始节点的坐标后，将其余节点依次放置在初始节点的同心圆上，这种方法的优点是具有层次特性并且空间利用率较高，不足之处在于如果同一个圆上的节点过多，则容易出现节点相互压盖情况。

网格型布局算法实现难度较低，从理论上来说，可以通过不断分割矩形网格来增加放置节点的位置，这在一定程度上解决了节点之间相互压盖和节点布局过于集中的问题。但该方法最大的缺陷在于，如果节点规模过大，或者放置节点的位置不规范，则各节点之间的连线会变得错综复杂，难以理解。

2.6 本章小结

本章介绍了网络拓扑推断和网络拓扑可视化的协议和技术。首先，对网络拓扑推断算法涉及的 IP 协议和子网进行说明。其次，介绍了几种常见的网络拓扑结构类型和常见的网络拓扑推断算法。接着阐述了三种可视化技术的概念。最后，分析了网络拓扑可视化的图布局算法，并说明各布局算法的优缺点。

第三章 基于被动测量的网络拓扑推断算法研究

3.1 引言

通过被动测量方式推断网络拓扑结构可以降低对实际网络的影响，因此受到了研究人员的关注。被动测量下基于特定协议的网络拓扑推断算法需要网络设备支持特定的网络协议，例如 OSPF 协议，这种方法无法推断不支持特定协议的网络的拓扑结构。基于生存时间的网络拓扑推断算法需要进行少量主动探测，例如 Traceroute，这种方法无法对禁止主动探测的网络进行推断。在实际环境中，可能存在某些安全级别较高的网络，所有网络设备均不支持特定的网络协议也不响应 ICMP 报文，已有的基于被动测量的网络拓扑推断算法并不能在该环境中推断出网络拓扑结构。

因此，本章对被动测量下基于生存时间的网络拓扑推断算法进行了研究。首先针对无法进行主动探测，网络设备不支持特定协议的问题，根据 IP 数据包首部的 TTL 字段，提出一种不需要进行主动探测的基于生存时间的网络拓扑推断算法，并对该算法进行实验验证。然后讨论网络允许进行主动探测的情况，旨在减少探测数据包数量的同时提高算法所推断的网络拓扑结构的完整度，在分析了文献[8]提出的 NDTTL 推断算法的基础上，指出该算法存在完整度较低的问题，最后给出 NDTTL 算法的改进方案并进行实验验证。

3.2 PFTTL:基于生存时间的网络拓扑推断算法

3.2.1 算法概述

本文在深入分析 IP 协议和路由寻址方式的基础上，综合考虑 IP 地址之间的关联性，提出一种基于生存时间的网络拓扑推断算法，即 PFTTL 算法。PFTTL 算法在确定网络拓扑结构中各节点之间的连接关系时，采用分而治之的方法，将整个网络拓扑结构分割成多个小型网络拓扑结构，然后根据 IP 地址前缀匹配长度确定每个小型网络拓扑结构中各节点之间的连接关系，最后将多个小型网络拓扑结构进行组合，组合结果即为算法推断的网络拓扑结构。PFTTL 算法在推断网络拓扑结构时不需要网络设备支持特定的网络协议，也不需要进行主动探测，可以很好的满足在 3.1 节所述的网络中推断网络拓扑结构的需求。

3.2.2 算法流程描述

在详细描述 PFTTL 算法的各流程之前，有必要对 PFTTL 算法在推断网络拓扑结

构时使用的网络数据进行说明。因此，本节首先对 PFTTL 算法使用的网络数据进行介绍，然后对 PFTTL 算法中路由跳数的计算、邻接矩阵的创建和节点连接关系的确定三个部分进行说明。PFTTL 算法推断网络拓扑结构的流程如图 3.1 所示。

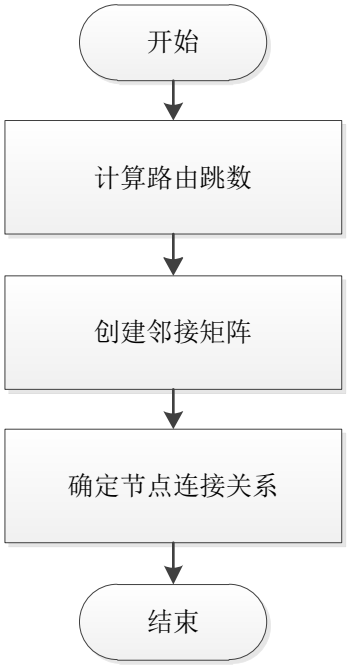


图 3.1 PFTTL 算法的流程图

（1）网络数据的获取与解析

在实际网络中，我们可以使用网络中 *K* 台网络设备，首先读取 *K* 台网络设备的 IP 地址，并将 *K* 台网络设备作为监测节点，然后在同一时间段内，每个监测节点以其自身的 IP 地址标记其监听的所有 IP 数据包，并将 IP 数据包以文件格式存储。

在解析 IP 数据包时，首先读取每一个 IP 数据包上的监测节点标记，从 IP 数据包首部提取源 IP 地址字段和 TTL 字段的值，并按照图 3.2 所示的数据结构进行存储。

```
Class Packet
{
        Int    ttl;
        String    srcIP;
        String    monitorIP;
}
```

图 3.2 数据包信息数据结构

（2）路由跳数的计算

从 2.1.1 节的 IP 协议中可知，IP 数据包首部的 TTL 字段表示该 IP 数据包在网络中可以经过的最大中间路由器数目。假设某个 IP 数据包被监测节点监听到时该 IP 数

据包首部的 TTL 值为T_{taken}，如果知道该 IP 数据包首部的 TTL 初始值T_{init}，则可以计算出该 IP 数据包从源网络设备到达监测节点的网络传输过程中经过的中间路由器数目，又称为路由跳数[7]。

因此，源网络设备与监测节点之间的路由跳数H计算公式如下：

$$H = T_{init} - T_{taken} \tag{3-1}$$

在实际网络中，IP 数据包首部的 TTL 初始值T_{init}根据网络设备操作系统的不同而各有差异。例如，Linux/Unix 类型的操作系统将 IP 数据包首部的 TTL 初始值设置为 64，Windows 系列的操作系统将 TTL 初始值设置为 128，其他类型的操作系统通常将 TTL 初始值设置为 32 或 255[8]。因此，本文将 IP 数据包首部 TTL 初始值的取值范围限定在 32、64、128 和 255。

此外，大量的网络数据统计结果表明，实际网络中任意两个网络设备之间的路由跳数小于 32[9]。因此，本文在计算路由跳数时，根据 IP 数据包抵达监测节点时的 TTL 值T_{taken}，从比T_{taken}大的所有限定值中选择最小的限定值作为该 IP 数据包首部的 TTL 初始值T_{init}。例如，某个 IP 数据包抵达监测节点时其首部的 TTL 值T_{taken}为 55，此时大于 55 的限定值有 64、128 和 255 且 64 是最小的限定值，所以该 IP 数据包首部的 TTL 初始值T_{init}为 64。图 3.3 为计算路由跳数的伪代码。

PFTTL 算法中计算路由跳数的伪代码	
输入：IP 数据包首部的 TTL 值T_{taken}	
输出：路由跳数	
1.	**If** $0 \le T_{taken} \le 32$ **then**
2.	$T_{init} = 32$
3.	**If** $32 < T_{taken} \le 64$ **then**
4.	$T_{init} = 64$
5.	**If** $64 < T_{taken} \le 128$ **then**
6.	$T_{init} = 128$
7.	**If** $128 < T_{taken} \le 255$ **then**
8.	$T_{init} = 255$
9.	$H = T_{init} - T_{taken}$

图 3.3 计算路由跳数的伪代码

在计算路由跳数后，将路由跳数、源 IP 地址和监测节点 IP 地址按照图 3.4 所示的数据结构进行存储。

```
Class Hoppacket
{
    Int    hop;
    String    srcIP;
    String    monitorIP;
}
```

图 3.4 网络信息数据结构

（3）邻接矩阵的创建

如果只关注网络拓扑结构中各节点之间的连接关系，则网络拓扑结构可以看作一个无向图。因此，可以用邻接矩阵表示网络拓扑结构。邻接矩阵是表示节点之间相邻关系的矩阵，通过查看矩阵中的元素，可以非常容易确定网络拓扑结构中任意两个节点之间是否存在连接关系。

在创建邻接矩阵时，需要知道网络拓扑结构中节点总数。为了方便计算，首先根据存储的网络数据为每个监测节点创建一个节点集合 A_i，然后计算所有节点集合 A_i 的并集，得到总节点集合 B，总节点集合 B 中的节点数目即为网络拓扑结构中节点总数。计算总节点集合 B 的公式如下：

$$B=\bigcup_{i=1}^{K}A_i \tag{3-2}$$

其中，K 表示监测节点的总数，A_i 表示第 i 个节点集合。

最后，根据总节点集合 B 中节点数目创建网络拓扑结构的邻接矩阵 D，并将邻接矩阵 D 中所有元素初始化为 0。

（4）节点连接关系的确定

每个监测节点都拥有一个与之对应的节点集合 A_i，节点集合 A_i 中所有节点与该监测节点之间的路由跳数是已知的。因此，采用分而治之的方法，将整个网络拓扑结构分割成 K 个小型网络拓扑结构，每个小型网络拓扑结构包含唯一的监测节点和该监测节点对应的节点集合 A_i 中的所有节点。然后分别确定每个小型网络拓扑结构中各节点之间的连接关系，根据节点之间的连接关系设置邻接矩阵 D 中对应元素的值，当确定所有小型网络拓扑结构中各节点之间的连接关系后，此时的邻接矩阵 D 就表示最终的网络拓扑结构。

对于每个小型网络拓扑结构，从树型拓扑结构的角度出发，首先将监测节点作为根节点，其余的节点作为子节点，然后根据各节点与监测节点之间的路由跳数，以监测节点为树型拓扑结构的第 0 层，依次设置其余节点在树型拓扑结构中的层次，每个层次包含多个子节点，得到多个层级节点集合。例如，某个监测节点对应的节点集合 A_i 中包含 5 个节点，其中，路由跳数为 0 的节点有 1 个，即监测节点，路由跳数为 1 的节点有 2 个，路由跳数为 2 的节点有 2 个。按照上述过程进行划分，则可以得到包含 1 个节点的第 0 层层级节点集合 C_0，包含 2 个节点的第 1 层层级节点集合 C_1 和包

含 2 个节点的第 2 层层级节点集合C_2。图 3.5 为层级节点集合示意图。

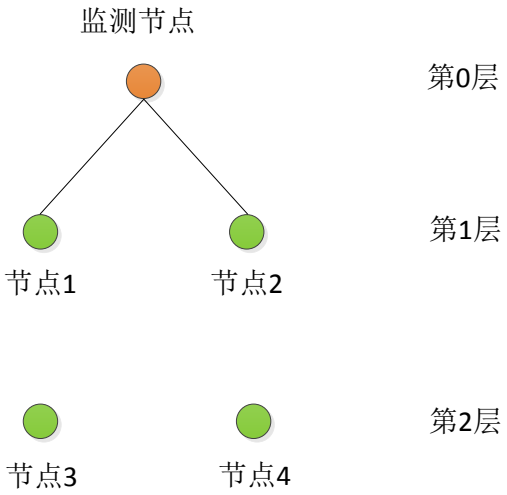


图 3.5 层级节点集合示意图

显然，第 0 层层级节点集合中的监测节点与第 1 层层级节点集合中的每个节点都存在连接关系。因此，在邻接矩阵 D 中设置监测节点与节点 1 和监测节点与节点 2 的对应元素的值为 1，表示它们存在连接关系。

对于其他层级节点集合中的节点，则需要根据 2.1.3 节中的最长前缀匹配原则确定各节点之间的连接关系，具体步骤如下：

第一步，选择第i层层级节点集合和第$i+1$层层级节点集合；

第二步，从第$i+1$层层级节点集合中任意选取一个节点，将该节点的 IP 地址逐一与第i层层级节点集合中的每一个节点的 IP 地址进行对比，在每一次对比过程中记录两个 IP 地址的第一个不相等的二进制位。图 3.6 为两个 IP 地址的第一个不相等的二进制位示意图。

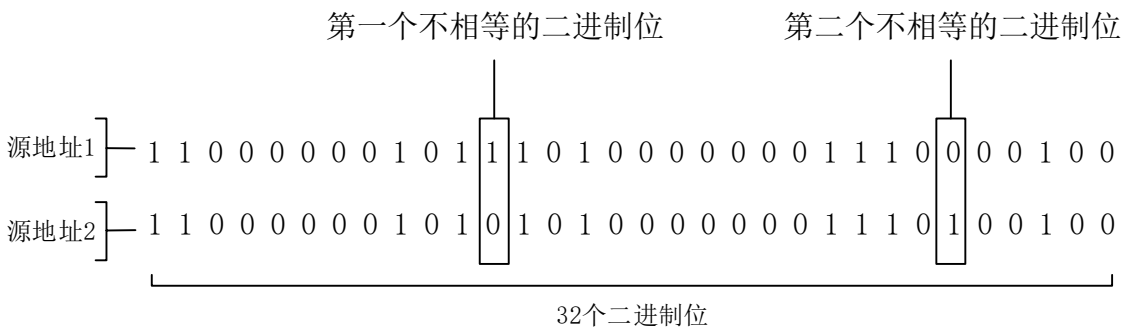


图 3.6 第一个不相等的二进制位示意图

第三步，根据第一个不相等的二进制位计算两个 IP 地址的前缀匹配长度。假设L

表示两个 IP 地址的前缀匹配长度，u 表示两个 IP 地址的第一个不相等的二进制位，则前缀匹配长度 L 的计算公式如下：

$$L = 32 - u \tag{3-3}$$

第四步，获得具有最大前缀匹配长度的两个 IP 地址所表示的节点，在邻接矩阵 D 中设置两个节点的对应元素的值为 1。图 3.7 为确定节点连接关系的伪代码。

PFTTL 算法中确定节点连接关系的伪代码

输入：k 个层级节点集合 $C_0, C_1, C_2, ..., C_{k-1}$

输出：邻接矩阵 D

```
1.  For  i = 0  to  k − 1  do
2.      If  i = 0  then
3.          For  ∀p ∈ C_{i+1}  do
4.              For  ∀q ∈ C_i  do
5.                  D_{p,q} = 1
6.      Else
7.          For  ∀p ∈ C_{i+1}  do
8.              Y_prefix = ∅
9.              For  ∀q ∈ C_i  do
10.                 L_{p,q} = PREFIX(p, q)
11.                 Y_prefix = Y_prefix ∪ {⟨⟨p, q⟩, L_{p,q}⟩}
12              ⟨x, y⟩ = MAX(Y_prefix)
13              D_{x,y} = 1
```

图 3.7 确定节点关系的伪代码

在图 3.7 的伪代码中，第 8 行表示将前缀匹配长度集合 Y_{prefix} 初始化为空集，第 10-11 行表示计算 p 和 q 的前缀匹配长度并将 p 、q 和前缀匹配长度加入集合 Y_{prefix}，第 12 行表示从集合 Y_{prefix} 中获取前缀匹配长度最大的节点对 $\langle x, y \rangle$。

3.2.3 算法验证与分析

科学技术在应用于现实环境之前需要进行大量的实验验证。本节首先介绍 PFTTL 算法的测试数据集，然后给出算法的评价指标，最后讨论 PFTTL 算法的实验结果。

（1）测试数据集

测试数据集主要分为模拟数据集和真实数据集两种。模拟数据集是指通过模拟工具构造的数据，模拟数据在一定程度上与真实数据具有相似的特性，但它并不能替代

真实数据，模拟数据集主要应用在真实数据采集困难的研究领域。真实数据集则是从现实环境中采集的具有实际意义的数据，在真实数据集上进行实验验证是评价算法效果的最佳选择。

为了保证实验结果的真实性，本论文采用 CAIDA 的 Ark 项目在 2014 年 4 月份采集的 IPv4 数据作为真实数据集。Ark 项目通过分布在世界各地的探测节点进行主动测量获取网络数据[33]。由于真实数据集较大，本论文从真实数据集中随机选取部分数据作为 PFTTL 算法的测试数据，该测试数据中包括 3276 个节点和 4531 个节点连接关系，为方便描述，以下称节点连接关系为边。

（2）算法评价指标

人们在对算法进行评价时，会根据关注角度的不同而使用不同的评价指标。本论文在对 PFTTL 算法进行实验验证时，主要关注算法的完整度和准确度。其中，完整度是指算法结果的边的总数与原始网络拓扑结构的边的总数的比率。完整度的计算公式如下：

$$Q = \frac{num_{pedge}}{num_{yedge}} \tag{3-4}$$

其中，Q表示完整度，num_{pedge}表示算法结果的边的总数，num_{yedge}表示原始网络拓扑结构的边的总数。

准确度是指算法结果中正确的边的总数与算法结果的边的总数的比率。准确度的计算公式如下：

$$Z = \frac{num_{pcorrect}}{num_{pedge}} \tag{3-5}$$

其中，Z 表示准确度，$num_{pcorrect}$表示算法结果中正确的边的总数。

（3）实验结果

在实验过程中，首先将测试数据中的 3276 个节点和 4531 个边作为原始网络拓扑结构，其次从 3276 个节点中随机选取 10 个监测节点、20 个监测节点、30 个监测节点、40 个监测节点、50 个监测节点和 60 个监测节点进行实验。为了减少实验中因随机选取节点造成的误差，将每个实验重复 20 次，然后将 20 次实验结果的平均值作为最终实验结果。

由图 3.8 可知，在本次试验中 PFTTL 算法的完整度约为 0.724，且小范围的增加监测节点的数目对提高完整度没有明显的效果，这是因为 PFTTL 算法只能确定路由跳数差值为 1 的节点之间的边，而无法确定具有相同路由跳数的节点之间的边。除此之外，PFTTL 算法每次只能确定一对节点之间的边，假设节点集合 A_i 中的节点总数为n，则 PFTTL 算法最终只能得到$n-1$个边。理论上，当监测节点的数目接近于网络拓扑结构中的节点总数时，每一个节点都可以经过一个中间路由器到达监测节点，

此时算法的完整度接近 1。

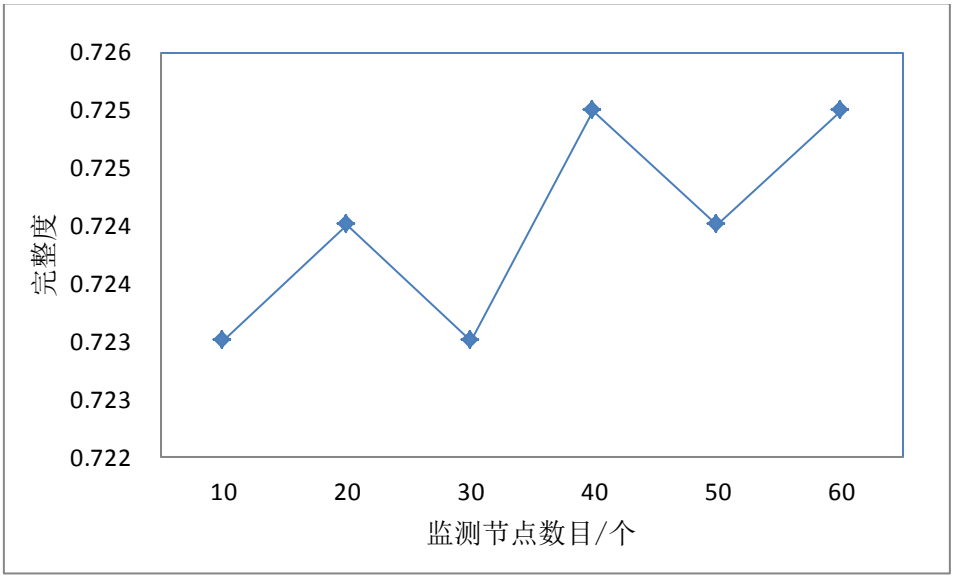


图 3.8 PFTTL 算法的完整度结果图

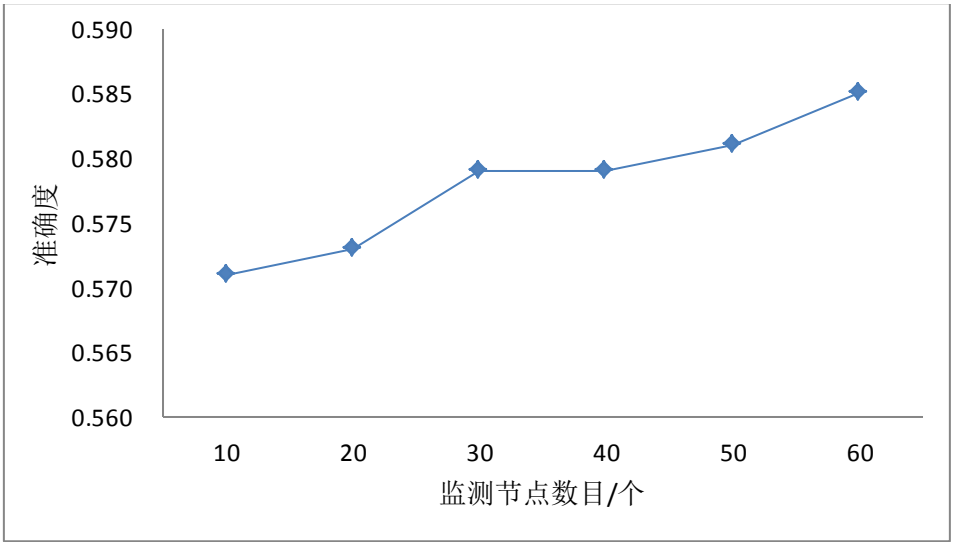


图 3.9 PFTTL 算法的准确度结果图

由图 3.9 可知，监测节点的数目越多，PFTTL 算法的准确度越高，当监测节点的数目接近于网络拓扑结构中的节点总数时，此时算法的准确度接近 1。

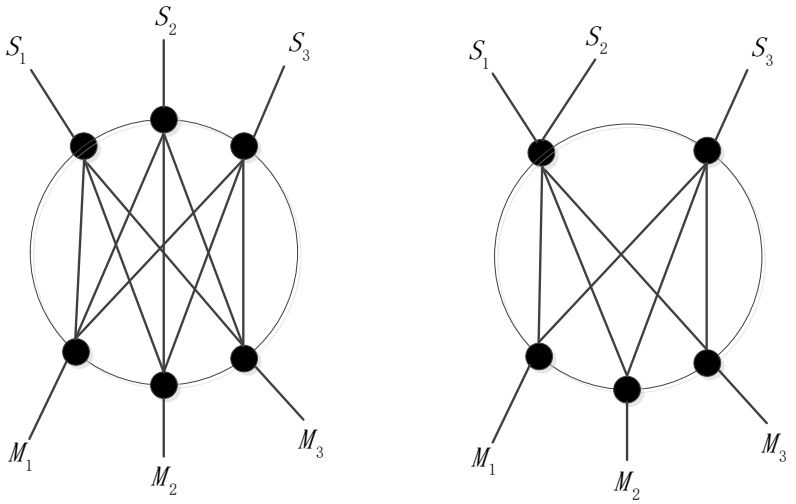
PFTTL 算法的优点是不需要进行主动探测便可以得到较为完整的网络拓扑结构。由于 PFTTL 算法利用 IP 地址的前缀匹配长度确定节点之间的边，因此该算法在推断 IP 地址分布集中，子网特性明显的局域网的拓扑结构，例如校园网等，可以获得较为准确的网络拓扑结构。如果用于推断主干网，则该算法的准确度有待提高。

3.3 NDTTL 算法的改进

在 3.1 节所述的网络中，如果网络设备可以响应 ICMP 报文，则可以通过少量主动探测手段提高网络拓扑推断算法的完整度和准确度。本节对文献[8]提出的 NDTTL 算法进行了分析，首先介绍该算法的理论模型，其次描述该算法的主要内容，然后指出该算法存在的问题，结合 PFTTL 算法的 IP 地址前缀匹配长度，给出改进算法，即 EPREDICT 算法，最后对改进后的 EPREDICT 算法进行实验验证。

3.3.1 NDTTL 算法的理论模型

在实际网络中，网络节点和监测节点之间可能存在多条网络路径，多条网络路径又可能相互交错，组成复杂的网络拓扑结构。然而，无论是多么复杂的网络拓扑结构，网络节点与监测节点之间的网络路径都是由一个或多个中间路由器连接而成。因此，在被动测量的基础上，网络拓扑结构可以划分为源节点层、中间路由层和监测节点层三个层次[34]。其中，源节点层包括被监测节点监听到的所有网络节点，以下称源节点。中间路由层包括源节点与监测节点之间的网络路径上的所有中间路由器，并且将距离源节点最近的中间路由器称为边界路由器。如图 3.10 所示，S_1、S_2 和 S_3 为源节点层中的 3 个源节点，M_1、M_2 和 M_3 为监测节点，图中的圆形区域表示中间路由器层。



(a)网络拓扑结构　　(b) 源节点共享边界路由的网络拓扑结构

图 3.10 网络拓扑结构示例

假设网络中有 K 个监测节点和 N 个源节点，第 i 个源节点 S_i 和第 k 个监测节点 M_k 之间的路由跳数为 $H_{i,k}$，由图 3.10 可知，路由跳数 $H_{i,k}$ 可以看作源节点 S_i 与边界路由器之间的路由跳数和边界路由器与监测节点 M_k 之间的路由跳数之和。假设源节点 S_i 与边界路由器之间的路由跳数为 x_i，边界路由器与监测节点 M_k 之间的路由跳数

为 $w_{i,k}$，则路由跳数 $H_{i,k}$ 可以表示：

$$H_{i,k} = x_i + w_{i,k} \tag{3-6}$$

对于两个源节点 S_i 和 S_j 共享一个边界路由器的情况，给出以下定理[8]：

定理 1、如果两个源节点 S_i 和 S_j 共享一个边界路由器，则这两个源节点与任意监测节点 M_k 之间的路由跳数差值为整型常数 E，即：

$$H_{i,k} - H_{j,k} = E \tag{3-7}$$

定理 1 证明：已知源节点 S_i 与监测节点 M_k 之间的路由跳数为 $H_{i,k}$，源节点 S_j 与监测节点 M_k 之间的路由跳数为 $H_{j,k}$，由于源节点 S_i 和源节点 S_j 共享一个边界路由器，因此对任意的监测节点 M_k，可以知道 $w_{i,k}$ 与 $w_{j,k}$ 相等，所以：

$$H_{i,k} - H_{j,k} = \left(x_i + w_{i,k}\right) - \left(x_j + w_{j,k}\right) = x_i - x_j = E \tag{3-8}$$

由此，定理 1 证明完毕。

定义 1、假设监测节点集合为 I_M，监测节点集合的一个子集为 $I_{M(D)}$，则源节点 S_i 和 S_j 在子集 $I_{M(D)}$ 上的均匀性测量[35]估计为：

$$D_{i,j}\left(I_{M(D)}\right) = \max_{k \in I_{M(D)}} \left(H_{i,k} - H_{j,k}\right) - \min_{k \in I_{M(D)}} \left(H_{i,k} - H_{j,k}\right) \tag{3-9}$$

定理 2、对于 $\forall M_k \in I_M$，若 $H_{i,k}$ 与 $H_{j,k}$ 的差值为整型常数 E，则：

$$\Delta_{i,j}\left(I_M\right) = 0 \tag{3-10}$$

定理 2 证明：若 $H_{i,k}$ 与 $H_{j,k}$ 的差值为整型常数 E，此时有：

$$\max_{k \in I_M} \left(H_{i,k} - H_{j,k}\right) = E \tag{3-11}$$

$$\min_{k \in I_M} \left(H_{i,k} - H_{j,k}\right) = E \tag{3-12}$$

根据定义 1 可知

$$\Delta_{i,j}\left(I_M\right) = \max_{k \in I_M} \left(H_{i,k} - H_{j,k}\right) - \min_{k \in I_M} \left(H_{i,k} - H_{j,k}\right) = E - E = 0 \tag{3-13}$$

由此，定理 2 证明完毕。

3.3.2 NDTTL 算法描述

NDTTL 算法在推断网络拓扑结构时，首先通过被动测量方式获取 IP 数据包，根据 IP 数据包首部的 TTL 值计算源节点与监测节点之间的路由跳数，并为每个源节点构造一个路由向量，该向量包括源节点与所有监测节点之间的路由跳数。其次将源节点的路由向量转换为差异向量，根据差异向量对所有源节点进行聚类，得到多个源节点簇，每个源节点簇中包含多个差异向量相等的源节点。然后从每个源节点簇中随机选取部分源节点作为目标节点，利用 Traceroute 工具确定监测节点与目标节点之间的网络路径，将所有网络路径的合并结果作为最终的网络拓扑结构。由此可知，NDTTL 算法的主要内容可以分为路由向量的构造、路由向量转换为差异向量、源节点聚类和网络拓扑结构的确定四个部分。

（1）路由向量的构造

源节点的路由向量是指源节点与所有监测节点之间的路由跳数所组成的向量。由3.3.1节可知，$H_{i,k}$表示第i个源节点S_i与第k个监测节点M_k之间的路由跳数，令HV_i表示第i个源节点S_i的路由向量，则有：

$$HV_i = \left[H_{i,1}, H_{i,2}, H_{i,3}, ..., H_{i,K} \right] \tag{3-14}$$

然而，被动测量方式并不能确保每个监测节点都能监听到网络中所有的源节点。因此，根据测量数据构造的路由向量可能存在部分路由跳数缺失的情况，从而导致无法对源节点进行聚类。

为了解决路由向量中部分路由跳数缺失的问题，Eriksson B等人讨论了均值填充法、ℓ范式填充法和网络中心填充法三种方法[35]。其中，均值填充法将未缺失的路由跳数的平均值作为缺失的路由跳数。ℓ范式填充法在计算源节点路由向量的缺失值时，首先找出距离该源节点最近的节点，然后将这些节点与监测节点之间的路由跳数的平均值作为缺失的路由跳数。网络中心填充法则考虑了网络拓扑结构特性，该方法在计算源节点路由向量的缺失值时，首先找出所有与源节点到监测节点上均匀性测量估计最小的节点，然后计算这些节点和源节点的路由向量中未缺失的路由跳数的平均差，最后根据平均差估计缺失的路由跳数。相比于其他两种填充方法，网络中心填充法更加具有优势。

（2）路由向量转换为差异向量

在获得源节点的路由向量后，并不能直接根据路由向量对源节点进行聚类，因为路由向量相近的源节点只是在欧几里得距离上相近，而并不代表源节点在实际网络拓扑结构中距离相近。如图 3.10(b)所示，假设源节点S_1、S_2和S_3 的路由向量相近，则直接根据路由向量对源节点进行聚类时，可能会将S_1、S_2和S_3 放置在同一个源节点簇中，但在实际的网络拓扑结构中，S_1和S_2属于同一个源节点簇，S_3 属于另一个源节点簇。

因此，在对源节点进行聚类之前需要对源节点的路由向量进行差异化处理，得到源节点的差异向量。源节点的差异向量计算公式如下：

$$HV_i' = HV_i - U_i e \tag{3-15}$$

其中，HV_i'表示第i个源节点S_i的差异向量，HV_i表示第i个源节点S_i的路由向量，U_i表示第i个源节点S_i与所有监测节点之间路由跳数的平均值，e表示与HV_i长度相等且元素值全为1的向量。

根据定理1，可以很容易知道源节点S_1和S_2的差异向量一定相等，而S_3 的差异向量不一定与S_1和S_2的差异向量相等。

（3）源节点聚类

在获得源节点的差异向量后，可以根据差异向量对源节点进行聚类。不同的聚类

算法会产生不同的聚类结果[36]。NDTTL 算法中分别使用了 K-means 聚类和高斯混合模型聚类两种聚类算法。K-means 聚类算法是一种基于距离的聚类算法，该算法将距离作为评价指标，两个节点之间的距离越近，则两个节点属于同一个簇的可能性越大[37]。高斯混合模型聚类算法是一种基于概率的聚类算法，高斯混合模型由多个不同的高斯分布组成，每一个高斯分布表示一个簇。利用高斯混合模型对源节点进行聚类的过程实质上就是通过差异向量估计每一个高斯分布的参数的过程[38]。

在此，只对 K-means 聚类算法的步骤进行具体说明。采用 K-means 聚类算法对源节点进行聚类的步骤如下：

第一步，从所有源节点中随机选取 k 个源节点作为中心节点，每个中心节点表示一个源节点簇；

第二步，根据源节点的差异向量，计算其他源节点与 k 个中心节点之间的距离，将源节点划分在与该源节点距离最小的中心节点所表示的源节点簇中；

第三步，对于每一个源节点簇，计算该簇中所有节点的差异向量的平均值，将该平均值作为新的中心节点；

第四步，重复执行步骤二和步骤三，直到各源节点簇不再发生变化或达到指定迭代次数。

（4）网络拓扑结构的确定

在获得多个源节点簇之后，通过使用少量的主动探测手段确定各源节点簇与监测节点之间的网络路径，然后根据网络路径确定网络拓扑结构。NDTTL 算法中分别使用了两种不同的利用网络路径确定网络拓扑结构的方法。第一种方法是直接将主动探测手段获得的网络路径作为网络拓扑结构，这种方法可以获得真实网络拓扑结构。

第二种方法首先给出两个节点之间共享路径长度的模型，其次将网络路径作为训练集对模型进行训练，然后根据训练好的模型预测两个节点与某个监测节点之间的共享路径长度，最后根据共享路径长度确定网络拓扑结构。这种方法在确定网络拓扑结构时加入了虚拟节点，因此，由第二种方法确定的网络拓扑结构并不是具有现实意义的网络拓扑结构。在此，只对第一种方法的步骤进行说明。通过第一种方法获得网络拓扑结构的步骤如下：

第一步，从每个源节点簇中随机选取部分节点作为目标节点；

第二步，以监测节点为探测源点，利用 Traceroute 工具确定各监测节点与所有目标节点之间的网络路径；

第三步，创建邻接矩阵，并将邻接矩阵初始化为 0；

第四步，对于每一条网络路径，依次获取网络路径上相邻的两个节点，并在邻接矩阵中将这两个节点的对应元素的值加 1。

3.3.3 NDTTL 算法存在的问题

文献[8]提出的 NDTTL 算法具有非常高的参考价值。该算法通过被动测量的方式获取推断网络拓扑结构所需要的 IP 数据包，降低了对特定网络服务的依赖。同时，该算法在确定网络拓扑结构时只需对聚类后的部分节点进行少量主动探测，减少了探测数据包的数量。该算法可以较好地应用于安全策略较高和限制网络数据流量上限的网络。

然而，NDTTL 算法在确定网络拓扑结构时仍然存在一些不足之处。如果直接采用第一种方法确定网络拓扑结构，则只能获得每个簇中部分节点与监测节点之间的网络路径，而无法确定簇中其他节点之间的边。假设某个节点簇中有 100 个节点，从簇中随机选择 10 个节点作为目标节点进行主动探测，得到这 10 个节点与各监测节点之间的网络路径。如果该簇中其他的 90 个节点都不在这些网络路径上，则无法确定这 90 个节点之间是否存在边，从而导致网络拓扑结构的完整度较低。

如果采用第二种方法，则只能获得逻辑上正确的网络拓扑结构，即网络拓扑结构中各节点之间的路由跳数是正确的，但除了少许真实的网络路径外，其他的中间节点都是虚构的。

3.3.4 EPREDICT:基于连接数预测机制的网络拓扑推断算法

针对 NDTTL 算法在确定网络拓扑结构时存在完整度较低的问题，本论文在 NDTTL 算法中引入连接数预测机制，结合 PFTTL 算法中的 IP 地址前缀匹配长度，提出基于连接数预测机制的网络拓扑推断算法，即 EPREDICT 算法。连接数预测机制是由 Roughan M 等人提出的预测网络拓扑结构中边的总数的方法，Roughan M 等人的实验结果表明，在 700 个监测节点的情况下，通过连接数预测机制可以预测 99.9% 的边[39]。

（1）连接数预测机制

在网络拓扑结构中，各节点之间的边只有存在和不存在两种情况，并且这两种情况相互对立。如果可以保证所有的监测节点是同质的，即各监测节点发现某个特定边的概率是相同的，同时，通过对边进行聚类的方式将网络拓扑结构分割成多个相互独立的小型拓扑结构，称之为连接簇，同一个连接簇中的各个边具有较大的关联性，而不同连接簇中各个边几乎不具有关联性。由此可知，单个连接簇中的边被监测节点发现的次数满足二项分布，因此，可以利用二项混合模型预测边的总数。

假设网络中存在 K 个独立的、同质的监测节点，某个连接簇为 F ，则可以知道连接簇 F 中的边被监测节点发现的次数满足二项分布，因此，连接簇 F 中的边同时被 K 个监测节点发现的概率为：

$$pro\{k\}=\binom{K}{k}p_F^k\left(1-p_F\right)^{(K-k)} \tag{3-16}$$

其中，p_F 表示连接簇 F 中的边被单个监测节点发现的概率。

如果知道连接簇 F 中的边的总数 E_F，则可以计算出 p_F：

$$p_F=\frac{\sum_{f\in F}obs_f}{E_F K} \tag{3-17}$$

其中，f 表示连接簇 F 中的一个边，obs_f 表示边 f 被监测节点发现的次数。

实际上，连接簇 F 中的边的总数 E_F 是未知的，但可以根据连接簇 F 中已经发现的边的次数，采用最大似然估计法，得到 p_F 的估计值 $\hat{p}_F$：

$$\hat{p}_F=\frac{\sum_{f\in F}obs_f}{E_F K} \tag{3-18}$$

由此，根据估计值 $\hat{p}_F$ 可以得到边的总数 E_F 的估计值 $\hat{E}_F$，然后将估计值 $\hat{E}_F$ 作为连接簇 F 中的边的总数。

然而，在现实网络中进行探测时至少需要一个监测节点，因此，连接簇 F 中的边同时被 k 个监测节点发现的条件概率为：

$$pro\{k\mid k>0\}=\binom{K}{k}\frac{p_F^k\left(1-p_F\right)^{(K-k)}}{1-\left(1-p_F\right)^K} \tag{3-19}$$

令 E_{obs}^F 表示连接簇 F 中已经被监测节点发现的边的数目，则有：

$$p_F=\frac{\left[1-\left(1-p_F\right)^K\right]\sum_{i=1}^{E_{obs}^F}obs_i}{E_{obs}^F K} \tag{3-20}$$

然后，通过迭代的方式求解估计值 $\hat{p}_F$：

$$\hat{p}_F{}^{(0)}=\frac{\sum_{i=1}^{E_{obs}^F}obs_i}{E_{obs}^F K} \tag{3-21}$$

$$\hat{p}_F{}^{(j+1)}=\frac{\sum_{i=1}^{E_{obs}^F}obs_i}{E_{obs}^F K}\left[1-\left(1-\hat{p}_F{}^{(j)}\right)^K\right] \tag{3-22}$$

根据估计值 $\hat{p}_F$ 得到边的总数 E_F 的估计值 $\hat{E}_F$：

$$\hat{E}_F=\frac{E_{obs}^F}{1-\left(1-\hat{p}_F\right)^K} \tag{3-23}$$

最后，将估计值 $\hat{E}_F$ 作为连接簇 F 中的边的总数。

（2）EPREDICT 算法描述

在确定网络拓扑结构时，如果能够保证各监测节点是同质的并且各监测节点之间相互独立，则可以通过连接数预测机制预测网络拓扑结构中边的总数，然后通过某种方法确定各节点之间的边，并且保证边的数目不超过连接数预测机制预测的边的总数，

从而提高网络拓扑推断算法的完整度和准确度。

通常情况下，距离相近的监测节点之间存在较大的关联性，而距离较远的监测节点之间关联性较小[36]。因此，可以通过对监测节点进行聚类的方式获得监测节点簇，同一个簇中的各监测节点存在较大的关联性，不同簇之间的监测节点几乎不存在关联性，然后从每个簇中选取一个监测节点，这些监测节点之间是相互独立的。

通过 EPREDICT 算法推断网络拓扑结构时，首先为每个源节点和每个监测节点构造路由向量，其次将路由向量转换为差异向量，根据差异向量分别对源节点和监测节点进行聚类，得到多个源节点簇和多个监测节点簇，接着利用 Traceroute 工具确定源节点与监测节点之间的网络路径，然后通过连接数预测机制预测各源节点簇中的边的总数，并根据 IP 地址前缀匹配长度确定簇中各节点之间是否存在边，最后用邻接矩阵表示网络拓扑结构。EPREDICT 算法的具体步骤如下：

第一步，构造源节点路由向量。通过被动测量方式获取 IP 数据包，利用 IP 数据包首部的 TTL 值计算各源节点与监测节点之间的路由跳数，将路由跳数进行组合得到路由向量，并采用网络中心填充法补全路由向量中缺失的路由跳数。

第二步，构造监测节点路由向量。利用 Traceroute 工具确定各监测节点之间的网络路径，根据网络路径获得各监测节点之间的路由跳数，将各监测节点之间的路由跳数和监测节点与其自身的路由跳数进行组合得到路由向量。其中，监测节点与其自身的路由跳数为 0。

第三步，路由向量转换为差异向量。利用差异向量公式将所有源节点路由向量和所有监测节点路由向量转换为差异向量。

第四步，对源节点和监测节点进行聚类。采用 K-means 聚类算法或高斯混合模型聚类算法分别对源节点和监测节点进行聚类，得到多个源节点簇和多个监测节点簇。

第五步，确定网络路径。从每个监测节点簇中随机选择一个监测节点作为探测源点，从每个源节点簇中随机选取部分节点作为目标节点，利用 Traceroute 工具确定所有探测源点与所有目标节点之间的网络路径。

第六步，创建邻接矩阵。根据所有网络路径、源节点和监测节点创建邻接矩阵。

第七步，预测源节点簇中边的总数。根据邻接矩阵统计各源节点簇中边被发现的次数，通过连接数预测机制预测各源节点簇中的边的总数。

第八步，确定源节点簇中的边。对于每一个源节点簇，首先根据源节点路由向量的层级关系将簇中所有节点两两配对，计算所有节点对的 IP 地址前缀匹配长度，其次按照前缀匹配长度从大到小的顺序对节点对进行排序，接着从 1 开始依次对排序后的节点对进行编号，然后将该簇的边的总数与已发现的边数的差值作为阈值，如果某个节点对的编号小于阈值，则在邻接矩阵中将对应元素的值加 1，否则，进行下一次判断。

（3）EPREDICT 算法的实现

EPREDICT 算法的第一步至第六步与 NDTTL 算法的步骤几乎一致，因此，本论文主要对 EPREDICT 算法中第七步的预测源节点簇中边的总数和第八步的确定源节点簇中的边这两个步骤的实现方式进行详细说明。

在图 3.11 的预测源节点簇中边的总数的伪代码中，第 2-4 行表示统计第i个源节点簇中的边a被各监测节点发现的次数，第 5-6 行表示统计第i个源节点簇中的所有边被各监测节点发现的次数，第 7-9 行表示通过迭代的方式计算第i个源节点簇中的边被单个监测节点发现的概率估计值，第 10 行表示连接数预测机制所预测的第i个源节点簇中边的总数。

EPREDICT 算法中预测源节点簇中边的总数的伪代码

输入：

- 邻接矩阵 D
- 监测节点数目 K
- k 个源节点簇 $SC_1, SC_2, SC_3, ..., SC_k$
- k 个源节点簇中已发现的边的数目 $E_{obs}^{SC_1}, E_{obs}^{SC_2}, E_{obs}^{SC_3}, ..., E_{obs}^{SC_k}$

输出：k 个源节点簇中边的总数 $\hat{E}_{SC_1}, \hat{E}_{SC_2} \hat{E}_{SC_3}, ..., \hat{E}_{SC_k}$

1. **For** $i=0$ **to** k **do**
2. **For** $\forall p \in SC_i$ **do**
3. **For** $\forall q \in SC_i$ **do**
4. $obs_a = D_{p,q}$
5. **For** $y=1$ **to** $E_{obs}^{SC_i}$ **do**
6. $edge_i = edge_i + obs_y$
7. $p_{SC_i}^{(0)} = edge_i / \left(E_{obs}^{SC_i} * K \right)$
8. **While** （ $j <$ 最大迭代次数） **or** （还未收敛） **do**
9. $p_{SC_i}^{(j+1)} = edge_i * \left[1 - \left(1 - p_{SC_i}^{(j)} \right)^K \right] / \left(E_{obs}^{SC_i} * K \right)$
10. $\hat{E}_{SC_i} = E_{obs}^{SC_i} / 1 - \left(1 - \hat{p}_{SC_i} \right)^K$

图 3.11 预测源节点簇中边的总数的伪代码

在通过连接数预测机制得到每个源节点簇中边总数后，在 PFTTL 算法的基础上，通过计算 IP 地址之间的前缀匹配长度确定源节点簇中各节点之间的边。在图 3.12 的确定源节点簇中边的伪代码中，第 2-3 行表示将第i个源节点簇的节点对集合和前缀匹配长度集合初始化为空集。第 4-10 行表示计算满足层级关系的节点对的前缀匹配长度，第 11 行表示对前缀匹配长度集合按照从大到小的顺序排序，第 12 行表示将节

点对编号，第 13-15 行表示在邻接矩阵中设置编号小于阈值的节点对中的节点之间存在边。

EPREDICT 算法中确定源节点簇中的边的伪代码

输入：

- 邻接矩阵 D
- k 个源节点簇 $SC_1, SC_2, SC_3, ..., SC_k$
- k 个源节点簇中边的总数 $\hat{E}_{SC_1}, \hat{E}_{SC_2} \hat{E}_{SC_3}, ..., \hat{E}_{SC_k}$
- k 个源节点簇中已发现的边的数目 $E_{obs}^{SC_1}, E_{obs}^{SC_2}, E_{obs}^{SC_3}, ..., E_{obs}^{SC_k}$

输出：邻接矩阵 D

1. **For** $i = 0$ **to** k **do**
2. $Pair_i = \varnothing$
3. $Y_{prefix}^i = \varnothing$
4. **For** $\forall p \in SC_i$ **do**
5. **For** $\forall q \in SC_i$ **do**
6. **If** p 和 q 路由向量存在差值为 1 的路由跳数 **then**
7. **If** $D_{p,q} < 1$ **then**
8. $Pair_i = Pair_i \cup \{\langle p, q \rangle\}$
9. $L_{p,q} = \text{PREFIX}(p, q)$
10. $Y_{prefix}^i = Y_{prefix}^i \cup \{\langle \langle p, q \rangle, L_{p,q} \rangle\}$
11. $\text{Sort}(Y_{prefix}^i)$
12. $\text{Mark}(Pair_i, Y_{prefix}^i)$
13. **For** $\forall < x, y > \in Pair_i$ **do**
14. **If** $mark_{<x,y>} \le \hat{E}_{SC_i} - E_{obs}^{SC_i}$ **then**
15. $D_{x,y} = D_{x,y} + 1$

图 3.12 确定源节点簇中边的伪代码

3.3.5 算法验证与分析

在真实数据集上对算法的完整度和准确度进行实验是必不可少的环节。本节同时对 NDTTL 算法和 EPREDICT 算法进行了实验，并对两个算法的实验结果进行比较。为了保证实验的真实性和一致性，本论文在实验过程中将文献[8]中验证 NDTTL 算法时所使用的数据作为真实数据集。真实数据集来源于 CAIDA 的 Skitter 项目在 2003 年 4 月 21 日至 5 月 8 日期间采集的网络路径。Skitter 项目通过分布在世界各地的探测节点利用 Traceroute 工具获取网络路径。真实数据集中包括 192,224 个节点和 609,066 个边[8]。由于真实数据集较大，本论文从真实数据集中随机选取部分数据作为测试数据，测试数据中包括 4639 个节点和 6581 个边。

NDTTL 算法通过两种不同的方法确定网络拓扑结构，由于第二种方法所确定的网络拓扑结构中存在虚拟节点，从而无法判断网络拓扑结构中边的准确性。因此，本论文在对 NDTTL 算法进行实验时采用第一种方法确定网络拓扑结构。在对 NDTTL 算法和 EPREDICT 算法的实验结果进行比较时，将 3.2.3 节中定义的完整度和准确度作为算法的评价指标。

在实验过程中，首先将测试数据中的 4639 个节点和 6581 个边作为原始网络拓扑结构，其次从 4639 个节点中随机选取 30 个监测节点进行实验。在利用 Traceroute 工具探测网络路径时，按照 10%、30%和 50%的比例从每个节点簇中随机选取节点，并将这些节点作为目标节点。为了减少实验中因随机选取节点造成的误差，将每个实验重复 10 次，然后将 10 次实验结果的平均值作为最终实验结果。图 3.13 是在 30 个监测节点的情况下，目标节点比例对算法完整度的影响的实验结果图。

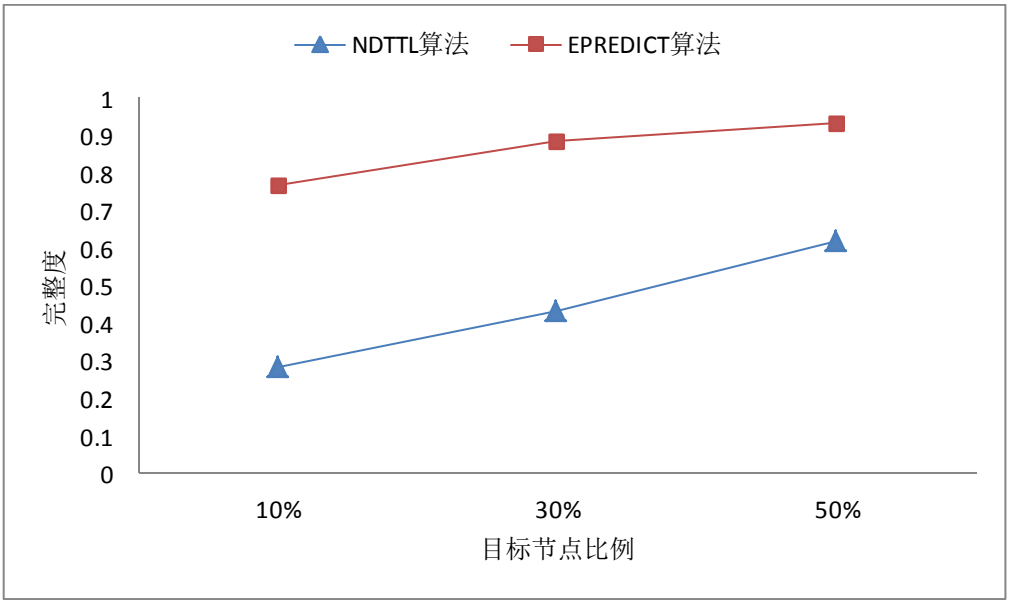


图 3.13 目标节点比例对算法完整度的影响

由图 3.13 可知，EPREDICT 算法比 NDTTL 算法具有更高的完整度。同时，目标节点比例对两个算法的完整度都有较大的影响，目标节点比例越高，算法的完整性越高。对于 EPREDICT 算法，目标节点数目越多，利用 Traceroute 工具获取的网络路径越多，连接数预测的边的总数也就更加准确，因此完整度越高。此外，NDTTL 算法在获取网络路径时，需要在 30 个监测节点上进行探测，但 EPREDICT 算法只需要在各监测节点簇中随机选取一个监测节点进行探测，从而减少了探测数据包的数量。

图 3.14 是在 30 个监测节点的情况下，目标节点比例对算法准确度的影响的实验结果图。由图 3.10 可知，NDTTL 算法比 EPREDICT 算法具有更高的准确度。同时，目标节点比例对 EPREDICT 算法的准确度影响较大，而对 NDTTL 算法的准确度没有

影响。这是因为 EPREDICT 算法在网络路径的基础上，通过连接数预测机制和 IP 地址之间的前缀匹配长度确定各节点之间是否存在边，这种方法存在一定的错误率。NDTTL 算法直接将网络路径作为网络拓扑结构，由于网络路径上的每个边都是原始网络拓扑结构中真实存在的，因此 NDTTL 算法的准确度为 1。

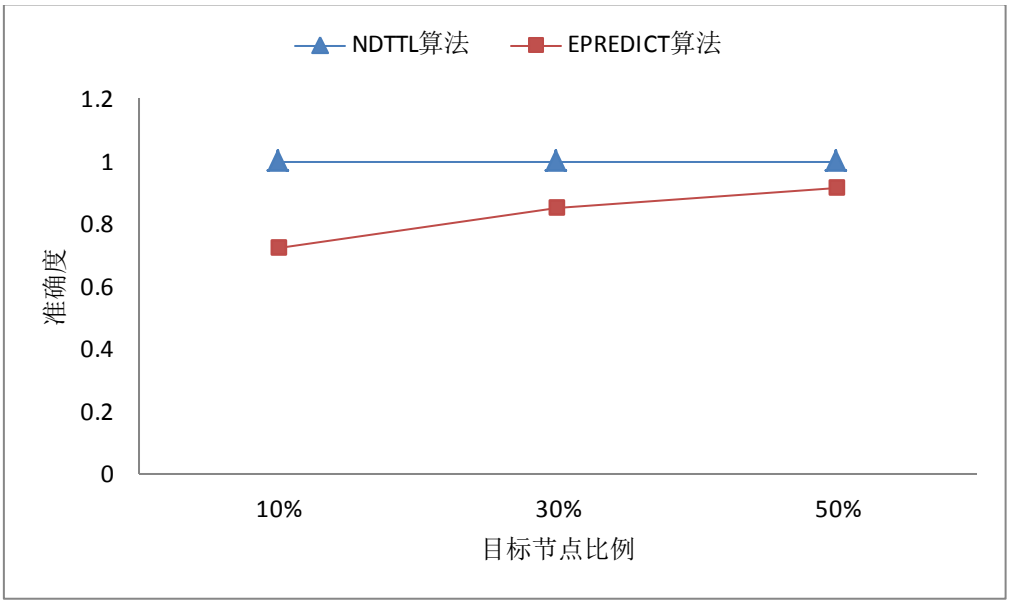


图 3.14 目标节点比例对算法准确度影响

由 3.2.3 节可知，完整度与准确度的乘积表示算法发现的正确的边数与原始网络拓扑结构中的边数的比率。根据上述实验结果可知，EPREDICT 算法在目标节点比例为 10%的情况下，完整度约为 0.76，准确度约为 0.7，完整度与准确度的乘积为 0.532。NDTTL 算法在目标节点比例为 10%的情况下，完整度约为 0.3，准确度为 1，完整度与准确度的乘积为 0.3。因此，在算法发现的正确的边的数量上面，EPREDICT 算法更具有优势。

3.4 本章小结

本章对被动测量下基于生存时间的网络拓扑算法进行了研究。首先，提出了一种不需要进行主动探测的基于生存时间的网络拓扑推断算法，并在真实数据集上验证算法的完整度和准确度。然后，分析了文献[8]提出的 NDTTL 算法，指出该算法存在完整性较低的问题。最后，引入连接数预测机制和 IP 地址前缀匹配长度对算法进行改进，在真实数据集上对改进后的算法进行实验验证。

第四章 拓扑可视化系统的设计与实现

如何利用可视化工具展现算法所推断的网络拓扑结构也是研究热点之一。虽然目前有许多开源的拓扑可视化工具，但它们需要特定格式的网络数据，不具有通用性且不易于扩展。因此，设计并实现一个适合本课题的拓扑可视化系统是非常必要的。

4.1 拓扑可视化系统需求分析

需求分析是设计拓扑可视化系统的重要环节，合理精确地分析用户需求可以有效提高系统开发效率，降低项目成本。在项目开发过程中，主要从功能性和非功能性两个方面分析用户需求。功能性需求是指系统应当具有什么样的功能，这是用户最基本的需求。非功能性需求主要考虑系统的稳定性和可靠性。因此，本论文将从功能性和非功能性两个方面分析拓扑可视化系统的需求。

4.1.1 系统功能性需求分析

功能性需求分析是针对用户的需求，分析拓扑可视化系统应当提供哪些功能。本系统的目的是从网络数据包中提取相关信息推断网络拓扑结构，最后以清晰、直观的形式展现网络拓扑结构。通过分析网络拓扑结构，统计网络的结构特性，如节点类型、节点分布情况等，并以图形的方式展示出来。

通过对拓扑可视化系统的功能分析，本系统需要提供的功能模块包括数据文件解析、数据查询、推断算法、拓扑结构布局和图表显示。因此，对网络拓扑结构进行可视化的流程为：首先通过数据文件解析模块从数据文件中提取网络拓扑推断算法需要的相关信息，接着将信息传输给数据查询模块，根据用户配置信息，数据查询模块选择直接将信息传输给推断算法模块或将信息写入数据库，然后将推断算法模块接收的信息作为算法输入，执行算法并将推断得到的网络拓扑结构传输给拓扑结构布局模块和图表显示模块，拓扑结构布局模块在用户界面以合适的布局算法展示网络拓扑结构，图表显示模块在用户界面将节点分布信息和节点类型信息以图表的形式进行展现。

4.1.2 系统非功能性需求分析

为了让用户在使用拓扑可视化系统时拥有良好的体验，本系统在满足了功能性需求的同时，也需要对非功能性需求进行分析。

1）用户界面的美观性。拓扑可视化系统的用户界面要简洁协调，在展示网络拓扑结构时，节点之间的连接关系要清晰明了，各个功能按钮的操作要简单便捷。

2）系统的可靠性。拓扑可视化系统需要保证当用户出现不致命的错误操作时，系统仍然能够正常提供服务，并且给出错误提示。

3）系统的响应时间。拓扑可视化系统在接收到用户请求时，必须第一时间做出响应，在节点规模为千级的情况下响应时间应少于 8 秒，如果因为数据量过大而导致系统响应时间长，则需要定时向用户反馈请求的处理进度。

4.2 系统总体架构设计

根据拓扑可视化系统的需求分析，本论文在设计拓扑可视化系统时采用浏览器/服务器结构，用户可以通过浏览器随时随地访问拓扑可视化系统的用户界面，并在用户界面上查看网络拓扑结构或进行交互操作，服务器负责响应用户的操作请求。此外，本论文在浏览器/服务器结构上增加一个资源服务器用于存储经过处理的网络数据信息。因此，拓扑可视化系统可以划分为展示层、业务逻辑层和资源层三层结构。拓扑可视化系统总体架构如图 4.1 所示。

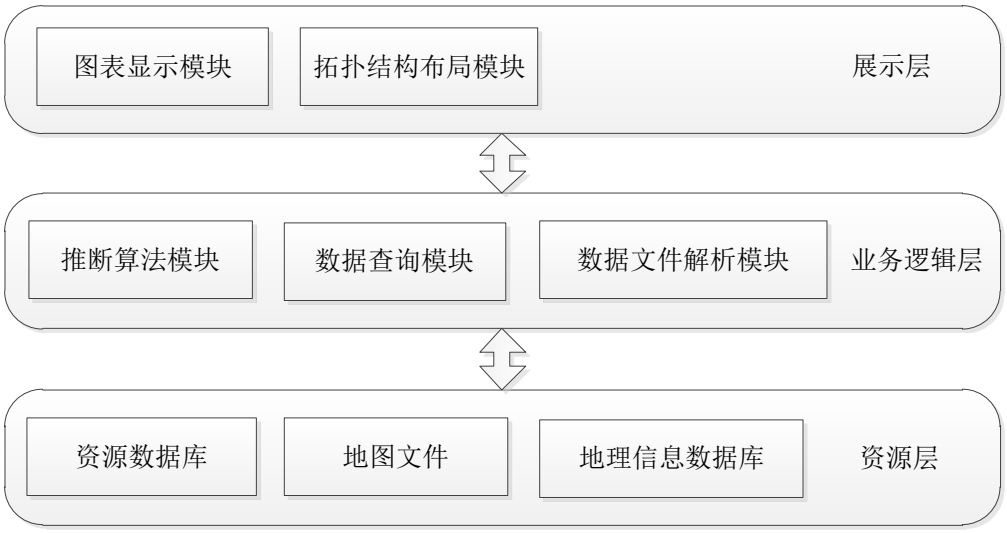


图 4.1 拓扑可视化系统的总体架构

资源层为拓扑可视化系统提供网络拓扑推断算法需要的网络数据。资源层包括资源数据库、地理信息数据库和图片格式的地图文件。其中，资源数据库用于存储经过业务逻辑层处理的网络数据，地理信息数据库用于查询节点的归属地和经纬度信息，图片格式的地图文件则用于拓扑可视化系统用户界面的背景地图展示。

业务逻辑层位于资源层和展示层之间，负责网络数据的解析、存储，上下层数据交换和业务功能的实现。在本论文设计的拓扑可视化系统中，业务逻辑层由推断算法模块、数据查询模块和数据文件解析模块三个功能模块组成。

展示层作为用户与拓扑可视化系统之间的窗口，负责向用户提供简洁协调的用户

界面。用户可以在拓扑可视化系统的用户界面上查看网络拓扑结构或进行交互操作。展示层由图表显示模块和拓扑结构布局模块组成。

4.3 系统数据库设计

数据库设计是系统开发过程中的重要环节。良好的数据库设计能够在一定程度上提高系统的运行效率。本系统采用第三方公开数据库作为地理信息数据库，详见 4.4.2 节。同时，根据拓扑可视化系统的需求分析，在满足数据完整性、数据安全性和数据一致性的前提下，本系统为资源数据库建立了如下的数据库表。

拓扑结构表用于存储网络拓扑结构信息，表中的每一条记录对应一个网络拓扑结构。包括网络拓扑结构 ID、网络拓扑结构中网络节点总数、网络拓扑结构中边的总数、时间戳和备注信息。详细设计如表 4.1 所示。

表 4.1 拓扑结构表

属性	属性说明	数据类型	是否允许为空	键类型
graphID	网络拓扑结构的 ID	int	否	主键
nodeNum	节点总数	int	否	-
edgeNum	边的总数	int	否	-
time	时间戳	timestamp	否	-
mark	备注信息	varchar	是	-

网络节点表存储了网络拓扑结构中的节点信息。包括网络拓扑结构 ID、节点的 IP 地址、节点对应的 IP 地址的所属国家、节点对应的 IP 地址所属城市、节点对应的 IP 地址的经纬度。详细设计如表 4.2 所示。

表 4.2 网络节点表

属性	属性说明	数据类型	是否允许为空	键类型
graphID	网络拓扑结构的 ID	int	否	外键
nodeIP	节点的 IP 地址	char	否	-
nodeCountry	节点所属国家	varchar	是	-
nodeCity	节点所属城市	varchar	是	-
nodeLatitude	节点的经度	double	是	-
nodeLongitude	节点的纬度	double	是	-

连接关系表存储了网络拓扑结构中各节点之间的边，每一条记录表示一个边。包

括网络拓扑结构 ID、边的一端的节点的 IP 地址、边的另一端的节点的 IP 地址和备注信息。详细设计如表 4.3 所示。

表 4.3 连接关系表

属性	属性说明	数据类型	是否允许为空	键类型
graphID	网络拓扑结构的 IP	int	否	外键
leftIP	边的一端 IP 地址	char	否	-
rightIP	边的另一端 IP 地址	char	否	-
mark	备注信息	varchar	是	-

网络数据表存储了原始的网络数据信息，表中的每一条记录对应一个 IP 数据包。包括网络拓扑结构 ID、IP 数据包源 IP 地址、监听到该 IP 数据包的监测节点 IP 地址、路由跳数。详细设计如表 4.4 所示。

表 4.4 网络数据表

属性	属性说明	数据类型	是否允许为空	键类型
graphID	网络拓扑结构的 ID	int	否	外键
srcIP	源 ID 地址	char	否	-
monitorIP	监测节点 ID 地址	char	否	-
hop	路由跳数	int	否	-

根据上述的资源数据库表，可以得到资源数据库表的关系如图 4.2 所示。

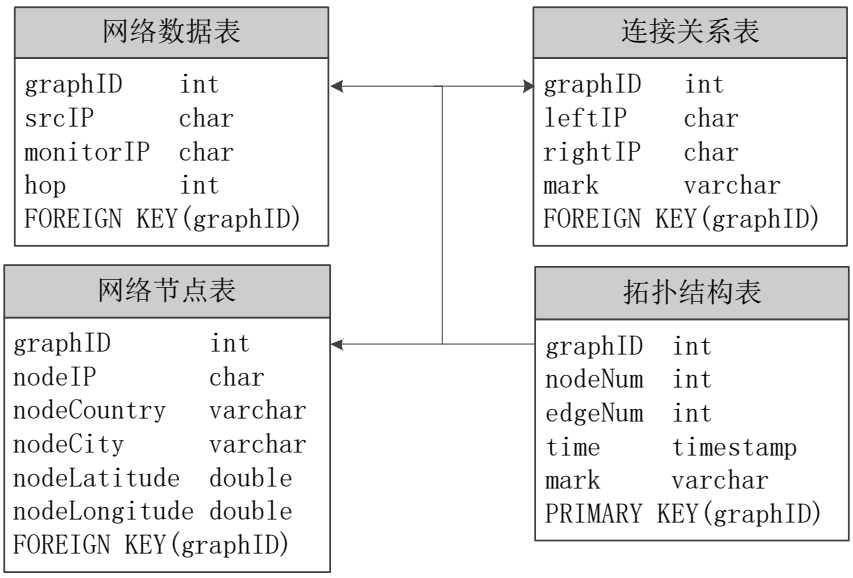


图 4.2 资源数据库表的关系

4.4 系统功能模块详细设计与实现

在完成拓扑可视化系统的总体架构设计后，本节主要阐述拓扑可视化系统中各个功能模块的详细设计与实现，包括数据文件解析模块、数据查询模块、推断算法模块、拓扑结构布局模块和图表显示模块。

4.4.1 数据文件解析模块

数据文件解析模块负责解析经过监测节点标记的 IP 数据包，每个监测节点都会以其自身的 IP 地址标记其监听的每一个 IP 数据包。该模块首先读取 IP 数据包的监测节点标记，然后从 IP 数据包首部提取源 IP 地址字段信息和 TTL 字段信息，最后将这些信息传输给数据查询模块。

在本系统中，采用 Java 语言的 I/O 工具包实现 IP 数据包的读取与解析工作。Java 语言的 I/O 工具包提供了大量的文件输入、输出操作所需要的类和功能函数接口，其中主要的类是 FileInputStream 类。FileInputStream 类通过字节流的形式读取文件内容。数据文件解析模块的流程如图 4.3 所示。

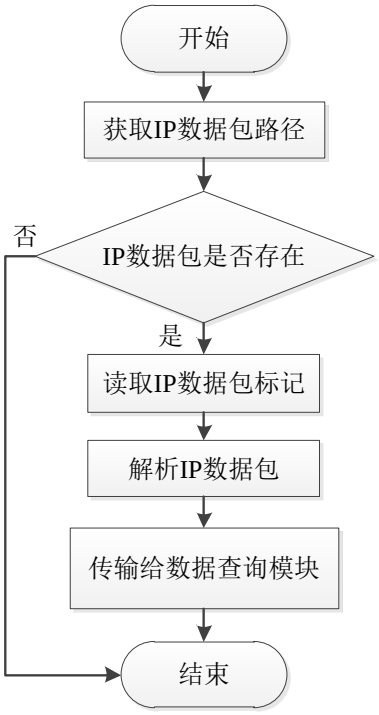


图 4.3 数据文件解析模块流程

数据文件解析模块的详细描述如下：

1）数据文件解析模块接收到用户提交的读取指定路径下的 IP 数据包请求，从该请求参数中获取 IP 数据包的路径。

2）判断指定路径下是否存在 IP 数据包，如果不存在，则结束。

3）如果存在 IP 数据包，则首先读取 IP 数据包上的监测节点 IP 地址，然后调用 FileInputStream 类的 read()方法读取 IP 数据包，并根据 2.1.1 节中 IP 数据包首部格式解析 IP 数据包内容，从中提取源 IP 地址字段信息和 TTL 字段信息。

4）数据文件解析模块将监测节点 IP 地址、源 IP 地址和 TTL 传输给数据查询模块。

4.4.2 数据查询模块

数据查询模块负责访问资源数据库、地理信息数据库和地图文件。通过该模块，用户可以对数据库进行增、删、改、查等操作。数据查询模块可以同时操作资源数据库、地理信息数据库和地图文件。

（1）资源数据库

资源数据库用于存储网络拓扑结构信息。根据 4.3 节的资源数据库设计，本系统采用 MySQL 作为资源数据库。MySQL 是 Web 应用中最流行的关系型数据库，它将数据存储在不同的关系表中，提高了访问速度[40]。同时，Java 语言提供了用于访问关系型数据库的接口 JDBC（Java Data Base Connectivity，Java 数据库连接），它是 Java 应用程序访问关系型数据库的标准方式[41]。因此，本系统通过 JDBC 与 MySQL 资源数据库建立连接，具体的连接方式如下：

```
Class.forName("com.mysql.jdbc.Driver");
String db_url = "jdbc:mysql://localhost/topologyDB";    //topologyDB 为数据库名
String db_user = "topology";                             //账号名
String db_pass = "topology";                             //密码
Connection conn = DriverManager.getConnection(db_url, db_user, db_pass);
```

在与 MySQL 资源数据库建立连接后，可以通过 Statement 类和 Connection 类的方法对数据库进行增、删、改、查操作。

（2）地理信息数据库

地理信息数据库存储着 IP 地址的归属地和归属地的经纬度信息。通过查询地理信息数据库，将地理位置布局算法应用在网络拓扑结构上，可以帮助用户更好的观察和理解网络拓扑结构。

本系统采用 Maxmind 公司的 GeoLite2-City 数据库作为地理信息数据库[42]。GeoLite2-City 是一个 MMDB（Main Memory Data Base，主存数据库）格式的文件，因此，可以采用文件读取的方式访问 GeoLite2-City 数据库。具体的访问方式如下：

```
File database = new File(mmdbPath);      //mmdbPath 为 mmdb 文件的存储路径
DatabaseReader reader = new DatabaseReader.Builder(database).withCache(new
CHMCache()).build();                     //创建地理信息数据库缓存
```

在创建地理信息数据库缓存后，可以通过地理信息数据库提供的 CityResponse 类获取 IP 地址的地理位置信息。

（3）地图文件

地图文件用于显示用户界面的背景地图。本系统使用的地图文件是指 JPG 格式的离线瓦片地图。瓦片地图是将巨型地图切片，分成多个尺寸相同的小型地图。用户在访问瓦片地图时，多个尺寸相同的小型地图会拼接组成一张巨型地图。瓦片地图的优势在于，用户不需要一次性加载整张地图，仅需要加载可见范围内的多个小型地图，从而减少用户加载地图的时间，提高了系统的访问效率[43]。

本系统通过第三方工具，例如太乐地图，下载 JPG 格式的瓦片地图，然后将瓦片地图导入本系统的工程文件中并设置瓦片地图的访问路径，用户界面可以根据瓦片地图的访问路径来加载瓦片地图。

4.4.3 推断算法模块

推断算法模块负责对本论文第三章所述的 PFTTL 算法和 EPREDICT 算法进行实现。通过该模块，用户可以根据网络数据中是否存在网络路径而选择不同的网络拓扑推断算法。

本系统以面向对象为指导思想，对网络拓扑推断算法的功能进行划分，在推断算法模块中建立四个主要的类，包括 TopologyAlog 类、PFTTLAlog 类、EPREDICTAlog 类和 Assistant 类。其中，TopologyAlog 类是一个抽象类，该类中定义了网络拓扑推断算法使用的公共属性，例如 TTL 属性和路由跳数属性等。PFTTLAlog 类是 TopologyAlog 类的一个子类，它对 PFTTL 算法的主要功能进行了封装，包括计算节点集合的方法和计算层级节点集合的方法。EPREDICTAlog 类同样是 TopologyAlog 类的一个子类，它对 EPREDICT 算法的主要功能进行了封装，包括连接数预测机制和节点对编号方法等。Assistant 类是辅助功能类，它提供了网络拓扑推断算法常用的功能，例如二进制的 IP 地址与十进制的 IP 地址相互转换的方法，计算路由跳数的方法和计算算法评价指标的方法等。此外，本系统通过 spark 的 clustering 工具包求解高斯混合模型聚类，降低了代码实现过程中的复杂性。

4.4.4 拓扑结构布局模块

拓扑结构布局模块负责在用户界面展示网络拓扑结构。为了帮助用户更好地观察和理解网络拓扑结构，本系统将节点以地理位置映射的方式布局在背景地图上。因此，拓扑结构布局模块可以从展示背景地图和布局节点两个层面实现具体功能。

（1）背景地图的展示

本系统通过 OpenLayers 加载离线瓦片地图的方式展示用户界面的背景地图。

OpenLayers 是一个著名的 WebGIS 引擎，它结合了 JavaScript、HTML5 和 CSS 等技术，支持 Google Map、OpenStreetMap 和离线/在线瓦片地图等多种网络地图访问方式，同时也提供了地图缩放、地图标记等交互功能[44]。

在利用 OpenLayers 展示用户界面的背景地图时，首先需要在用户界面创建一个 div 容器并通过 CSS 设置容器的样式，包括容器的大小、容器在用户界面中的位置等。其次通过 OpenLayers 的 ol.Map 类创建基础图层对象，该对象用于统筹管理所有的资源对象。然后通过 ol.layer.Tile 类创建瓦片图层对象，在该对象中通过 ol.source.XYZ 类加载 4.4.2 节中获取的离线瓦片地图。接着将瓦片图层对象填充至基础图层对象中，设置基础图层对象的 ol.interaction.defaults()属性以提供地图移动、缩放和标记等交互功能。最后将填充完毕的基础图层对象存入 div 容器。用户通过浏览器访问用户界面时，浏览器会自动完成背景地图的渲染工作。使用 OpenLayers 加载离线瓦片地图的流程如图 4.4 所示。

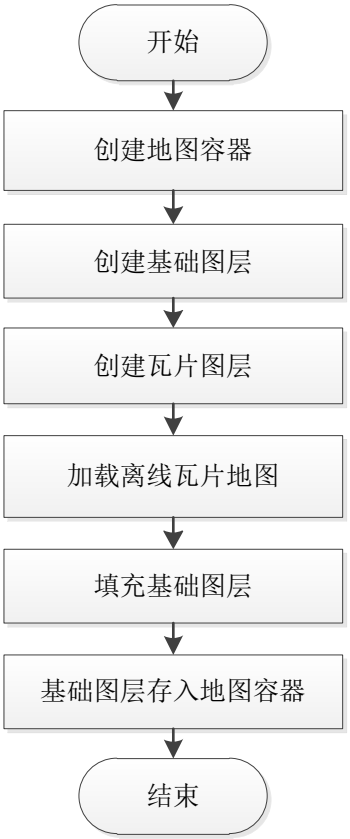


图 4.4 OpenLayers 展示背景地图流程

（2）节点的布局

通过地理信息数据库查询节点的经纬度信息，可以帮助我们使用地理位置布局算法对节点进行布局。然而，拓扑可视化系统用户界面的背景地图是平面形式，节点的经纬度不能直接作为节点在背景地图中的坐标。因此，我们需要将节点的经纬度转换

成平面地图坐标。

本系统采用墨卡托投影将经纬度转换为平面地图坐标。墨卡托投影是一种等角正切圆投影，它将本初子午线作为标准经线，赤道作为标准纬线，两者交点作为平面地图坐标原点，以东为横轴正方向，北为纵轴正方面[45]。墨卡托投影是平面地图中应用最广泛的投影方式，例如，Google Map 和百度地图都采用墨卡托投影将经纬度转换为平面地图坐标。

为了简化计算，将地球视为标准球体，假设节点的经纬度坐标为（J，W），对应的平面地图坐标为（X，Y），则采用墨卡托投影计算平面地图坐标的公式为：

$$X = R * J \tag{4-1}$$

$$Y = R * \ln(\tan(\frac{\pi}{4} + \frac{W}{2})) \tag{4-2}$$

其中，R 表示地球半径，ln 表示自然对数函数，tan 表示正切函数，π 表示圆周率。

4.4.5 图表显示模块

图表显示模块负责将节点分布信息，节点类型信息以饼图、柱状图的形式进行展示。本系统通过 ECharts 插件实现图表显示模块的功能。ECharts 是一个 JavaScript 可视化库，提供了丰富多彩的数据可视化图表[46]。在通过 ECharts 展示统计结果图表时，首先需要在用户界面创建 div 容器存放 ECharts 图表，然后设置图表的参数信息，包括图表类型、图表尺寸、图表数据来源等。ECharts 设置图表参数的格式如图 4.5 所示。

```
option({
        title: {    text:'节点数量 Top10'
            }
        tooltip : {
            xAxis: { type: 'value'},
            yAxis: { type: 'category', data: tempname},
            series: [{ name: 'IP 地址数量', type: 'bar', stack: '总量',
                label: { normal: { show: true, position: 'insideRight' }},
                data: tempvalue
               }]
            }
    });
```

图 4.5 ECharts 设置图表参数的格式

以展示节点分布情况的统计结果为例，首先，用户界面以 ajax 形式向服务端发送查询请求，服务端返回 JSON 格式的节点数据，包括节点总数排名前十的归属地、每个归属地拥有的节点数目等信息。用户界面接收到 JSON 格式的网络数据后，在 ECharts 图表中的 series.type 属性设置图表类型为条形图，series.data 属性设置图表的数据来源，以及在 tooltip 属性中设置提示信息。最后调用 ECharts 的 setOption(option) 方法绘制条形图。ECharts 绘制图形的流程如图 4.6 所示。

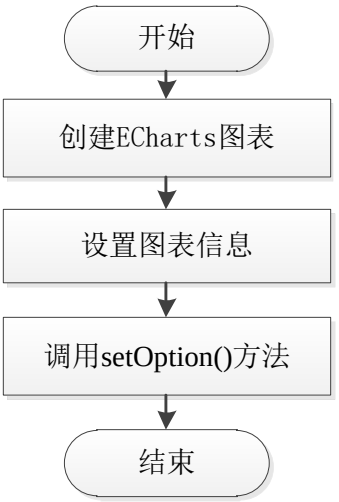


图 4.6 ECharts 绘制图形流程

4.5 系统开发、运行与测试

系统开发过程中可能存在不可知的缺陷，这些缺陷会导致系统无法正常运行。因此，在交付系统之前必须对系统进行全方面的功能测试和性能测试，确保用户的功能需求完全实现，编写的代码符合系统开发的编码规则，同时，观察系统的运行情况，验证系统的稳定性。通过本阶段的实施，可以有效的发现系统中存在的缺陷，从而对系统进行不断的完善。

4.5.1 开发环境

拓扑可视化系统的客户端采用 Windows 7 操作系统，浏览器可以选择谷歌浏览器和 IE 等。服务端采用 Windows 7 操作系统，将 Tomcat 7.0 作为 Web 服务器，MySQL5.6 作为资源数据库。

4.5.2 系统测试

（1）功能测试

功能测试用于检测拓扑可视化系统的功能模块是否能够正确处理业务，得到正确的结果。本系统采用黑盒测试，首先针对拓扑可视化系统的各功能模块设计相应的测

试样例，然后根据测试样例进行测试，对比系统的运行结果与预期结果，判断系统功能是否正常实现，运行结果是否正确，从而发现系统的缺陷。为避免篇幅过大，下面仅介绍拓扑结构布局模块的测试情况，具体的测试内容和测试结果如表 4.5 所示。

表 4.5 拓扑结构布局模块的功能测试表

序号	功能操作	测试内容	结果
1	显示地图	是否能够在用户界面上正常显示背景地图。	正常
2	缩放地图	当用户点击放大或缩小背景地图的控件时，是否能够正确的放大或缩小背景地图。	正常
3	移动地图	当用户点击向左、向右、向上、向下移动背景地图的控件时，是否能够正确的移动背景地图。	正常
4	显示节点	是否能够在背景地图上显示节点，并且节点按照地理位置布局算法布局在背景地图上。	正常
5	获取节点信息	当用户选择某个节点时，是否正常显示节点的信息，包括节点的 IP 地址、经纬度等。	正常

（2）性能测试

性能测试用于检测拓扑可视化系统的可靠性。可靠性是指系统在各种非正常情况下依然能够正确的为用户提供服务。本系统采用 LoadRunner 进行性能测试。LoadRunner 是一个自动化测试工具，它可以提供详细的性能测试结果，并且缩短了性能测试的时间[47]。通过模拟用户使用拓扑可视化系统，测试用户进行功能操作的成功率、最大响应时间、系统的 CPU 占用率和内存占用率。测试结果如表 4.6 所示。

表 4.6 性能测试结果表

测试内容	期望结果	测试结果	是否合格
功能操作成功率	100%	100%	合格
最大响应时间	<=8 秒	6.75 秒	合格
CPU 占用率	<60%	55.269%	合格
内存占用率	<50%	42.837%	合格

测试结果表明，拓扑可视化系统具有良好的性能状态，可以满足用户对系统的性能要求。

（3）测试问题与解决方案

在对拓扑可视化系统测试的过程中，发现了系统存在的若干问题。通过对系统功

能代码的检查和修改，最终解决了这些问题。下面对几个典型的问题进行详细说明。

1）数据格式错误。用户在使用拓扑可视化系统时存在输入操作。在测试过程中，发现用户输入的数据格式存在非法情况，包括输入数据为空，IP 数据包路径格式不正确，IP 地址不合法等。本系统通过对输入数据进行判断，验证输入数据是否为空，输入格式是否正确，利用 Java 语言的正则表达式判断 IP 地址是否合法，最终解决该问题。

2）中文乱码。用户向服务端发出的请求参数可能存在中文字符，由于编码方式的不一致，用户界面的中文字符传递到服务端时会产生中文乱码的问题。本系统通过对用户请求参数的编码方式进行转换，统一采用 UTF-8 编码方式，解决了中文乱码的问题。

4.5.3 系统运行截图

拓扑可视化系统的用户界面是观察网络拓扑结构和与拓扑可视化系统进行交互的主要窗口，简洁明了的界面布局能够明显提高界面的美观性。拓扑可视化系统的用户界面如图 4.7 所示。

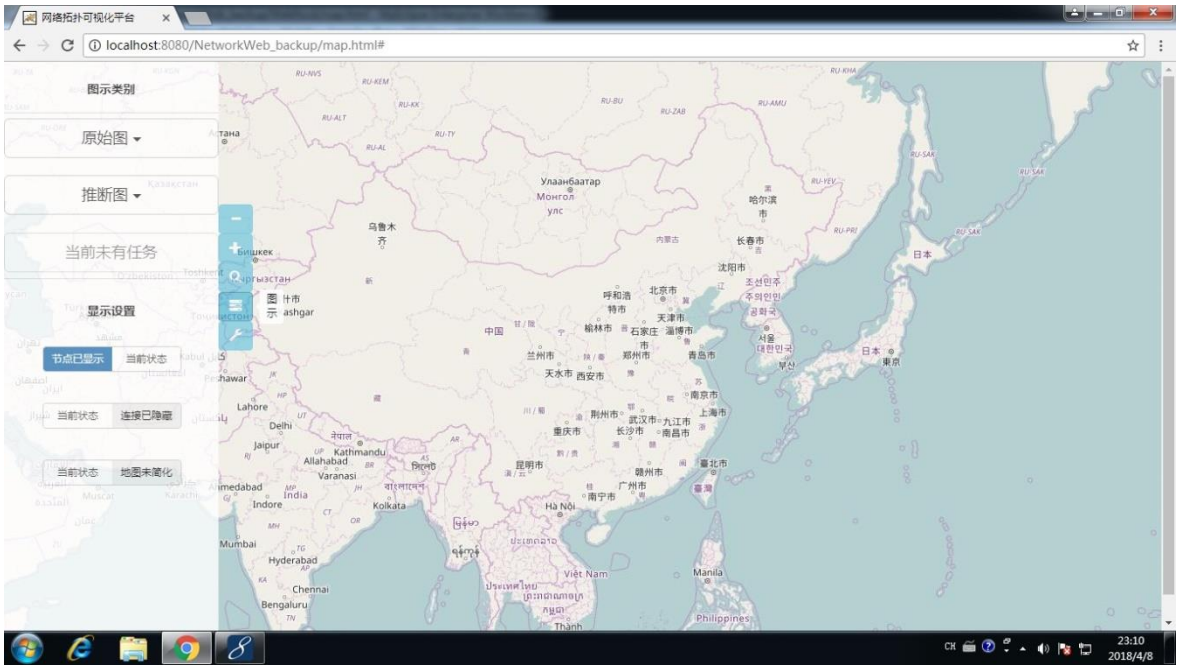


图 4.7 拓扑可视化系统用户界面

通过将节点的经纬度转换为平面地图坐标，将节点以地理位置映射的方式布局在用户界面的背景地图上，可以很好的帮助用户观察网络拓扑结构中各节点的分布情况。节点的地理位置布局图如图 4.8 所示。

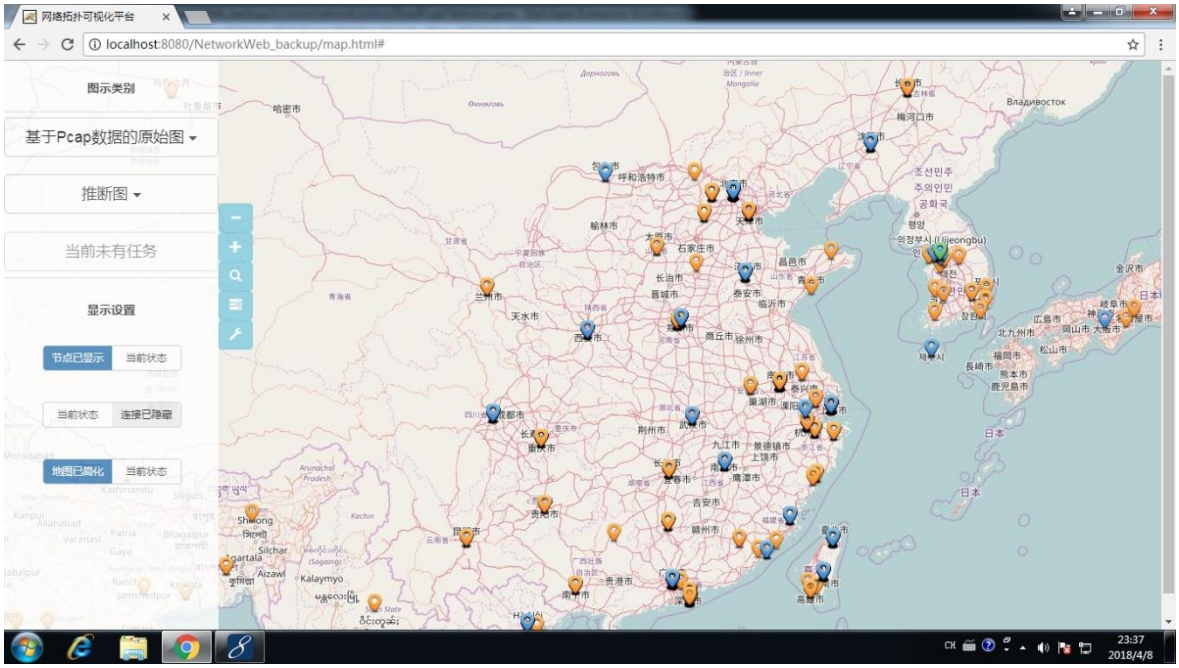


图 4.8 节点地理位置布局图

节点之间相互连接组成网络拓扑结构，在对节点进行布局之后，根据节点之间的连接关系绘制各节点之间的边。节点连接图如图 4.9 所示。

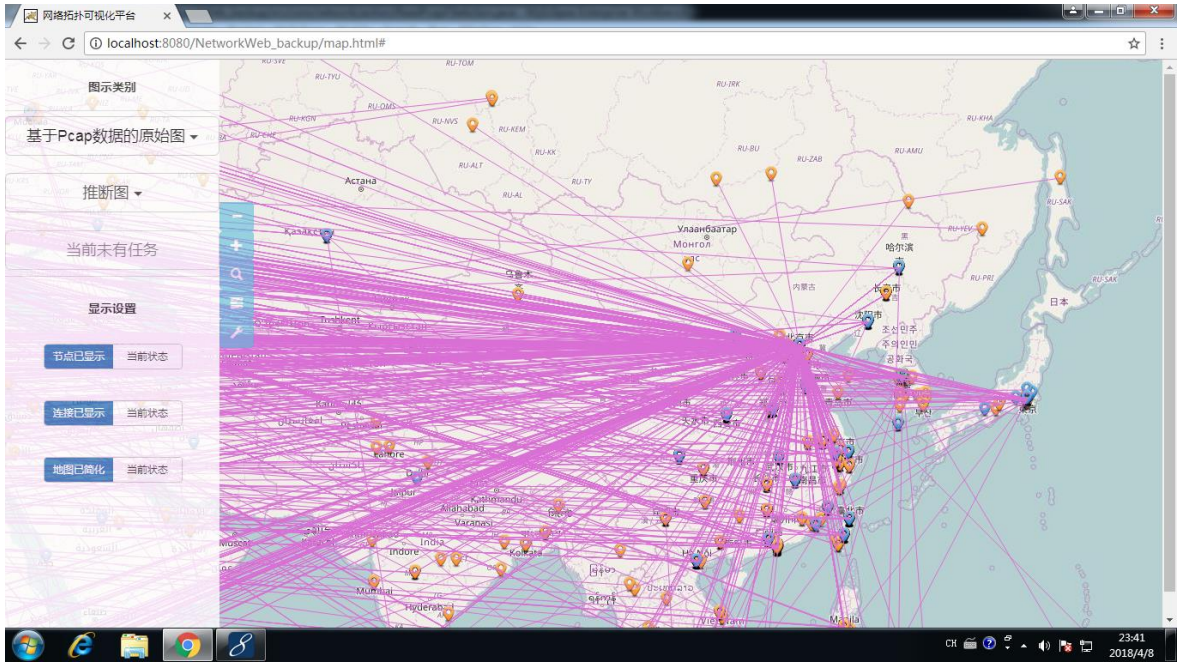


图 4.9 节点连接图

当点击用户界面上的某个节点时，能够显示节点的具体信息，包括节点 IP 地址、节点的地理位置、节点的经纬度和节点类型。同时，与该节点连接的其他节点会以关联节点的形式展现在用户界面的左下方。节点信息图如图 4.10 所示。

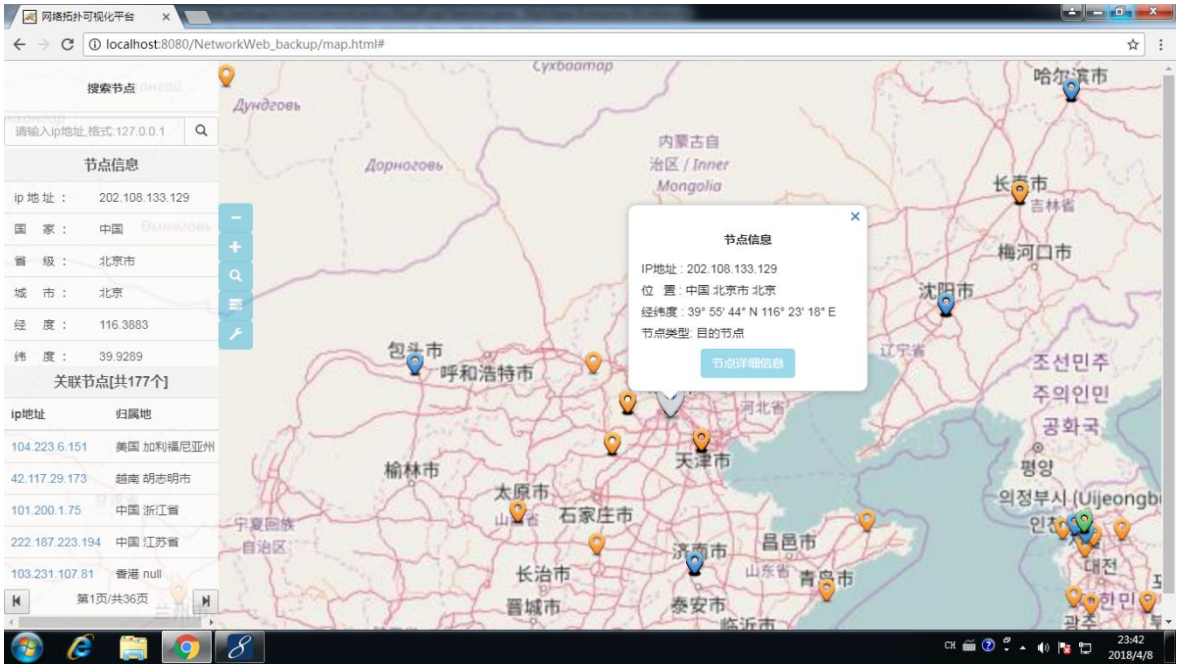


图 4.10 节点信息图

用户可以通过节点归属统计图了解网络拓扑结构中各节点的归属地统计信息。节点归属地统计图如图 4.11 所示。

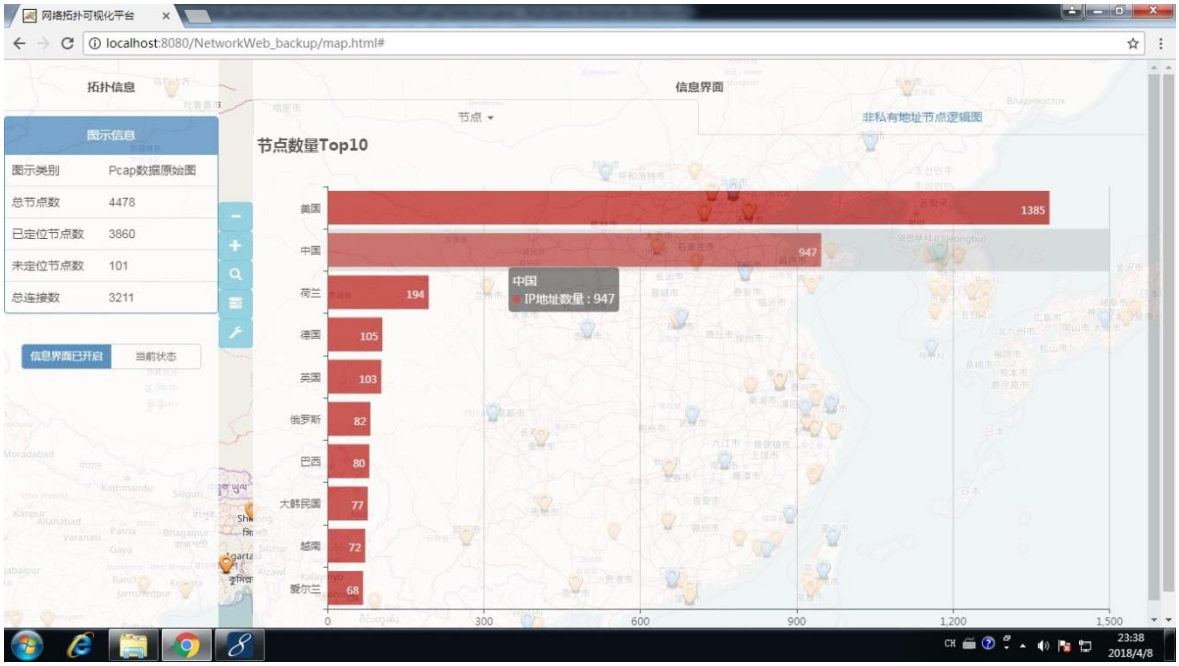


图 4.11 节点归属地统计图

用户可以通过查看节点类型图了解网络拓扑结构中哪些节点主动向网络中发送了数据包。节点类型图如图 4.12 所示。

图 4.12 节点类型图

4.6 本章小结

本章设计并实现了一个拓扑可视化系统。首先，从功能和性能两个方面对拓扑可视化系统的需求进行分析。其次，介绍了拓扑可视化系统的总体架构设计，包括资源层、业务逻辑层和展示层，给出系统的数据库设计，对数据库表进行说明。然后，详细阐述了系统各核心模块的设计与实现，包括数据文件解析模块、数据查询模块、推断算法模块、拓扑结构布局模块和图表显示模块。最后，在真实环境中对拓扑可视化系统进行功能测试和性能测试，测试结果表明，该系统可以较好的满足用户的需求。

第五章 总结与展望

5.1 本文总结

网络规模的迅速增长使得网络结构日益复杂。掌握网络拓扑结构对管理网络、分析网络性能和定位网络故障至关重要。如何在无法进行主动探测的情况下推断网络拓扑结构，如何使得推断的网络拓扑结构更加完整、准确，如何清晰直观的展示网络拓扑结构都是具有现实意义的问题。

已有的被动测量下基于生存时间的网络拓扑推断算法主要存在两个问题，一是推断的网络拓扑结构存在完整度较低的问题，二是如果目标网络禁止主动探测，则该算法无法推断网络拓扑结构。同时，越发复杂的网络拓扑结构难以让人们观察和分析，已有的拓扑可视化工具存在功能单一，只能使用特定格式数据的问题。本文旨在推断网络拓扑结构的过程中，降低对特定网络协议的依赖，减少探测数据包的数量或在不进行主动探测的情况下推断网络拓扑结构，并以清晰直观的方式展示网络拓扑结构。因此，本文对基于生存时间的网络拓扑推断算法和可视化进行了研究，在此基础上主要完成了以下工作：

1）介绍了网络拓扑推断和可视化的研究背景与意义，对它们的研究现状进行说明，总结了现实网络中存在的网络拓扑结构类型，介绍了几种常见的网络拓扑推断算法，分析了常用的可视化布局算法，并比较了每个布局算法的优缺点。

2）提出了一种被动测量下基于生存时间的网络拓扑推断算法 PFTTL，详细描述了算法的流程，并在真实数据集上对算法进行实验验证，实验结果表明，PFTTL 算法可以在不进行主动探测的情况下推断网络拓扑结构并且具有较高的完整度。

3）分析了已有的基于生存时间的网络拓扑推断算法 NDTTL，指出该算法在推断网络拓扑结构时存在完整度较低的问题，针对该问题，引入连接数预测机制，并结合 PFTTL 算法的 IP 地址前缀匹配长度，提出了改进的算法 EPREDICT，并在真实数据集上对 NDTTL 算法和 EPREDICT 算法进行实验验证，实验结果表明，EPREDICT 算法在减少了探测数据包的同时，具有更高的完整度。

4）设计实现了一个展示网络拓扑结构信息的可视化系统。分析了系统需求，介绍了系统的总体架构设计和数据库设计，详细阐述了系统各功能模块的实现方式，在真实环境中对系统进行了测试，测试结果表明，该系统运行良好，可以满足用户需求。

5.2 展望

虽然目前的研究成果可以满足用户的需求，但仍然有许多内容需要完善。未来可

以从以下 3 个方面进行更深入的研究：

1）本文所提出的 PFTTL 算法在推断网络拓扑结构时只考虑了将 IP 数据包中的 TTL 信息作为推断依据，这种方法虽然可以在不向网络中注入探测数据包的情况下推断网络拓扑结构，但算法的准确度有待进一步提高，下一步可以研究如何结合更多的网络信息提高算法的准确度。

2）本文所提出的 EPREDICT 算法在推断网络拓扑结构时，虽然减少了探测数据包的数目并提高了算法的完整度，但该算法在获取网络路径时要求监测节点之间相互独立，下一步可以研究如何寻找合适的监测节点使得算法的适用范围更广，算法的完整度和准确度更高。

3）拓扑可视化系统在总体架构设计上只考虑了读取本地数据文件和从数据库中获取网络信息，不具有动态展示网络拓扑结构的功能。因此，未来可以在拓扑可视化系统与监测节点之间增加通信模块，系统实时接收监测节点发送的数据并且动态地将网络拓扑结构展示在用户界面上。

参考文献

[1] Ambure R, Karve S. Generating Router Level Topology Using Dns And Ip Identifier[J]. International Journal of Engineering Research & Applications, 2015, 5(4).

[2] Yin J, Cai Z, Zhao W, et al. Passive calibration of active measuring latency[C]// NETWORKING - Icn 2005, International Conference on Networking, Reunionisland, France, April 17-21, 2005, Proceedings. DBLP, 2005:746-753.

[3] 唐东明. 网络编码关键问题研究[D]. 电子科技大学, 2013.

[4] 潘胜利. 多径路由网络层析成像[D]. 电子科技大学, 2016.

[5] Pandey S, Choi M J, Lee S J, et al. IP network topology discovery using SNMP[C]// International Conference on Information NETWORKING. IEEE Press, 2009:33-37.

[6] 潘楠, 王勇, 陶晓玲. 基于 OSPF 协议的网络拓扑发现算法[J]. 计算机工程与设计, 2011, 32(5):1550-1553.

[7] Jin C, Wang H, Kang G S. Hop-count filtering: an effective defense against spoofed DDoS traffic[C]// ACM Conference on Computer and Communications Security. ACM, 2003:30-41.

[8] Eriksson B, Barford P, Nowak R. Network discovery from passive measurements[C]// ACM SIGCOMM 2008 Conference on Data Communication. ACM, 2008:291-302.

[9] 蒋锦林. 基于 hop-count 的局域网网络层被动拓扑发现算法研究[D]. 电子科技大学, 2012.

[10] 张伟明, 罗军勇, 王清贤. 网络拓扑可视化研究综述[J]. 计算机应用研究, 2008, 25(6):1606-1610.

[11] 刘杰. 网络拓扑可视化研究[D]. 西安电子科技大学, 2013.

[12] Cheswick B, Burch H, Branigan S. Mapping and Visualizing the Internet[C]// Proc. of Usenix Technical Conference. 2000:1--12.

[13] 蒲涛. 网络拓扑可视化技术的研究[D]. 西安电子科技大学, 2010.

[14] Au S C, Leckie C, Parhar A, et al. Efficient visualization of large routing topologies[J]. International Journal of Network Management, 2004, 14(2):

[15] 程远, 严伟, 李晓明. 基于斥力-张力模型的网络拓扑图布局算法[J]. 计算机工程, 2004, 30(3):104-105.

[16] JamesF.Kurose, KeithW.Ross. 计算机网络:自顶向下方法与 Internet 特色:A Top-Down Approach Featuring the Internet[M]. 机械工业出版社, 2005.

[17] 樊爱京, 张志立. IP 地址在子网中的划分[J]. 许昌学院学报, 2003, 22(2):94-95.

[18] 吴卫东. 高速路由器的数据转发技术研究[D]. 华中科技大学, 2005.

[19] 孙岩, 张楠. 网络拓扑结构研究与分析[J]. 计算机光盘软件与应用, 2013(17):55-56.

[20] 赵婧如. 网络拓扑发现技术研究[D]. 西安电子科技大学, 2009.

[21] W.RichardStevens. TCP/IP 详解. 卷 1, 协议[M]. 机械工业出版社, 2006.

[22] 张卓, 宣蕾, 郝树勇. 可视化技术研究与比较[J]. 现代电子技术, 2010, 33(17):133-138.

[23] 林茂松. 科学计算可视化的应用研究[D]. 西南交通大学, 2006.

[24] Robertson P K, Earnshaw R A, Thalmann D, et al. Research issues in the foundations of visualization[J]. IEEE Computer Graphics & Applications, 1994, 14(2):73-76.

[25] Cleveland, William S. Visualizing Data[J]. Technometrics, 1993, 36(3).

[26] 任东怀. 高维数据可视化研究[D]. 北京交通大学, 2007.

[27] Inselberg A, Dimsdale B. Parallel coordinates: a tool for visualizing multi-dimensional geometry[C]// Visualization, 1990. Visualization '90. Proceedings of the First IEEE Conference on. DBLP, 1990:361-378.

[28] Keim D A. Designing Pixel-Oriented Visualization Techniques: Theory and Applications[J]. IEEE Transactions on Visualization & Computer Graphics, 2000, 6(1):59-78.

[29] Shneiderman B. Tree visualization with tree-maps: 2-d space-filling approach[J]. Acm Trans on Graphics, 1992, 11(1):92-99.

[30] Card S K, Mackinlay J D, Shneiderman B. Readings in information visualization - using vision to think.[C]// Series in Interactive Technologies. Morgan Kaufmann Publishers Inc. 1999.

[31] 陈谊, 胡海云, 李志龙. 树图布局算法的比较与优化研究[J]. 计算机辅助设计与图形学学报, 2013, 25(11):1623-1634.

[32] Wills G J. NicheWorks — Interactive visualization of very large graphs[C]// Graph Drawing '97 Conference. 1997:403-414.

[33] Canbaz M A, Thom J, Gunes M H. Comparative Analysis of Internet Topology Data sets[C]// IEEE Global Internet Symposium. IEEE, 2017.

[34] Barford P, Bestavros A, Byers J, et al. On the marginal utility of network topology measurements[C]// 2001:5-17.

[35] Eriksson B, Barford P, Nowak R, et al. Learning network structure from passive measurements[C]// ACM SIGCOMM Conference on Internet Measurement 2007, San Diego, California, Usa, October. DBLP, 2007:209-214.

[36] Baralis E, Bianco A, Cerquitelli T, et al. NetCluster: A clustering-based framework to analyze internet passive measurements data[J]. Computer Networks, 2013, 57(17):3300-3315.

[37] Hastie T, Tibshirani R, Friedman J. The Elements of Statistical Learning[J]. Journal of the Royal Statistical Society, 2001, 167(1):267-268.

[38] Alderson D, Li L, Willinger W, et al. Understanding internet topology: principles, models, and validation[C]// IEEE/ACM TRANSACTIONS ON NETWORKING. 2005:1205--1218.

[39] Roughan M, Tuke S J, Maennel O. Bigfoot, sasquatch, the yeti and other missing links:what we don't know about the as graph[C]// ACM SIGCOMM Conference on Internet Measurement 2008, Vouliagmeni, Greece, October. DBLP, 2008:325-330.

[40] Widenius M. Mysql Reference Manual[M]. O'Reilly & Associates, Inc. 2002.

[41] Cattell R, Fisher M. Jdbc database access with java[J]. Isbn, 1997.

[42] Bischof Z S, Otto J S. Distributed systems and natural disasters:BitTorrent as a global witness[C]// Proceedings of the Special Workshop on Internet and Disasters. ACM, 2011:1-8.

[43] 苏旭明，谭建成. WebGIS 中瓦片地图关键技术研究[J]. 北京测绘, 2012(2):9-12.

[44] Santiago P A. OpenLayers Cookbook[M]. Packt Publishing, 2012.

[45] 少倩. 墨卡托投影[J]. 地图, 1986(2):36.

[46] 王子毅，张春海. 基于 ECharts 的数据可视化分析组件设计实现[J]. 微型机与应用，2016, 35(14):46-48.

[47] Yang P, Jie L I. Using LoadRunner to Test Web's Load Automatically[J]. Computer Technology & Development, 2007.

致谢

三年的硕士生活转眼间步入尾声，在这段美好的时光里，自己收获颇多。在书写这段感谢的时候，脑海中不时的闪现出一个个身影，那些在讲台上讲授知识的老师，那些共同努力，一起奋斗的同学，此时此刻，我衷心的向你们说一句，谢谢！

首先，我要特别感谢姜奇老师对本论文的悉心指导与倾力相助。从研究课题的选择、项目实施直至论文完成，姜奇老师始终给予我耐心的指导和支持，他严谨的治学态度、精益求精的工作作风深深感染着我，在此谨向姜奇老师致以衷心的感谢！

其次，我要感谢实验室的同学们，特别是李兵妍、钱媛媛、钱少锋、陈智仁、张欣、张玲、黄晓涵和吕龙伟，是你们在我失落的时候带给我欢笑，是你们陪伴我不断地在前进的道路上披荆斩棘，这些单纯、奋斗、美好的时光，我会永远的记在心里！

最后，我要向百忙之中参与评审本论文和参与本论文答辩的各位老师表示由衷的感谢！

西安電子科技大學
XIDIAN UNIVERSITY

地址：西安市太白南路2号
邮编：710071
网址：www.xidian.edu.cn