密级：　　保密期限：

北京郵電大學

硕士学位论文

题目：基于 OpenFlow 的发布/订阅系统中拓扑和路由子系统的研究与实现

学　　号：2012111559

姓　　名：王双锦

专　　业：计算机科学与技术

导　　师：章洋

学　　院：网络技术研究院

2014 年 12 月 26 日

独创性（或创新性）声明

本人声明所呈交的论文是本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京邮电大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

本人签名：_____ 日期：_____

关于论文使用授权的说明

学位论文作者完全了解北京邮电大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属北京邮电大学。学校有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许学位论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存、汇编学位论文。（保密的学位论文在解密后遵守此规定）

保密论文注释：本学位论文属于保密在　年解密后适用本授权书。非保密论文注释：本学位论文不属于保密范围，适用本授权书。

本人签名：_____ 日期：_____

导师签名：_____ 日期：_____

基于 OpenFlow 的发布/订阅系统中拓扑和路由子系统的研究与实现

摘　要

发布/订阅是一种分布式系统的事件驱动过程范式，对消息的生产者与消费者进行空间、时间和控制的解耦，它的异步和多点通信的特点增强了分布式系统的灵活性和可扩展性。

在现有应用于物联网的发布/订阅系统中，对拓扑的有效维护和大规模系统的扩展有待优化，拓扑的收敛性保证、路由算法的改进、信息膨胀带来的负担也都是目前亟需解决的问题；另外由传统网络向 SDN 迁移也须深入研究。

基于现状，本文首先在传统网络中设计和实现发布/订阅系统的拓扑和路由方案，通过应用 OSPF 链路协议提供收敛性好、易于扩展的拓扑维护架构，通过应用 Dijkstra 算法和网络流量的结合设计策略驱动的主题聚合路由方法。此外设计策略驱动的主题树使路由树得到精简，临近地域的集群设定使转发路径得到优化，从而提高系统的整体效率。

改进后的发布/订阅系统在性能上有所提升，但由于其建立在传统 IP 网络之上，底层转发过程对应用层透明，很难有效控制消息传播过程，因此本文又提出了发布/订阅系统在 SDN 上的迁移。SDN 技术使得路由器/交换机可以受控于控制器，以此控制消息流的转发，提高消息传播效率。此外控制器的拓扑维护和对 OpenFlow 交换机流量的采集应用，也是本文研究的重点之一。

本文将从传统网络和到 SDN 的迁移两个方面对比验证拓扑和路由方面设计的有效性，并分别进行实验。实验结果表明，这些设计的引入，提高了系统的效率和鲁棒性。

关键词 发布/订阅系统 SDN floodlight 拓扑 路由

RESEARCH AND IMPLEMENTATION OF TOPOLOGY CONSTRUCTION AND MESSAGE ROUTING OF OPENFLOW-BASED PUBLISH/SUBSCRIBE SYSTEM

ABSTRACT

Distributed Event-Based Systems (DEBS) adopt Publish/Subscribe paradigm which decouples producers and consumers on space， time and control dimensions. Their decoupling feature improves the flexibility and scalability of distributed systems.

When they are used in IoT (Internet of Things) scenarios, there are some problems to be considered such as wide-area topology maintenance and real-time event delivery to lots of event subscribers. For wide-area publish/subscribe systems, there are heterogeneous networks, computation nodes and applications, direct topology construction may result in data congestion in some local area or redundant data forwarding. Furthermore, the publish/subscribe systems over traditional networks can not completely satisfy the real-time requirement of event delivery for IoT applications. So migration from traditional Network to SDN is needed.

We propose a data-localized topology maintenance scheme and a policy-driven name tree aggregation routing scheme to address the above issues. In IoT applications, the location of sensors and devices can often be predicted such that we can optimize the topology of Publish/Subscribe system. Besides, we also offer the design of policy-driven name tree to reduce the scale of routing tree, clusters generated by adjacent regions to enhance forwarding paths, and thus raise the efficiency of the whole system.

The performance of Pub/Sub system is improved by measures above. But it does not completely satisfy the IoT applications' requirements because it cannot control the process of underlying message forwarding in traditional networks. So we adopt Software Defined Network (SDN) as

the underlying physical networks to support our solution. The switches/routers can be controlled by controllers over SDN which can work according to the applications' requirements such that the real-time dispatching algorithm can be adopted.

We do experiments to check the correctness and efficiency of our solution. The results show that our solution can promote the efficiency of Pub/Sub systems to support IoT applications.

KEY WORDS publish/subscribe system, SDN, floodlight, topology, message routing

目录

第一章 绪论

1.1. 研究背景

互联网倡导建立在“自律”基础上的“开放”、“平等”和“创新”，允许每个人参与建设和发展的精神理念，因此“端到端透明性”油然而生。端到端原则的设计思想是：能在端系统实现的就不应该在网络核心中实现，网络核心提供通用服务，应避免具体功能的出现。发布/订阅（Publish/Subscribe）系统即是一种端到端的体系结构。

发布/订阅是一种消息范式，实现了消息发布者和订阅者的松耦合，与传统的客户端/服务器相比，具有更好的可缩放性。发布/订阅系统中的信息交换是由发布者和订阅者的分布式布局传播的，订阅者通过发布兴趣包接收相应消息，消息的转发由发布者和订阅者之间关系的匹配来触发。

在物联网的应用中，我们已有一个基于 servicemix 的发布/订阅系统实现版本，但其事件传播速度较慢（低于 50 条/秒），针对于大规模广域网尚未有较为成熟的设计，在订阅同步、实时性提高、效率提高、底层转发路径方面均有待提升；同时，一些相似订阅的冗余、所有节点没有限制的接收也给发布/订阅系统的管理和消息转发带来很大的不便。因此对网络拓扑的有效性和可扩展性管理、高效的路由和路由查找算法、对相同和相似主题合并的主题树和针对某些主题对一些节点做限制接收的策略消息的引入在此显得尤为重要。

传统的发布/订阅系统工作于应用层，信息的层层传递和对实际网络的未知使实时性和可靠性无法得到保障。而 SDN（Software Defined Network，软件定义网络）则可以将其扩展至网络层和数据链路层，为消息的传播提供便利。SDN 用“流交换”代替“包交换”，用“集中管理”取代单独配置，在网络流量灵活性方面有良好的作用，可以实时调节发布/订阅系统流量的走向[1]。SDN 的三大关键要素为：转发与控制分离；OpenFlow 协议；具有一致性、全系统范围的网络操作系统可编程接口。基于这三大要素，SDN 就可以解决传统网络中一些无法避免的问题，如缺乏灵活性、对需求变化响应速度较慢、无法实现网络虚拟化等。[2]

对 SDN 网络的控制需要一定的协议与交换机/路由器进行交互，而 OpenFlow 便是这一协议。OpenFlow 交换机的核心思想是用 OpenFlow 交换机和控制器协同完成报文转发来取代传统的由交换机/路由器控制，从而将数据转发和路由控

制分离开来。控制器可以控制 OpenFlow 交换机的流表，从而达到控制数据转发的目的。OpenFlow 目前已经有广泛的应用，如校园网中的应用、在广域网和移动网络中的应用、在网络管理和安全控制中的应用和在数据中心网络中的应用。OpenFlow 也有相应的软件由工程师们开发和维护，如 Open vSwitch、Beacon、floodlight，其开发环境、语言和功能的多样化给相关研究人员提供了便利。

1.2. 研究内容

本文的研究目的在于对基于普通网络的发布/订阅系统的拓扑和路由进行设计，并进一步将其移植到 SDN 网络上，主要实现目标为：设立中心管理器，完成拓扑加入和丢失的判断，路由计算；完成各用户 hello 消息的发送、传输和邻居维护；完成对 OpenFlow 交换机/路由器相关数据的收集和应用。

在现有 IP 网络中，路由决策在各个路由器上分布式计算，对应用层透明；但在 OpenFlow 网络中，OpenFlow 交换机/路由器受控于控制器，对应的路由决策也须有控制器决定。这一特点导致在 OpenFlow 网络上实现网络层路由服务方式与现有的 IP 网络大相径庭。我们在深入研究现有 IP 网络中网络路由功能和实现方式的基础上，需要结合 OpenFlow 网络的特点，找出符合 OpenFlow 网络的路由解决方案。

为使具体路由方案能够适应各节点负载和网络状况，控制器（代表）将考虑各节点的流量状况综合计算转发路径，以使 OpenFlow 交换机/路由器的转发效率达到最大值。为维护整个网络中的拓扑结构，各物理相邻或逻辑相邻的节点在构建邻居之后须定时发送 Hello 消息来维护。当接收消息超时时则向控制器报告，并由控制器对此消息进行判定和处理。

在本系统的拓扑和路由工作中主要的研究内容包括以下四部分。

1、 研究现有 IP 网络技术和 quagga 拓扑维护机制

为了在 OpenFlow 网络上提供网络层路由服务，首先需要对现有 IP 网络的路由技术进行研究，理解其实现原理和机制，从中找出其与 OpenFlow 交换技术的异同点，考虑 IP 路由选择与底层数据转发的联系和分离方案。

Quagga 的主要功能是提供基于 TCP/IP 的路由服务，它是一款开源软件。它在拓扑维护方面机制较为成熟，对局域网内和网间的拓扑均能很好的维护。在本系统开发时需要相对准确和一致的拓扑以便发送消息，因此以其作为参考模板。

2、 OpenFlow 技术分析与应用

OpenFlow 技术作为整个 OpenFlow 网络的核心，主要的标准是 OpenFlow 协议。因此系统的开发需要仔细研究此协议并加以应用。

使用新技术代价往往十分高昂，但 OpenFlow 具有足够的开放性，因此给开发带来了可能性。本系统在传统的以太网之上，通过部分引入的方式引入了基于 OpenFlow 协议的控制层，显著降低了构建成本。

3、 OpenFlow 网络中控制器的研究和扩展

控制器在整个系统中作为中心计算和控制单元，在控制 OpenFlow 交换机的同时与代表功能集成，负责整个系统的路由计算和拓扑维护。控制器保存所有代表信息，当有节点加入时控制器端则增加相应的记录，当有节点报告丢失时，控制器将判断丢失集群邻居状况和通过向其发送试探消息的方式判定其是否丢失；当整个网络中拓扑发生变化时，控制器将重新计算路由，综合考虑网段、网络状况等因素，计算出最佳路径。当控制器直接相邻的拓扑有变化时，则通过洪泛的方式告知全网。

4、 拓扑维护和路由计算新方案

拓扑维护由各集群的代表和代理进行，同一集群的代表和代理之间、相邻集群的代表之间均会通过发送 Hello 消息的方式告知对方自身的存在，当 Hello 消息超时时则通过 TCP 试探来判断对方是否确实不可达，并对此作出相应的动作。邻居的选择将综合考虑节点之间的距离、物理连接情况和邻居个数，当新加入或邻居个数不足预设时进行选择。

路由计算基于拓扑之上，考虑各节点的实际负载进行动态调整。整体考虑使用 Dijkstra 算法，在其中将 OpenFlow 交换机/路由器的各方向的流量模型作为参数附加到该方向上的距离，以此综合计算出最佳路径。

1.3. 论文组织

本文研究基于主题的发布/订阅系统在 SDN 网络中拓扑和路由的设计与实现，重点放在它的拓扑组织结构和路由算法上面。鉴于此，本文的组织结构如下：

第一章为绪论，概要介绍了本文研究的背景和主要内容；

第二章为发布/订阅系统及 OpenFlow 关键技术的概述，主要介绍发布/订阅系统的拓扑结构、路由算法，OSPF 的链路维护方法和路由算法，OpenFlow 的相关技术；

第三章为系统的需求分析，包括主题树、策略、订阅管理、拓扑维护、路由计算、SDN 网络拓展、高效性和可靠性需求，为系统的设计与实现提供了目标；

第四章为系统的概要设计，简要描述了与需求相对应的主题树、策略处理、订阅管理、拓扑维护、路由方法和 SDN 移植的设计蓝图；

第五章叙述发布/订阅系统的拓扑和路由在传统网络中的实现，包括各个功能模块中的类图设计、消息流程等；

第六章为发布/订阅系统在 SDN 上迁移的实现过程，详细描述了物理设施的移植、对 OpenFlow 交换机的控制和对拓扑流量信息的收集和应用；

第七章讨论系统的测试和验证，主要包括实际应用场景的描述、网络的搭建，以及测试用例和结果分析。

第二章 发布/订阅系统及 OpenFlow 关键技术

发布/订阅系统又称基于事件（Event-based）的消息中间件，其中发布者发布事件（Event），订阅者订阅事件消息，而发布/订阅系统则负责将发布者发布的消息按照一定的路径传送给所有相关订阅者。为了适应复杂的应用和实际网络的不断变化，大规模的发布/订阅系统通常采用分布式的系统结构，其中事件代理（event broker）为一定数量的本地客户端服务。这些代理按照一定的拓扑结构连接，实现消息在各节点之间的转发。[3]

2.1. 发布/订阅系统的关键技术

发布/订阅系统有拓扑结构、主题匹配、路由策略、消息转发的可靠性保证等关键技术，以下介绍与本文研究相关的拓扑结构和路由计算两个方面。

2.1.1. 拓扑结构

发布/订阅的拓扑结构根据其设计思想的不同可以分为集中型、层次型、无环图型、环型、混合拓扑等方式。其中集中式需要中心的存储转发服务节点，层次型由上到下呈树状结构，混合拓扑为多种结构的混合。具体拓扑结构应结合各结构的特点和实际应用场景进行相应的选择。[4]

集中型和层次型拓扑介绍

集中型拓扑结构如图 2-1 所示，所有的事件发布者和接收者都需要依赖中间服务器进行转发，电子商务、银行等通常对可靠性、数据一致性或者事物支持有很高要求的系统适合这种范型，而本系统设计的发布/订阅系统为分布式架构，不要求高可靠性，对数据的转发效率有较高要求，因此并不适用；层次型拓扑是对集中型拓扑的一种扩展，通知服务器之间存在层次关系是它的特色。[5]如图 2-2 所示，每个服务器服务可以连接一组客户端，客户端可以同时充当发布者和订阅者。代理网络分为两层，最顶层为核心层，核心层的每个节点都是超级节点，本组所有订阅和发布都需要经过他的超级节点；在核心层与子网层之间设置网关，可以增强安全性和便于管理网络[6]。支持这种拓扑结构的系统有 JEDI[7]和 SIENA[8]。该结构中，路由较为简单，但层级越高的服务器节点负载越重，容易

产生性能瓶颈，因此对于本系统的高性能需求也不适用。

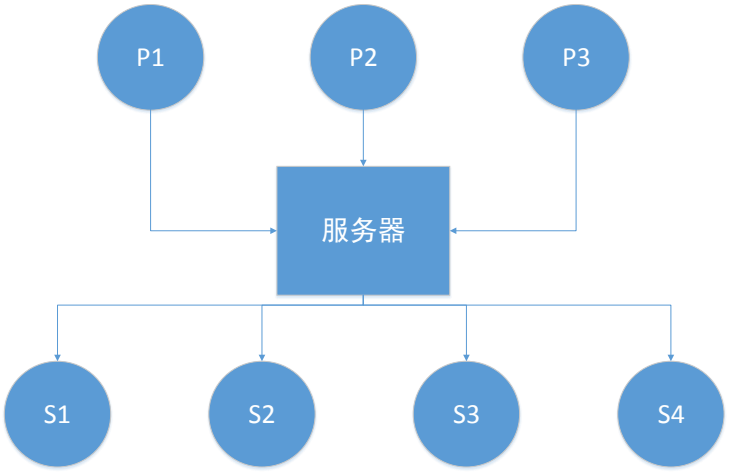

图 2-1 集中型拓扑

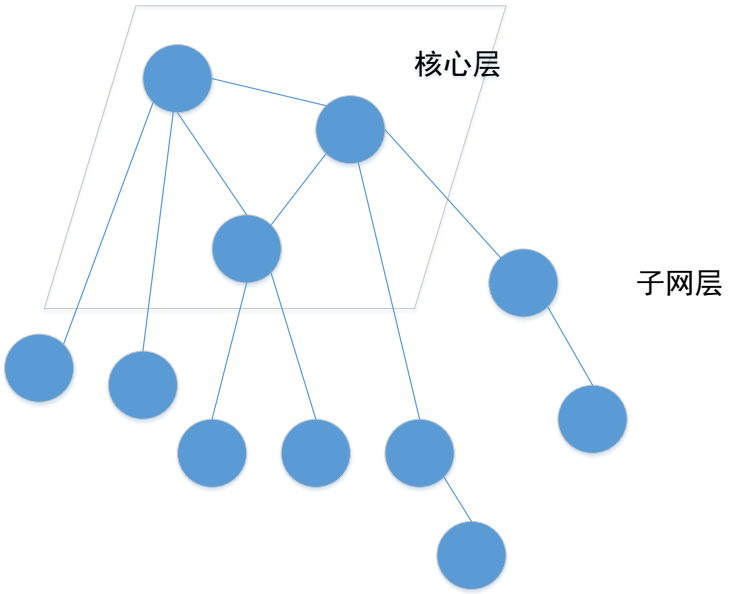

图 2-2 层次型拓扑

本系统相关拓扑介绍

集中型和层次型拓扑在大规模应用时均会产生负载不均衡导致消息传输瓶颈的问题，因此本系统的设计采用混合型拓扑，以相邻地域的方式构建各个 cluster，cluster 中以一个对外代表和其他节点直连的形式连接，Cluster 之间以无环形拓扑或有环形拓扑相连，这样可以在大规模系统中方便扩展，同时保证大量事件传播的实时性。

本系统结合各拓扑结构的特性，选用无环图拓扑、有环图拓扑、混合拓扑相结合的架构，以下分别介绍这几种拓扑。

1、 无环图拓扑

无环图拓扑的所有代理服务器组织成一个无环图，如图 2-3 所示，其中 S 代表代理服务器，负责消息的转发，C 代表客户端，负责发布和接收订阅事件。每个代理服务器收集管理自己客户端发出的订阅，并根据拓扑图转发给其他代理服务器；当客户端发布事件时，其代理服务器会将该事件接收后根据拓扑图计算出转发路径进行相应的转发。SIENA 和 Gryphon 都支持这样的拓扑。此种方法可以有效防止路由环路，节约网络资源，在构建本系统拓扑时可以应用。

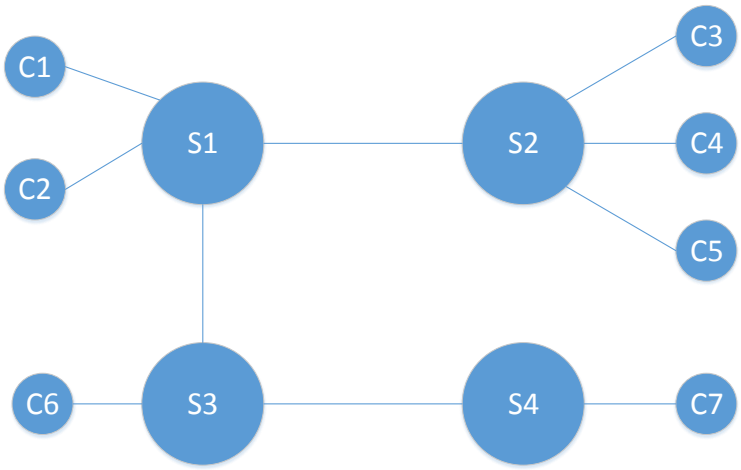

图 2-3 无环图拓扑

2、 环形拓扑

在环形拓扑中，服务器呈环形互相连接，他们之间是 peer-to-peer 的关系。如图 2-4 所示，在这种拓扑中，服务器之间和服务器与客户端之间的协议在类型和信息量上有所不同，服务器之间使用双向通信协议交换订阅和通知，客户端是请求订阅的发生者，也是发布通知的最终接受者。服务器需要维持彼此的信息，包括拓扑和订阅的信息，作为访问点或者传递消息的路由器而存在。消息传递的路径需要各服务器在环形拓扑的基础上计算得出，并避免循环转发。这种拓扑相对于无环图可以更加有效，与事件的初始转发节点关系不大，使转发效率更高，可以应用于本系统的拓扑构建中。

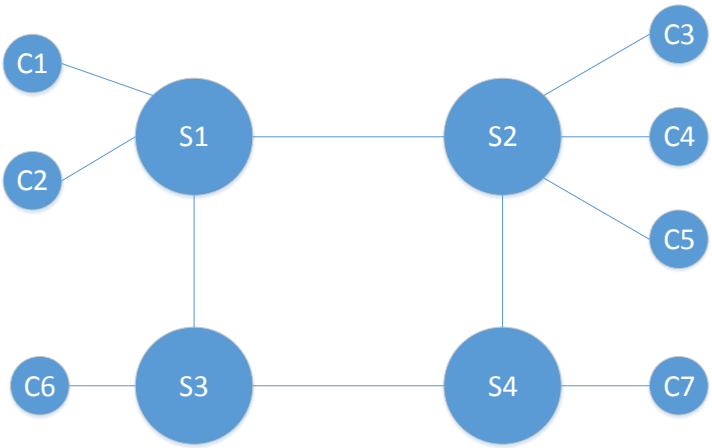

图 2-4 环形拓扑

3、 混合拓扑

在实际应用中，因为物理连接或分类管理方便的考虑，网络可能被分割成一个个较小的子网（Cluster），这些子网分别有自己的组网方式，子网之间以一定的拓扑连接，这便是混合拓扑。[9]如图 2-5 所示，每个集群内可能采用了不同的拓扑，每个集群通过一个节点与外部连接，从整个网络的角度观察，每个集群又都可以看作是拓扑中的一个节点。这种拓扑方式适合应用于较大规模且在地域上有分片划分现象的系统，方便管理，在区域内消息的转发效率也因拓扑的简化而变高。[10]

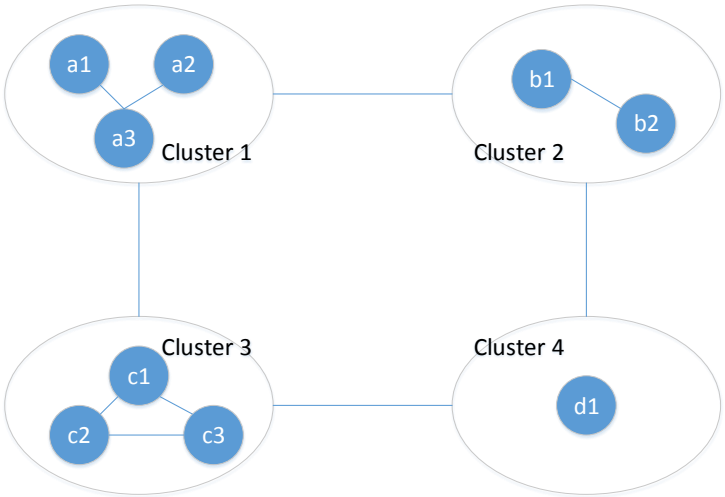

图 2-5 混合拓扑

2.1.2. 基于事件的路由

发布/订阅系统的路由是对于每个发布事件在发布者与订阅者之间建立路径，使得该事件可以正确地到达其订阅者。系统的路由算法与拓扑结构紧密相关，对于不同的拓扑结构应结合其特点选择相应的路由策略。[11]发布/订阅系统的路由与传统网络路由的思想基本一致，主要的区别是在传统网络中依据目的 IP 来进行寻址，而在发布/订阅系统中依靠事件的内容，事件的内容中除标题外还包含目的 IP。[12]

对转发路由的选择很大程度上依赖于以事件为基础的中间架构的稳定性，Hermes 使用了一种比其他研究更稳定的算法，且不需要全局的广播。[13]在本文的研究中也主要以 Hermes 范式为参考模板，设计路由的选取和转发策略的选择，因此将着重介绍 Hermes 的相关技术。

Hermes 的路由应用于基于类型的发布/订阅系统和支持主题过滤的基于类型和参数的发布/订阅系统，[14]考虑到本文中系统的具体应用场景，将主要介绍在基于类型的发布/订阅系统中的应用。

在事件的发布端和订阅端交换四种信息：[15]

1、**类型消息。**在系统中增加新的事件类型并为类型设置聚集存储点。一条类型消息包含在聚集点存储的事件类型的定义，发布的事件可以通过该定义进行类型检查。

2、**广告消息。**代表一个发布者发布一种特定类型事件的能力。通过这种消息在代理网络中设置事件的传播路径，并沿路径传播至聚集点。

3、**订阅消息。**代表一个订阅者对于某些类型的事件感兴趣。此类消息通过与广告消息建立联系，创建事件在网络中的传播路径，传播至聚集点。

4、**发布消息。**在消息体中承载发布事件。由事件发布者发送，然后通过广告消息和订阅消息创建的路径进行传播。

基于类型的路由算法遵循的规则是：[16]

1、发布者发布类型消息；

2、设置聚集点；

3、发布者发布广告消息至聚集点；

4、订阅者通过将订阅信息路由至聚集点订阅该类型（主题）；

5、订阅消息沿途代理记录该订阅转发路径；

6、发布者发布事件；

7、事件沿广告消息的路径发送至聚集点，沿订阅路径发送至订阅者。

2.2. OSPF 的拓扑和路由

OSPF（Open Shortest Path First）在 Internet 技术飞速发展的今天已成为 Internet 广域网和 Intranet 企业网采用最多、应用最广泛的路由协议之一。[17]基于它的应用范围（自治系统）、拓扑维护有效方法（LSA）和高效的路由计算（SPF，Shortest Path First），适合于本系统的应用，因此对其加以分析和使用。

2.2.1. OSPF 在自制系统中的协议

OSPF 是一种典型的内部网关路由协议（Interior Gateway Protocol, IGP），一般在一个自治系统（Autonomous System, AS）内决定路由。这里的自治系统指的是该网络中所有通信通过统一的路由协议进行路由信息的交互，并且在此自治系统中，所有 OSPF 路由器维护一份一样的存放系统中链路信息的数据库，以此计算出相应的路由表。

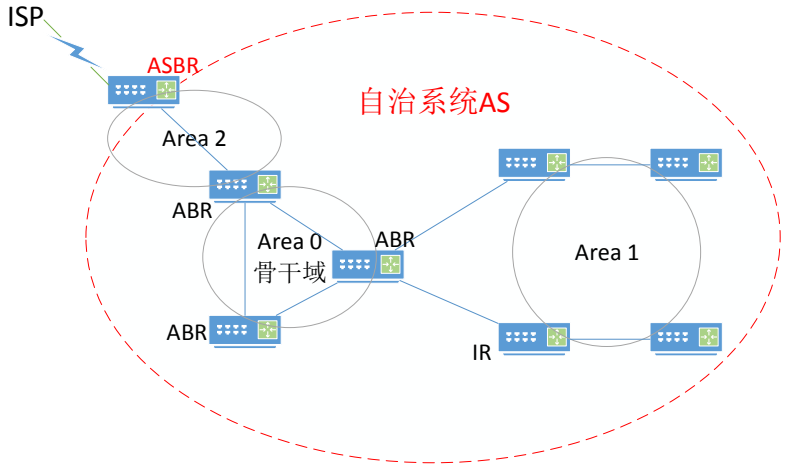

图 2-6 自治系统组成

图 2-6 所示为一个普通的自治系统的组成部分，在自治系统中有多个区域，其中 Area 0 为骨干区域，负责连接所有的区域，作为中间节点负责其他区域之间的信息传输。每个区域有独立的链路数据库，在区域内独立计算路由。[18]每一个区域内部的链路状态对外是不可见的，区域之间通过 ABR 来进行连通，这样可以有效减少全网范围内 LSA 的广播。

图中的 IR 为内部路由器，负责区域内的通信，其所有直连的链路都与它位于同一区域内，内部路由器也只须计算区域内的 OSPF 路由。ABR 为区域边界路由器，负责区域之间的通信。一个 ABR 与多个区域相连，保存着与它相连的每个区域的拓扑结构，同时也指导如何将消息传送至骨干域。ASBR 为自治区域边界路由器，是自治区域内部与外界通信的媒介，自治区域边界路由器可以同时是区域内部路由器或区域边界路由器，它会将外部收集的路由信息在自治区域内广播。

2.2.2. 链路状态路由协议

OSPF 是一种链路状态路由协议。网络中的路由以路由器的物理连接和网络速度为基础，并随之变化。链路状态是指路由器的端口信息、链路连接信息、外部状态信息等。有效的路由计算需要一致的链路状态数据库为基础，因此在所有路由器中维护相同的一个链路状态数据库。[19]

OSPF 使用 LSA（Link State Advertisement）来广播数据给某一区域的所有路由器。此 LSA 定时同步，同时有链路更新时也会有相应的路由器进行发送。[18]LSA 中包含发送源路由器上的所有端口状态的信息。路由器会根据收到的链路状态库计算出到每一个可达目的的代价和要到达该目的地实际上要转发的下一跳。网络中的 LSA 通过刷新（Flooding）的方式来交换 LSA，首先路由器会将自身的 LSA 数据包发送至所有直接相连的路由器，邻居路由器会根据此 LSA

更新自身的链路状态数据库，并将此 LSA 接着传送给与他相连的路由器，直到网络中的链路状态已经统一。

两个相邻路由器之间要经过邻居发现、双向通信、数据库同步和相邻关系建立阶段。如图 2-7 所示，两个路由器初始状态为 Down，即没有从邻居出接收到信息，但会尝试与其建立连接；init 阶段为收到邻居路由器的 Hello 包但本身并未在该 Hello 包中出现；当路由器可以互发 Hello 消息时便进入 2-way 阶段，然后路由器会发送 DD 包来确定主/辅路由器，此为 Exstart 阶段；在 Exchange 阶段，路由器互发 DD 包描述链路信息，也可以发送 LSA 请求信息；在 Exchange 阶段未收到的 LSA 会在 Loading 阶段继续请求和接收；最后当所有信息都已交换完成后到达最终状态 Full。

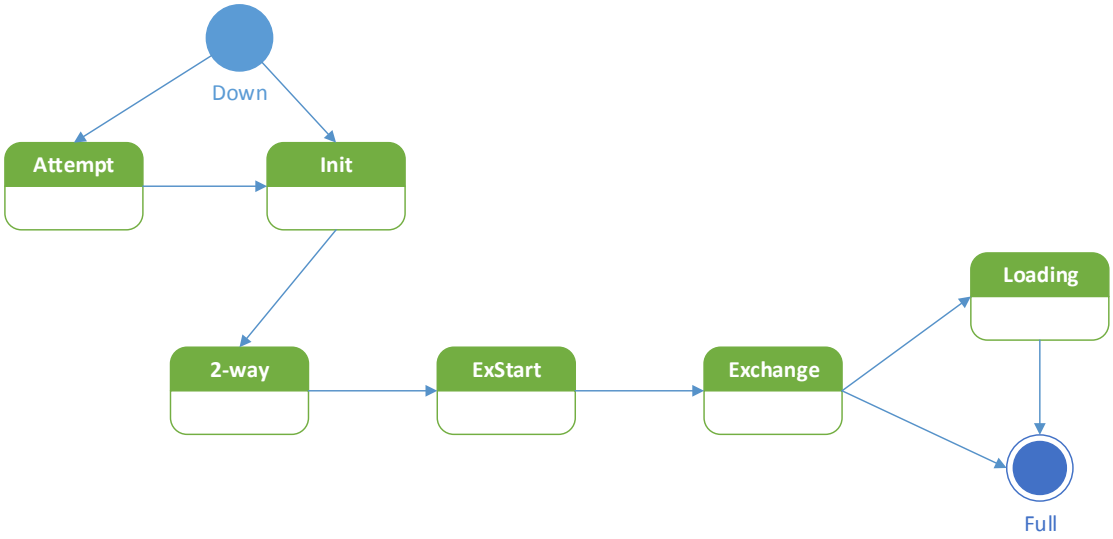

图 2-7 邻居建立状态机

2.2.3. SPF 路由算法

SPF 算法又称为 Dijkstra 算法，以该路由器的链路状态数据库构造出以自身为根节点的最短路径树，并由这个最短路径树生成路由表。所有的路由器并行运算 SPF 算法，计算网络路由。在 OSPF 路由协议中，使用开销（Cost）作为距离的度量值，节点之间的长度，也是最短路径树树干的长度，计算公式为$\text{Cost} = 100 \times 10^6/\text{链路带宽}$，其中链路带宽以 bps 表示。链路带宽与 OSPF 的 Cost 成反比，例如，2M 串行链路的 Cost 为 48，计算公式为$100 \times 10^6/(2 \times 1024 \times 1024)$；FDDI 或快速以太网的 Cost 为 1；10M 以太网的 Cost 为 10 等。这种度量的开销被分配到路由器的每个接口，两个节点之间的路径开销是这台路由器到目的路由器之间所有经过链路接口开销的和。

当链路状态路由算法构建完 LSDB 后，接下来便要调用 SPF 算法对 LSDB 内的 LSA 进行处理，计算出所有路径。

在本系统中，考虑到应用的可扩展性，两个 Cluster 之间的链路带宽有时因为相距较远而未可知，因此仅考虑两者之间的数据传播量对链路的影响以及 Cluster 之间的物理距离。

2.3. OpenFlow 网络中的控制

OpenFlow 网络由 OpenFlow 交换机、FlowVisor 和 Controller 组成。在这种结构中，原来完全由交换机和路由器控制的报文转发过程变为了由 OpenFlow 交换机和 Controller 共同完成，将数据转发和路由控制分离，这也是 OpenFlow 技术最大的特点。OpenFlow 交换机可以实现数据层的转发，FlowVisor 负责对网络进行虚拟化，Controller 则实现对网络的集中控制。[20]常用的 SDN 控制器有 NOX、floodlight、OpenDaylight 等，鉴于 NOX 和 OpenDaylight 实现较为繁琐，因此本文选取 floodlight 作为控制器。控制器的控制和交换机的转发均服从 OpenFlow 协议，因此可以达到互通消息的目的。

2.3.1. OpenFlow 的控制指令

OpenFlow 标准定义了控制器与交换机之间的信息交互协议，以及一组交换机操作。此协议运行在安全传输层协议或无保护的 TCP 连接之上。控制器可以向交换机发送指令，控制数据包的转发规则、配置参数（如 VLAN 优先级）。[21]交换机在代转发的链路中断或者接收到未指定转发指令的数据包时向控制器发送通知消息。

每个 OpenFlow 交换机上维护着一个或多个流表（flow table），其中每个表包含多条路由记录，每个记录又包含一个匹配域，定义流、指令集和计数器。匹配域可能包含跟接收数据包对应的比较参数或一个流的参数集中不存在此记录的指示值。[22]数据包到达后从第一个流表开始匹配，可能会匹配多个流表，这样可以允许数据包在下一个流表中进一步进行处理或者元数据信息在表内流动。[23]首先找到一个流表中优先级最高的流条目完成匹配，如果匹配成功，则更新这个流条目计数器值，同时这个流条目的指令集操作将被应用生效。

OpenFlow 支持三类消息：控制器到交换机、交换机到控制器和两者都可以发起的消息。控制器可以向交换机发送查询交换机能力、设置或查询配置参数、增删改流表等消息；交换机可以向控制器发送数据包不匹配、流表移除、端口配置变化等消息；两者均可不通过协商向对方发送 Hello 消息和 Echo 消息等。一

个 OpenFlow 交换机可以连接多个控制器，但其中只有一个为主控制器。

2.3.2. Floodlight 控制器在 SDN 中的应用

Floodlight 是目前主流的 SDN 控制器之一，它由于自身的稳定性和易用性已经获得了 SDN 专业人士以及爱好者们的一致好评，而且其为开源软件，使得开发更为便利。

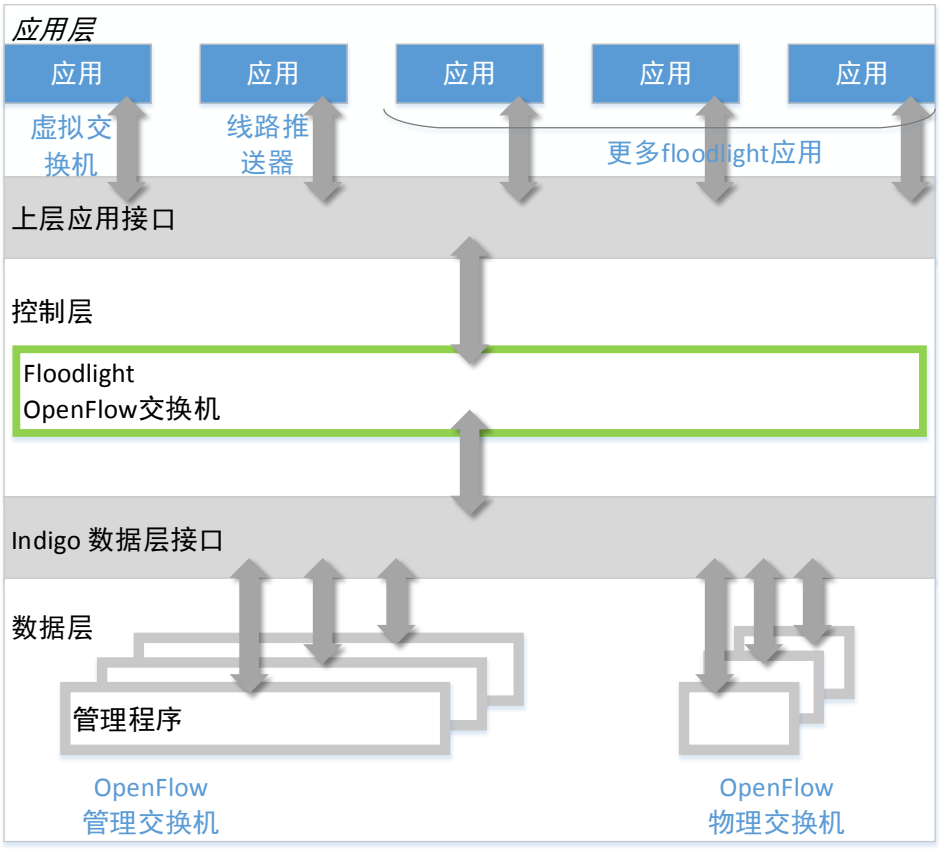

图 2-8 floodlight 框架图

Floodlight 具有发现网络拓扑和事件、控制网络交换机、提供 web 界面和 debug 服务器等功能。如图 2-8 所示，在 SDN 的三层架构中，控制器从数据转发层分解出来，使得数据的转发和路由的控制分离开来。[18]这种分离使得转发控制可以由一个分布式模型来求解取代传统的数据转发，同时也使得控制层面的开发和运行可以在比传统的低性能 CPU 的交换机/路由器更好的平台上实施。

Floodlight 实现了一系列的常规功能来控制和检测 OpenFlow 网络，同时向上提供一系列接口来满足不同用户的需求。当控制器运行时，由 Java 实现的 REST API 也可以通过指定端口来进行服务，任何语言书写的 REST 应用均可以通过发送 REST 请求给 REST 端口来获得所需信息。

2.4. 本章小结

本章主要针对课题所使用的具体技术进行了介绍，包括发布/订阅系统拓扑结构、路由算法、OSPF、floodlight 等技术。其中发布/订阅系统的拓扑结构是系统运行的基础，路由算法为事件转发提供路径，floodlight 为基于 SDN 的系统提供基础信息。

第三章 需求分析

本系统的设计基于发布/订阅模式，为消息的订阅和发布提供了服务平台和完整的转发功能。本系统为基于主题的发布/订阅系统，并在设计的过程中对主题进行分级，拓扑以混合模式构建，将订阅和链路状态信息一起同步，在此基础上以 Dijkstra 方法为主计算出正确路由以供消息转发。本文主要研究拓扑和路由部分以及发布/订阅系统在 SDN 上的移植，但基于系统的总体设计依赖于主题树和策略，而订阅管理则与路由直接相关，因此将着重介绍主题树、策略、订阅管理、拓扑维护、消息路由和向 SDN 的移植部分。

3.1. 需求概述

本论文中提到的基于主题的发布/订阅系统短期目标是探索 SDN 应用环境，在可编程路由器/交换机的设施之上实现发布/订阅系统，探索发布/订阅系统性能提高途径和 SDN 的应用环境，在保证系统平稳运行的同时实现服务的高效性和可扩展性。长期目标是扩大系统的应用范围，为实际工程项目（广告发布、消息交换等）提供可靠的消息传播平台。上层的业务类型对系统的要求是需要按照一定的主题或内容进行事件发布的识别过程。业务需求事件发布量比较大且对实时性要求较高，点对点的传播无法满足其需求，且对消息的识别在应用层，对网络的利用率也有待提高。因此，需要对系统进行进一步的扩展，满足其高效和节约带宽的需求，避免人为配置资源的局限性，同时实现消息发布服务平台资源利用最大化。

发布/订阅系统自适应地根据网络状况计算拓扑和路由，为消息的发布和传输提供基础。同时拥有对于主题树的匹配技术，方便对于用于重复主题的情况进行改善和快速处理。实现对计算机资源和网络资源管理的前提是能实时掌握网络传输状况和网络拓扑状况，在此基础上进行智能分析和决策，完成策略的动态调整和路径选择等。

综上所述，我们在系统设计和开发之前要明确系统的功能集，按照用户的需求详细描述基于主题的发布/订阅系统的功能需求，如表 3-1 所示。

表 3-1 功能需求描述

功能名称	详细描述
主题树操作	可以查看主题树结构和对其进行增加、修改和删除
事件订阅	可以根据主题树选择主题进行订阅
事件发布	可以指定事件主题进行消息发布
事件接收	对于订阅的主题可以正确地接收
事件传播	对于某一主题的事件可以正确发送至订阅集群并避免提交至未订阅集群
事件优先级设定	对于不同级别的事件按照其优先级优先发送高优先级事件
策略添加和删除	可以对一些集群添加和删除策略信息以控制其事件接收
客户端程序	可在一台或多台机器上同时启动一个或多个客户端程序
系统可扩展	支持系统中的客户端或集群随时间加入或退出

3.2. 主题树

随着信息的膨胀，主题的相关性会越来越大，特别是在某些特定领域，很多主题意义相近或具有一定联系，且他们的订阅者较为集中。因此，若仍将每个主题作为一个单独的个体，系统中就会存储很多内容、意义都相近的主题，从而产生大量冗余，此外，并列主题的增多对于路由表的查找也会造成负担。

在工程项目中，经常会有父子包含的主题，例如有两个主题分别是“计算机技术”和“计算机软件技术”，则对“计算机技术”感兴趣的订阅者往往也会对“计算机软件技术”感兴趣。基于此需求，本系统设计使用主题树的方式来存储主题。

路由的转发需要遵循主题树结构，在代表或代理端，需按照管理员定义的主题树进行事件转发的匹配、查找和路由转发。

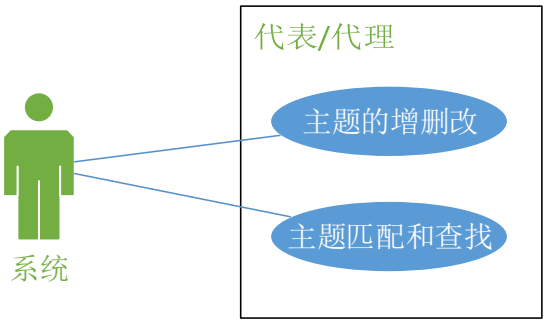

图 3-1 主题树用例

主题进行分级后，一个主题树内的多个主题就会在逻辑上出现包含的关系，在主题树上反应出来就是父子节点的关系，如图 3-1 所示为主题树的用例图。例如在图 3-2 中，根节点为“GLData”，其中订阅“digitalData”的订阅者也会对“万科”、

“fangDanYuan”和“maDian”感兴趣，所以，所有后三者的通知信息，都会发送给“digitalData”的订阅者，但是这个关系是不可逆的。因此在转发通知消息时，可以从上至下逐级查找和发送信息。

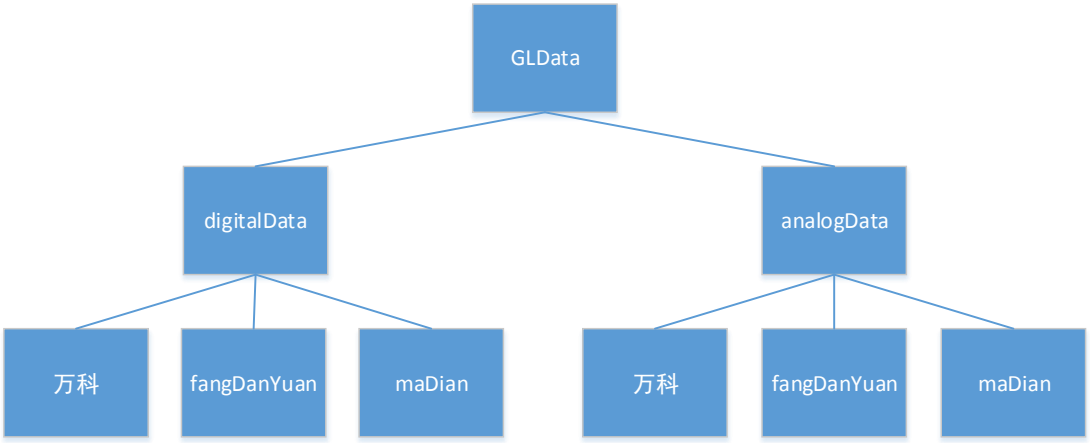

图 3-2 主题树

3.3. 策略处理

在实际应用中，信息的发布有时会需要对特定的订阅者进行约束，使其不能接收某些特定消息，此时就需要策略信息来定义这种约束。

策略信息是针对某一类主题，对系统中的某些客户端或客户群体的行为约束。主题的行为约束也同样遵循订阅时的上下级包含关系，如在图 3-2 中限制接收“digitalData”主题的客户群同样也被限制接收其下“万科”、“fangDanYuan”和“maDian”相关主题。限制的客户端可以是单个，可以是物理上相近的群体，也可以是无关的集合，这几种情况需要不同的设计对其进行限制。

为使消息能够准确地传送到所有未被策略消息限制的节点处，所有节点要保存同一个策略库。这样可以有效减少不必要的信息传输，缓解系统压力，同时也可以适应系统的保密性需求，防止节点任意订阅主题。

主题的策略信息应统一进行管理，不能由订阅者或发布者任意更改，以此对主题消息的传播范围做统一限制。系统的各参与者可以通过路由计算和向客户端提交限制的方法来实施策略信息。

图 3-3 所示为策略的用例图，用户可以通过管理员进行策略的添加、修改、删除操作，这些操作都会在生效后在系统中产生相应的作用。

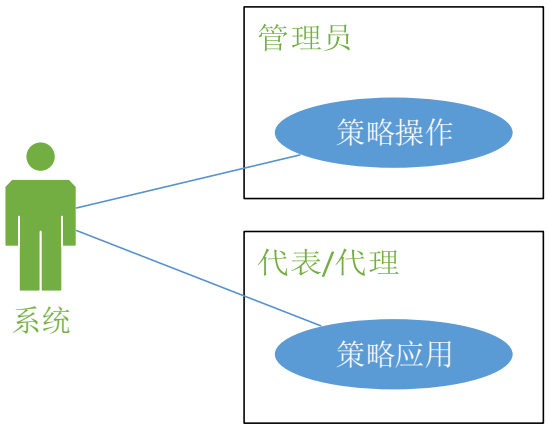

图 3-3 策略用例

3.4. 订阅管理

用户需要根据主题树的定义，选取主题进行相应的订阅。在订阅的过程中，需要遵循主题树中上下级包含的关系，当已经订阅父主题时，对子主题的订阅系统将不做处理，取消订阅时也是此种规则。

用户生成订阅后，发布/订阅系统需要对所有主题进行统一管理，保证系统中主题订阅表的一致性，以便计算出正确的路由转发结构，使订阅用户能够正确收到相应消息。

订阅主题的操作者是用户，用户可以选择主题树中的任意主题进行订阅和取消订阅，由系统对其合法性进行判定和传播。同时系统中若有针对该主题的策略并不影响订阅主题的传播，仅在路由计算和事件传播时有效。

系统接收到订阅主题后会对订阅和取消订阅分别进行处理，并执行相应的订阅传播和订阅存储，这些对用户不可见，仅为系统内部为保证正常运行而进行的工作。如图 3-4 所示为系统对订阅管理的用例图。

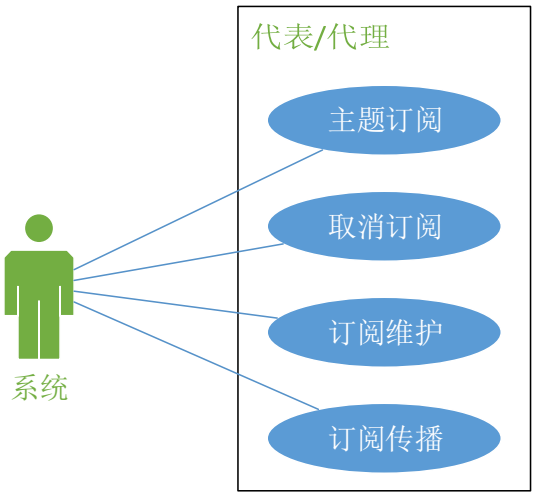

图 3-4 订阅主题用例

3.5. 拓扑维护

为保证系统的正常运行，需要拓扑维护机制对系统的网络状态进行统一管理。订阅的有效性依赖于客户端所属代表或代理的有效工作，路由的正确计算依赖于链路状态的正常维护和存储，数据的转发同样需要路由路径的与路由计算中链路状态的一致，因此拓扑维护是系统运转的基础。

拓扑维护包括集群内维护和集群间维护，其中集群内的维护主要为代表和代理之间的维护，集群间为各代表之间的维护。拓扑信息应在所有代表和代理中存储与实际链路相一致的信息。

节点之间的有效性探测主要依靠节点间的 Hello 探测机制维护，当链路状态变化时则需要 LSA 消息对变化链路进行广播，另外每个节点还需要相应的拓扑存储来提供路由基础信息。

如图 3-5 所示为代表的拓扑维护用例。

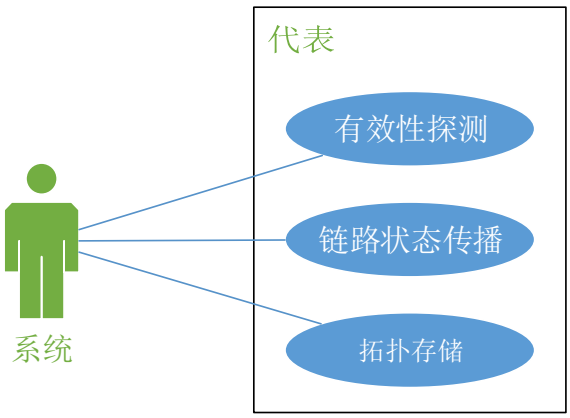

图 3-5 代表拓扑维护用例

3.6. 消息路由

在有代表或代理处接收到客户端的发布事件时，应按照一定的路径转发至所有订阅关于该事件主题且未被策略限制的客户端处，同时不应有回路或重复发送的情况产生。

为订阅客户端能够尽快收到发布事件，路由转发的策略应尽量高效，转发路径应在网络拓扑的基础上进行合理选择，综合考虑节点间的物理距离和网络中流量信息。

如图 3-6 所示为系统中消息路由的用例图，消息路由对用户透明，是系统为保证有效传输计算的转发路径。为保证转发效率，需要先将计算结果存储，在事件转发时再进行路由表的查询。由于路由是以集群为单位的，因此集群中的代理不需要计算路由，仅接收代表消息即可，代表需要路由的计算和查询。

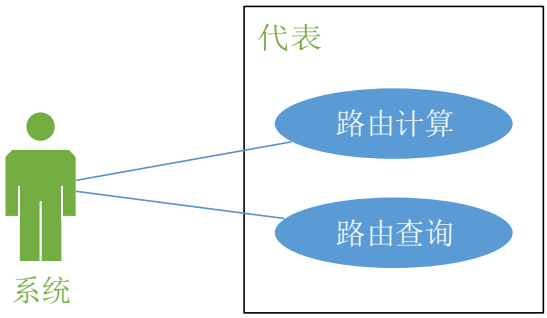

图 3-6 消息路由用例

3.7. SDN 移植

为使发布/订阅系统的传输效率得到进一步的提升，我们提出了向 SDN 的迁移。在迁移过程中需要在保持原有发布/订阅功能的前提下融入 SDN 元素，使用 OpenFlow 交换机或路由器，应用 OpenFlow 网络中 island 概念区分集群，实现对 Controller 的应用等。

为使 SDN 与传统网络相比的优势得到体现，在设计和实现当中应充分考虑 Controller 的控制和作用范围，优化数据流向，结合通过 OpenFlow 交换机的数据信息和链路状态进行路由选择的优化。

图 3-7 为 SDN 中路由用例。系统可以通过 Controller 收集其管理的 OpenFlow 交换机中的流量信息，并以此作为路由计算的参考值。

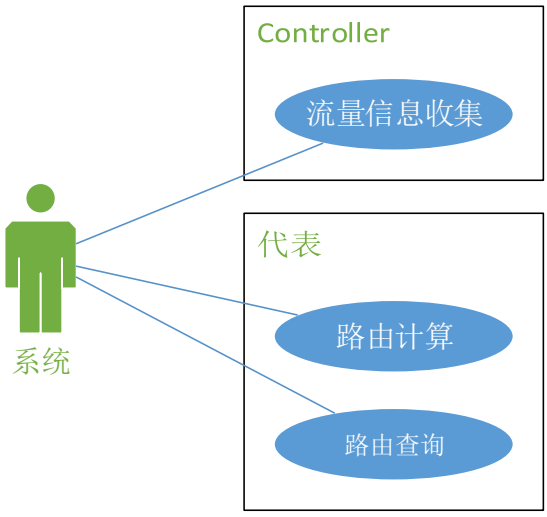

图 3-7 SDN 路由用例

3.8. 高效性和可靠性需求

现有的基于 wsn 组件开发的发布/订阅系统性能较弱，不能有效地进行拓扑

的维护和收敛性处理，路由转发较不合理，且未添加任何限制策略，所有主题均为平级主题，这给系统的工作带来沉重的负担。因此，本系统的一个主要目标就是综合运用各种技术，从拓扑维护、路由计算、主题树管理、策略设置、数据转发等方面进行效率的提升。

在效率提高的同时，系统的可靠性也必不可少，拓扑的可靠性可以给路由计算和数据转发提供基础，路由的可靠可保证系统内事件转发的正确性和防止环路的产生。由于系统在实际生产环境中需要长期运行，因此系统在长时间内的可靠传输也是衡量系统优劣的关键指标。原有系统由于设计和实施的缺陷，并不能坐到长时间稳定运行，因此可靠性的提高也是本课题的一项重要任务。

3.9. 本章小结

本章对发布订阅系统的订阅管理、拓扑维护和路由转发方案进行了详细的需求分析。我们分别阐述了主题树、策略库、订阅管理、拓扑维护、消息路由和 SDN 迁移部分的需求，并给出了详细的用例说明。此外，我们也提到了系统的稳定性方面的需求。最后，对本章做了小结。

第四章 概要设计

本系统可以向客户端提供完整的发布/订阅能力，提供发布和订阅的接口并设计主题树结构以供客户从中选择订阅。如图 4-1 所示，本课题主要研究发布/订阅系统的订阅管理、路由计算和拓扑维护模块，各个模块协同合作，保证整个系统工作的可靠性和实时性。

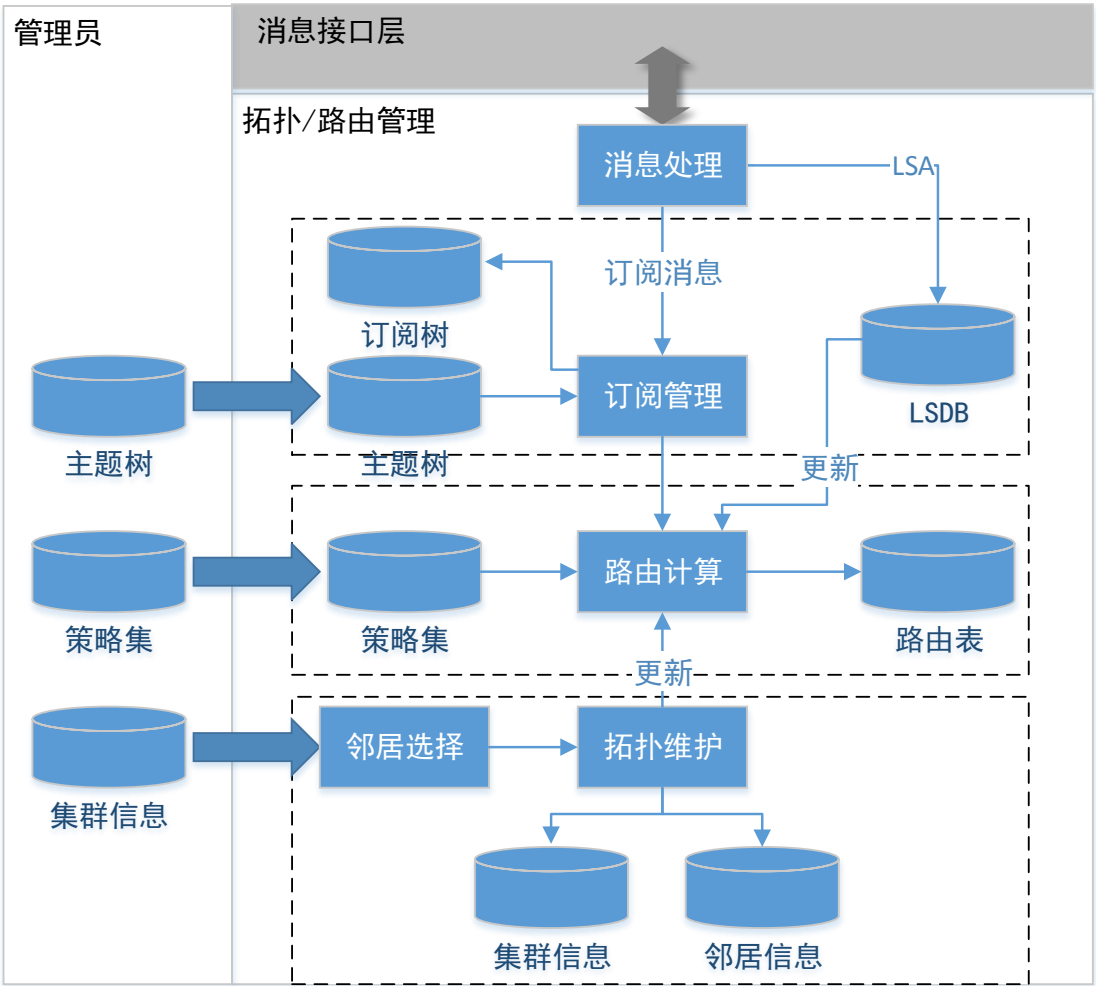

图 4-1 拓扑/路由子系统工作模块

其中主题树以 LDAP 的方式存储，当管理员启动时即向 LDAP 请求所有的主题树信息，并可以通过界面的方式展示、增加、删除、修改主题树信息。当有新节点请求时，将已经缓存的主题树信息下发。

4.1. 策略驱动的主题树设计

主题信息在大多数情况下都是独立的，但随着信息膨胀，主题持续增加，越

来越多的主题具有相关性。尤其是在某些特定的领域中，相关主题出现更加频繁，而且他们的订阅者也比较集中。在这种情况下，如果仍把每个主题看作单独的个体，系统就会存储很多内容相近的冗余信息，在订阅时也会由于主题种类太多而产生不便。故在本系统中采用主题树的概念，将主题分级，订阅父主题的订阅者默认其对父主题之下所有的子孙主题也感兴趣。同时引入的策略信息与主题有对应关系，由于策略信息是针对每个主题进行的限制，因此策略应与主题树保持一致。

主题是由客户通过管理员进行增加、删除和修改的，客户端订阅和发布相关消息时，消息的标题也是从用户定义的主题树中选取节点来命名，如果命名不在主题树中则无法成功订阅和发布。主题树是任意多叉树，只有父子节点之间有直接连线，其他节点之间没有相互关联。当删除主题树中某一节点时，其子孙节点也会相应地被删除。

当管理员启动时会从 LDAP 请求获得所有主题树，并在管理员的界面中展示，在本地内存中存储相应信息。其他集群代表向其注册的同时也会申请获得该主题树，用于订阅和发布消息的生成。管理员可以通过界面对主题树进行相应操作，当操作生效时，系统会在向 LDAP 更新的同时洪泛主题树，使主题树在全网范围内更新，代表收到更新消息后会重新向管理员进行主题树的请求。

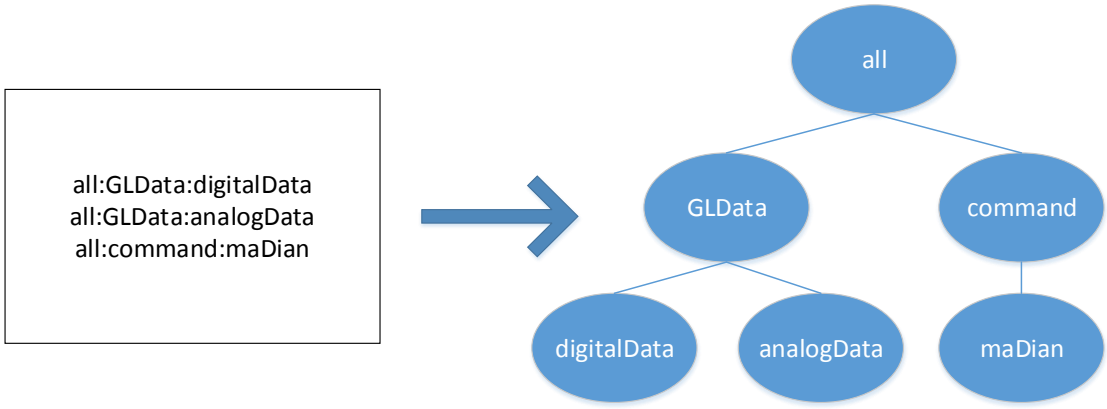

图 4-2 主题树转换结构

客户端订阅、取消订阅和发布事件的主题均从主题树中获得，若主题不在该树中则会订阅或发送失败。由于主题树是树状结构，客户端的主题消息在匹配时从树的根部开始逐级匹配，直到找到相应节点。如图 4-2 所示对应的主题和存储结构的关系。在本系统中为方便表示，字符串形式的主题以“：”来区分不同层级的主题，在匹配时则由“：”分割该字符串，获得真正的主题层级结构。

4.2. 策略处理设计

在系统中同时引入了“策略”的概念，定义关于某些主题限制某集群、代理或客户端的接收。策略的引入可以有效区分不同集群的权限，便于实现管理，防止信息泄露、网络流量不必要增大等状况发生。策略限制的主题也会从完整的主题树中进行选取，若该主题更名或被删除时，该策略信息也会有相应的改变。策略的功能体现在路由计算和向上层转发时，路由计算应尽量避免把策略主题转发至策略集群，若无法避免则应在该策略集群中做进一步判断，避免向上层提交该消息。

策略的配置通过管理员进行，相应的策略信息将会以 XML 文件的形式存储在物理磁盘上，其他集群的代表和代理也会以相同的存储方式存储策略信息。在代表进行关于某一主题的路由计算时会考虑针对于该主题的策略信息，在路径选择时避免传播到该集群处，但若策略仅针对某集群的某一代理或客户端则不影响路由的计算，会在被限制集群内由被限制者自主选择是否接收该类消息。

消息处理时遇到一些策略消息，则将其以一定形式封装至本地 XML 文件存储；拓扑类的消息则会在初步处理和判断后选择性地交付拓扑模块或丢弃。策略限制可以针对某些客户端、代理、代表或集群，根据需求的不同产生不同限制范围的策略消息。策略库的主要操作包括策略信息的添加、删除和修改。

如表 4-1 所示即为一个典型的策略信息 XML 存储结构，该策略信息表示对于标题 all:command:maDian，对集群 G1 的所有节点进行限制接收。

表 4-1 策略信息

```
<?xml version="1.0" encoding="UTF-8"?>
<WsnPolicyMsgs>
  <policyMsg targetTopic="all:command:maDian">
    <array as="targetGroups">
      <TargetGroup allMsg="true" name="G1"/>
    </array>
  </policyMsg>
</WsnPolicyMsgs>
```

策略的添加、删除和修改操作在管理员处进行统一实现，管理员进行相应操作后会通知所有集群，集群代表在接收到相应消息后会以相同的形式在本地进行消息存储，并进行必要的路由计算。

在本地接收策略消息后会对本地已有的策略消息进行判断，与新消息进行融

合，若策略消息有更新，则须进行相应的路由计算，图 4-3 描述了这一过程。策略对主题的识别也遵循主题树的构造，在策略合并时同样是上级包含下级的关系。

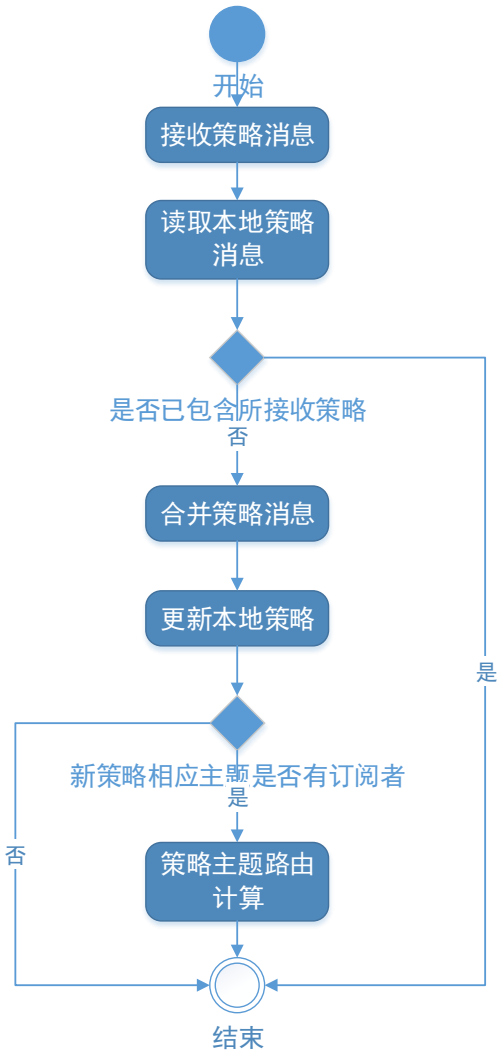

图 4-3 策略处理流程

4.3. 订阅管理设计

在发布/订阅系统中，对订阅表的一致性维护对路由的正确计算和消息的正常传播提供了基础，其完整性和一致性在系统维护中也至关重要。参照 quagga 对拓扑维护的设计，在本系统中对订阅表的一致性维护采用了 LSA（Link State Advertisement）发送并周期性同步的机制，对每条 LSA 编上序号，以此识别其到达顺序，防止循环传播带来的消息冗余。针对于每个集群的 LSA 会集中存储于 LSDB（Link State Database）中，并周期性地对 LSDB 中的信息进行失效检测，当某 LSA 超时未更新时则判断该集群丢失。

LSA 的更新具有交互性，一个集群的邻居信息也会在对应邻居的 LSA 中更

新信息，此过程发生在代理服务器将接收到的 LSA 存储 LSDB 中时。如有一集群发送消息中携带新加另一集群为邻居的消息，则也在 LSDB 中另一集群的 LSA 邻居信息中增加该集群信息，但邻居集群的序列号并不会因此改变。当有集群发送的消息中携带删除邻居的信息时过程与此类似。此机制的设计主要为避免消息到达延时或邻居集群已丢失带来的路由计算不一致造成回路的状况，对于一条链路从左右两端到达对方的路程一样。整个网络的拓扑结构为一个带权的无向图。

LSA 消息记录关于一个集群的基本状态消息，包括名称、与邻居之间的距离、订阅，其消息结构如表 4-2 所示。代理服务器会根据该条 LSA 消息的集群名称检查本地 LSDB 中是否有该集群的消息，若有则执行合并操作，没有则直接将其存储。每条 LSA 拥有自己的序号，每个集群的 LSA 序号从零开始，在整数范围内递增。

表 4-2 LSA 消息结构

序号	标识一个集群的 LSA 唯一编号，从 0 开始递增
是否同步 LSA	判断是消息同步 LSA 还是普通消息更新
集群名称	发送 LSA 源的集群名称
代表 IP	发送 LSA 源的集群代表 IP
失效邻居	发送 LSA 的集群新失去连接的邻居
订阅主题	LSA 发送者新增加的主题（普通 LSA）或所有订阅（同步 LSA）
取消订阅主题	LSA 发送者取消的订阅集合
与各邻居距离	Map 形式存储 LSA 发送源的邻居和与其距离

4.4. 拓扑维护设计

在系统中考虑到一些代理节点物理位置上的相近和便于扩展的因素，在系统设计中引入集群的概念，即一个集群中可以有多个代理服务器，其中有一个为代表，作为本集群的代表与外界通信，其余为代理。代理和代表本质相通，都可以为若干客户端提供消息的订阅、发布和接收服务，代理是备选的代表，当代表不可达或程序停止时代理中会选举新任代表并通知其他集群和管理员关于本集群代表信息的更新。

此模块主要用于维护系统正常运作所需要的拓扑结构，使得订阅消息的洪泛、路由的一致性计算和通知事件的正确传播能够顺利进行。拓扑是整个网络的基础，又由于其动态性，网络传输很难得到保障，所以考虑必须周详。

拓扑包括邻居的选择与构建、邻居之间的维护、集群内的拓扑信息维护。邻

居选举时综合考虑待选举集群中的代表与本代表的距离和其已有邻居个数，选择适当的目标进行邻居构建请求，当获得肯定反馈时开始邻居维护。邻居是路由计算的基础，在实际环境中，由于路由计算代价、拓扑维护代价和实际物理连接等方面的考虑，不能构建一个全连通的拓扑图，必须根据实际情况选择个数合适的邻居进行维护。邻居个数的最大值可以事先根据实际应用环境在管理员处进行设置，每个集群的邻居个数可以不同。在集群开始加入时，代表会选择最大值/2 个邻居作为默认构建邻居，当邻居个数少于最大值/3 时重新选择新邻居构建，此构建过程当邻居个数已满足要求或已无可探测备选邻居时结束。

如图 4-4 所示，在横坐标轴上的红点表示新加入的集群，其初始情况下邻居个数为零，其余各点为网络中已存在的集群代表对应的信息。若允许构建邻居的最大值设为 6，则初始加入时默认构建的邻居数为 3，选择如图 3-6 中半圆内的三个蓝点所示，若这三个点构建邻居不成功则会将其排除然后继续扩大半圆的范围找到下一待构建节点。图中纵坐标为各集群的邻居个数，横坐标表示各集群代表的相对距离，此距离在当前版本中仅由代表的 IP 相差绝对值来模拟，在今后扩展中可以加入实际物理位置的考虑。

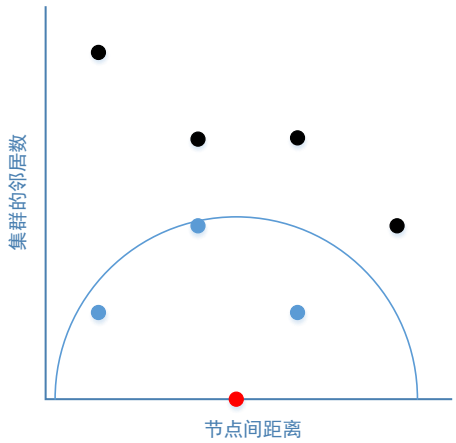

图 4-4 邻居选择

当新的代理服务器启动时会向管理员请求当前网络的集群信息，若已有本集群信息则自动成为集群内代理成员，并与集群代表进行交互和维护；若无本集群信息或原有代表已失效则自动成为新代表，作为新集群加入到网络中。当代理服务器作为代表加入时首先要构建邻居，构建方法综合考虑集群距离和已有邻居数，按照图 4-4 所示构建模型，选择距离较近的若干点进行请求探测和接收待选邻居的反馈消息，并向其中一个邻居请求全网的拓扑信息和订阅信息，接收后按照本地格式进行存储。建立邻居之后即开始邻居间的 Hello 维护，如图 4-5 所示，此消息采用 UDP 方式传输，周期性发送（目前为三十秒），并设定接收阈值（目前为两分钟），若超过此阈值则探测该节点是否可达。加入完成后生成关于本集

群的 LSA 通过邻居扩散至全网。

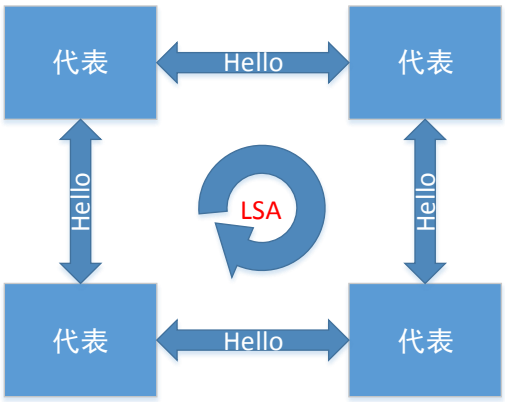

图 4-5 集群间 Hello

集群内的 Hello 消息是每个代理向代表发送 UDP 消息，而代表通过组播的形式在集群内发送，如图 4-6 所示。当一个集群内的普通代理失效时，集群代表会由于其 Hello 消息超时而知道其失效，将其订阅消息取消并通知本集群其他代理后判断该代理的失效是否影响了原有集群订阅表的内容，若有更改则须发送本集群 LSA 通知其他集群新的订阅信息。当集群的代表失效时，其相邻集群会因为其 Hello 消息的超时而将其加入 waiting list，等待它的重新选举；集群内部若有有效的其它代理则会选择一个来成为新代表，新代表会通知集群内其他成员以及通过该集群的邻居告知全网新代表的信息，最后向管理员处更新本集群的代表信息。相应的，当代表检测一个邻居 Hello 消息失效时也会等待其重新选举，超过等待时间才会删除邻居关系。

拓扑模块维护的“邻居”信息与代表实际的邻居有所差别，拓扑维护的邻居包括集群邻居中 Hello 消息未超时的邻居和集群内的其他代理，其中集群的邻居以集群名的形式传给拓扑模块，集群内代理则以 IP 的形式载入。拓扑模块对这两种邻居的失效判定条件相同，不同的是调度模块会对集群邻居保留等待选举的时间，对代理则不会。

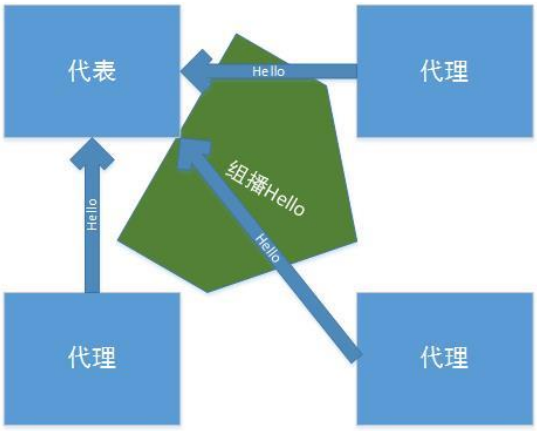

图 4-6 集群内 Hello

拓扑维护包括集群内、集群间、管理员与备份管理员之间的拓扑构建和维护，他们之间的联通关系如图 4-7 所示。其中备份管理员保存管理员的所有信息，并与管理员保持与其他集群代表之间一样的 Hello 消息维护，当管理员失效时进行角色替换和通知所有集群。在局域网内可以只通过代理服务器的 IP 来进行识别和传递消息。通常一个集群在同一局域网的同一网段中，互相能够通过组播的方式传递消息。

各个集群的名字可以由用户自己定义，每个代理服务器在启动时就已在本地设置好要加入的集群名称，然后向管理员报告。管理员通过查找本地的配置信息判断是否已对该集群是否有预定设置，查找成功即可将配置信息返回，若未找到该集群配置信息则返回默认配置，并将该集群的实际配置以文件的形式保存至本地。

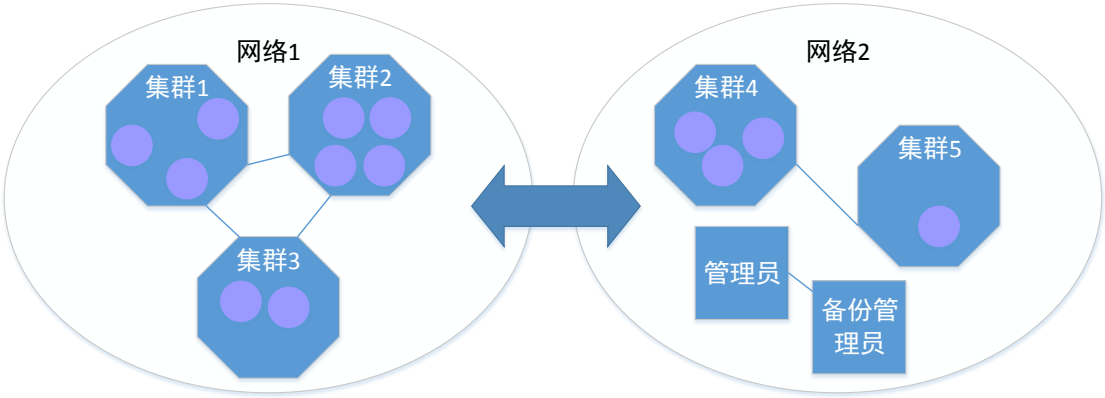

图 4-7 集群拓扑图

调度模块与系统总线的作用类似，接收所有的消息然后将其发送到对应的处理程序中去。这些接口包括 TCP、UDP 的远程连接和系统内部接口的调用，与管理员、其他代理服务器、本系统内各模块（数据转发、拓扑模块等）都有信息的交互。通过对消息类型和接收时间、来源等信息做初步判断后提交至其他接口做进一步处理或直接给出反馈。调度模块的正常工作对于系统的运行来说至关重要，对于消息种类的划分、优先级的设定、多线程的考虑等都需要在设计和测试过程当中不断完善，将实际应用环境模拟和试验，对代码细节进行不断修订，使系统能在真实环境下正常运行。

集群邻居个数的设置需要综合考虑集群之间相连的物理链路状况和集群的总数目，避免不必要的构建或是网络中邻居相连太多带来的拓扑维护代价增加。在现有版本中邻居个数是在管理员处对所有集群进行手动限定的，需要根据实际网络状况进行人为的判断和设置，但这并不利于网络的扩展，可在未来加入个数动态变化的设定，仍然由管理员限定个数，但数字由目前网络状况分析之后算出，在新集群不断加入的过程中，若有增加原集群邻居数的需求时也可以重新计算出

数值后通知该集群代表。通知消息的转发需要稳定的拓扑结构做基础，因此对邻居的构建和他们之间的维护显得尤为重要，需要在设计和实验的过程中不断完善，适应实际网络状况。

4.5. 策略驱动的主题树聚合路由设计

路由计算用于设计路径以供各个主题消息的转发，高效的路由算法是路由模块甚至整个系统的关键问题。路由模块提供接口查询路由信息，同时屏蔽计算细节，接口依据需求而定，在接口不变的情况下具体实现可以有多个版本。路由计算建立在拓扑结构和主题订阅的基础之上，在本系统中，利用最短路径算法，同时设置最多下一跳个数，同时应用策略信息，求出针对于每一个主题的路由树。在计算时首先对于集群按照其名字排序，选取第一个为根节点，此后按照 Dijkstra 算法依此计算，直到关于此主题的订阅集群全部计算完毕，以此保证所有集群计算出的路由树一致，完成消息的正确转发。

路由转发时，通知事件可能是源于本集群也可能是其他集群，同一棵路由树对于这两种情况有不同的下一跳节点，同时如果本集群也有相关订阅的话也需要在群内组播。通知事件携带源集群的信息，对于各事件只须根据它的主题和发送源即可查找出相应的下一跳，这个功能在数据转发阶段，通过调用路由提供的查询接口来完成。路由模块的内部结构如图 4-8 所示。路由计算模块由订阅管理模块存储的订阅表和 LSDB 中存储的拓扑信息共同计算出路由树的集合。当通知事件到达时，数据转发模块通过调用路由查询的接口查询路由表即可获得对于该事件的下一跳，将其放到发送队列当中。图中 LSDB 中的与各邻居间的距离是计算过程中需要用到的，如有碰到链路记录不对称的（如 A 集群记录与 B 的距离为 1，B 集群记录与 A 集群距离为 2），则取其中 LSA 接收时间较近的一个为准；订阅表即为已在订阅管理阶段描述的树状结构，每个节点记录本级名称和所有的订阅集群；流量信息以集群名-流量的方式记录，每隔一段时间（如半小时）扫描一次，计算平均的流量，若与之前值相差较大则须更新拓扑信息，同时清空记录开始重新统计；路由查询接口提供传入参数事件的上一跳转发者和事件主题，通过查询路由表来返回需要转发的下一跳集合，此接口会在通知事件到达时被数据转发模块调用。

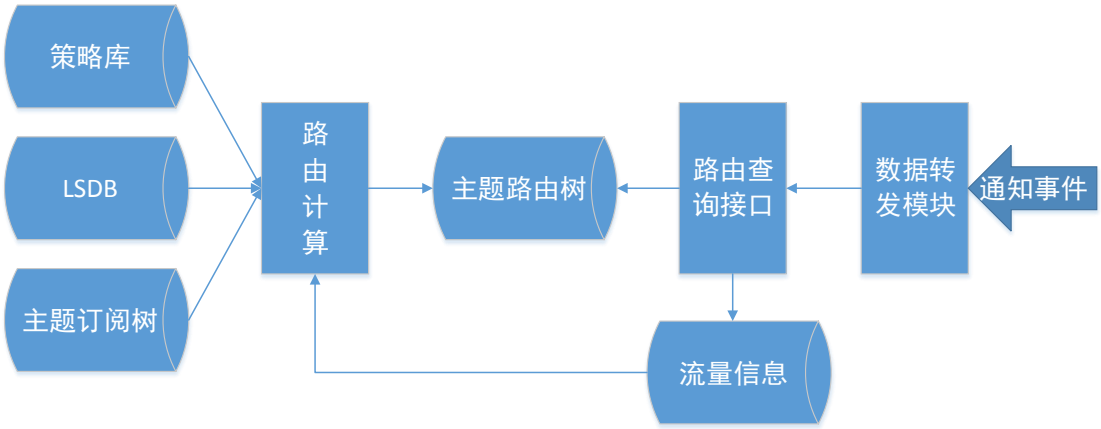

图 4-8 消息路由过程

数据转发模块在转发时对于每条通知事件调用路由查询接口获得转发的下一跳，查询接口在返回其下一跳的同时会记录本集群至下一跳集群的总转发量，由于路由树是基于拓扑结构的计算，此总转发量也就是通知事件在本集群与邻居集群之间事件的流量。在路由计算的过程中从订阅表中获得所有的主题，对于每一个主题计算其路由树，其中将会用到完整网络的拓扑信息，而拓扑信息的更新则与邻居间的流量统计有关，因此流量信息也会间接影响路由的计算结果。在统计流量信息时主要检测其单位时间内的流量，每当单位时间内流量增加或减少一定值（如 1 秒 100 个事件）时便将本集群与该邻居集群之间的距离加一然后通过 LSA 更新拓扑消息。

路由计算以 Dijkstra 算法为基础，将目标节点由一个扩展为 N 个，其中 N 为对于某一主题的所有订阅集群。在计算过程中首先对所有集群按照名称进行排序，之后的计算匹配也是按照这个序列进行，选取订阅此主题的最前一个集群作为此路由树的根节点，之后从根节点出发依此找到到所有订阅集群的最短路径，构建转发树。在计算的过程中可以认为设定一个父节点最多对应的子节点个数，防止由于某集群邻居个数较多带来的路由压力，当所有订阅集群均在此路由树时停止计算。如图 4-9 所示，左边为网络拓扑结构，集群之间有连线代表邻居集群，线上数字为他们之间的距离，距离初始值为 1，其后按照上述流量统计的方法增加或减少。对于某主题 A，图中深色部分集群（G1、G2、G5、G8、G9）为订阅集群，在路由计算时首先排序后选取 G1 为此主题路由树的根节点，其后根据最短路径原则选取路径，依此选取了 G2、G5，然后又以 G7 为中间节点，间接到达 G8、G9，至此计算结束。若此集群为 G7，则针对于此主题树只需要存储其根节点 G1 和对应的下一跳节点 G8 和 G9 即可，其他中间信息在计算之后便可忽略。在通知事件 A 到达时，集群 G7 通过匹配事件 A 从路由表中查找到下一跳集合为 G1、G8、G9，若其中有上一跳转发者，如 G1 转发事件 A 给 G7，则从下一跳集合中删除 G1，然后将结果返回。

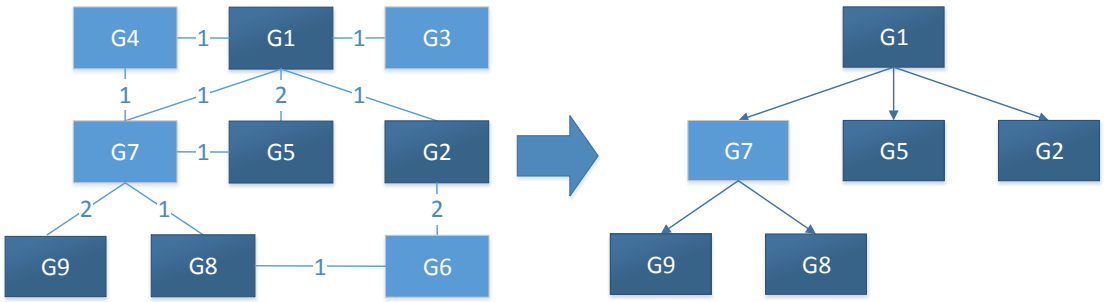

图 4-9 路由计算过程

对于集群内的通知事件转发则比集群间要简单得多。为使集群内的消息流量得到控制，也避免对代表产生过大的负载，当集群内其他代理有订阅某事件时，代理通过组播的方式传递给集群内代理，由代理接收事件后自行判断是否为本代理订阅。当代理主动发送通知事件时则只须将消息发送给集群代表，由其代为转发即可。集群的组播地址在每个代表或代理加入集群时从管理员处获得，获得后加入这一统一的组播群，即可在组播群中收发消息。

4.6. SDN 移植设计

在系统向 SDN 移植中，包括物理设施的移植、对控制器的应用、拓扑维护的扩展和路由计算的优化。

OpenFlow 网络中有各种开源的 Controller 版本，本系统中选取 floodlight 作为控制器原型。Floodlight 遵循 OpenFlow 1.0 协议，配合网络中的 OpenFlow 交换机使用。如图 4-10 所示，在拓扑维护时，引入 Controller 的消息反馈机制，当 OpenFlow 交换机直连的机器失效时则不必等待 Hello 消息的探测即可直接判定该机器上运行的代表或代理失效，提高系统的收敛效率。

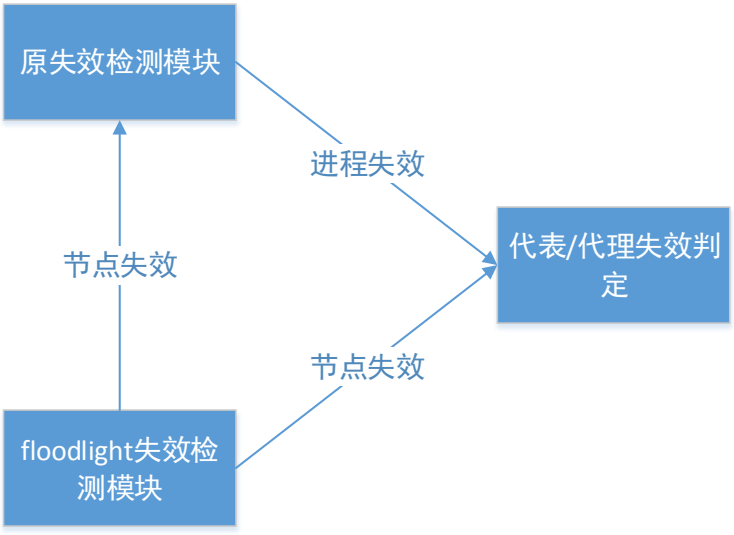

图 4-10 SDN 中的失效判定

按照 OpenFlow 网络中的定义，以集群为 OpenFlow 网络中的 island 进行统一管理，将 OpenFlow 交换机取代原有交换机连接集群和普通网络。Floodlight 中有 TopologyService 模块，为 floodlight 提供拓扑服务，维护拓扑信息，同时也可以发现网络中的路由。该服务从 IlinkDiscoveryService 中获取链路信息并计算出网络拓扑，在拓扑服务中有一个很重要的概念叫做 openflow 区域。每一个 openflow 区域内的交换机与同一个 floodlight 实例相连，openflow 区域之间可以用二层的非 openflow 交换机相连，如下为一个 openflow 区域的示例：

[OF switch 1] – [OF switch 2] – [traditional L2 switch] – [OF switch 3]

为适应 SDN 网络，在移植过程中也将按照 openflow 区域的划分方式来划分各个集群，方便发布/.订阅系统和 floodlight 对网络拓扑的协同工作。

应用控制器收集信息的功能，收集交换机上的数据流量信息，应用到路由计算中，同时在拓扑维护方面结合控制器的探测函数，发现失效代理。

4.7. 本章小结

本章主要介绍了发布/订阅系统中消息事件的发布流程、订阅管理、拓扑维护、路由计算策略处理和 SDN 移植的具体功能，并根据系统中对于各模块的需求进行详细地设计。同时分析了各个设计模块采用设计方法的原因和预计效果。

第五章 拓扑和路由实现

本章主要介绍发布/订阅系统中策略处理、订阅管理、拓扑和路由部分的实现。其中拓扑和路由部分是本文关注的重点，也是为保证正确、高效的消息传输的核心。

5.1. 策略处理的实现

用户从管理员处可以添加、删除和修改策略消息，如图 5-1 所示即为管理员处的操作界面。其中主题显示为从 LDAP 中读取的主题结构，用户可以从下拉框中进行选取，进行限制的集群则从当前已经在管理员处中注册了的集群。在实现中暂时只针对整个集群进行策略的限制，未具体到代理或客户端。

在每个集群代表启动时向管理员注册，即可从管理员处获得当前所有的策略信息，动态变化的策略则须管理员主动通知各集群。

策略信息在合并时遵循主题树上下级关系，如对 G1 关于主题 all:command 进行策略限制，则若新添加对 G1 关于主题 all:command:maDian 的策略则并不实际生效，其消息会被融合进前一条策略消息中。策略生效后会在路由计算时在订阅列表中去除策略限制的集群，同时每个集群的代表在接收到关于某个主题的事件时也会进行策略判断，若本身集群被限制则不在本集群内组播也不向客户端提交。

图 5-1 策略操作界面

5.2. 订阅管理的实现

为正确计算出针对于某一主题的正确路由，全网的订阅表必须保持一致，邻居状态也应同步，此过程都由 LSA 实现。由于网络消息有一定的丢失率以及拓扑变化等问题，链路消息不一定完全被接收到，因此设置周期性同步机制，通过可配置的时间（目前为半小时）同步本集群的链路信息。与此同时设置相应的超时时间（目前为四十五分钟），周期性（10 分钟）检测本代理服务器存储的所有 LSA，若有超时或某一集群的邻居数为零则认为该集群已丢失，删除其在本代理处的所有注册信息，全网中有若干集群检测到这一丢失信息后会通知管理员该集群的丢失。

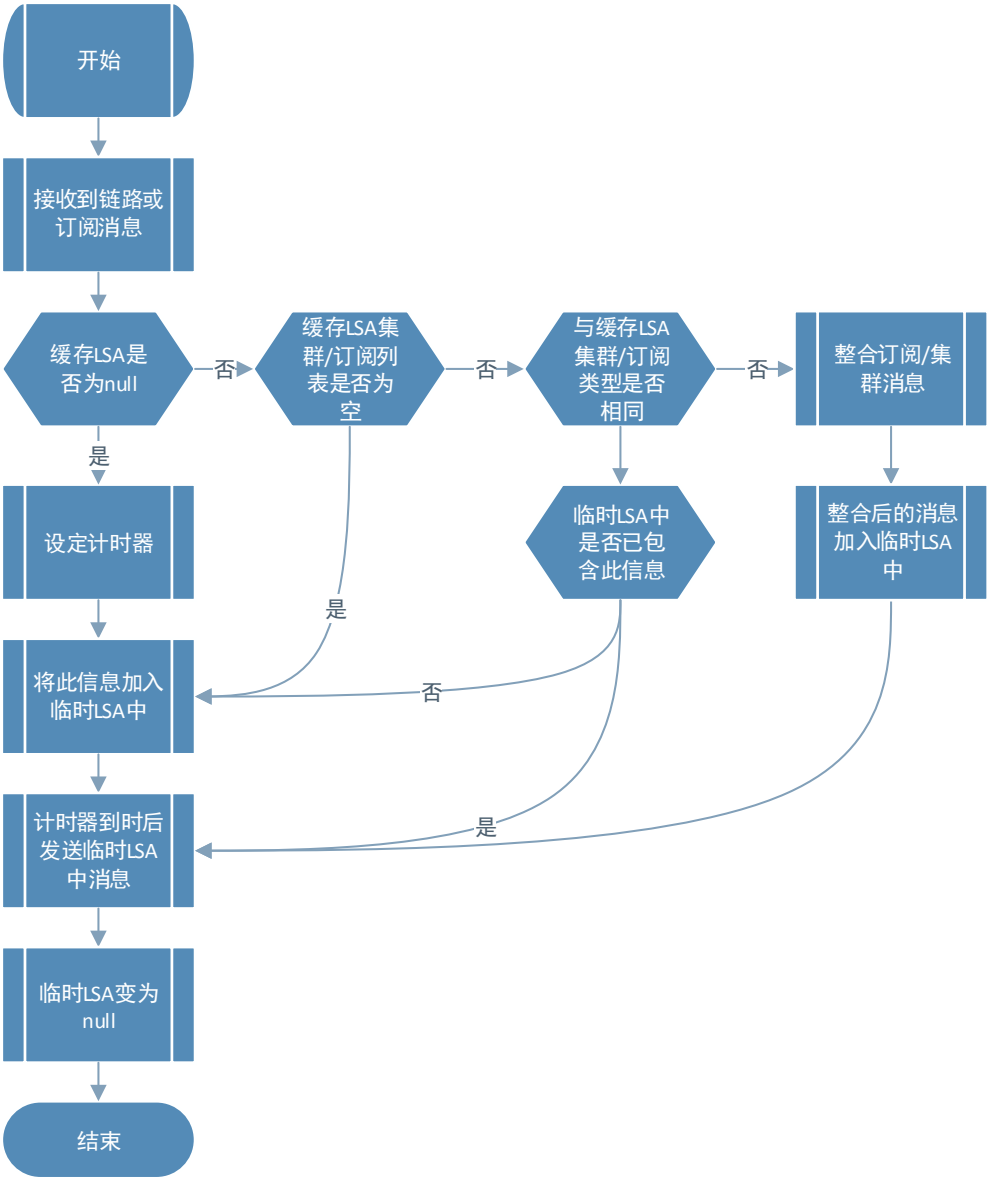

图 5-2 LSA 发送过程

LSA 的传播通过邻居进行洪泛，每个集群会对新收到的 LSA 进行有效性判

断，对于新 LSA 进行相应的操作和转发，并忽略过时消息。当有链路状态或订阅消息需要更新时，会经过如图 5-2 所示的过程。在消息发送的时候设置缓存 LSA，收集一段时间内的更新信息后一起发送，每次发送缓存 LSA 时均在之前已发送的 LSA 序列号上加一作为新的序列号。缓存 LSA 在发送完毕后再次置为空，以供下次更新消息的转发。消息的合并包括丢失邻居和新增邻居的更新、订阅和取消订阅的更新，对于缓存 LSA 的更新需要设置同步机制，使其串行进行，对消息的合并以新消息为准。

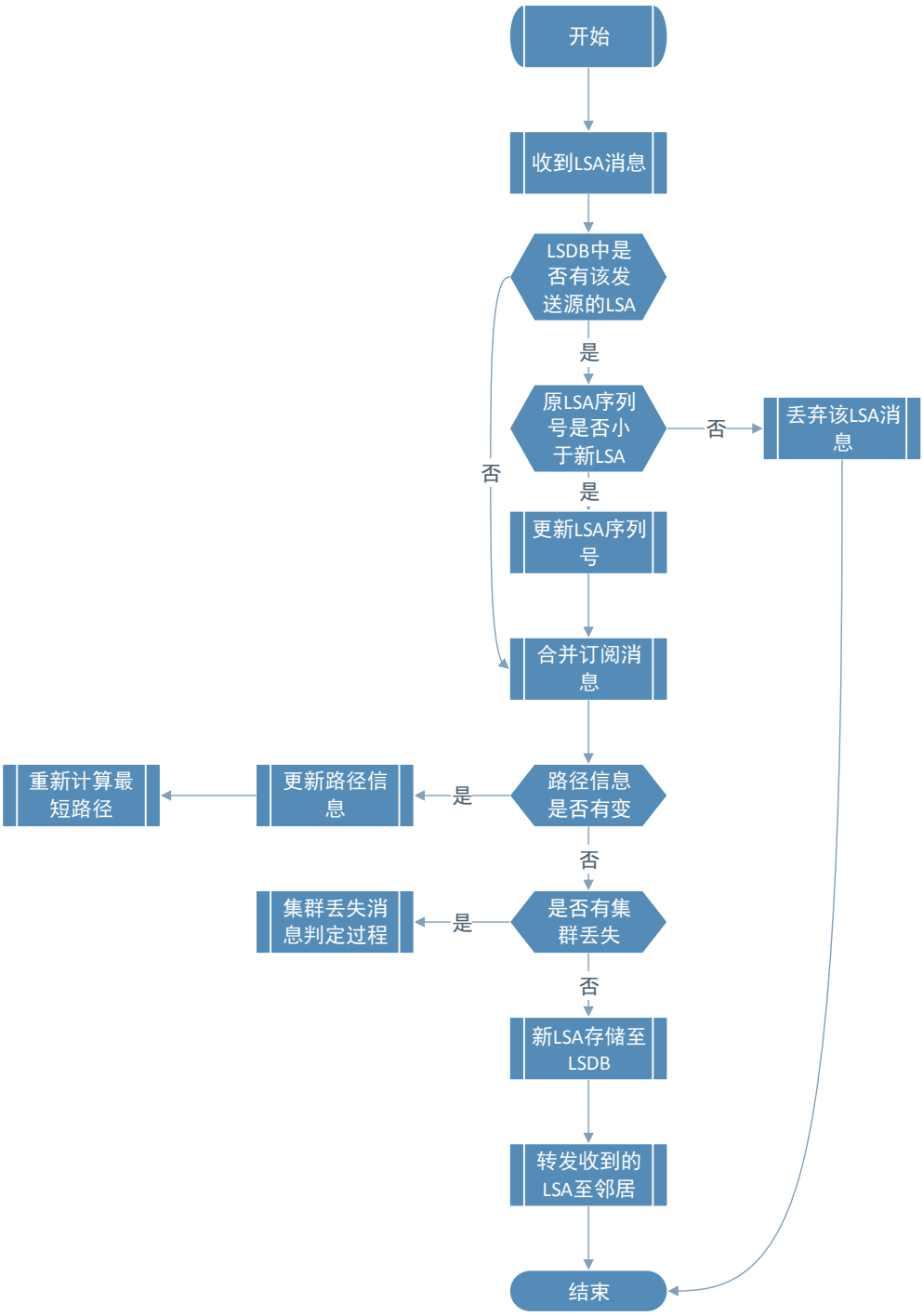

图 5-3 LSA 接收过程

图 5-3 所示为 LSA 的接收过程，当其他集群接收到一条 LSA 时会通过查找自身的 LSDB 判断是否有已经大于或等于来自该集群的 LSA，若该 LSA 为过期消息则直接丢弃，否则即进行进一步处理。与 LSA 分为同步与非同步链路状态广播，对于同步类的消息，代理服务器会删除原有记录，将同步消息重新存储；对于非同步类消息，代理服务器则会根据其具体的携带内容进行相应操作。若该消息传递失效的邻居信息不为空，则说明该集群与一个或多个邻居的链路失效，在更新该集群的 LSA 与邻居距离信息的同时也要查找其邻居的 LSA 将该集群从邻居列表中删除。订阅和取消订阅主题也通过 LSA 进行传播，若其为订阅消息，则与原有订阅相对比，取最大父类存储；若为取消订阅消息，则检查原有订阅中是否有该类或该类子类的主题并对之执行删除操作。

对于订阅的管理，代表和代理的处理机制是一样的，区别是代表会在订阅更新完成之后对于更新订阅的路由树进行重新计算，而代理只须进行订阅的存储即可。在存储时，订阅会根据其主题在主题树中的位置进行合并更新。如新增订阅时，若新增订阅为已存储某一订阅主题的子树节点，则无须更改订阅；若新增订阅为已存储某些订阅主题的父节点，则须将之前的子树中订阅删除，并将新订阅加入存储。在取消订阅时类似，对父类订阅取消时会删除原有所有的子树中订阅，对子类订阅取消时则无须更新订阅表，只须在 LSA 发送源处添加拒收主题，即当收到该类主题时拒绝向上层提交。

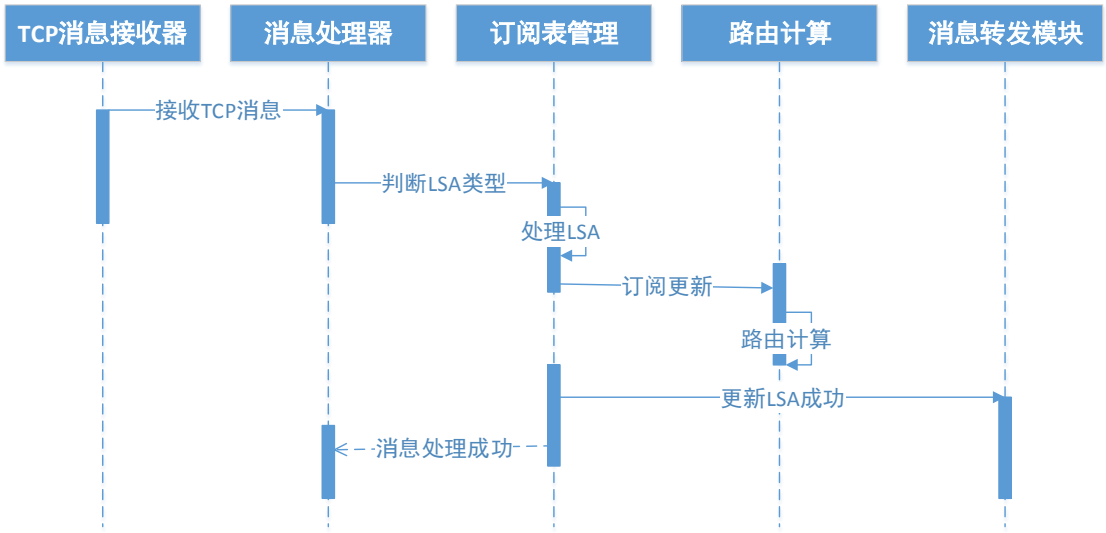

图 5-4 LSA 订阅处理时序图

如图 5-4 所示，代表对于 LSA 消息中订阅的处理过程为代表的 TCP 消息接收器接收到外界发送的消息后会转给消息处理器对其进行类型的判断，判断其为 LSA 类型后转给订阅表的管理者进行 LSA 消息的处理。订阅表管理者首先判断该 LSA 的序列号是否为新的，若不是则直接忽略，若是则继续订阅表的更新操

作。订阅表的订阅主题按照上述规则更新，若有订阅改变时通知路由计算模块进行相应的路由计算，路由计算模块在计算出路由树后会在自身模块中返回和存储。当 LSA 序列号更新成功后，订阅表管理模块会将此消息转给消息转发模块，由其将此新 LSA 转发给所有的本集群代理和本集群的邻居代表处。处理完成之后订阅表管理模块将消息处理成功的信息返回。

5.3. 拓扑维护的实现

在拓扑维护的实现中，包括节点的加入、删除、邻居的构建、邻居的维护等，目的是为维护统一的网络拓扑信息，给统一的路由计算提供基础条件。以下分别介绍这些功能的实现过程。

5.3.1. 节点的加入

当某代理服务器加入网络时，首先根据本地的配置信息向管理员注册，管理员返回该集群的信息，代理服务器会根据返回信息成为代表或代理，并将自身信息告知本集群其他成员或其他集群。此后开始与相邻节点的 Hello 消息维护。

如图 5-5 所示，节点在系统启动时读取本地的配置文件：configure.txt，其中包括的设置有管理员 IP、备份管理员 IP、管理员 TCP 端口号、加入试探次数、本地集群名称、本地 IP、本地网关、本地 TCP 端口号。读取到配置信息后代理便按照配置信息中的管理员 IP 和 TCP 端口号向管理员发送本集群名称请求信息，管理员根据其集群名称按照当前记录中是否有该集群的名称来返回其代表的信息，若无该集群相应信息则返回结果中代表的 IP 为空，请求的代理方自动成为本集群代表；若返回代表 IP 不为空则向已有代表发送加入请求，当该代表正常运作时会返回当前网络状况（集群信息、LSDB 等），若连接失败该请求代理会当做该集群代表已失效自己成为代表。请求代理成为代表后按照上章描述的选择邻居算法进行邻居集群的选择， 并向其发送请求建立邻居的请求，建立完成后将本集群新的代表信息发送给管理员，并发送本集群的第一个 LSA 至全网。

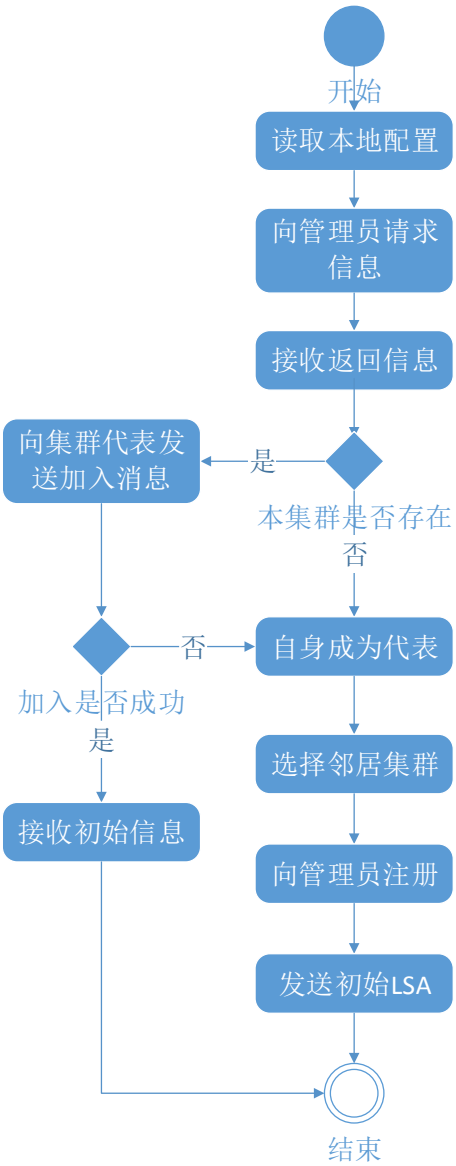

图 5-5 节点加入流程图

5.3.2. 节点的删除

节点的失效和删除分为代理的失效和代表的失效，检测服务器也分为代表检测和代理检测，由代表检测本集群代理和邻居集群代表的失效，由代理检测本集群代表的失效，这三种检测起初都是由 Hello 消息探测超时引发的，但相应的后续处理不尽相同。以下以代表的节点失效检测为例，显示节点失效时代理服务器的处理过程。

在代表的拓扑探测模块，会周期性地发送 Hello 消息给自己的邻居集群代表，并在集群内组播 Hello 消息，同时也会对每一个邻居集群代表和本集群的代理设置定时器，定时器时长为预设阈值，每一次收到新的 Hello 消息便会对该邻居的

定时器重新开始计时。当超过阈值还未收到某一邻居的 Hello 消息时便会启动节点失效处理程序。

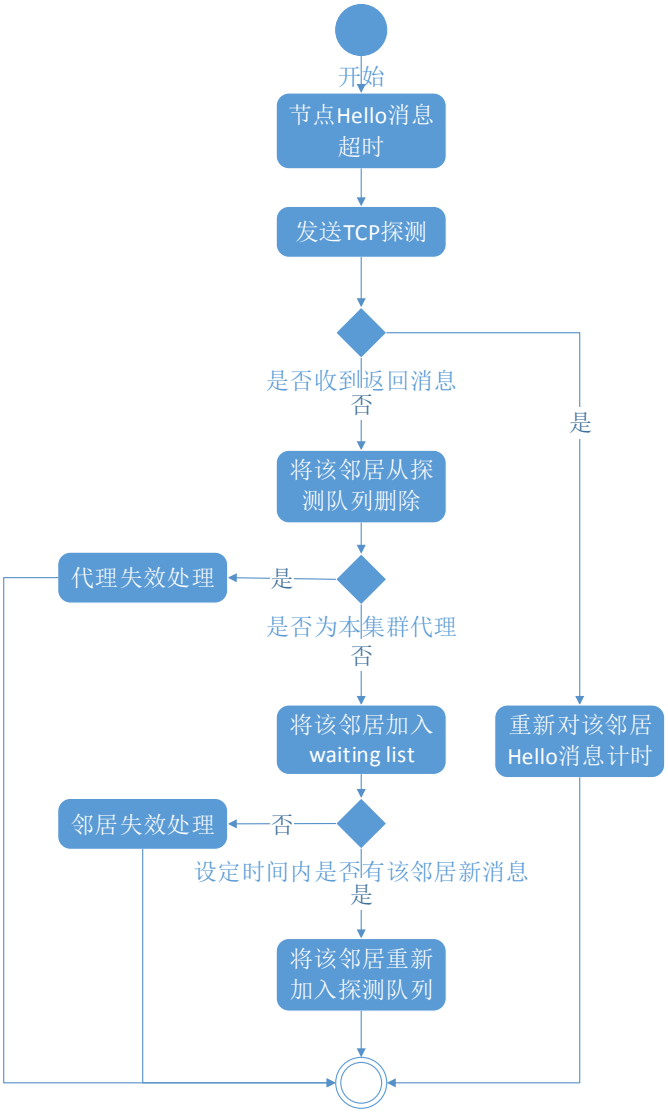

图 5-6 代表对节点失效的处理

如图 5-6 所示，拓扑探测模块在 Hello 消息超时后会先向其发送 TCP 探测消息，判断是否会收到反馈消息，若收到反馈消息则说明只是网络问题未接收到 UDP 方式发送的 Hello 消息，拓扑模块会继续对其进行探测和重新设定计时器；若未收到反馈消息则默认该邻居对于本代表来说已不可达，将其从探测队列中删除，并通知调度程序其不可达，开始失效的处理流程。若失效节点为本集群的代理，则直接认为其失效，通知集群内其他代理后删除失效代理的订阅；若失效节点为邻居集群的代表，则会先将其加入 waiting list，等待该集群重新选举代表与之建立连接。若在设定选举时间内仍未收到该集群的消息，则认为本集群与该邻居集群之间链路已经中断，无法到达，将其从邻居列表中删除后更新邻居信息并发送新拓扑 LSA 给全网，所有集群代表在收到拓扑更新 LSA 后会重新计算所有

的路由树。

5.3.3. 邻居的构建

在网络拓扑中，集群内代表和代理之间自动成为邻居关系，并互相进行拓扑探测，此处不需要进行邻居的选择，而对于集群之间的邻居构建，则需要一定的选择、加入和维护程序，其个数可以变化，最大值可以设置，最小值和默认构建值均与最大邻居数有关。默认值是当该代表新加入，没有邻居时选择的邻居个数，最小值是指当邻居个数小于此值时要开始启动邻居重新选取程序。邻居数的设定与网络拓扑的具体状况和网络中集群的个数有关，在设置时需要根据实际情况和消息发送的实验结果进行综合选择。以下先介绍邻居的构建过程。

通知事件的转发需要依赖拓扑的完整，集群之间的转发也需要知道彼此可达的链路。对于全连通的网络，可以只根据网络链路状况计算出只包含订阅集群的路由树，删除中间节点；但对于不全连通的网络则需要通知事件的传播依据已构建的网络邻居拓扑进行传播，会有一些没有订阅的集群作为中间节点进行消息的转发。邻居之间的维护不仅是有效维护订阅表，也是为路由树的构建提供拓扑结构。因此为使消息在转发时能够尽量高效，邻居之间的维护成本也须考虑，因此应综合集群之间的实际物理距离和邻居已有的邻居个数来进行选择，选择过程如图 5-7 所示。首选将所有非邻居也非本身的集群加入备选集合里，作为所有备选邻居的集合。然后判断本集群邻居的个数是否已达到预设值并且此时备选集合是否为空（最小值=最大值/3），若条件满足则可结束邻居的选择过程，若没有则继续。按照设计阶段所说的邻居选择算法对备选集合中的集群进行选择，选择的个数为 min(邻居最小值-现有邻居个数，备选集合中集群个数)，对选择的集群发送邻居建立的请求信息，若得到肯定的反馈则可以将其加入邻居列表并进行拓扑维护，对探测之后的集群就可从备选邻居的集合中删除。每次选择完成之后都会进行邻居个数是否达标的判断。

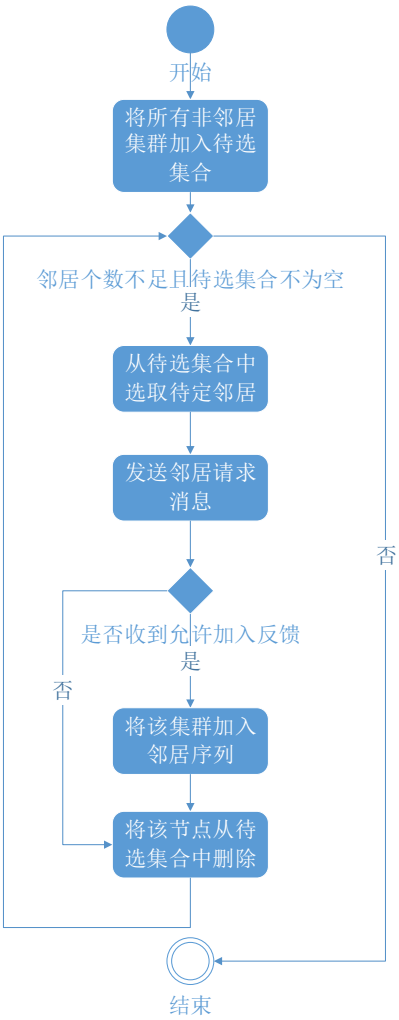

图 5-7 邻居的构建过程

5.3.4. 邻居的维护

邻居的维护分为代表对代理的维护、代表对代表的维护和代理对代表的维护，但它们所用的拓扑探测框架是一样的，只是在探测集合中加入的选项不同。对代表来说，定时器集合中需要包含所有的邻居集群和代理，对于每一个邻居集群的代表都要定期发送 UDP 消息，而对于集群内只须组播该消息；对于代理来说，要探测的节点只有本集群的代表，向其发送 UDP 消息，同时接收其发送的组播 Hello 消息。

邻居维护需要调度模块和拓扑维护模块的共同合作。拓扑维护模块需要调度模块提供基础配置（如新增维护节点、发送周期、阈值等），拓扑维护某块也需要向调度模块发送节点失效消息，其相关类的结构如图 5-8 所示。拓扑维护类继承探测接口，实现添加探测目标、移除探测目标、设置失效阈值等函数，并在本类中设置对于每个探测目标的定时器，实现节点失效时需要执行的失效函数、发送 Hello 消息函数、对超过阈值的目标进行 TCP 探测函数等，在失效时由参数

routeManager 调用 RouterManager 中的 lostTarget 函数。RouterManager 类继承 HelloMsgHandler 接口，实现节点失效处理的函数 lostTarget，并在类中初始化 DetectManager 类的实例，向其传输参数并接收其反馈消息。RouterManager 只是作为消息失效处理的中间节点，最终将消息发送给代表或代理的处理程序，即 RepState 或 RegState，在 RouterManager 中则通过声明变量 AState 并进行代表或代理的实现来进行消息的交付处理。

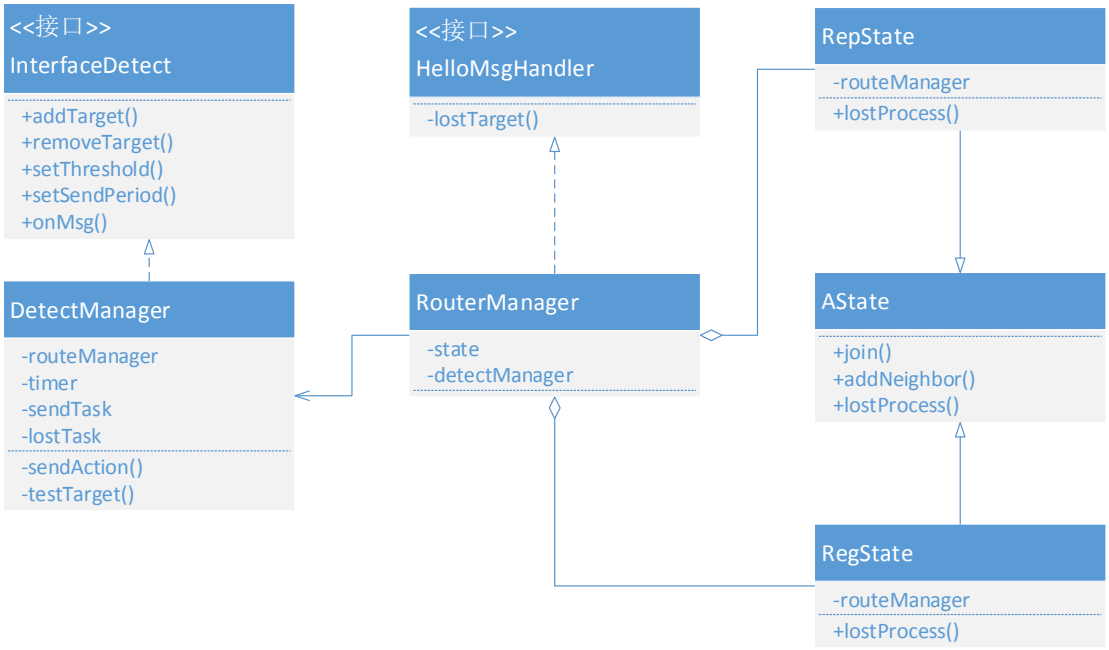

图 5-8 拓扑检测相关类图

5.4. 策略驱动的主题树聚合路由的实现

路由计算的源数据从与主题树相对应的订阅树和 LSDB 中获得，计算时针对于订阅树中的每个主题提取订阅集群放入目标集合，然后查找策略信息，去除被限制的集群，找到该主题的根节点后利用 LSDB 里的信息对路由树进行计算，当路由树中已容纳所有订阅集群时停止计算。目前的算法中以实际物理拓扑为依据，假设网络不是全连通的，事件的转发都在邻居之间进行。

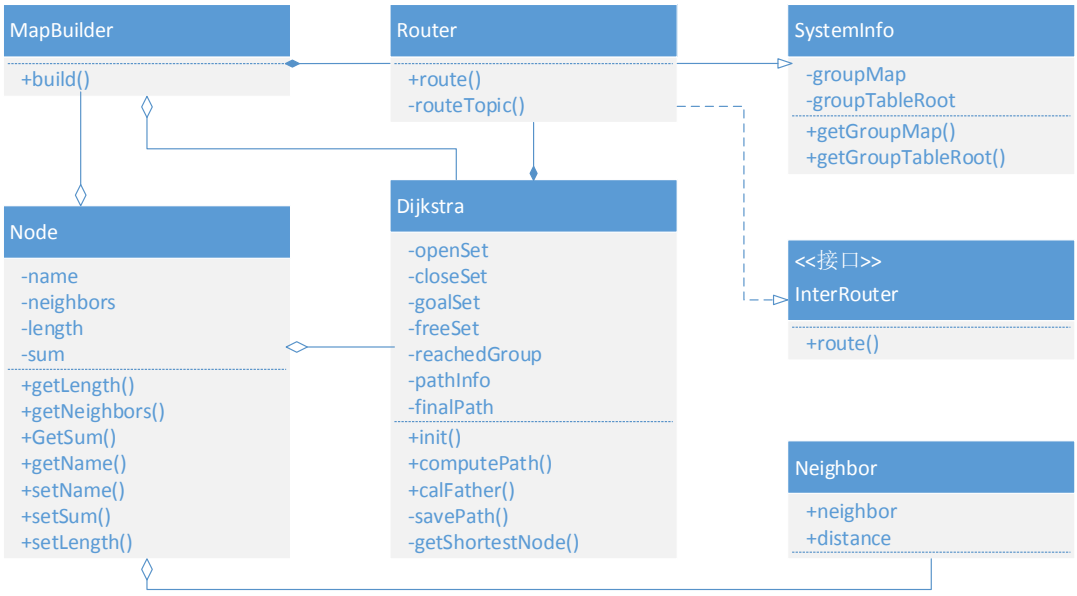

图 5-9 路由计算相关类图

图5-9为路由计算的相关类图，其中Node和Neighbor为存储网络拓扑的类，MapBuilder将所有集群排序和找出路由树的根节点，Dijkstra为核心的路由计算部分，Router为路由计算的接口部分，从SystemInfo中获取基础信息并根据调用者的请求进行路由计算和存储。

在路由计算时使用 Dijkstra 算法，该算法的核心思想是设定源点V_s，S 为已经求得的最短路径的终点集合，在此系统中即为路由树中的各节点，开始时$S = \{V_s\}$。当求出第一条最短路径（V_s，V_i）后，S 为$\{V_s, V_i\}$。以此类推，直到求出所有的目的节点的最短路径。定义一个数组 dist[n]，保存集群名称和到源点的距离的映射关系，其上一跳为 S 集合中的点。每次选择新节点进入 S 时即选择

$$\text{dist[i]} = \min\{\text{dist[k]}, V_k \in (V - S)\} \qquad [1]$$

重复上述步骤直到到达所有的目的节点。在计算时集群间的距离与他们之间的流量有关，此附加值在统计流量时就已计算入邻居之间的距离，并以 LSA 的方式发送至全网。如从集群 A 到集群 B 和从集群 B 到集群 A 发送的所有数据包为平均 200 包/秒，而两集群之间的平均带宽为 2M，则按照每包平均 300K 的长度计算，他们之间要增加的距离为 200*300*0.1/1024/2=2，其中 0.1 为加权因子，可以根据实际的发送结果和实验状况进行调整。另外在实际计算过程中，每当新加入 S 一个节点，即对其父节点进行判断，子节点是否已超出预设的最多下一跳值，若超出则将其从 S 中删除。结合以上改进，对路由算法的计算过程如下所示：

1、令$S = \{V_s\}$，$R = V - S$，R 表示待计算的节点，Q 为所有订阅该主题的集群集合，带权的邻接矩阵图 E 可从 LSDB 中获得，对图中每个节点V_i按照以下原则进行初始化：

$$dist[i] = \begin{cases} 0 & , i = s \\ D_{si} & , i \neq s, (V_s, V_i) \ni E \\ \infty & , i \neq s, (V_s, V_i) \notin E \end{cases}$$

2、选择节点V_j，使得

$$\text{dist[j]} = \min\{\text{dist[k]} | V_k \in R\}$$

3、将V_j放入 S 中，同时从 R 中删除，若$V_j \in Q$也将其从 Q 中删除。更新所有与V_j相连的节点，修改其 dist[k]，修改公式如下

$$\text{dist[k]} = \min\{\text{dist[k]}, \text{dist[j]} + D_{jk}\}, \forall V_k \in R$$

4、对于V_j的父节点V_i，计算其在 S 中的孩子数目 n，若$\text{n} \geq N_{max}$则将V_i从 S 中删除；

5、重复 2-4 步骤，直到 Q 为空。

5.5. 本章小结

本章主要介绍了在设计章节提到的各模块的具体实现。其中对于订阅表维护的方法、拓扑维护中节点的增加删除、路由算法的改进等均有描述，策略的处理方面针对具体的实施环境做了相应的设计和实现，同时分析了在各个模块中相应设计的原因和起到的作用。

第六章 发布/订阅系统在 SDN 上的移植

IP 网络与 SDN 网络相比，主要有物理设施的不同、对 OpenFlow 交换机/路由器控制的不同和信息收集的不同，以下分别就此三方面进行详细说明，同时在测试章节对本移植方案进行功能和性能上的具体实验。

6.1. 物理设施的移植

在物理设施方面，传统的 IP 网络是由普通的交换机和路由器构成的网络，在本系统中设计为SDN网络与传统网络的组合。OpenFlow网络中包括OpenFlow交换机和 Controller，其中 OpenFlow 交换机负责转发数据，Controller 负责对网络进行集中控制。OpenFlow 交换机在接收到数据包后，会经过本地流表匹配和转发的阶段，在此过程中若流表项匹配失败则将该数据包发送至 Controller，由 Controller 决定下一步动作。

本系统中的 OpenFlow 交换机遵从 OpenFlow 1.0 协议，同时也具有普通交换机的功能。在应用中控制器可以根据消息的优先级选择其相应的发送队列，控制发送速度。在拓扑方面，由于所有同集群的代表和代理都与一 OpenFlow 交换机直连，因此当一个节点整体失效时可先于上述传统 Hello 探测知道，进一步提高了拓扑维护的时效性。

OpenFlow 交换机中维护一个 FlowTable，并且只按照 FlowTable 进行转发，Controller 负责向 FlowTable 执行生成、维护、添加、删除等操作，完成控制和转发的分离。FlowTable 的操作主要由数据转发阶段来完成，路由阶段需要向数据转发阶段提供相应的下一跳信息，帮助其完成高效的数据转发。

FlowTable 由很多流表项组成，每项对应一个转发规则。当交换机接收数据包时通过查询 FlowTable 中的每一项规则来获取转发的目的端口。流表项包括头域、计数器和操作，定义了头域匹配的数据包该执行的操作。

OpenFlow 网络由多个 OpenFlow Island 构成，各 OpenFlow Island 之间可能有非 OpenFlow 网络，OpenFlow Island 和非 OpenFlow Island 之间不能形成回路，一种典型的连接如图 6-1 所示。在本系统中一个集群即为一个 OpenFlow Island。

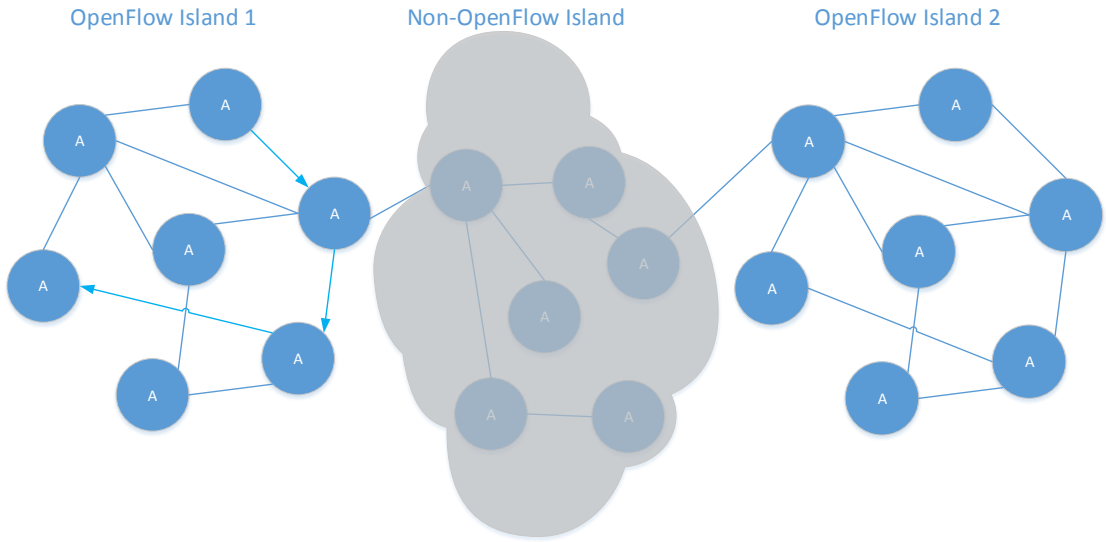

图 6-1 OpenFlow Islands

在实验阶段，我们在一个集群内仅使用一台 OpenFlow 交换机与外界相连，所有的代表和代理与该 OpenFlow 交换机直连，代表是 OpenFlow 交换机的主控制器，对其执行信息采集、转发控制等命令。如图 6-2 所示，该拓扑结构由四个集群组成，每个集群由一个交换机与外界相连，集群之间的传统网络对各集群透明。与集群相连的 OpenFlow 路由器为方便局域网内 IP 的识别和消息传播降级为交换机使用，在今后的拓展中可以将路由器引入网络。

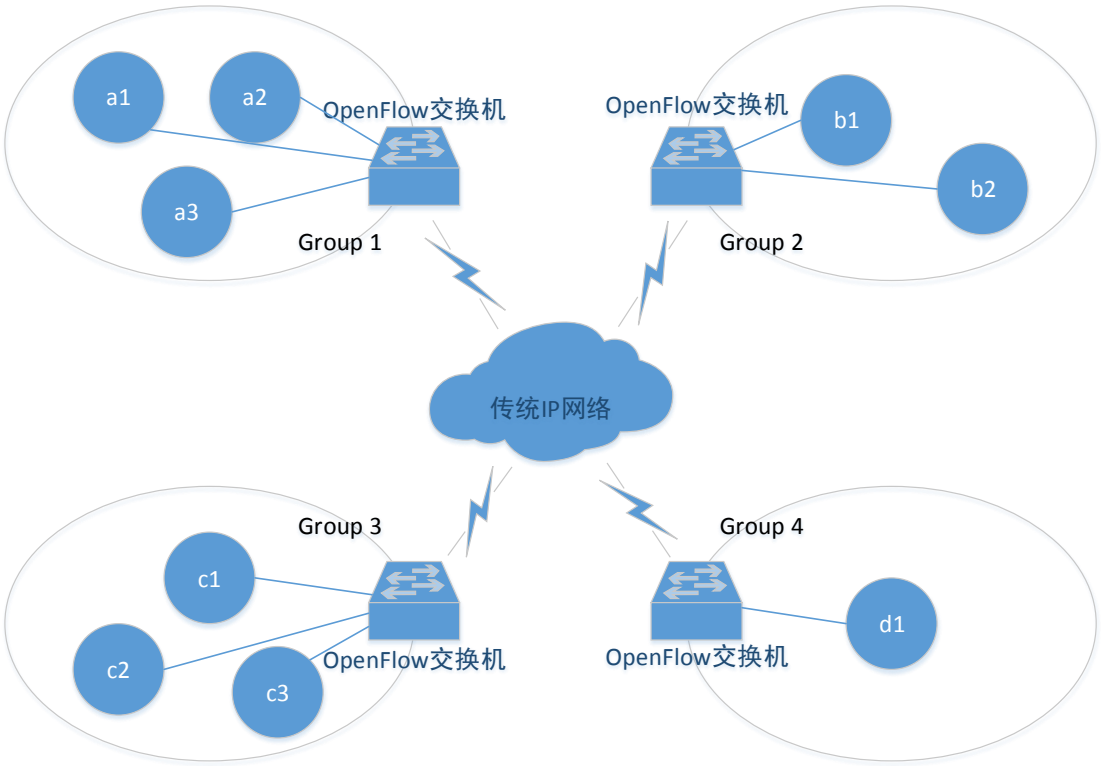

图 6-2 SDN 网络上的发布/订阅拓扑

OpenFlow 交换机分为专用交换机和支持 OpenFlow 的交换机。其中专用 OpenFlow 交换机不支持现用商用交换机的处理流程，所有数据包都需要基于 OpenFlow 协议进行匹配和转发；支持 OpenFlow 的交换机是只在现有商业交换机的基础上添加支持 OpenFlow 协议的操作来获得的具有 OpenFlow 匹配功能的交换机，可以在支持商业交换机的传统转发规则的同时支持 OpenFlow 转发逻辑。本实验中因为需要与传统网络直连，且在转发过程中除发布事件的消息并不多，因此尚未进行 SDN 移植，因此需要对传统转发机制的支持，所以用到的为支持 OpenFlow 的交换机。

6.2. 对 OpenFlow 交换机的控制

我们采用开源软件 floodlight 作为 OpenFlow 交换机的控制器，floodlight 为运行在 Linux 系统的用 Java 书写的开源代码，可以支持我们的物理设施——基于 OpenFlow 1.0 协议的交换机。

在对 SDN 的移植过程中，我们并未将全部消息的发送都按照 OpenFlow 的模式进行，仅事件的发布规则通过 OpenFlow 模式进行，其他消息类型仍按照传统网络的发送规则。事件发布时根据事件的优先级将其放入不同队列，按照不同速度进行发送，而发送的下一跳需要由路由表查找得出。

Floodlight 作为本系统中 OpenFlow 交换机的 Controller，可以

1、发现网络状态和事件（拓扑、设备、流量等）；

2、控制网络交换机通信（基于 OpenFlow 1.0 的流表规则）；

3、提供 web 界面和 debug 服务器。

其中网络拓扑事件可以给拓扑维护提供帮助，网络流量的监测可以给路由的计算提供参考，流表的添加功能可以在数据转发阶段设置对事件的不同处理，web 界面可以直观地看到与 OpenFlow 交换机直连的所有设备信息。

在 floodlight 启动时将发布/订阅系统作为其一个子模块启动，在 floodlight 中按照官方指南添加 module，这样在 floodlight 启动时就可以同时并行启动发布/订阅系统。以下为添加发布/订阅系统的具体步骤：

1、新建名为 WSNStart 的类继承 IOFMessageListener 和 IfloodlightModule；

2、声明变量 floodlightProvider、macAddresses 和 logger：

```
protected IFloodlightProviderService floodlightProvider;
protected Set<Long> macAddresses;
protected static Logger logger;
```

3、实现 getModuleDependencies 方法，把依赖关系告诉模块加载系统；

```
@Override
```

```
public Collection<Class<? extends IFloodlightService>>
getModuleDependencies() {
    Collection<Class<? extends IFloodlightService>> l =
        new ArrayList<Class<? extends IFloodlightService>>();
    l.add(IFloodlightProviderService.class);
    return l;
}
```

4、实现 init 方法，在 floodlight 初始启动时运行加载依赖和数据结构，同时将发布/订阅系统的启动入口加入其中；

```
@Override
public void init(FloodlightModuleContext context) throws
FloodlightModuleException {
    floodlightProvider =
    context.getServiceImpl(IFloodlightProviderService.class);
    macAddresses = new ConcurrentSkipListSet<Long>();
    logger = LoggerFactory.getLogger(WSNStart.class);
    net.floodlightnet.floodlightcontroller.wsn.Start start = new
    net.floodlightnet.floodlightcontroller.wsn.Start();
}
```

5、实现 startUp 方法，订阅接收 PACKET_IN 消息，实现基本的监听程序，在此之前其他依赖模块都已初始化完毕；

```
@Override
public void startUp(FloodlightModuleContext context) {
    floodlightProvider.addOFMessageListener(OFType.PACKET_IN, this);
 }
```

6、为 OFMessage 监听器添加 ID，并通过 getName()返回；

```
@Override
public String getName() {
    return WSNStart.class.getSimpleName();
}
```

7、定义 PACKET_IN 消息的处理方法，在该方法结束时返回 Command.CONTINUE 以使其他模块也可以对该消息进行处理；

```
@Override
public net.floodlightcontroller.core.IListener.Command receive(IOFSwitch
sw, OFMessage msg, FloodlightContext cntx) {
```

```
        Ethernet eth = IFloodlightProviderService.bcStore.get(cntx,
    IFloodlightProviderService.CONTEXT_PI_PAYLOAD);
        Long sourceMACHash = eth.getSourceMACAddress().getLong();
        if (!macAddresses.contains(sourceMACHash)) {
            macAddresses.add(sourceMACHash);
            logger.info("MAC Address: {} seen on switch: {}",
            eth.getSourceMACAddress().toString(),
            sw.getId().toString());
        }
        return Command.CONTINUE;
    }
```

8、向 src/main/resources/META-INF/services/net.floodlight.core.module.Ifloo-dlightModule 中添加创建的类的路径：net.floodlightcontroller.wsn.WSNS-tart，注册新加模块；

9、在默认配置文件 src/main/resources/floodlightdefault.properties 中加入新增类的路径，使其作为默认启动模块启动。

经过上述步骤后已成功将发布/订阅系统融入 floodlight 中，两者同时运行。同时发布/订阅模块可以通过 floodlight 的相应功能接口与其信息进行交互，完成系统功能。

Floodlight 在工作时有多种参数可以配置，如 TCP 端口、OpenFlow 交换机地址、REST API 监听端口等，具体可见表 6-1，可以在配置文件中通过类似以下标示名进行相应的配置，默认情况下监听的 TCP 端口号为 6633。

表 6-1 floodlight 配置

net.floodlightcontroller.restserver.RestApiServer.host
net.floodlightcontroller.restserver.RestApiServer.port
net.floodlightcontroller.core.internal.FloodlightProvider.openflowhost
net.floodlightcontroller.core.internal.FloodlightProvider.openflowport
net.floodlightcontroller.jython.JythonDebugInterface.host
net.floodlightcontroller.jython.JythonDebugInterface.port

可以在控制端远程登录 OpenFlow 交换机，添加网桥和控制器 IP 进行控制识别，同时可以对 OpenFlow 路由器进行降级设置，使其成为交换机，方便实验的进行。

通过对 OpenFlow 交换机的配置可以进行代表和代理对其主控制权的切换，同步监听端口，在其中添加原发布/订阅系统的启动接口，使他们并行运行。

6.3. 拓扑和流量信息的收集应用

在传统网络中，拓扑信息依靠 Hello 消息来进行维护，平均代理有 2 分钟的失效判定时延，代表有 2.5 分钟的失效判定时延，在失效判定时还需要 TCP 探测和定时器等待等相关措施，此维护方法在 SDN 网络中仍然有效，但同时加入对 OpenFlow 交换机相连的设备信息收集，可以先于 Hello 消息更准确地判定一个节点的丢失。OpenFlow 交换机可以通过控制器显示所有与其直连的设备的信息，若某设备丢失则该设备上的发布/订阅程序也应失效，但发布/订阅程序的失效并不总等同于设备的失效，因此应两种拓扑维护方式相互配合使用。

在拓扑管理模块（TopologyManager）启动后会随时监听拓扑更新消息，由于代表和代理都是与 OpenFlow 交换机直连，因此当代理失效时很容易由拓扑管理模块根据链路更新获得。链路更新消息首先存储在 LDUpdates 队列中，由拓扑管理模块从中依此取出，判断其类型，进行相应处理。发布/订阅系统在此时即可根据链路更新类型判断是否为集群代理丢失消息即可。如图 6-3 所示为 floodlight 探测失效链路的时序图，失效信息会传送至发布/订阅系统的拓扑维护模块，拓扑/维护模块通过对失效目的节点的 IP 进行判断是否为本系统内的代表和代理，若是的话就会按照拓扑模块自己探测失效的。

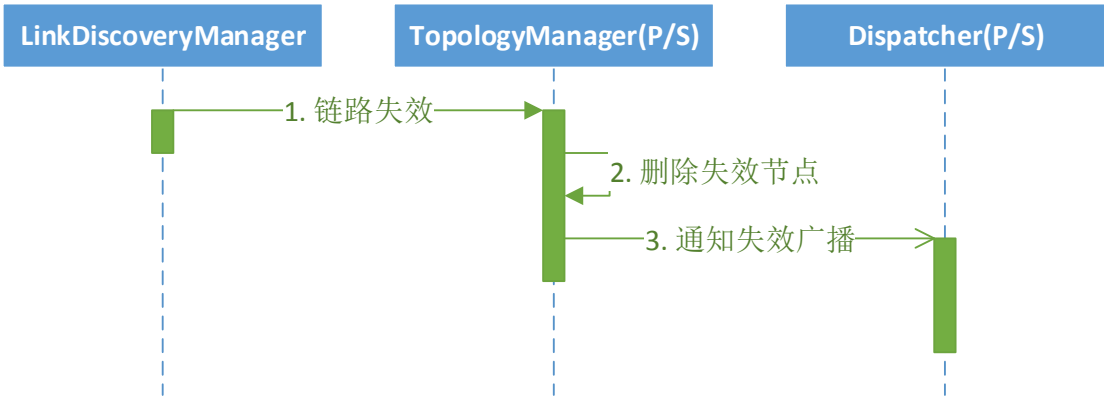

图 6-3 floodlight 探测链路失效时序图

Floodlight 提供 REST API 给上层接口，可以通过一定的命令来获取所需消息，包括每个端口的流量信息、物理连接信息等。这些获取命令以 URI 的形式给出，可以通过在终端中输入命令、在程序中嵌入命令和在浏览器中输入 URI 的方式获得所需信息。如表 6-2 所示为部分的 URI 及其对应的功能。

表 6-2 部分 REST API 命令

URI	方法	描述	参数
/wm/core/switch/all/<statType>/json	GET	获取所有交换机的状态集	**statType**: port, queue, flow, aggregate, desc, table, features
/wm/core/switch/<switchId>/<statType>/json	GET	获取每个交换机状态	**switchId**: Valid Switch DPID (XX:XX:XX:XX:XX:XX:XX:XX) **statType**: port, queue, flow, aggregate, desc, table, features
/wm/topology/links/json	GET	展示由 LLDP 发现的所有直连链路	none

在传统的 IP 网络上，通过数据发送阶段的统计来获取两个节点之间的流量，并将此流量赋以权值加到两点之间的距离上，以供路由计算。但这种流量统计并不完全，还有很多除发布事件的之外消息也会构成两个节点之间的流量，因此在 SDN 中 OpenFlow 交换机获取的流量信息显得尤为重要。如图 6-4 所示，应用 REST API 周期性获取交换机的流量信息，通过运算作用于拓扑信息，判定与之前的拓扑是否有改变，若有改变则还需发送 LSA 通知全网，同时运行路由计算模块，进行所有转发路径的重新计算，将结果存储至路由树。

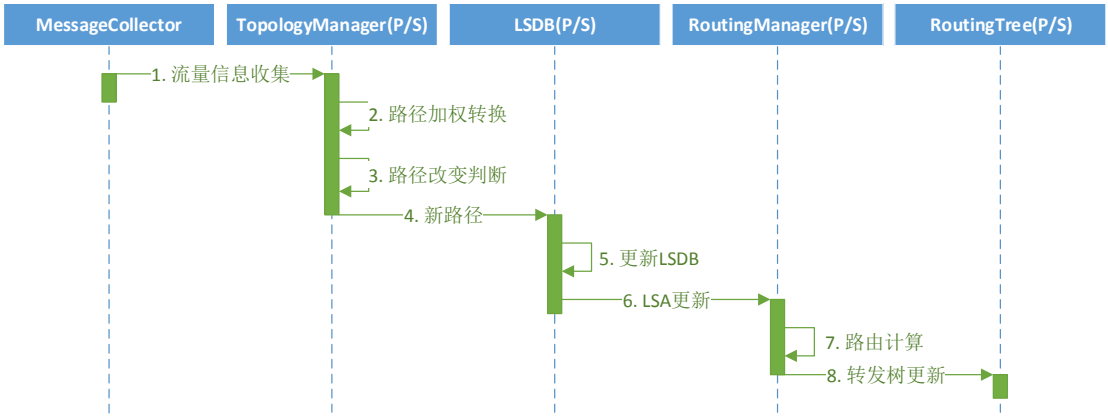

图 6-4 流量影响路由时序图

在 SDN 网络中，在 Controller 端可以通过表 6-2 所示的 URI 获得当前时间在 OpenFlow 交换机的每个端口的流量信息，在实际的路由计算中，可以将此流量信息加到所有当前与该节点相连的邻居距离中，定期判断是否需要更新连接距离，间接影响事件发布时的传播路径。

6.4. 本章小结

本章主要从物理设施、对 OpenFlow 交换机的控制、拓扑及流量信息的收集三个方面阐述了发布/订阅系统在 SDN 上关于拓扑和路由方面的移植部分。在阐述的过程中详细说明了移植的原因、方法和作用，说明在 SDN 上移植发布/订阅系统的研究价值和意义。

第七章 系统测试和验证

本系统的测试主要分为功能测试和性能测试部分，主要对于拓扑维护、路由计算以及路由效率进行测试。

7.1. 测试目标

为了保证基于 SDN 的发布/订阅系统的需求都能得到正常的满足，我们对系统的各个功能模块进行了单元测试和集成测试。本研究主要针对其中的拓扑和路由方面进行相应测试。在此基础上，基于 SDN 的发布/订阅系统验证了新型网络中发布/订阅的有效性。功能测试的对象是在设计和实现部分列出的各功能模块，即验证调度模块对不同消息的处理功能、订阅表构建的正确性、拓扑维护的有效性、路由计算的合理性。

7.2. 测试环境

我们使用的测试环境为普通 IP 网络和 SDN 网络混合的拓扑结构。使用 40 台虚拟机来模拟普通的 IP 网络，再与 OpenFlow 交换机相连，模拟两个通过 OpenFlow 交换机与外界相连的集群。虚拟机的配置指标部分如下所示：

- CPU：Intel® Xeon® CPU E5640，主频 3.00GHz+2.99GHz；
- 内存：2GB；
- 软件平台：JDK1.7；
- 操作系统：Windows xp。

两个与 OpenFlow 交换机相连的集群代表与上述配置基本一致，只是操作系统为 Windows 7。系统部分的测试 IP 和集群分布如表 7-1 所示。表中所示集群在测试过程中保持其配置属性，只有代表和代理的属性在其加入和退出的过程中可能会有所变化，邻居之间的关联也会有所变化。

表 7-1 测试系统概况

集群名称	IP 地址	虚拟机/实体机	代理/代表
G1	10.109.253.15	虚拟机	代表
	10.109.253.16	虚拟机	代理
	10.109.253.17	虚拟机	代理
G2	10.109.253.18	虚拟机	代表
	10.109.253.19	虚拟机	代理
G3	10.109.253.20	虚拟机	代表
	10.109.253.21	虚拟机	代理
G4	10.109.253.22	虚拟机	代表
G5	10.109.253.23	虚拟机	代表
	10.109.253.24	虚拟机	代理
	10.109.253.25	虚拟机	代理
G6	10.109.253.26	虚拟机	代表
	10.109.253.27	虚拟机	代理
	10.109.253.28	虚拟机	代理
	10.109.253.29	虚拟机	代理
G7	10.109.253.30	虚拟机	代表
	10.109.253.31	虚拟机	代理
	10.109.253.32	虚拟机	代理
G8	10.109.253.33	虚拟机	代表
	10.109.253.34	虚拟机	代理
G9	192.168.1.151	实体机	代表
	192.168.1.152	实体机	代理
G10	192.168.1.235	实体机	代表

7.3. 系统功能测试

本小节主要描述系统中拓扑和路由部分的功能测试，预期的测试结果应与设计和实现中描述的一致。

7.3.1. 调度模块测试

调度模块主要是集群的加入、集群之间和集群内拓扑的维护。

1）配置功能测试

测试项目：新集群配置功能测试。

测试目的：验证新集群产生也就是某个代理作为代表新加入时，能否正确配置。

测试步骤：启动管理员程序，一台机器即一个集群产生时是否出错，多台机器多个集群同时启动时配置是否出错。可以通过管理员来查询其配置信息是否同步和准确

预期结果：一台机器配置过程不报错，同时从管理员方位可以查询到其代理地址、TCP 端口号、组播号、子节点数、组播端口号等信息。这些信息与代理处配置一致；多台机器配置过程不报错，同时从管理员方位可以查询到各个集群代理相互独立工作，可单独查询某个集群代理地址、TCP 端口号、组播号、子节点数、组播端口号等信息。这些信息与集群代理处配置一致。

实际结果：与预期相同，在管理员处查询到的集群信息如图 7-1 和图 7-2 所示，可以看出其 TCP 端口号、UDP 端口号、组播地址等与管理员处的配置相同，代表的 IP 也已更换为新注册的代表 IP。对于多个代表加入的情况也与之类似，可以在管理员处查看每个集群的信息。

图 7-1 管理员查询集群信息

图 7-2 管理员处配置文件

2）集群内配置功能测试

测试项目：新代理的配置和集群内代表和代理连接情况的测试。

测试目的：验证在某个已经存在的集群中加入一个新代理，能否正确配置；是否保持一个集群中只有一个代表的状态。

测试步骤：集群 G9 中已经存在一个代表，此时启动另外一个相同集群的程序，看是否会成为代理，配置过程是否会出错。通过管理员来查询加入后的信息。

预期结果：新代理加入配置过程中不报错，同时从管理员处可以查询到其代理地址、TCP 端口号、组播地址等信息，这些信息与代表处一致。新代理可与代表之间维护 Hello 消息。

实际结果：与预期相同。管理员控制台处显示收到代理 192.168.1.152 的配置请求，集群 G9 的代表 192.168.1.151 控制台处收到来自代理 192.168.1.152 的请求加入消息，此后代表开始收到来自代理的 Hello 消息，代理也可以收到来自代表的组播 Hello 消息。从管理员处可以查询到当前 G9 的成员。

3）订阅提交功能检测

测试项目：测试本地订阅能否正确提交给发布/订阅处理系统。

测试目的：验证代理/代表所连接的本地客户端提供的订阅信息能够正确被接收并对其作出相应处理。

测试步骤：集群 G9 的代表 192.168.1.151，在本地客户端发布订阅消息 all:alarm；集群 G10 的代表 192.168.1.235 发布关于 all:alarm 的消息。

预期结果：集群 G9 的代表接收到来自 G10 的发布消息，同时在其管理下的订阅客户端处也显示收到发布消息。

实际结果：与预期相同。G9 代表的本地客户端在发送订阅消息后代表

192.168.1.151 过十秒后开始发送新的集群 LSA，并对 all:alarm 主题进行路由计算；G10 接收到来自 G9 的 LSA 后也重新计算 all:alarm 路由树；当 G10 代表发布关于 all:alarm 主题的消息时 G9 的代表和其订阅客户端可以接收到该消息。

7.3.2. 拓扑维护测试

拓扑维护的测试主要包括居的建立和维护、代理和代表丢失时对网络拓扑影响的测试。

1） 邻居构建功能测试

测试项目：检测集群代表之间邻居的构建。

测试目的：检测网络中已有集群时新加入的集群代表能否按照规定的规则进行邻居的选择和构建。

测试步骤：设置集群最多构建邻居数为 4，在启动管理员后依此启动集群 G1、G2、G3、G4、G5、G6、G7、G8，查看每个集群在此过程中构建的邻居。

预期结果：在集群启动的过程中会根据邻居选择的规则（如实现阶段描述）进行邻居的选择和构建，在初始启动时选择 2 个，最终最多构建 4 个。

实际结果：与预期相同。G1 首先启动，G2 启动后向 G1 发送邻居构建请求建立邻居成功，G3 启动后向 G1 和 G2 发送构建邻居请求并成功建立邻居，G4 启动后向 G2、G3 发送构建邻居请求并构建邻居成功，G5 启动后向 G3、G4 发送构建邻居请求并构建邻居成功，G6 启动后向 G4、G5 发送构建邻居请求并成功建立邻居，G7 启动后向 G5、G6 发送构建邻居请求并成功建立邻居，G8 启动后向 G6、G7 发送构建邻居请求并成功建立邻居。其构建完成后的拓扑如图 7-3 所示。

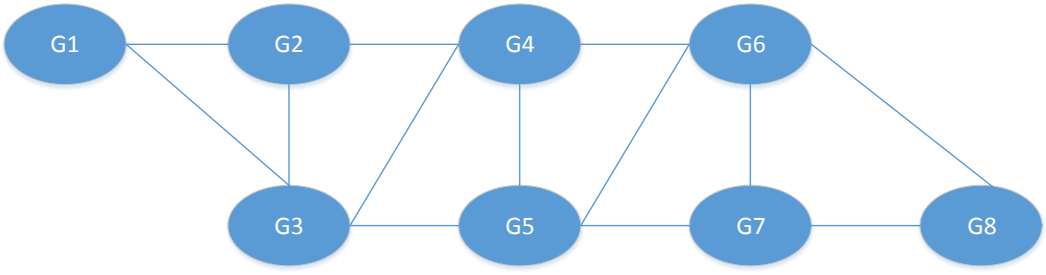

图 7-3 邻居构建拓扑图

2） 代理失效功能检测

测试项目：集群代表对本集群内代理失效的检测和处理。

测试目的：验证代理失效后代表能够检测到并对其进行相应的处理（集群内通知和订阅的处理）。

测试步骤：对于集群 G9 已有的代表和代理，代理发送订阅 all:alarm，代表无订阅。在正常维护 Hello 消息的过程中关闭代理程序，查看集群代表是否在控制台处打印其丢失消息，并将取消订阅 all:alarm 的消息以 LSA 的方式发送给其他集群。

预期结果：代表 192.168.1.151 可以在 2 分钟后检测到代理 192.168.1.152 失效，并在等待若干时间后发送新的集群 LSA 给其他集群代表，同时会重新计算 all:alarm 对应的路由树，G10 的代表 192.168.1.253 收到来自 G9 的新 LSA，然后重新计算 all:alarm 的路由树。

实际结果：与预期相同。集群 G9 的代表在 2 分钟后检测到代理不可达，在发送新 LSA 的同时重新计算 all:alarm 的路由树，集群 G10 的代表收到该 LSA 后也重新计算 all:alarm 路由树。从管理员处查询 G9 即可看到该集群已无代理信息。

3） 邻居维护功能测试

测试项目：测试已经构建好的邻居之间维护的稳定性。

测试目的：验证系统可以在各代表和代理都有效工作的情况下能够保持稳定的拓扑。

测试步骤：在测试用例 1 的基础上进行各代表和代理控制台的信息打印，看随时间的变化集群之间的邻居关系和集群内的拓扑关系是否会改变。

预期结果：邻居集群之间和集群内代表和代理之间的拓扑维护关系不会因时间而改变。

实际结果：与预期相同。在八个集群建立好邻居之后，发送订阅和发布消息，在分别等待半小时、1 小时、10 小时、24 小时、48 小时后，集群之间邻居关系和集群内代表和代理之间仍然有有效的 Hello 消息发送，与拓扑刚建立时的关系一致。

7.3.3. 路由转发测试

路由转发的测试包括路由计算的测试、路由表查找的测试、消息转发的测试和策略信息影响的测试。以下简要列出相应的几个测试用例。

1） 路由计算的测试

测试项目：系统中各代表对同一主题的路由计算。

测试目的：验证系统中不同集群代表对同一主题的路由树计算的一致性、有效性和合理性。

测试步骤：按照表 5-1 所示集群组成依此启动各个代理和代表，全部集群订阅 all:alarm，G1、G3、G5、G7、G9 订阅 all:notify，在启动完所有订阅进程

后观察各集群代表控制台处打印信息，记载的下一跳信息和根节点信息，将所有集群代表的计算结果综合对比。

预期结果：在完成订阅后所有集群代表计算出的每个主题的路由根节点分别一致，根据各代表计算信息，all:alarm 的路由树为一棵包含所有集群的转发树，all:notify 的路由树为一棵包含 G1、G3、G5、G7、G9 的转发树。

实际结果：与预期相同。所有集群计算出来的 all:alarm 路由树的根节点相同，all:notify 的根节点相同。根据所有集群计算出的下一跳信息可以构建出如图 5-3 和 5-4 所示的路由树。

2） 路由表查找的测试

测试项目：在发布事件的过程中，各个集群代表对路由表的查找。

测试目的：验证发布事件过程中，各集群代表可以根据该事件对应的主题路由树查找到正确的下一跳。

测试步骤：以路由测试用例中描述的集群启动过程和订阅完成后，分别由 G1 发布关于 all:alarm 主题的消息，由 G2 发布关于 all:notify 主题的消息。在事件转发时可以从各代表的控制台处看到打印转发下一跳的消息，比对之前构建的路由树，其中发布源的下一跳还应包括路由树的根节点。

预期结果：转发阶段查询到的下一跳与路由计算阶段构建的路由树一致。

实际结果：与预期相同。

3） 路由转发的测试

测试项目：系统对主题发布事件的选择性转发。

测试目的：验证系统可以将发布事件正确地转发到所有订阅者处，同时无环路形成。

测试步骤：按照表 5-1 所示集群组成依次启动各个代理和代表，全部集群订阅 all:alarm，G1、G3、G5、G7、G9 订阅 all:notify，然后由 G1 和 G2 分别发送 all:alarm 和 all:notify 主题的通知事件，观察各个代表和代理收到的消息。

预期结果：所有集群均可收到 G1 发送的通知事件，G1、G3、G5、G7、G9 集群的客户端可以收到 all:notify 的发布事件，且当发布事件关闭后，所有集群也在延时若干时间后停止接收消息。

实际结果：与预期相同。当集群 G1 开始发布 all:alarm 之后所有集群依此开始接收此类消息，当 G1 停止发送后，所有集群在不同延时之后开始停止接收消息；当集群 G2 开始发布 all:notify 通知事件时，G1、G3、G5、G7、G9 开始接收到消息，当 G2 停止发布时，上述集群在延时若干秒后停止接收。

4） 策略消息测试

测试项目：系统对策略消息的识别和处理。

测试目的：验证系统可以有效识别策略消息，并在路由计算阶段和事件转发阶段根据该策略进行相应的屏蔽。

测试步骤：按照表 5-1 所示集群组成依次启动各代理和代表，全部集群订阅 all:alarm，发布关于 all:alarm 主题的消息，然后对 G1、G3 添加策略 all:alarm，观察各代表和代理的路由计算；发送关于 all:alarm 主题的消息，重新观察接收者。

预期结果：第一次发布消息时所有集群均可收到消息并向上提交；增加策略后，各代表本地的 policy.xml 中增加了该策略，并在控制台处重新计算路由，第二次发布消息时 G1、G3 虽可收到消息但不向上转发。

实际结果：与预期相同。第一次发布消息时所有集群开始不同时延后接收该消息，在相应有订阅的客户端处显示收到消息；当管理员处添加策略后各代表收到策略并对 all:alarm 重新计算路由，在本地策略信息文件 policy.xml 中也有相应新增记录；第二次发布消息时 G1、G3 作为中间节点可以接收到消息，但不再向上层提交，其他集群仍可正常接收和提交消息。

7.3.4. SDN 扩展功能测试

1） 代理和代表的 web 界面展示

测试项目：在代理和代表端通过 floodlight 提供的 web 界面查询代理和代表的连接信息。

测试目的：验证系统可以在 floodlight 的 web 端展示 OpenFlow 交换机、相连的代理和代表信息。

测试步骤：启动 G9 的代表和代理以及它们的 OpenFlow 交换机，注册完成后打开 floodlight 的 web 界面观察其拓扑结构。

预期结果：可以从 G9 的代表和代理处的 floodlight 的 web 界面看到 OpenFlow 交换机的信息以及两个代表代理的信息。

实际结果：与预期相同。可以从 web 界面看到 OpenFlow 交换机的 IP 地址 192.168.1.1、代表的 IP 地址 192.168.1.151 和代理的 IP 地址 192.168.1.152。

2） 代理丢失时的 web 界面测试

测试项目：在代表端通过 floodlight 提供的 web 界面查看代理的丢失。

测试目的：验证 floodlight 可以通过 OpenFlow 交换机有效发现与其相连的节点失效，并在 web 端反映出来。

测试步骤：在启动管理员和 G9 的代表、代理后，在代表的 floodlight 提供的 web 界面进行观察；将代理的网线从 OpenFlow 交换机中拔出，观察代表端的 web 界面。

预期结果：在 G9 的代理和代表启动后坑从代表处观察到 OpenFlow 交换机和代理、代表的连接情况；在代理的链路中断后可以从代表的 web 端观察到代理的信息消失。

实际结果：与预期结果相同。在 G9 的代理和代表都启动成功后能在代表处观察到 OpenFlow 交换机 IP、代表 IP、代理 IP 以及它们之间的连接情况；在代理连接中断后经过 5 秒左右即可观察到代表处 web 界面代理信息已消失。

7.4. 系统性能测试

本文所描述的性能方面指标包括拓扑的收敛速度、路由计算的速度、基于 OpenFlow 的系统跟基于普通网络相比效率的提升和根据路由发布事件的效率，总体从拓扑和路由两个方面来检测本系统这两方面设计的有效性。

7.4.1. 拓扑收敛速度

集群的增多给新加入代表的邻居计算带来麻烦，新加入代表需要从更多的代表中选择和建立邻居，并通知其他集群。新节点向管理员请求信息和下载主题树部分主要与网络状况和主题树本身有关，与拓扑收敛无关，故此部分在此不作分析。邻居的计算和选择时构建拓扑的先决条件，构建完成后由 LSA 传播新集群信息。如图 7-4 所示为随着集群的增加邻居计算和邻居构建所需要的时间。从图中可以看出，当网络中仅有一个集群时，第二个集群代表的加入由于只需要与第一个集群构建邻居，因此比第三个集群的加入费时要少；在此后的集群代表加入时，由于按照默认设定，初始构建的邻居个数均为 2 个，故构建费时相差并不太多，但随着集群的增加，代表的计算和反应的及时性都将降低，故大体上仍然是用时增多的方向。

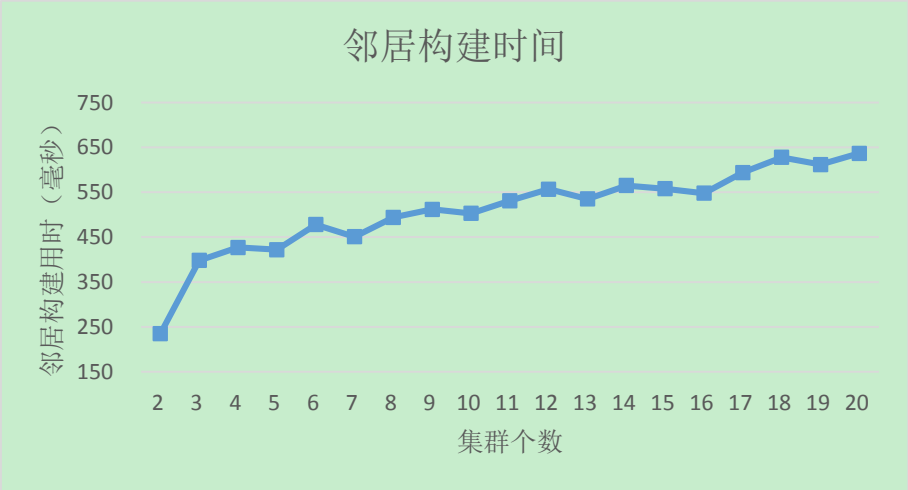

图 7-4 邻居初始构建时间统计

当集群加入网络的数量逐渐变多时，维护的成本也相应变大，当链路发生变化时将消息传达至全网的时间也会相应变长。图 7-5 所示为随着网络内集群数量的增加，新节点从启动到发送自身第一条 LSA 至全部集群所需的时间。由图可知，LSA 的传播延时与集群的数量正向相关，与新节点与最远节点之间的距离有关系。因为此系统中 G2、G3 与 G1 的距离相仿，G4、G5 与 G1 的距离相仿，G6、G7 与 G1 的距离相仿，故在 G2 与 G3 加入时 LSA 传播时延相差不大，但由于 G4 的加入，网络中最远的两个节点 G1 和 G4 距离增大，因此传播时延也有了明显增加，之后的情况可以此类推。

图 7-5 LSA 传播时延统计

节点的失效对于拓扑的收敛性有很大的考验，尤其是代表的失效，为了使验证具有多方面的考量，分别以集群内无其他代理的代表失效和集群内有其他代理的代表失效做实验，检测网络对该失效信息的反应灵敏度。

1、 代表失效 10 点测试

在 10 个集群的网络拓扑中，不同的集群代表失效有不同的传播时延，如图 7-6 所示为 10 个集群的拓扑结构。

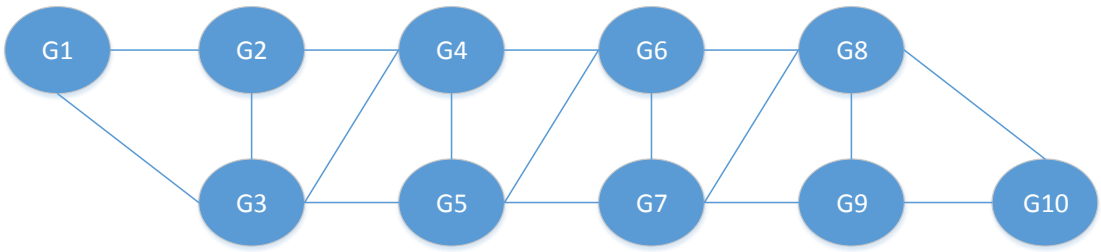

图 7-6 10 点拓扑结构

对于每个集群的代表丢失都需要一定的时间传到网络中每个集群代表处，而在网络中，不同集群由于位置不同所需传播时间也不同，对其中的 G1、G3、G5、

G7、G9 分别做失效测试，由于每个集群中代理取代代表的时间基本相同，为设置失效值 2 分钟，因此仅对代理成为代表后开始到所有集群代表均接收到新代表信息位置统计用时，得出结果，如图 7-7 所示。从图中可以看出，失效集群代理成为代表后通知网络所需时间与其在拓扑中的位置有关系，在相对边缘位置的集群，如 G1、G9，消息到达全网所需时间较长；在相对中心位置的集群，如 G5、G7 所需时间则相对较短。

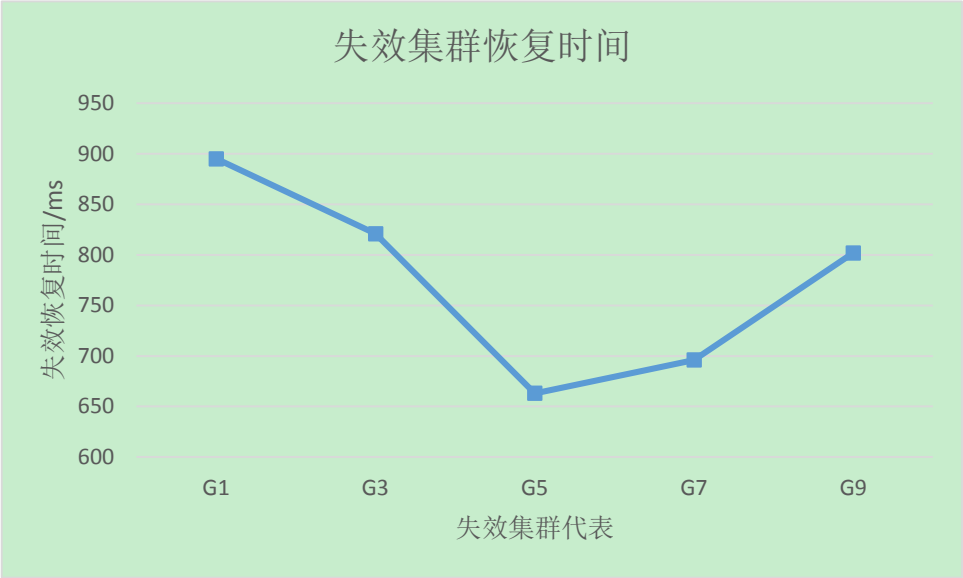

图 7-7 10 点失效恢复时间

当集群中只有一个代表，无代理时，则当该代表失效时，其所有邻居无法到达该集群后广播链路失效的消息，该代表已无邻居后则判定该集群失效，并有集群通知管理员。在此过程中，邻居判定链路失效的机制是一样的，为 2 分钟收不到 Hello 消息后再等待 30 秒，若仍无新代表产生则判定该链路失效，故只记录从邻居判定其失效后发送丢失邻居的 LSA 开始，到管理员接收到该集群失效消息为止所需时间。如图 7-8 所示为 G1、G3、G5、G7、G9 失效时管理员收到消息所需时间。由图可知，在网络中不同位置的集群失效，管理员和网络内其他成员接收到其失效信息所需时间也不同。图中 G3、G9 失效时，网络边缘位置的 G1、G10 需要将邻居失效消息传播至全网，因此费时较多，其他集群失效时，由于其邻居位于较为中间的位置，因此传播时差并不大。

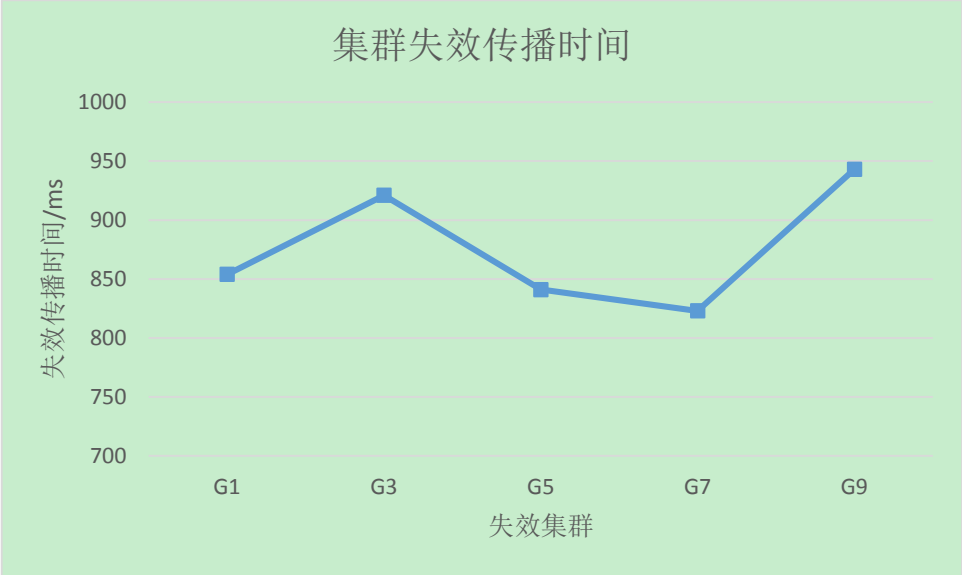

图 7-8 10 点集群失效传播时间

2、 代表失效 20 点测试

当网络中有 20 个集群时，网络规模相较于 10 个点有所增加，但由于规模有限，因此在测试时仍然将最多邻居数设为 4。其构建成功的拓扑如图 7-9 所示。

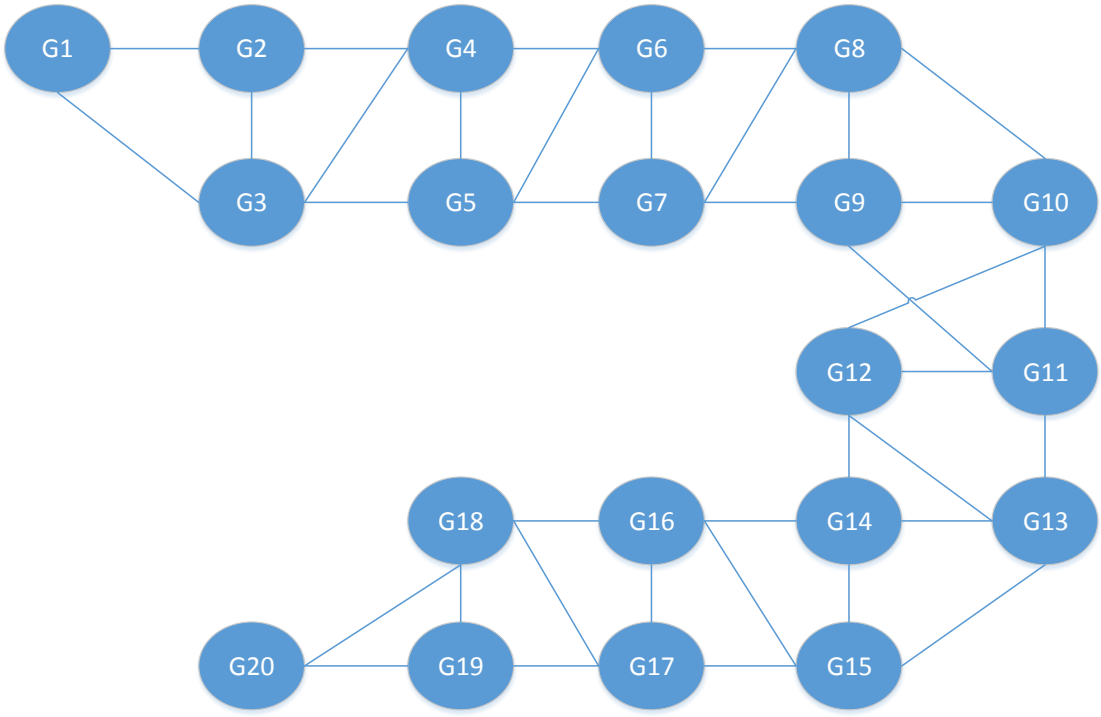

图 7-9 20 点拓扑结构

在网络中若每个集群内均有代表和代理，分别关闭 G1、G3、G5、G6、G7、G9、G11、G13、G15、G17、G19 的代表，测试其代理成为代表后通知全网所用的时间。

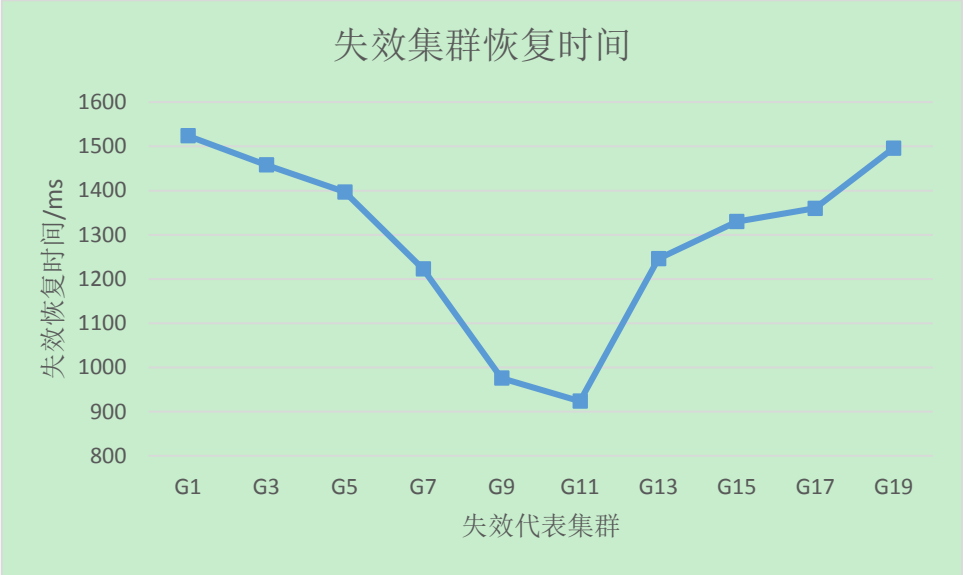

图 7-10 20 点集群失效恢复时间

与 10 点测试中相似，在 20 个集群的网络拓扑中，对于每个集群没有另外代理的情况进行代表的失效测试。如图 7-10 和 7-11 所示，集群失效的传播时间也与集群在网络拓扑中的位置有关。可以看出，失效集群由邻居探测其失效后到最终判定失效的时间与其邻居在网络中的位置有关，因为需要所有邻居将其失效的消息传播全网，得出最终失效的判断。

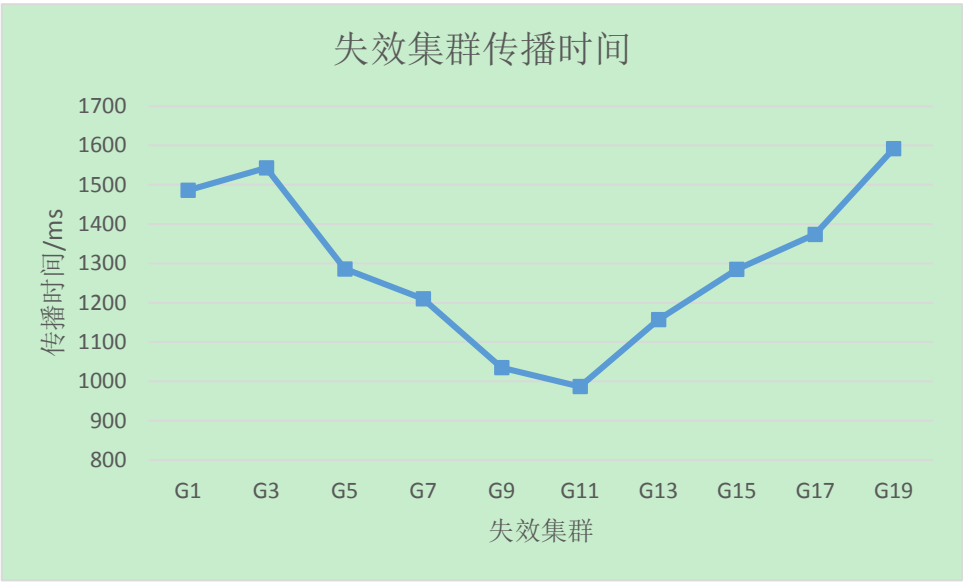

图 7-11 20 点失效集群传播时间

3、 代表失效 30 点测试

为保持与前面结果一致，且考虑实际网络状况，对于 30 个集群的拓扑我们测试时仍然将最多邻居数设为 4。其构建成功的拓扑如图 7-12 所示。

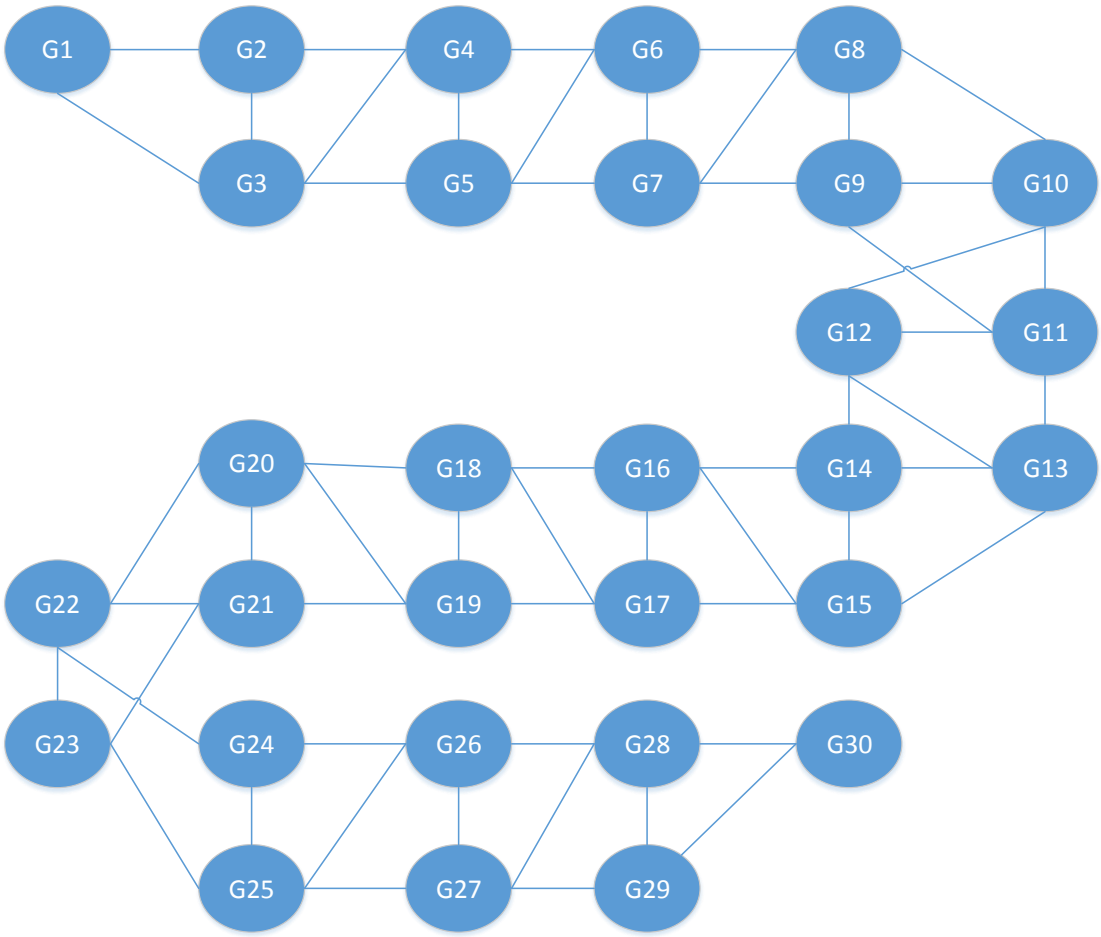

图 7-12 30 点拓扑结构

在网络中若每个集群内均有代表和代理，分别关闭 G1、G5、G9、G13、G17、G21、G25、G29 的代表，测试其代理成为代表后通知全网所用的时间。

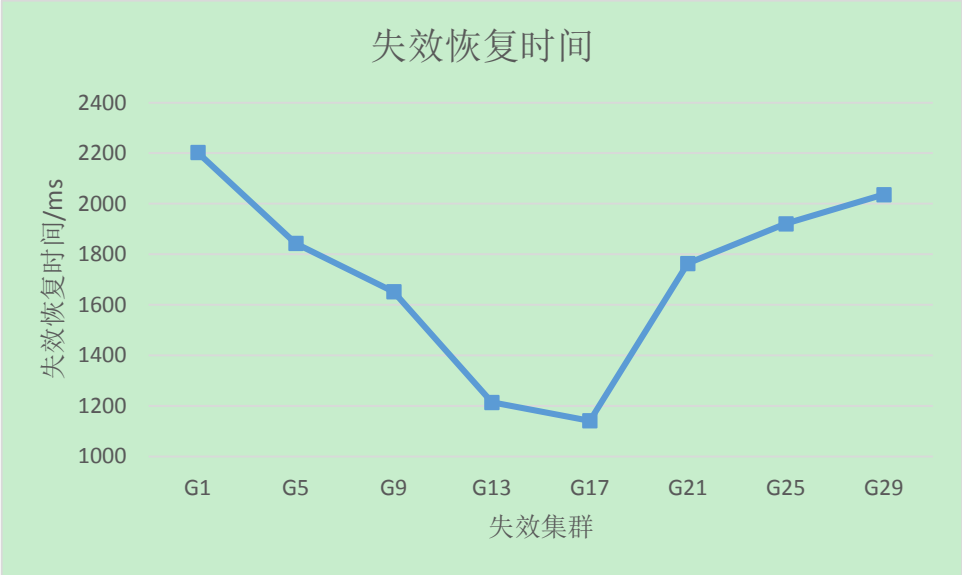

图 7-13 30 点集群失效恢复时间

与 10 点和 20 点的测试中相似，在 30 个集群的网络拓扑中，对于每个集群没有另外代理的情况进行代表的失效测试。如图 7-13 和 7-14 所示，集群失效的

传播时间也与集群在网络拓扑中的位置有关。可以看出，失效集群由邻居探测其失效后到最终判定失效的时间与其邻居在网络中的位置有关，因为需要所有邻居将其失效的消息传播全网，得出最终失效的判断。

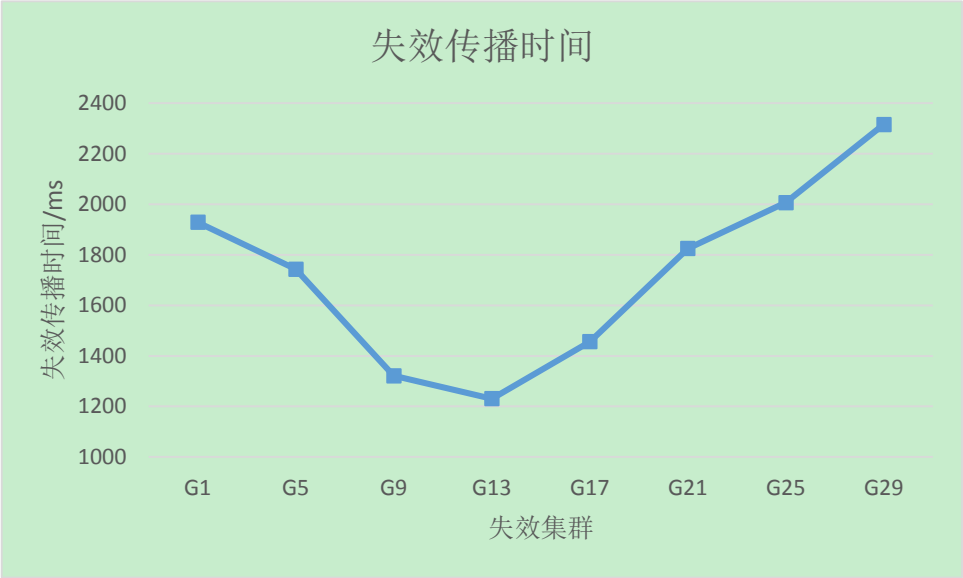

图 7-14 20 点失效集群传播时间

7.4.2. 路由的计算与查询

随着集群的增加、网络拓扑变大，路由计算的负担也会变重，所需时间也相应增加。路由计算是事件转发的基础，因此路由计算的速度也是衡量网络设计是否具有可扩展的依据之一。路由的计算与事件的订阅者数量有关，为方便单项对比，在路由计算的测试时从 G1 到 G8 依次订阅，防止未订阅的中间转发节点影响结果对比。如图 7-15 所示，分别为有 2 个集群到 20 个集群订阅路由计算的平均时间。由图可知，在集群数量不多的情况下，路由计算所需时间在 15ms~25ms 之间波动，其变化较为随机，路由计算与集群的增长无明显的正向相关，变化不大，可以预见在有限的集群增加范围内其计算代价也不会显著增长，但更多的集群情况还有待验证。

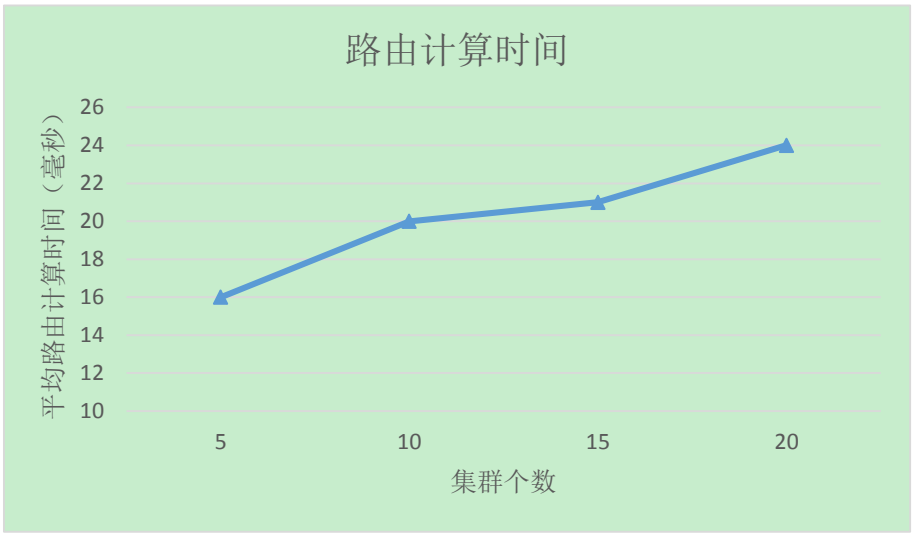

图 7-15 路由计算时间统计

路由计算所需时间反映出算法的计算效率，但事件转发时只须对路由表进行查询，因此对路由表的查询效率才是关键。事件转发过程中，每一个事件都需要根据其主题进行路由查阅，返回下一跳集合，路由查询的效率与事件转发的效率息息相关。

由于路由的单次查找用时非常少，不能明显反映出查询效率的趋势，因此在测试中连续查询 10000 次主题，统计出查询效率进行分析。路由查询效率统计如图 7-16 所示，统计的主题分别为一级、二级、三级、四级主题，由图可知路由查询所用时间与主题的深度成正相关，但相比于转发速度来说其查询效率增加并不明显。

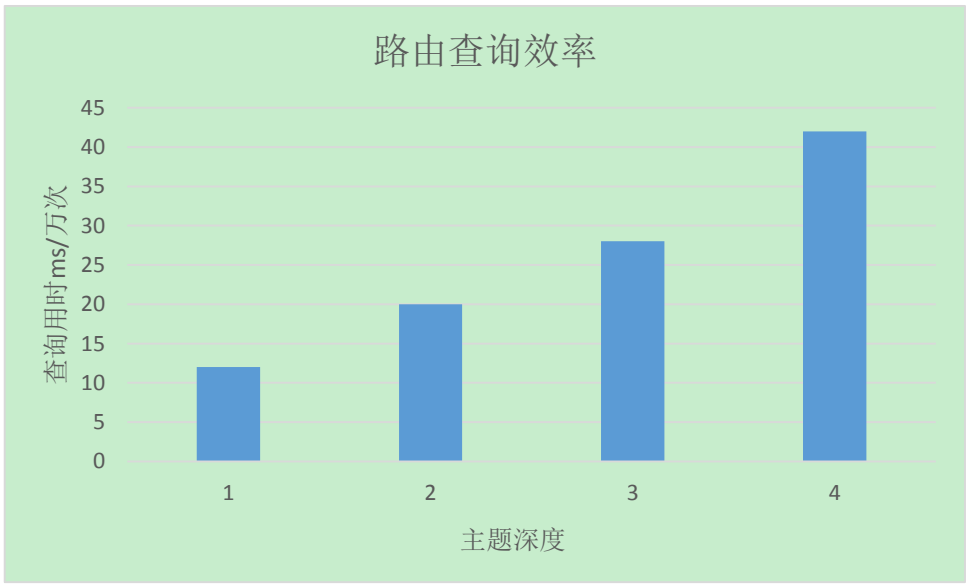

图 7-16 路由查询平均效率

7.4.3. 事件转发效率的对比

OpenFlow 网络在拓扑维护、路由计算和数据转发方面对发布/订阅系统均有改进，在拓扑收敛性、事件转发效率方面理论上均应有提升，以下就对其进行相应的验证。但由于实验环境原因，未能在较大规模上进行对比。

为使传统网络和 SDN 网络在相同环境下对比，下面采用三个集群的方式对事件转发效率进行对比。在测试中由一个集群发布事件，另外两个订阅事件，测试系统处理 10 万条信息的平均用时。发送节点每发送 5000 条信息休息 2ms。在测试中事件发布者共发送 20*5000 条信息，所有的接受者共接收 2*20*5000 条信息，即总共有 20 万条信息通过本系统。以下为 SDN 网络和传统网络每个节点全部完成 10 万条数据处理所需时间。如图 7-17 所示，分别为 SDN 网络中的 G1、G2 代表和传统 IP 网络中的 G1、G2 代表接收 G3 发送的 10 万条消息接收和处理所用的时间，可以看出在 SDN 网络中由于前面论述的改进，对消息处理的用时有了显著的改善。

图 7-17 SDN 与 IP 网络集群代表处理消息对比

7.5. 本章小结

本章对文中的设计实现方案做了详细的测试。首先对系统的各项基本功能（如系统启动、加入、获取策略等）进行了测试，保证了系统的可用性。其后对拓扑的构建和维护、路由的转发以及在 SDN 上的扩展进行了功能测试，实验表明该系统能够进行有效的拓扑构建和维护，有良好的拓扑收敛性，可以按照预设链路进行路由计算和数据转发，在 SDN 上与 floodlight 的结合对系统的拓扑维护有着有效的作用。其后针对拓扑收敛速度、路由计算和查询效率进行了量化测试，

测试范围最多为 30 个点进行，避免测试节点太少带来的随机性结果。最后将在 SDN 网络中的事件转发效率测试结果与传统网络中的事件转发效率结果进行对比，论证了发布/订阅系统在 SDN 网络上移植的有效性和必要性。

第八章 总结与展望

本章节对本系统的实现中的调度、拓扑和路由部分进行总结和未来的展望，以已经取得的成果对未来后续的开发和拓展提供一些经验和构思。

8.1. 工作总结

本文所论述的方案开始准备于 2012 年 11 月，在之前已经实现的版本上进行拓扑、路由等方面的修改。刚开始 5 个月对原版代码进行分析、理解和调试，并对相关技术进行学习，对实现方法和框架进行初步设计。相关技术包括 OSPF、WSN、网络编程、floodlight 等。结合之前版本的测试结果分析了其拓扑不稳定、容错性差、路由计算方案效率低下等不足，针对这些问题我们设计了如本文中所描述的较为稳定高效的拓扑路由方案。

在拓扑维护方面，我们对之前的心跳消息、心跳队列进行改进，参照 OSPF 设计规则，使用三十秒一次的 UDP 消息取代心跳消息的发送，并对该 UDP 消息进行实时处理，通过对其设定计时器检测其是否超时，有效检测失效节点；在路由计算方面，我们对之前完全由 IP 排序构建的二叉树进行改进，基于实际拓扑，同时结合链路上的转发压力，利用最短路径算法构建路由树；在 OpenFlow 的应用方面，对之前单纯的 IP 网络进行 SDN 网络的拓展，利用开源软件 floodlight 作为拓扑探测的辅助工具和数据转发的控制器；在订阅表维护方面，取代之前单纯的主题名称和集群关系的对应，针对分层主题进行主题树的设置，在树中的每个节点处存储该节点的订阅，方便路由计算和路由查找；引入策略消息，每当有新的策略更新时，都会更新相应的本地记录和路由表信息，从而为信息的正确传播奠定基础。

8.2. 未来展望

本文论述了发布/订阅系统在拓扑和路由方面的改进设计，同时提出了向 SDN 网络的改进方案。在本系统中，用户可以实现出松耦合、高时效、异步的可靠应用，但同时本系统也有可以改进和扩展的地方，例如：

1、进一步完善拓扑的维护规则。由于目前的拓扑维护中缺乏防止外界干扰、相同系统不同作用域之间的干扰以及其他恶意破坏的机制，因此在接下

来的工作中可以进一步提高系统的安全性和可靠性，在防干扰和攻击方面做进一步的设计和实现；

2、本系统在拓扑维护方面与 floodlight 结合不够紧密，未充分利用 floodlight 的所有功能模块，如主控制器和辅控制器的交互等，因此可以进一步在 floodlight 中拓展其功能模块，以使拓扑维护更高效；

3、目前系统的运行规模较小，其在大规模网络中的应用效果尚未可知，因此在接下来的工作中可以在传统网络和 SDN 网络方面同时扩大规模，以验证系统的可扩展性，并在实验中对其进行相应改进。

参考文献

[1] Carzaniga A, Rutherford M J, Wolf A L. A routing scheme for content-based networking[A]. Proceedings of the 23rd AnnuralJo int Conference of the IEEE Computer and Communications Societies[C] . USA: IEEE INFOCOM, 2004. 918- 928.

[2] Fengyun Cao, Jaswinder Pal Singh, MEDYM: match-early with dynamic multicast for content-based publish-subscribe networks [A]. The 6th ACM/ IFIP/ USENIX International Middleware Conference [C]. Berlin: Springer, 2005. 292 -313.

[3] Rowstron A, Kermarrec A M. SCRIBE: the design of a large-scale event notification infrastructure [A]. Proceedings of the Third International COST264 Workshop on Networked Group Communication[C]. London: Springer-Verlag, 2001. 30- 43.

[4] Gianpaolo Cugola, Davide Frey. Content-based routing for publish-subscribe on a dynamic topology concepts, protocols, and evaluation [EB/OL]. http://dit.unitn.it/~picco/papers/psReconf.pdf, 2005.

[5] Eiko Yoneki, Jean Bacon. An adaptive approach to content-based subscription in mobile Ad Hoc networks [A]. Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops[C]. USA: IEEE Computer Society, 2004. 92- 97.

[6] Emmanuelle Anceaume, Ajoy K. Datta. Publish/ Subscribe scheme for mobile networks [A]. Proceedings of the Second ACM International Workshop on Principles of Mobile Computing [C]. New York: ACM Press, 2002. 74-81.

[7] 孙其博，刘杰，黎葬，等. 物联网:概念，架构与关键技术研究综述[J]. 北京邮电大学学报，2010，33(3):1-9.

[8] 沈苏彬，范曲立，宗平，等. 物联网的体系结构与相关技术研究[J]. 南京邮电大学学报(自然科学版)，2009，29(6):1-11.

[9] 郝光星. 基于主题的 DDS 系统的扩展性设计与实现[J]. 软件，2012，33(12): 41-46.

[10] 汪锦岭. 面向 Internet 的发布/订阅系统的关键技术研究[D]. 北京:中国科学院软件研究所，2005.

[11] Pietzuch P R, Bacon J M. Hermes: A distributed event-based middleware architecture[C]//Distributed Computing Systems Workshops, 2002. Proceedings.

22nd International Conference on. IEEE, 2002: 611-618.

[12] Chockler G, Melamed R, Tock Y, et al. Spidercast: a scalable interest-aware overlay for topic-based pub/sub communication[C]//Proceedings of the 2007 inaugural international conference on Distributed event-based systems. ACM, 2007: 14-25.

[13] Mühl G, Fiege L, Pietzuch P. Distributed event-based systems[M]. Heidelberg: springer, 2006.

[14] Topic detection and tracking: event-based information organization[M]. Springer, 2002.

[15] Pietzuch P R, Bacon J. Peer-to-peer overlay broker networks in an event-based middleware[C]//Proceedings of the 2nd international workshop on Distributed event-based systems. ACM, 2003: 1-8.

[16] Fiege L, Gärtner F C, Kasten O, et al. Supporting mobility in content-based publish/subscribe middleware[C]//Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware. Springer-Verlag New York, Inc., 2003: 103-122.

[17] Network M Y O. OSPF network design solutions[J]. 2003.

[18] Fortz B, Thorup M. Internet traffic engineering by optimizing OSPF weights[C]//INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE. IEEE, 2000, 2: 519-528.

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

[20] Gajic B, Riihijarvi J, Mahonen P. Intra-domain topology manager for publish-subscribe networks[C]//Telecommunications (ICT), 2011 18th International Conference on. IEEE, 2011: 394-399.

[21] Jacobson V, Mosko M, Smetters D, et al. Content-centric networking[J]. Whitepaper, Palo Alto Research Center, 2007: 2-4.

[22] Gajic B, Palenzuela J, Riihijarvi J, et al. Mobility-Aware Topology Manager for Publish-Subscribe Networks[C]//New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on. IEEE, 2012: 1-5.

致谢

文章即将收尾，在此我对研究生阶段所有帮助、鼓励、支持和批评我的人表示衷心的感谢。

首先，我要感谢我的导师章洋副教授。章老师常常以身作则，与我们一同对科研项目的设计和实现反复推敲，精益求精。章老师的专业能力和对科研方向的洞察力令人心生佩服，同时对于实践的指导也使我受益匪浅。在项目协作中，我与其他项目开发人员的合作互助也与章老师的谆谆教导息息相关。章老师对学术的态度使我在学习的过程中有了深切体会，同时也有了奋斗的目标。

其次，我要感谢和我一起并肩作战的战友们。感谢温鹏、左文峰学长，感谢肖丹学姐，他们的指导和帮助让我对项目从初步认识到深入了解，上一届的学术和人文关怀令人难忘；感谢陈天宇、郭成、苟粲然、范淋淋、陈小杰等同届学生，与他们一起合作开发项目、思想碰撞、互帮互助的日子让我倍感珍惜；感谢下一届师弟臧亚强、陈岩，你们的帮助和新鲜想法的加入使项目的实现更加充实，也使我的思维更加开阔；感谢博士学姐王亚丽，你带给我们的设计思想和学术论文让我在科研的方向迈出了新的一步。

感谢我的家人、朋友对我的支持、鼓励和无微不至的关怀，让我在学习、奋斗之时有了思想动力，也有了感情的依托。

最后，感谢各位参加评审的评委老师们在百忙之中抽出宝贵时间来评阅本文。你们的肯定是我努力的目标。谢谢！

攻读学位期间发表的学术论文目录

[1] 王双锦、章洋。基于 SDN 的发布/订阅系统中的路由计算，中国科技论文在线，2014。（已发表）

[2] S.J. Wang, Y. Zhang. Evaluation of publish/subscribe based routing over SDN [C], 2014 International Conference on Future Communication Technology and Engineering, Shenzhen China, 2014.（已录用，EI检索）