**密级： 保密期限：**

xm 拷贝

**硕士学位论文**



**题目： 基于SDN发布订阅系统中多媒体**

**传输质量保证方案的设计与实现**

**学 号：**

**姓 名：**

**专 业： 计算机科学与技术**

**导 师：**

**学 院： 网络技术研究院**

**2019年 2 月 14 日**

**Confidentiality level:**

**Confidentiality period:**

**Beijing University of Posts**

**and Telecommunications**

**Master’s Thesis**



**Title： RESEARCH AND DESIGN OF**

**\_MULTIMEDIA TRANSMISSION**

**\_\_\_OVER PUB/SUB SYSTEM\_\_**

**\_\_\_\_\_ IN SDN NETWORKS**

**No. ：**

**Name：**

**Major： Computer Science and Technology**

**Tutor：**

**College： Institute of Network Technology**

**2019.2.14**

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

本人完全了解并同意北京邮电大学有关保留、使用学位论文的规定，即：北京邮电大学拥有以下关于学位论文的无偿使用权，具体包括：学校有权保留并向国家有关部门或机构送交学位论文，有权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，有权允许采用影印、缩印或其它复制手段保存、汇编学位论文，将学位论文的全部或部分内容编入有关数据库进行检索。（保密的学位论文在解密后遵守此规定）

本人签名： 日期：

导师签名： 日期：

**基于SDN发布订阅系统中多媒体传输质量保证方案的**

**设计与实现**

**摘 要**

发布订阅系统作为一种分布式消息中间件，实现了消息的发送者和接收者之间的解耦；而SDN网络具备可控制、可编程的特性，将发布订阅系统与SDN网络相结合，可以充分发挥这两者的特点，实现传统网络难以完成的功能。

物联网中传输的数据不仅仅包含文本数据，还包含图片、监控视频等多媒体数据，传统的发布订阅系统采用层叠网络架构，无法控制底层的路由通信设备，因此很难为多媒体数据传输提供质量上的保证。SDN“控制-转发”相分离的结构支持系统在运行时对网络设备进行编程，因此基于SDN网络的发布订阅系统使得多媒体数据的传输成为可能。

为了实现多媒体数据在SDN发布订阅系统中的传输，本文从基于SDN的发布订阅系统架构、路由转发、质量保证等方面进行设计，本文的工作和贡献包括：

1. 采用分群分层的方式对系统进行划分，这样便于网络的大规模拓展，集群由交换机、主机、集群控制器组成，集群控制器负责集群内的发布订阅管理，集群与集群之间通过边界交换机连接，在集群间通过管理员进行管理
2. 发布订阅系统中存在多源多播问题，系统使用改进的路由算法，计算出代价最小的转发路径
3. 提供两层质量保证方案，系统接收用户时延需求后，在管理员处分析决策，在控制器处执行调整，两者相互依赖相互限制，共同保障系统的传输质量
4. 主机上多个用户的订阅会带来高并发问题，系统在用户端设置WSN层，在数据转发时采用改进的数据转发方案和负载均衡策略，提高了消息转发的稳定性

本系统在SDN交换机构建的真实网络环境中实现了发布订阅系统的所有功能，并且提供了完善的传输质量保证方案。经过大量的测试，基于SDN的发布订阅系统在物理环境下稳定运行，并且支持多媒体传输和用户需求协商功能，测试结果验证了本系统质量保证方案的有效性。

**关键词** SDN网络 发布订阅系统 多媒体传输 质量保证方案 WSN层

**RESEARCH AND DESIGN OF MULTIMEDIA TRANSMISSION OVER PUB/SUB SYSTEM**

**IN SDN NETWORKS**

**ABSTRACT**

As a distributed message middleware, publish-subscribe system realizes the decoupling between sender and receiver of message. SDN network has the characteristics of controllability and programmability. Combining publish-subscribe system with SDN network can give full play to the characteristics of both and realize the functions that traditional network not accomplish.

The data transmitted in the Internet of Things includes not only text data, but also multimedia data such as pictures, surveillance videos. The traditional publishing and subscribing system adopts a cascade network architecture, which does not control the underlying routing and communication equipment. Therefore, it is difficult to provide quality assurance for multimedia data transmission. SDN "control-forwarding" separated structure support system programmed network devices at runtime, and SDN-based publishing and subscribing system made multimedia data transmission possible.

In order to realize the transmission of multimedia data in SDN publishing and subscribing system, this paper designs the publishing and subscribing system architecture, routing and forwarding, and quality assurance based on SDN. The innovations of this paper include:

1. The system divided into clusters and layers, which is convenient for the large-scale expansion of the network. Clusters are composed of switches, hosts and cluster controllers. Cluster controllers are responsible for publishing and subscribing management in the cluster. Clusters connect other Clusters by boundary switches. Clusters are controled by administrators.
2. Multi-source multicast exists in publish-subscribe system. The system uses improved routing algorithm to calculate the least cost-forwarding path.
3. Provide a two-tier quality assurance scheme. After receiving the user's delay requirements, the system analyses the decision-making at the administrator and executes the adjustment at the controller. The two depend on each other and restrict each other to ensure the transmission quality of the system.
4. The subscription of multiple users on the host will lead to high concurrency. The system sets up WSN layer on the client side and adopts improved data forwarding scheme and load balancing strategy to improve the stability of message forwarding.

This system realizes all functions of the publish/subscribe system in the real network environment built by SDN switching agency, and provides a perfect transmission quality assurance. After a lot tests, the publish/subscribe system based on SDN runs stably in the physical environment, and supports multimedia transmission and user requirement negotiation. The test results verify the effectiveness of the system's quality assurance scheme.

**KEY WORDS** SDN network publish/subscribe system multimedia transmission system quality assurance WSN layer

**目录**

[**第一章** **绪论** 1](#_Toc2443884)

[**1.1** **研究背景** 1](#_Toc2443885)

[**1.2** **研究内容** 2](#_Toc2443886)

[**1.2.1** **SDN发布订阅系统的架构设计** 2](#_Toc2443887)

[**1.2.2** **多媒体传输** 3](#_Toc2443888)

[**1.2.3** **质量保证方案** 4](#_Toc2443889)

[**1.3** **论文组织** 4](#_Toc2443890)

[**1.4** **本章总结** 5](#_Toc2443891)

[**第二章** **相关技术概述** 7](#_Toc2443892)

[**2.1** **软件定义网络技术** 7](#_Toc2443893)

[**2.1.1** **广义SDN定义** 7](#_Toc2443894)

[**2.1.2** **OpenFlow协议** 8](#_Toc2443895)

[**2.1.3** **SDN控制器** 9](#_Toc2443896)

[**2.1.4** **SDN交换机** 9](#_Toc2443897)

[**2.1.5** **流表** 9](#_Toc2443898)

[**2.2** **基于SDN的发布订阅系统** 10](#_Toc2443899)

[**2.3** **web service** 12](#_Toc2443900)

[**2.4** **OSPF协议** 13](#_Toc2443901)

[**2.5** **RTP协议** 14](#_Toc2443902)

[**2.6** **Reactor模式** 14](#_Toc2443903)

[**2.7** **本章总结** 15](#_Toc2443904)

[**第三章** **需求分析** 17](#_Toc2443905)

[**3.1** **SDN控制器需求** 18](#_Toc2443906)

[**3.1.1** **拓扑管理需求** 19](#_Toc2443907)

[**3.1.2** **路由计算需求** 20](#_Toc2443908)

[**3.1.3** **流表管理需求** 21](#_Toc2443909)

[**3.2** **发布订阅系统需求** 23](#_Toc2443910)

[**3.2.1** **主题管理需求** 23](#_Toc2443911)

[**3.2.2** **消息接口需求** 25](#_Toc2443912)

[**3.3** **用户需求** 26](#_Toc2443913)

[**3.3.1** **多媒体传输需求** 27](#_Toc2443914)

[**3.3.2** **用户协商需求** 28](#_Toc2443915)

[**3.4** **本章总结** 29](#_Toc2443916)

[**第四章** **系统概要设计** 31](#_Toc2443917)

[**4.1** **系统架构设计** 31](#_Toc2443918)

[**4.1.1** **已有系统架构** 31](#_Toc2443919)

[**4.1.2** **系统架构改进** 32](#_Toc2443920)

[**4.2** **控制器上发布订阅功能设计** 33](#_Toc2443921)

[**4.2.1** **拓扑管理** 34](#_Toc2443922)

[**4.2.2** **发布订阅管理** 38](#_Toc2443923)

[**4.2.3** **路由管理** 39](#_Toc2443924)

[**4.2.4** **流表管理** 42](#_Toc2443925)

[**4.3** **管理员功能设计** 44](#_Toc2443926)

[**4.3.1** **主题树管理** 45](#_Toc2443927)

[**4.3.2** **用户请求处理** 46](#_Toc2443928)

[**4.3.3** **ui展示** 48](#_Toc2443929)

[**4.4** **WSN功能设计** 48](#_Toc2443930)

[**4.4.1** **消息接口** 49](#_Toc2443931)

[**4.4.2** **数据转发方案** 50](#_Toc2443932)

[**4.5** **发布订阅系统的QoS保证方案设计** 52](#_Toc2443933)

[**4.5.1** **队列信息获取** 53](#_Toc2443934)

[**4.5.2** **队列带宽调整算法** 54](#_Toc2443935)

[**4.6** **用户功能设计** 55](#_Toc2443936)

[**4.6.1** **多媒体传输** 56](#_Toc2443937)

[**4.6.2** **用户协商** 57](#_Toc2443938)

[**4.7** **本章总结** 59](#_Toc2443939)

[**第五章** **系统详细设计与实现** 61](#_Toc2443940)

[**5.1** **系统架构实现** 61](#_Toc2443941)

[**5.2** **消息的定义与实现** 64](#_Toc2443942)

[**5.2.1** **拓扑消息** 66](#_Toc2443943)

[**5.2.2** **主题树消息** 67](#_Toc2443944)

[**5.2.3** **发布订阅消息** 67](#_Toc2443945)

[**5.2.4** **用户请求消息** 67](#_Toc2443946)

[**5.2.5** **集群状态消息** 68](#_Toc2443947)

[**5.3** **控制器上发布订阅功能实现** 68](#_Toc2443948)

[**5.3.1** **拓扑管理** 68](#_Toc2443949)

[**5.3.2** **发布订阅管理** 72](#_Toc2443950)

[**5.3.3** **路由管理** 74](#_Toc2443951)

[**5.3.4** **流表管理** 80](#_Toc2443952)

[**5.4** **管理员功能实现** 83](#_Toc2443953)

[**5.4.1** **主题树管理** 83](#_Toc2443954)

[**5.4.2** **用户请求处理** 90](#_Toc2443955)

[**5.4.3** **ui展示** 91](#_Toc2443956)

[**5.5** **WSN功能实现** 93](#_Toc2443957)

[**5.5.1** **消息接口** 93](#_Toc2443958)

[**5.5.2** **数据转发方案** 94](#_Toc2443959)

[**5.6** **发布订阅系统的QoS保证方案实现** 95](#_Toc2443960)

[**5.6.1** **队列信息获取** 95](#_Toc2443961)

[**5.6.2** **队列带宽调整算法** 96](#_Toc2443962)

[**5.7** **用户功能实现** 96](#_Toc2443963)

[**5.7.1** **多媒体传输** 96](#_Toc2443964)

[**5.7.2** **用户协商** 99](#_Toc2443965)

[**5.8** **日志功能实现** 100](#_Toc2443966)

[**5.9** **本章总结** 101](#_Toc2443967)

[**第六章** **系统测试** 103](#_Toc2443968)

[**6.1** **测试环境** 103](#_Toc2443969)

[**6.1.1** **硬件环境** 103](#_Toc2443970)

[**6.1.2** **软件环境** 104](#_Toc2443971)

[**6.2** **功能测试** 104](#_Toc2443972)

[**6.2.1** **拓扑功能** 105](#_Toc2443973)

[**6.2.2** **控制器功能** 107](#_Toc2443974)

[**6.2.3** **主题树管理功能** 111](#_Toc2443975)

[**6.2.4** **发布订阅功能** 114](#_Toc2443976)

[**6.2.5** **用户协商功能** 116](#_Toc2443977)

[**6.2.6** **多媒体传输功能** 117](#_Toc2443978)

[**6.2.7** **功能测试小结** 118](#_Toc2443979)

[**6.3** **性能测试** 118](#_Toc2443980)

[**6.3.1** **丢包率测试** 120](#_Toc2443981)

[**6.3.2** **时延测试** 124](#_Toc2443982)

[**6.3.3** **性能测试小结** 129](#_Toc2443983)

[**6.4** **本章总结** 129](#_Toc2443984)

[**第七章** **总结与展望** 131](#_Toc2443985)

[**7.1** **工作总结** 131](#_Toc2443986)

[**7.2** **工作展望** 131](#_Toc2443987)

[**参考文献** 133](#_Toc2443988)

[**攻读学位期间发表的学术论文目录** 135](#_Toc2443989)

1. **绪论**
   1. **研究背景**

随着互联网技术的不断发展，网络带宽不再成为限制数据传输的瓶颈[1]，多媒体数据的传输日益成为人们生产生活中的常态，传统网络中，存储转发的传输方式对链路整体情况没有把控，对数据的时延、丢包率没有保证，因而无法为多媒体传输提供可靠性保障，本文需要设计一种稳定、高效的新型网络传输模式，从而提升用户的服务体验。

SDN（Software Defined Networking）是一种新型的网络架构[2]，它在传统匹配、转发模式的基础上，将数据传输中的控制层与转发层剥离开来，网络层设备只负责流量的转发，而网络拓扑的收集、路由计算、流表下发等功能则由控制器实现。当一条无匹配项的数据包到达交换机时，传统网络设备的处理方式是采用默认转发机制[3]，但每个交换机不会对全网拓扑有完整的认知，因此采取的默认转发方案很大概率是错误的，会浪费链路资源，SDN交换机的解决方法则是将该条数据流上传至控制器，由控制器计算相关路径再将流表下发到交换机中，从而提高转发的准确率。传统网络中，对数据的转发、控制功能都是由网络设备提供的，这些设备是各个厂家自己开发生产，集成了大量与业务特性相关的操作系统和硬件，不同厂家生产的网络设备很难做到互相通用。在SDN网络中，网络设备只需要提供转发功能，因此可以采用通用的硬件，而不同厂家生产的操作系统由控制器负责，与底层硬件的通信通过编程实现[4]，这样可以面向不同的业务特性提供定制的控制功能，从而降低了对硬件的依赖性[5]。

传统的消息传输过程中，消息的发送方首先需要建立与接收方的通信连接，然后将消息直接发送给接收方，这要求通信双方构建紧密联系。发布-订阅（publish–subscribe）是一种新型消息传输模式[6]，发布者不需要知道消息的具体传输、接收过程，发布者将发布的消息进行主题分类，把带有主题的消息传送给网络节点，同时订阅者可以表达对该主题的兴趣，订阅了该主题后就可以收到相应的内容，而无需对发布者有所了解，这样就大大降低了发布者和订阅者之间的耦合度[7]，消息的生产者与消费者实现了脱离，系统具有更好的可扩放性，能够支持动态的网络拓扑。

基于SDN的发布订阅系统结合了发布订阅系统和SDN网络的特点，其优点在于借助SDN网络控制转发分离、网络可编程的特点，解决传统网络中的实时性、安全性等问题，同时结合发布订阅系统的特点，将消息通信双方解耦，具体的传输转发、路由计算功能由系统提供，最终构建出一个稳定、高效的消息中间件网络。

近年来，虽然SDN技术变得越来越火热，但是真实应用场景却很少，这是因为SDN交换机的造价昂贵，只有大型公司才能生产，如华为、思科等，因此不少高校、研究机构采用虚拟化技术，比如Open vSwitch[8]、Mininet等，这给SDN研究带来了极大的便利性和可行性，但是虚拟化技术在设计之初就存在性能上的瓶颈，实际环境与虚拟环境的测试结果无法同步，由仿真环境带来的研究弊端不容忽视，严重阻碍了发布订阅系统在SDN网络中的应用。

针对上述问题，本课题基于SDN网络下的发布订阅系统，为用户提供多媒体传输服务。使用SDN交换机加快传输速度，交换机只负责匹配转发，而控制器根据网络负载情况，使用改进的路由算法计算转发路径，为多媒体数据的传输提供质量保证；编码实现多媒体数据传输，使用数据转发方案保证数据的可靠转发；同时系统还支持用户需求协商，为用户提供更好的消息传输服务。

* 1. **研究内容**

本文的研究内容是基于SDN发布订阅系统中多媒体传输质量保证方案的设计与实现，系统在SDN交换机上部署、运行、测试，避免了虚拟环境的影响，同时基于发布订阅系统实现，在传输层面上解耦消息的发送者、接受者，能够为多媒体传输带来性能上的提升。系统整体的研究内容包括以下三个方面：一是SDN发布订阅系统架构的设计，实现发布订阅系统与SDN物理交换机的结合，为数据转发提供保障；二是多媒体数据的传输，结合发布订阅系统的特性，实现视频数据端到端的传输；三是质量保障方案，提供性能上的保证。下面进行具体介绍。

1. **SDN发布订阅系统的架构设计**

已有系统使用的是模拟环境，结合OpenDayLight[9]控制器软件监测虚拟网桥、虚拟交换机，这给大规模测试提供了便利，但是虚拟环境下的测试结果与真实环境差异很大，遇到的问题也各有不同，因此本系统的核心目标是开发一款发布订阅系统，实现与SDN物理交换机的结合。就本系统而言，基于SDN的发布订阅系统核心架构主要分为拓扑管理、发布订阅管理、路由管理、流表管理、主题管理、消息接口这几个方面。

拓扑管理分为集群内拓扑管理和集群间拓扑管理，集群内的拓扑由集群控制器负责，相应的连接信息保存在集群控制器中；集群间拓扑是指不同集群的连接关系，运行时集群可以进行添加、删除等操作，因此集群间拓扑具有动态性，集群控制器需要对邻居集群进行拓扑探测以及拓扑维护管理。系统采用了OSPF协议，邻居集群间会通过三次握手的方式进行感知，收集到的邻居信息保存在本地数据库中，这样经过拓扑探测，所有集群都将获得全网拓扑信息。

路由计算分为管理路径的计算和主题路径的计算，管理员通过管理路径与控制器交互，主题消息在主题路径上传输。系统使用Dijkstra[10]算法计算管理路径，使用Steiner[11]算法计算主题路径，计算出的结果是逻辑上的链路转发路径。

流表管理是发布订阅系统与SDN网络结合的重要一环，系统将消息转发路径转换为具体的流表并下发给SDN交换机，这样当消息到达SDN交换机后会执行相应的匹配转发操作，实现“控制-转发”[12]的功能。

主题管理的功能是通过编码的方式，快速区分不同主题，这样消息的发布者、订阅者通过指定主题的方式构建联系。系统在主题内容与IPv6地址间形成一一映射关系，将主题消息编码为IPv6组播地址[13]的形式，这样既实现了主题的区分功能，又可以在交换机匹配时指定目的地址，方便消息的转发。

消息接口是系统与用户之间的消息交互接口，用户通过消息接口调用系统的发布订阅的功能，同时也可以提出时延带宽上的需求，系统通过消息接口，将主题消息、需求计算结果反馈给用户。

上述模块共同完成了系统的基本功能，拓扑管理模块用于邻居集群的发现，构建出全局链路连接情况，发布订阅管理模块维护了用户的订阅信息，用户消息经过路由模块的计算得到转发路径，并转换为流表下发至SDN交换机，主题管理模块是发布订阅的核心功能，用户间通过主题进行联系，消息接口模块是用户与系统交互的方式，模块间相互合作，为用户提供稳定高效的数据传输服务。

1. **多媒体传输**

在基于SDN的发布订阅系统中，已有的研究内容专注于传输文本数据，这并不符合真实网络环境中用户的需求，在真实网络环境中，音频、视频等多媒体数据正逐渐成为互联网中信息交互的主体，同时用户对于数据的实时性也有进一步的要求。

RTP[14]协议是由IETF[15]开发的实时传输协议，提供面向连接和无连接两种方式，通常和UDP协议一起使用。在RTP协议中定义了两种报文格式：RTP报文和RTCP报文[16]，RTP报文用于传送多媒体数据，比如音频和视频信息，RTCP报文用于传送控制信息，以实现协议控制功能，在RTCP报文中包含了网络传输QoS有关信息[17]，如平均时延、数据包丢失率等，发送者可以通过这些信息监测和评价网络传输QoS状况，并采取适当的策略实施同步。而基于SDN的发布订阅系统，提供了硬件转发、集中控制的功能，可以根据RTCP中链路状态信息，重新计算当前网络环境中发布者、订阅者间的转发路径，缓解链路压力，这使得RTP协议和本课题所在的系统环境得到了很好的契合，有助于提升用户的服务体验。

在实际传输过程中，接收方订阅相应主题，发送方发布匹配的主题，而系统会结合主题编码、拓扑信息、订阅情况计算出转发路径，并作为流表的匹配项下发至交换机，这样发送方将封装好的数据包发送到系统，经过路由转发到达接收方，接收方再根据协议格式拆包处理，就能得到视频流数据，从而实现多媒体数据的传输功能。

1. **质量保证方案**

系统的质量保证方案包括队列带宽调整算法、数据转发方案、用户协商机制。SDN网络具有可编程、可控制的特性，数据包在SDN交换机端口的不同队列上分别传输，系统通过相应的接口可以获取到队列运行信息，进而得到网络转发状态。系统结合SDN队列信息使用队列带宽调整算法，保证发布订阅系统的高质量传输，当网络数据包过多、队列拥塞时，动态协调不同队列间的带宽分配策略，达到整体时延的优化效果。

数据转发方案在接收到主题消息后调用，由于网络状态的复杂性，数据可能存在丢失、阻塞的问题，系统使用阻塞队列缓存消息[18]，并通过分发的方式指定后续处理线程，同时我们可以在处理流程中添加负载均衡策略[19]，在主队列缓存数据过大时丢弃一些耗时长的任务，进一步提升系统性能。

系统还提供了用户协商机制，用户发起时延带宽请求，系统结合网络整体运行状况进行分析处理，并将计算结果下发给用户所在集群的控制器，若可以满足，则控制器下发相应的流表，更改端口带宽等参数；若无法满足，则将处理结果反馈给用户，由用户自行决定是否接受当前处理结果。由用户发起的请求更加具有真实性，进一步完善了发布订阅系统的质量保证方案。

队列带宽调整算法结合SDN网络的特性，为用户提供了高效的传输服务；数据转发方案是系统传输过程中的自我调节机制，能够缓解下游接收处理端的压力，并结合负载均衡策略提供性能上的保障；用户协商机制是在用户层面的质量保证，具有真实性和有效性。

* 1. **论文组织**

本章是绪论，主要提出SDN发布订阅系统的架构，多媒体传输的方式，以及相应的质量保障方案，相关章节组织结构如下：

第一章是绪论，介绍了本文的研究背景和主要研究内容；

第二章是相关技术概述，介绍了本文涉及的相关技术知识，包括软件定义网络、基于SDN的发布订阅系统、web service、OSPF协议、RTP协议、Reactor模式等；

第三章是需求分析，主要介绍了系统各个核心模块的组成，其中核心参与角色包括集群控制器、管理员、用户，以及各自的需求；

第四章是系统概要设计，这里简要论述了各个模块的核心架构、基本流程和涉及的关键数据结构；

第五章是系统详细设计与实现，详细说明了消息的定义、各个模块的实现方式、核心算法等；

第六章是系统测试，该章节介绍了测试环境、测试方案，并得出了具体的测试结果，证明本系统的可靠性，以及质量保障方案的有效性；

第七章是总结与展望，主要包括本文的工作总结、心得体会和对未来的目标展望等。文末是参考文献和致谢词。

* 1. **本章总结**

本章是绪论，说明了本文的研究背景和主要研究内容，介绍了关于SDN软件定义网络的相关知识，介绍了发布订阅系统的概念和作用，同时对本文的研究内容进行了简要阐述，包括SDN发布订阅系统的架构设计、多媒体传输方式、质量保障方案的设计等，最后对论文的组织结构进行了概括，方便读者了解文章的结构和写作目的。

1. **相关技术概述**

本文的研究内容是将SDN网络与发布订阅系统结合起来，实现多媒体传输的功能，并且提供完善的质量保障方案，因此相关的技术要求和背景知识包括SDN网络、发布订阅技术、多媒体传输协议、质量保障策略等，下面将对这些技术内容进行简要介绍。

1. **软件定义网络技术**

软件定义网络技术是一个抽象的概念，从广义上讲，具备可编程特性、控制与转发分离的网络架构都可以称为软件定义网络，而在狭义概念上，软件定义网络是指基于OpenFlow协议的南向接口网络。下面分别进行介绍。

1. **广义SDN定义**

软件定义网络（Software Defined Network，SDN）是由美国斯坦福大学CLean State课题研究组提出的一种新型网络创新架构，是网络虚拟化的一种实现方式。其核心技术在于使用OpenFlow将网络设备的控制层面与数据层面分离开来，从而实现了网络流量的灵活控制，使得网络具备更加智能的转发功能，为应用的创新、网络的拓展提供良好的平台[20]。SDN的架构图如下所示。

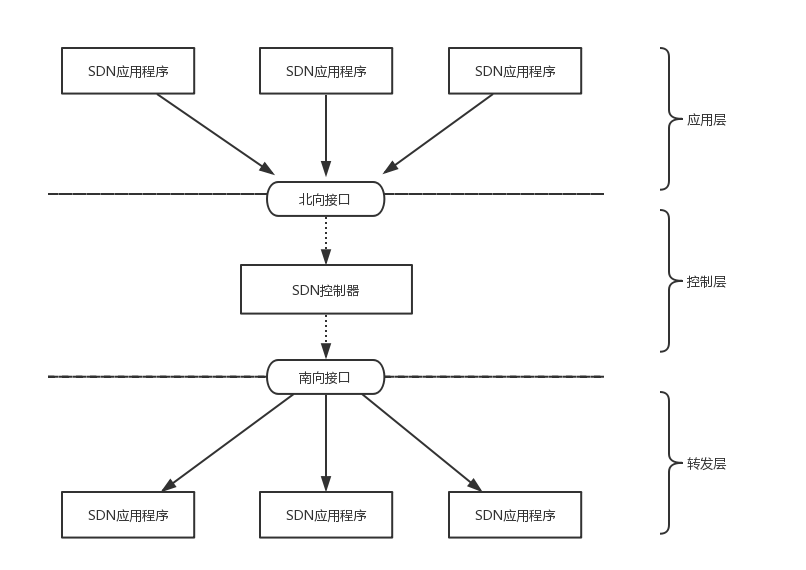


图2-1 SDN架构图

SDN的核心架构分为三层：最上面的是应用层，负责接收用户的业务需求，并为用户提供SDN应用服务；中间是控制层，具有拓扑管理、路由计算等功能，消息的转发控制也是在该层实现的；最底层是转发层，也称为数据层，负责执行匹配、转发操作，同时可以将设备的运行状态反馈给上层用户。

我们将应用层与控制层之间的接口称为北向接口，负责提供控制层到应用层的API（应用程序编程接口），便于实现用户对网络的管理；将控制层和转发层之间的接口称为南向接口，负责定义控制和转发之间的协议规范，能够完成对底层硬件设备的控制。实现南向接口功能的协议有很多，目前比较通用的是OpenFlow协议[21]。经过上面的描述，我们可以知道，SDN网络与OpenFlow协议是两个不同的概念，SDN是一种网络架构，而OpenFlow是南向接口协议的一个实现，制定了控制层和转发层之间的通信规则，广义上来讲，SDN包含了OpenFlow协议。

1. **OpenFlow协议**

斯坦福大学在2008年最先提出了OpenFlow的概念，经过两年的努力后发布了OpenFlow协议1.0版本，该协议自发布以来，一直由开放网络基金会（ONF）管理[22]，ONF是一个致力于开发SDN应用以及开放标准的用户主导型组织。由于OpenFlow协议发展最久，设计最为完善，因此从众多的南向接口协议标准中脱颖而出，被SDN网络使用者广泛接受。OpenFlow协议的设计理念是将流表的维护过程放在SDN交换机中，消息到达交换机后，会进行相应的匹配操作，查看包头域中的关键字段是否一致，如果匹配成功就执行相应的转发操作，如果匹配失败，会将该消息通过安全信道发送出去，由控制器负责流表的生成、下发和维护管理。

随着SDN网络的日益成熟，OpenFlow协议本身也在不断地更新、完善，目前已经发展到了1.5版本，相应的功能也得到了进一步的加强。OpenFlow协议在移动互联网、数据网络等领域中的作用越来越重要，为SDN网络的推广和发展提供了有力的支持。

1. **SDN控制器**

SDN控制器是软件定义网络的控制中心，负责网络拓扑的管理、路由的计算、流表的下发与维护等。控制器通过南向接口与交换机进行交互，从而获得SDN网络的控制权，同时在北向接口，提供相应的API接口供开发人员使用，丰富系统的功能。目前，市面上存在多款开源的控制器软件，比如OpenDayLight、Floodlight、Ryu、ONOS等[23]，已有的系统采用的就是OpenDayLight软件，作为一款JAVA版本的开源控制器，OpenDaylight支持跨平台操作，为用户提供了充足的RestAPI，方便我们二次开发，在搭建虚拟网桥后，就可以生成虚拟网络环境，从而进行测试。但是上述技术在真实环境下无法使用，SDN物理环境与仿真环境下的测试结果差异很大，本系统在开源控制器的设计思路基础上，经过架构设计、编码实现，开发出了一款独立的SDN基础上的发布订阅系统。

1. **SDN交换机**

SDN交换机，也称为OpenFlow交换机，位于SDN架构中的最底层。SDN交换机需要支持OpenFlow协议，能够与控制器进行通信，一方面可以接受控制器发来的执行语句，下发匹配流表；另一方面，交换机需要具备队列、流量查询的功能，能够通过安全信道向控制器上报设备信息，以及转发过程中的运行状态等。与传统交换机相比，SDN交换机功能更加强大，可以根据用户指定的匹配规则下发流表，同时能够收集交换机运行信息。

1. **流表**

流表的基本功能是对数据包进行匹配，为数据提供转发依据，并对匹配成功的数据包执行相应的操作。流表由多种多样的流表项组成，流表项是数据匹配的最小单位，与传统路由层的路由表相比，流表项是由控制器生成并下发的，这也是SDN网络中控制层与转发层相隔离的重要体现，除此之外，流表还能够整合网络中不同层次的配置信息，为用户提供更加完善的匹配规则[24]。流表项的基本结构如下所示。

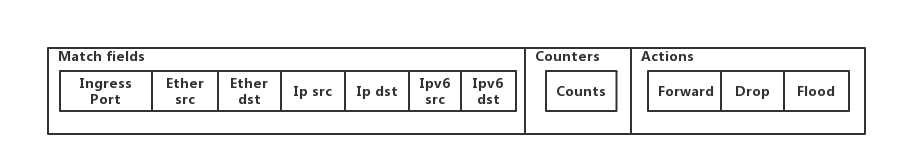


图2-2 流表项结构图

图中列出了流表项的基本构成，各字段作用见表2-1。

表2-1 流表项各字段名称及作用

|  |  |  |  |
| --- | --- | --- | --- |
| 域名 | 功能 | 字段名 | 字段描述 |
| Match fields | 匹配域，将各字段分别与数据包首部进行匹配，匹配成功则执行相应的操作 | Ingress Port | 交换机入端口 |
| Ether src | 源Mac地址 |
| Ether src | 目的Mac地址 |
| IP src | 源IPv4地址 |
| IP dst | 目的IPv4地址 |
| IPv6 dst | 源IPv6地址 |
| Counters | 计数，统计队列发送的数据包数量和字节数量 | Counts | 计数器 |
| Actions | 动作，规定了数据包的转发行为 | Forward | 转发，向目的端口发送 |
| Drop | 丢弃，不处理 |
| Flood | 洪泛，向所有端口转发 |

流表项定义了充足的字段来实现匹配、转发功能，正是由于其丰富的匹配项和转发动作，SDN交换机能够提供路由表基本功能之外的操作，使得SDN网络更加高效、灵活。

1. **基于SDN的发布订阅系统**

发布订阅系统是一款消息中间件，用于解耦通信双方，消息作为通信交互的承载单元，将发布请求、订阅请求、主题消息发送到网络中。消息使用主题进行分类，不同的主题表示不同的类别，针对同一主题进行发布订阅的用户才能建立联系。消息的发送方称为发布者（publisher），消息的接受者称为订阅者（subscriber），发布者、订阅者之间无需建立直接的联系，消息的路由、转发由发布订阅节点（broker）提供。发布者、订阅者之间存在多对多的映射关系，一旦某个主题发生改变，所有订阅该主题的用户都将收到系统通知。发布订阅消息交互图如下所示。

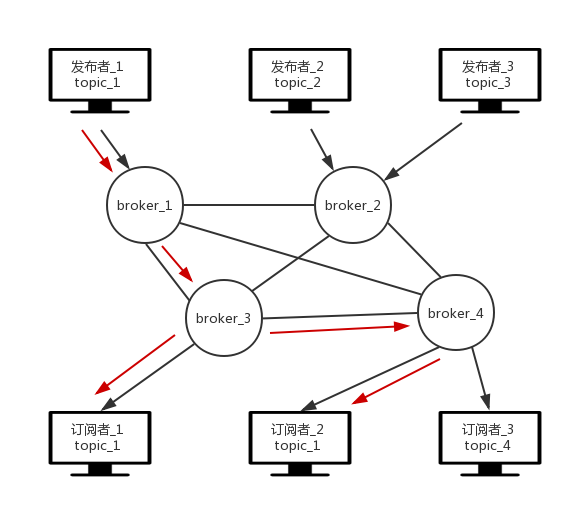


图2-3 发布订阅消息交互图

图中发布者\_1、订阅者\_1、订阅者\_2通过相同的主题topic\_1建立了联系，这样发布者\_1发布主题消息时，经过节点的路由转发功能，最终订阅者\_1、订阅者\_2都将收到主题消息，相应的转发路径用红色箭头进行了标注。

SDN网络、发布订阅系统都是分层结构，包含的中间层负责联系上下层系统，那么我们可以进一步将两者结合起来，充分发挥这两个系统的优势。基于SDN的发布订阅系统，是在SDN网络中部署发布订阅系统，这样在控制层由发布订阅系统负责用户的数据通信，将计算出的转发路径下发到SDN交换机，在数据层由SDN交换机负责消息的转发，SDN网络为发布订阅系统提供了传输支持，而发布订阅系统是在SDN网络上的控制实现，两者间的关系如下所示。

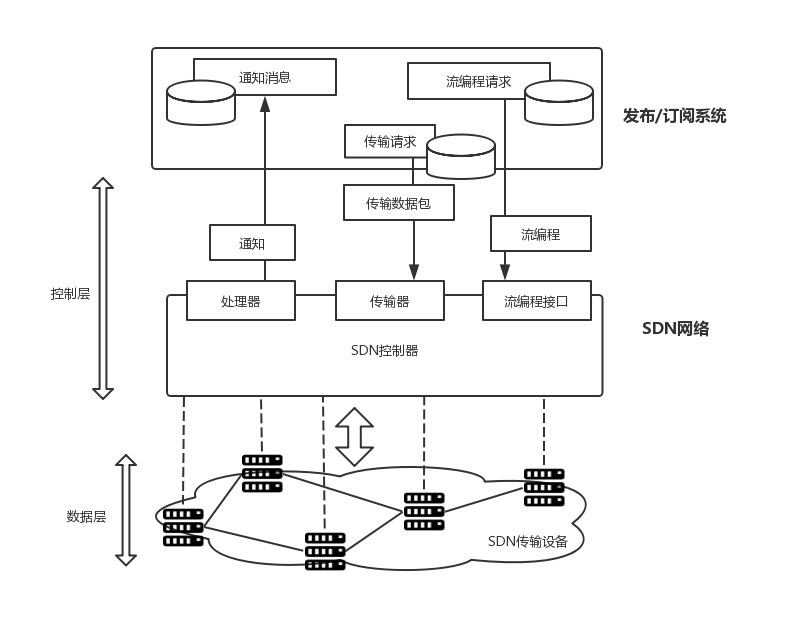


图2-4 发布订阅和SDN关系图

基于SDN的发布订阅系统使用了消息解耦机制，通信双方无需事先建立传输链路，甚至不知道对方的的存在，发布者只需要向特定的发布地址传送消息，订阅者监听主题对应的地址就能收到消息，实现了通信角色的解耦，这对于大规模分布式环境有着显著的性能提升效果。此外，基于SDN的发布订阅系统可扩展性好，底层硬件设备只负责消息的匹配转发，路由机制由控制器负责，这样能够充分发挥“软件定义网络”的功能，适应不同的工作场景。

1. **web service**

Web Service是一种远程调用技术，能够在不同的操作系统上使用[25]。SOAP（简单对象访问协议）提供了标准的远程方法调用机制，简单来说：SOAP协议 = HTTP协议 + XML数据格式[26]，web service是SOAP协议的具体应用，在使用时，web service通过HTTP协议在通信双方间传送消息，发送请求时采用XML格式封装，并增加了特定的HTTP消息头部，以说明请求的内容格式，在收到消息后也需要进行相应的解析，包括去除HTTP头部和XML格式等,在本系统中，使用web service来实现发布订阅服务的调用。

1. **OSPF协议**

OSPF（Open Shortest Path First）开放最短路径优先协议，是由Internet工程任务组开发的路由选择协议，它为网络间路由器互相发现、交换信息并最终生成路由表提供服务[27]。它是一个内部网关协议(Interior Gateway Protocol），用于在单一自治系统（Autonomous System）中进行路由决策[28]。OSPF交互流程如下所示。

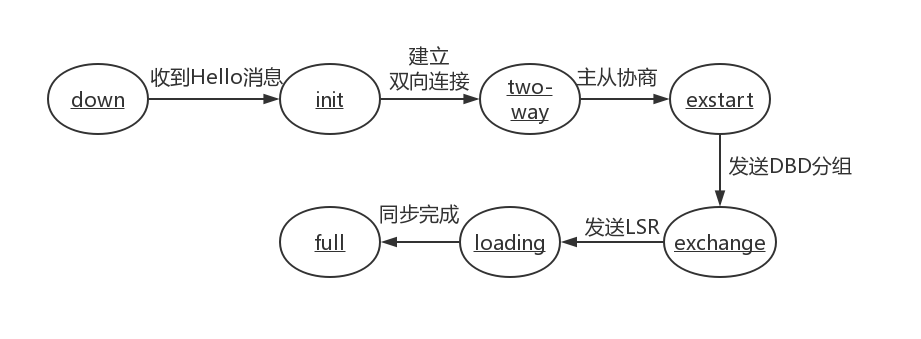


图2-5 OSPF交互流程图

图中列出了OSPF交互流程中可能存在的几种状态，包括新建（down）状态，此时路由器刚接入网络，需要发送Hello消息，向网络中的其他路由器介绍自己，当收到Hello消息后进入初始化（init）状态，说明此前发送的Hello消息被接收，邻居路由将新增路由器添加至本地邻接表中，邻居路由器发送回应hello信息，包含自己的router Id和所有邻居列表，此时为单向连接，新路由器收到回复消息后进入双向连接（two-way）状态，新增路由器收到回复Hello消息，发现自己的router Id在邻居表中，建立双向连接，新建路由器通过主从协商的方式进入信息交换初始（exstart）状态，在这个状态下，相邻路由器之间建立主--从关系，然后双方交换链路分组，进入信息交换（exchange）状态，本地路由器和邻居交换一个或多个链路分组，链路分组中包含有关LSA条目的摘要信息，然后交换机通过LSR请求，进入信息加载（loading）状态，收到链路消息后，将会同LSDB中的数据进行比较，如果链路消息中有更新的链路状态数据，则向对方发送一个LSR请求，用于获取最新的LSA，然后更新完成后进入完全邻接（full）状态，至此邻接的链路状态数据库同步完成[29]。

OSPF协议应用于系统的拓扑探测阶段，基本的操作流程包括发送Hello报文、建立邻接关系、生成并维护链路状态数据库、路由计算、生成路由表。经过OSPF拓扑探测阶段，集群控制器获得了全网链路连接信息。

1. **RTP协议**

RTP协议是一种实时传输协议，可以在面向连接或无连接的底层协议上工作，通常和UDP协议一起使用[30]；RTP定义了两种报文：RTP报文和RTCP报文，RTP报文用于传送多媒体数据，如音频和视频数据，RTCP报文用于传送控制信息，以实现协议控制功能，在消息的传输过程中，将数据包的丢失情况、时延情况封装在RTCP数据包中，发送者可以通过这些信息分析网络传输QoS状况，并采取适当的策略实施同步。RTP包格式如下所示。

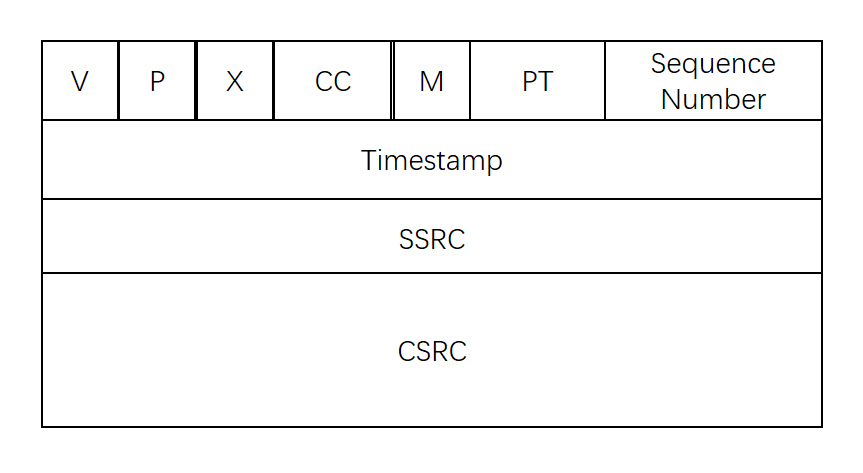


图2-6 RTP数据包格式图

图中比较重要的几个字段及其含义为见表2-2。

表2-2 RTP数据包各字段名称及作用

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段含义 | 字段功能 |
| CC | CSRC标识的数目 | CSRC标识紧跟在RTP固定头部之后，用来表示RTP数据报的来源 |
| PT | 负载类型 | 标明RTP负载的格式，包括所采用的编码算法、采样频率、承载通道等 |
| sequence number | 序列号 | 用来为接收方提供探测数据丢失的方法 |
| timestamp | 时间戳 | 记录了负载中第一个字节的采样时间 |

1. **Reactor模式**

Reactor模式是事件驱动编程的一种实现方式，每种服务在服务端可能由多个方法组成，Reactor会解耦接收到的请求，并分发给对应的事件处理器来处理，该模式适用于高并发、大数据量的分布式场景。目前，许多流行的开源框架都用到了Reactor模式，如：Netty、Node.js、JAVA中的NIO等[31]。Reactor模式的处理机制如下所示。

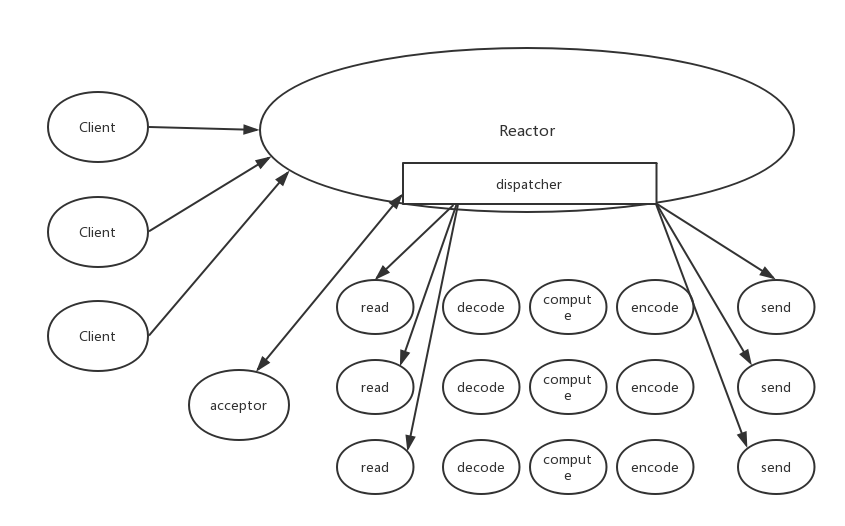


图2-7 Reactor模式处理机制

当消息到来时，系统先使用阻塞队列将消息缓存起来，进一步通过“事件”触发分派器取出消息进行分发处理，同时，由于分派器的逻辑是系统自行定义的，还可以添加负载均衡策略，从而提升处理效率。

1. **本章总结**

本章主要对本文涉及的相关技术知识进行了简单的介绍，包括软件定义网络技术、基于SDN的发布订阅系统、web service、OSPF协议、RTP协议、Reactor模式等。软件定义网络是本系统实现的物理基础，发布订阅系统在SDN基础上实现了消息的解耦，OSPF协议完成了拓扑探测功能，RTP协议保障了视频的实时传输，而Reactor模式提升了系统的处理性能。其中，实现SDN发布订阅系统中多媒体数据的传输是本系统的功能目标，传输质量保证是本系统的性能目标，详细内容将在后续章节中说明。

1. **需求分析**

本系统需要达到的目的是构建一套基于SDN物理交换机的发布订阅系统，因此基本的需求包含三个方面：SDN控制器需求、发布订阅需求和用户需求。系统具备SDN控制器功能，能够对交换机进行信息获取、流表下发等操作，具体包括拓扑管理、路由计算、流表管理等需求，从而达到SDN“控制-转发”的目的；系统实现发布订阅功能，能够在用户间构建关联，为用户提供发布、订阅功能，以及时延需求协商功能，具体包括主题管理、消息接口等需求；用户通过消息接口调用系统的发布订阅功能，提出时延带宽协商的需求。系统整体用例图如下所示。

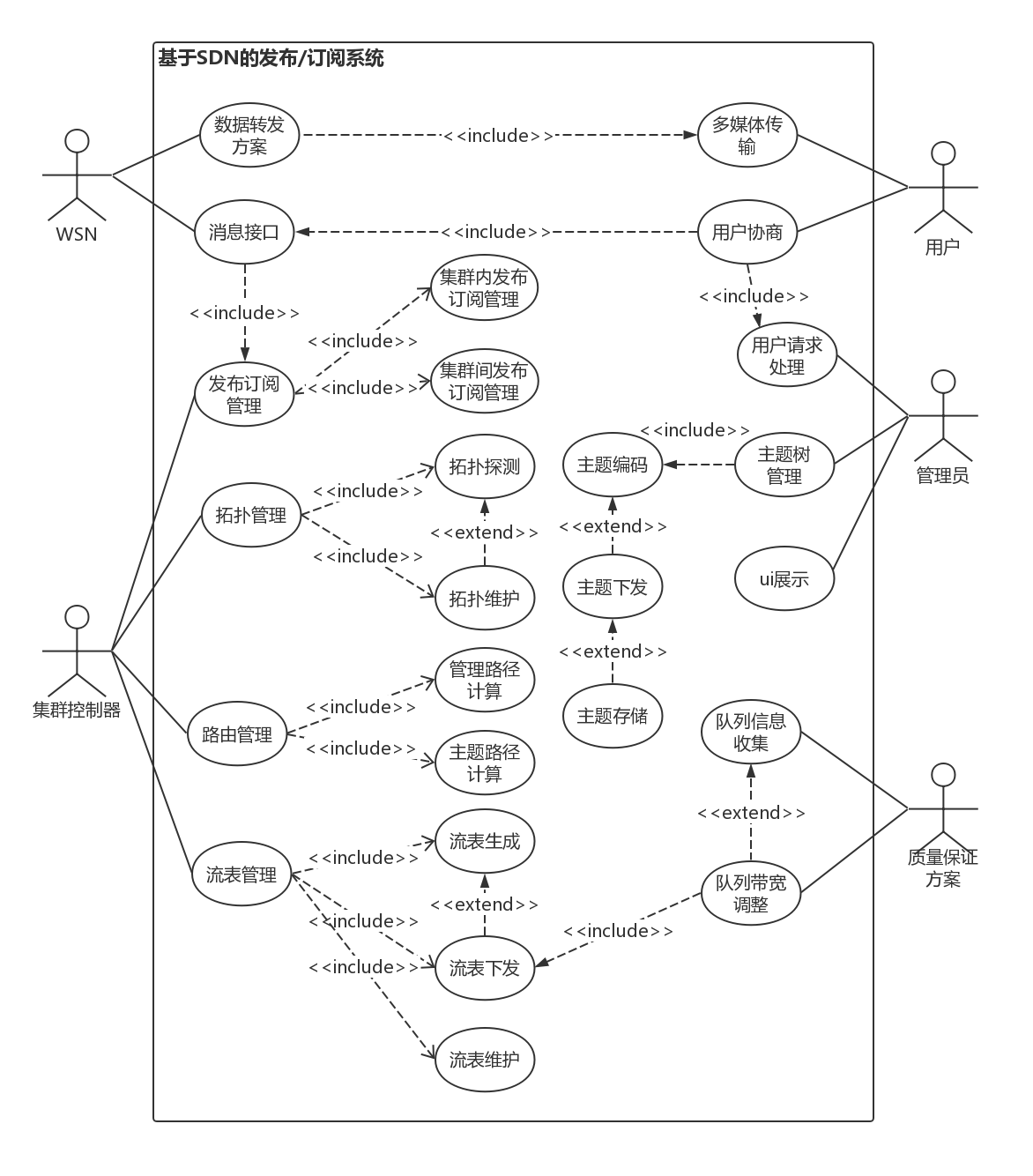


图3-1 系统整体用例图

1. **SDN控制器需求**

本系统需要完成SDN控制器的功能，包括系统运行过程中能够进行拓扑探测、调用路由算法计算出转发路径、下发转发流表等，因此需要完成多种功能需求，下面进行详细介绍。

1. **拓扑管理需求**

拓扑管理分为集群内拓扑管理和集群间拓扑管理，集群内拓扑指的是SDN交换机的连接情况，可以通过交换机相关指令进行查询；集群间拓扑指的是不同集群的分布、连接情况，具有动态变化的特点。这里讨论的是集群间拓扑的探测和维护，即通过拓扑管理功能，系统能够感知邻居集群的存在，进而构建全网链路信息。为实现拓扑管理需求，需要完成拓扑探测、拓扑维护功能，拓扑管理的用例如图3-2所示。

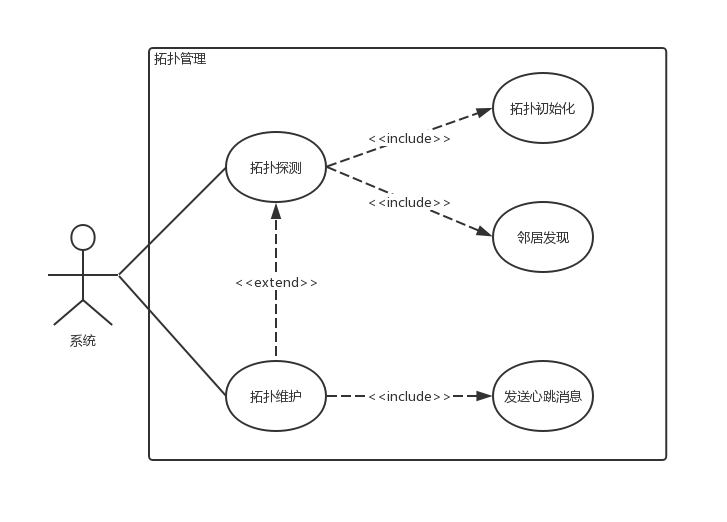


图3-2 拓扑管理用例图

拓扑探测：需要首先对本集群进行初始化，获取集群配置信息，然后下发初始化流表，并向对外端口发送探测消息，用例详细情况见表3-1。

表3-1 拓扑探测用例详细情况

|  |  |
| --- | --- |
| 参与者 | 系统 |
| 用例描述 | 拓扑探测 |
| 前置条件 | 系统启动，获取本集群配置信息 |
| 详细描述 | 1、根据配置信息，获取集群名、集群交换机地址等信息  2、下发初始化流表  3、向所有对外端口发送探测消息 |
| 后置条件 | 系统得到邻居集群的消息反馈，拓扑探测完成 |

拓扑维护：在构建了邻居集群间的连接关系后，系统还需要定时向邻居集群发送心跳维护消息，防止因故障导致的集群失效，用例详细情况见下表。

表3-2 拓扑维护用例详细情况

|  |  |
| --- | --- |
| 参与者 | 系统 |
| 用例描述 | 拓扑维护 |
| 前置条件 | 系统启动后，与邻居集群构建了连接关系 |
| 详细描述 | 1. 完成拓扑探测功能 2. 定时向邻居集群发送心跳消息 3. 若消息超时或未回复，则认为邻居集群已经失效，将该集群从拓扑中删除 |
| 后置条件 | 系统维护了全网拓扑，并将失效集群删除 |

1. **路由计算需求**

经过拓扑管理模块，我们获得了全网拓扑信息，进一步可以得到网络中各个集群的连接、分布情况，那么在路由计算时就可以根据这张分布图，得到相应的转发路径，系统需要完成管理路径和主题路径的计算。路由计算的用例图如下所示。

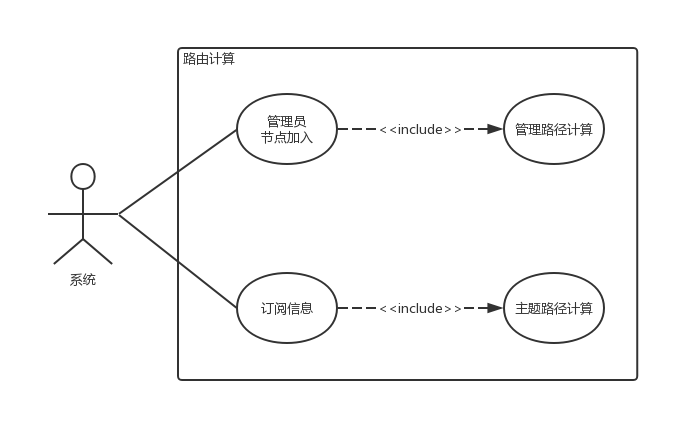


图3-3 路由计算用例图

管理路径计算时：若当前拓扑环境中存在管理员节点，则需要调用路由算法，计算出一条管理路径，使得管理员可以将消息下发至每个控制器，用例详细情况见下表。

表3-3 管理路径计算用例详细情况

|  |  |
| --- | --- |
| 参与者 | 系统 |
| 用例描述 | 管理路径计算 |
| 前置条件 | 经过拓扑探测，链路中存在管理员身份的节点 |
| 详细描述 | 1. 调用拓扑探测模块 2. 分析处理后，发现链路中存在管理员节点 3. 计算出管理路径 |
| 后置条件 | 经过相关路由算法，计算出管理路径 |

主题路径计算：当集群中产生新的发布、订阅消息后，需要结合全网拓扑计算出一条转发路径，用例详细情况见下表。

表3-4 主题路径计算用例详细情况

|  |  |
| --- | --- |
| 参与者 | 系统 |
| 用例描述 | 主题路径计算 |
| 前置条件 | 用户发起发布、订阅请求 |
| 详细描述 | 1. 用户发起请求 2. 结合订阅信息、全网拓扑，重新计算主题路径 |
| 后置条件 | 经过相关路由算法，计算出主题转发路径 |

1. **流表管理需求**

经过路由计算，系统得到了逻辑链路转发路径，与已有系统不同，本系统运行在物理SDN网络上，因此逻辑上的转发路径无法直接使用，还需要借助流表管理模块实现逻辑路径到物理流表的转换，最终得到SDN交换机识别的流表指令，实现“控制-转发”的功能。该模块需要完成流表生成、流表下发、流表维护等功能。流表管理的用例图如下所示。

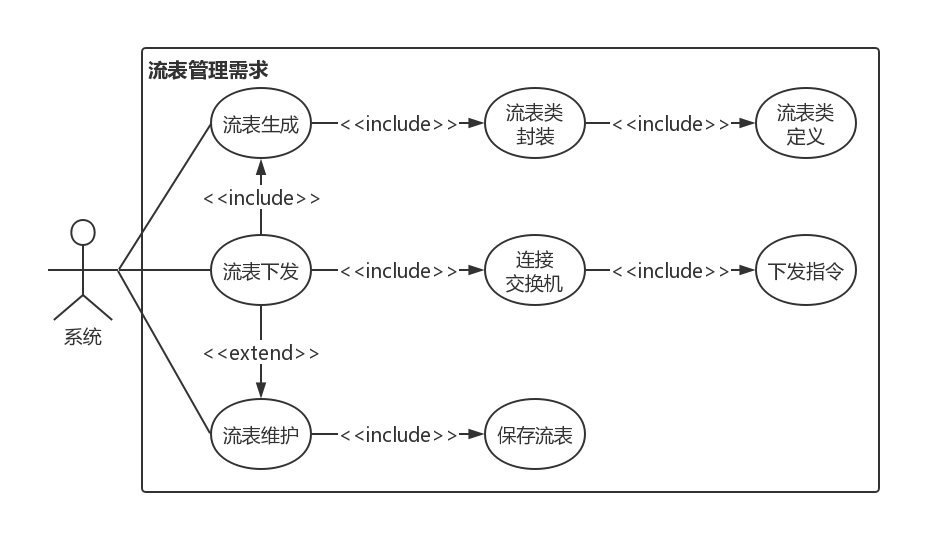


图3-4 流表管理用例图

流表生成：首先需要结合流表格式，给出流表类的定义，然后根据相应的数据依次填充，封装成流表类的形式，详细的用例情况见下表。

表3-5 流表生成用例详细情况

|  |  |
| --- | --- |
| 参与者 | 系统 |
| 用例描述 | 流表生成 |
| 前置条件 | 系统计算出转发路径 |
| 详细描述 | 1. 系统给出流表类的详细定义 2. 根据相应字段，封装为流表类的形式 |
| 后置条件 | 经过转换，得到SDN交换机识别的流表 |

流表下发：系统需要控制SDN交换机，执行下发的流表指令，从而达到控制转发的功能，详细的用例情况见下表。

表3-6 流表下发用例详细情况

|  |  |
| --- | --- |
| 参与者 | 系统 |
| 用例描述 | 流表下发 |
| 前置条件 | 系统得到流表 |
| 详细描述 | 1. 系统生成相应的流表指令 2. 控制器与SDN交换机构建通信通道 3. 执行相关流表 |
| 后置条件 | 交换机下发流表成功，执行系统定义的匹配规则 |

流表维护：下发的流表存储在交换机中，执行匹配转发功能，但是这些流表本身也会存在重复、冲突的问题，因此内存中还需要对下发的流表进行保存、维护，详细的用例情况见下表。

表3-7 流表维护用例详细情况

|  |  |
| --- | --- |
| 参与者 | 系统 |
| 用例描述 | 流表维护 |
| 前置条件 | 系统执行流表下发功能 |
| 详细描述 | 1. 系统下发相关流表 2. 内存中保存流表数据 3. 当有新的流表到来时，判断是否重复、冲突，并调用维护模块处理 |
| 后置条件 | 内存中保存流表数据，并对其进行管理 |

1. **发布订阅系统需求**

发布订阅需求包括主题管理需求和消息接口需求，用户调用消息接口进行发布、订阅注册，不同用户通过主题管理联系在一起，主题消息在SDN网络中转发，从而为用户提供数据传输服务。

1. **主题管理需求**

主题是发布订阅系统连接的纽带，不同用户通过相同的主题进行匹配，关注的主题不同，则系统不会将相应的数据转发给用户，因此系统中需要对主题进行管理。为实现主题管理需求，需完成主题存储、主题编码、主题树下发等功能，主题管理的用例图如下所示。

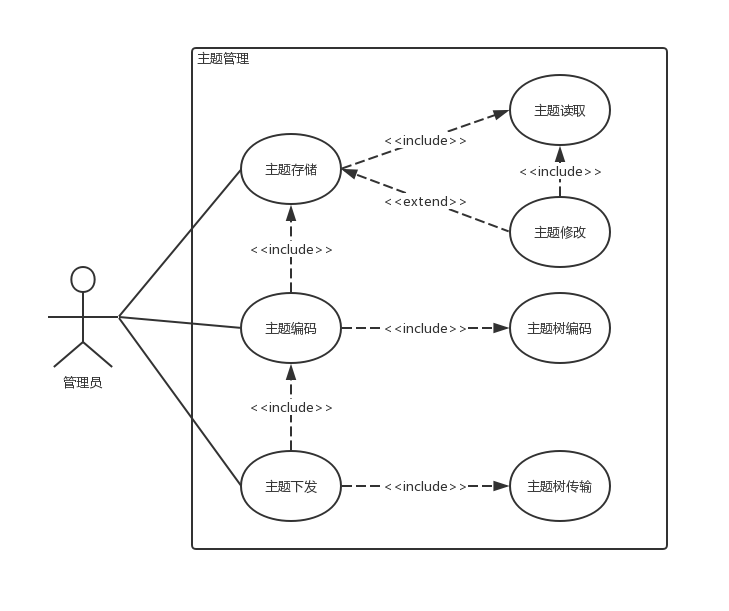


图3-5 主题管理用例图

主题树存储：管理员可以获取存储的主题树信息，对主题树进行修改操作后，需要将修改后的主题树保存到XML文件中，用例详细情况见下表。

表3-8 主题树存储用例详细情况

|  |  |
| --- | --- |
| 参与者 | 管理员 |
| 用例描述 | 主题树存储 |
| 前置条件 | 管理员启动，调用主题树获取接口 |
| 详细描述 | 1. 管理员通过接口将主题树读取到内存中 2. 管理员对主题树进行修改 3. 系统将修改后的主题树保存到XML文件中，便于再次读取 |
| 后置条件 | 修改后的主题树保存在XML文件中，管理员能够再次获取 |

主题树编码：通过主题编码方式，将每个主题节点编码成IPv6组播地址的形式，用例详细情况见下表。

表3-9 主题树编码用例详细情况

|  |  |
| --- | --- |
| 参与者 | 管理员 |
| 用例描述 | 主题树编码 |
| 前置条件 | 系统将主题树读取到内存中 |
| 详细描述 | 1. 管理员读取主题树信息 2. 调用编码方式，将每个节点编码成为IPv6地址 3. 将每个编码主题连接在一起，封装为编码主题树 |
| 后置条件 | 经过编码，系统获得编码主题树 |

主题树下发：管理员需要将编码主题树下发给各个集群控制器，用例详细情况见下表。

表3-10 主题树下发用例详细情况

|  |  |
| --- | --- |
| 参与者 | 管理员 |
| 用例描述 | 主题树下发 |
| 前置条件 | 主题树编码完成 |
| 详细描述 | 1. 系统获得编码主题树 2. 使用特定消息格式进行封装 3. 沿着管理路径下发给各个集群控制器 |
| 后置条件 | 经过传输，集群控制器获得编码主题树 |

* + 1. **消息接口需求**

消息接口是用户调用系统发布订阅服务的接口，为了提高转发效率，系统将服务抽象为一种资源，向用户暴露服务接口，用户向服务地址发起请求，系统接收后进行处理，从而提升系统的处理性能和扩展性。消息接口的用例图如下所示。

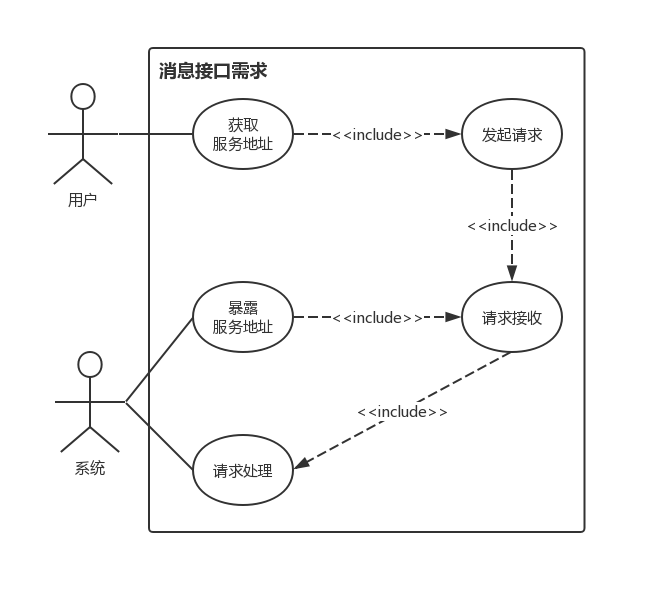


图3-6 消息接口用例图

用例详细情况见下表。

表3-11 消息接口用例详细情况

|  |  |
| --- | --- |
| 参与者 | 用户、系统 |
| 用例描述 | 消息接口 |
| 前置条件 | 系统对外暴露服务，用户获得服务地址 |
| 详细描述 | 1. 系统对外暴露服务地址 2. 用户获取服务地址，并向服务地址发起服务调用请求 3. 系统接收请求并处理 |
| 后置条件 | 用户可以得到系统提供的服务 |

1. **用户需求**

用户是系统服务的使用者，通过消息接口，可以获得多媒体传输服务和用户协商服务，因此该模块包含多媒体传输需求和用户协商需求。

1. **多媒体传输需求**

本系统的创新点之一是支持多媒体数据的传输，在完成SDN控制器、发布订阅系统功能后，就可以在这两者的基础上实现视频的播放。发布者和订阅者通过消息接口获得系统的服务，主题管理模块将相同主题的用户关联起来，经过路由计算、流表下发，最终在用户间建立通信链路，这样发布者将视频数据发送给订阅者，订阅者调用播放功能就能实现多媒体数据的传输。多媒体传输的用例图如下所示。

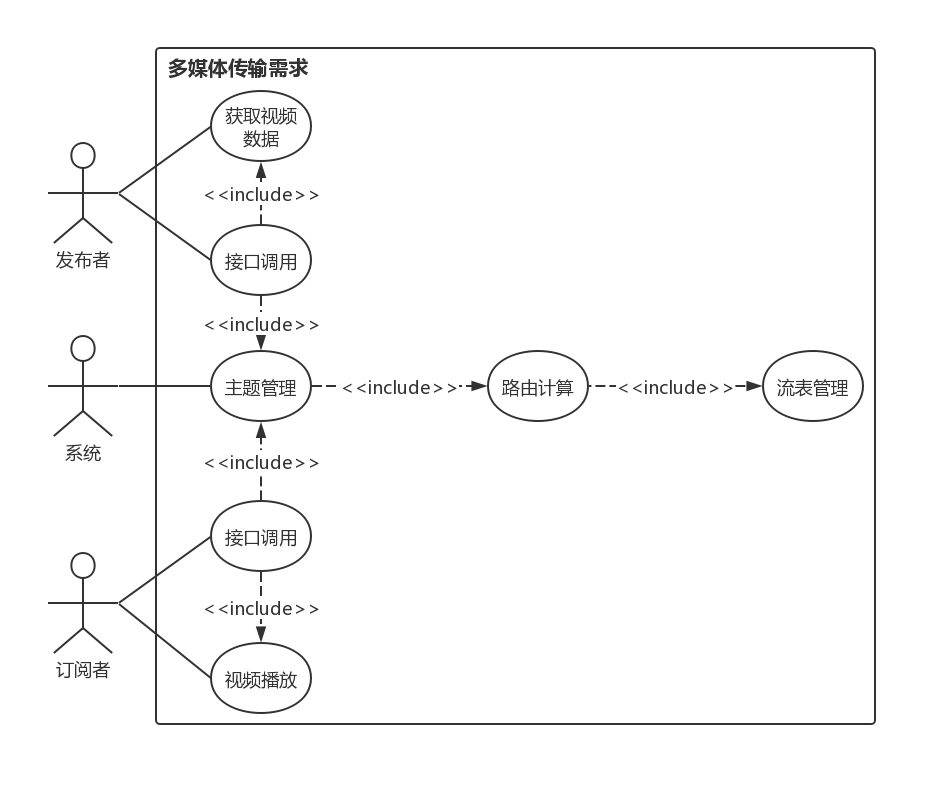


图3-7 多媒体传输用例图

多媒体传输的用例详细情况见下表。

表3-12 多媒体传输用例详细情况

|  |  |
| --- | --- |
| 参与者 | 发布者、订阅者、系统 |
| 用例描述 | 多媒体传输 |
| 前置条件 | 发布者、订阅者通过消息接口关注相同的主题 |
| 详细描述 | 1. 发布者、订阅者通过消息接口获取服务，并关注相同的主题 2. 发布者获取视频数据流 3. 发布者通过系统进行传输 4. 订阅者获得视频数据 5. 订阅者进行视频播放 |
| 后置条件 | 用户实现视频数据的播放功能，完成多媒体数据的传输 |

1. **用户协商需求**

为了给用户提供更好的服务，系统支持用户主动提出对时延、带宽的需求。运行时，若当前系统提供的时延、带宽方案无法满足用户要求，用户将相应的需求通过控制器发送给管理员，由管理员计算后下发处理策略，若计算结果可以满足，则将相应的计算结果下发至集群控制器；若无法满足，则通过控制器通知用户，由用户决定是否重新发起请求。具体用例图如下所示。

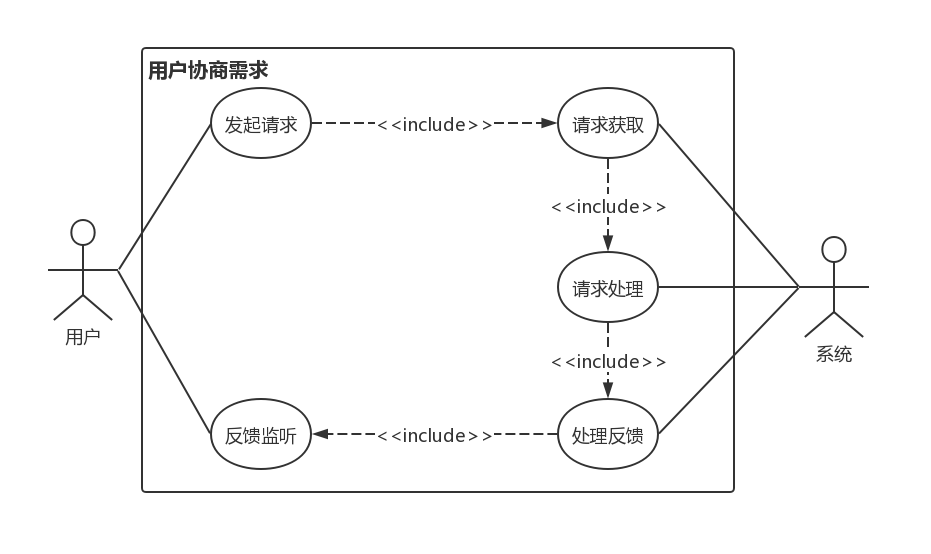


图3-8 用户协商用机制例图

用户协商机制的用例详细情况见下表。

表3-13 用户协商机制用例详细情况

|  |  |
| --- | --- |
| 参与者 | 用户、系统 |
| 用例描述 | 用户协商机制 |
| 前置条件 | 用户订阅某个主题 |
| 详细描述 | 1. 用户发起时延、带宽需求 2. 控制器上传给管理员 3. 若可以满足，则管理员将计算结果下发给控制器，由控制器进行后续处理；若无法满足，则将处理结果告知用户 4. 用户决定是否发起新的请求 |
| 后置条件 | 将用户的请求上传、分析，最终用户将得到需求的反馈结果 |

1. **本章总结**

本章主要对系统涉及的相关需求进行描述，分为SDN控制器需求、发布订阅系统需求和用户需求。SDN控制器需求包括拓扑管理需求、路由计算需求、流表管理需求，发布订阅系统需求包括主题管理需求、消息接口需求，用户需求包括多媒体传输需求、用户协商需求。关于上述需求的解决方案将在第四章概要设计中给出。

2. **系统概要设计**

第三章介绍了完成本系统需要实现的需求目标，本章将对系统各模块的设计思路进行详细阐述，介绍系统的架构设计和模块功能。

1. **系统架构设计**

系统架构是系统功能的直观体现，通过架构分析，读者可以对系统的整体架构、模块划分有一定的了解。本文结合了SDN物理网络的概念，在已有系统的基础上进行改进，下面介绍架构设计思路。

1. **已有系统架构**

在设计之初，首先需要对系统架构有一个清晰的表述，在已有系统中，采取的是虚拟环境下划分集群的方式，已有系统架构如下图所示。

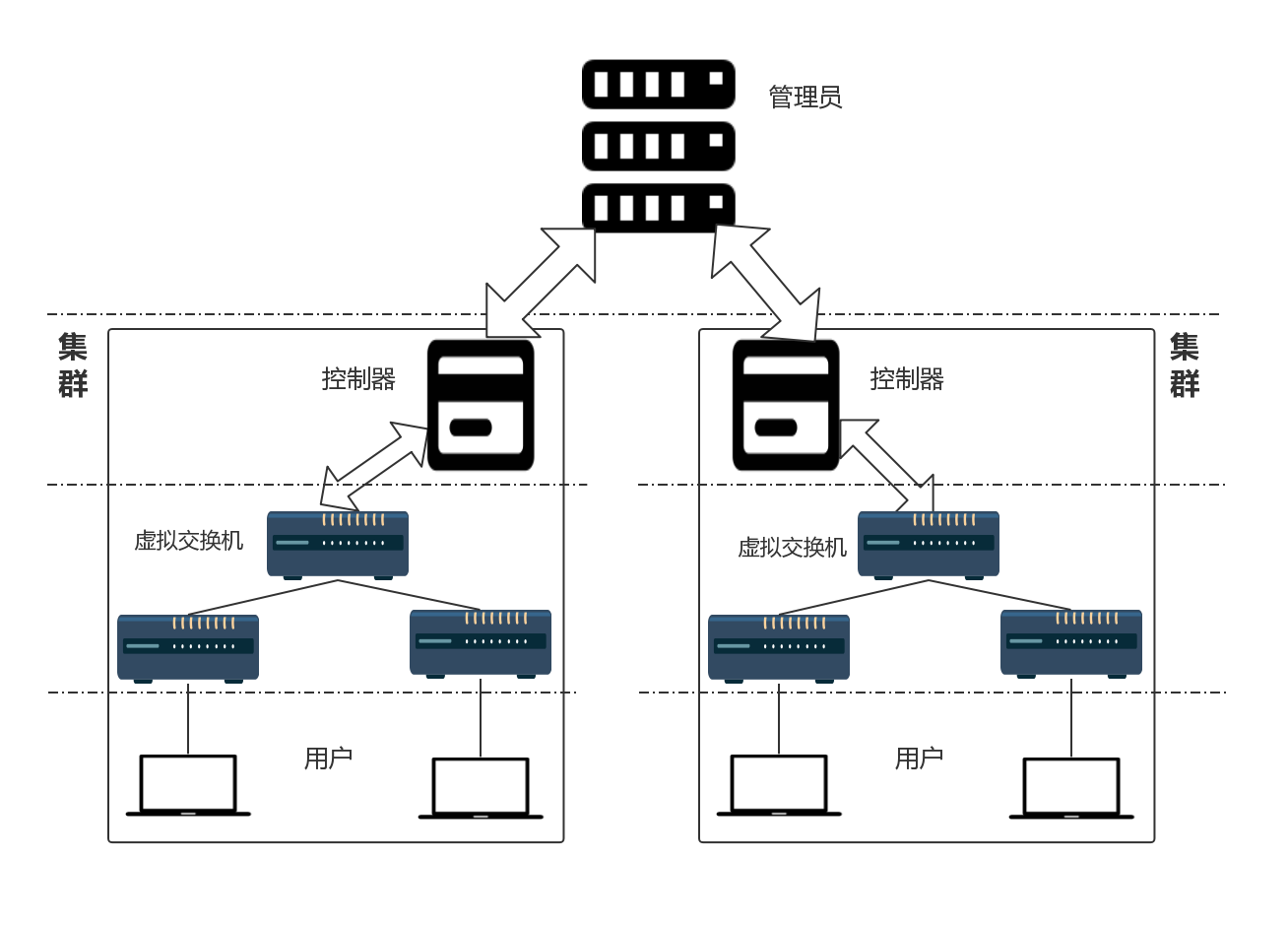


图4-1 已有系统架构图

从架构图中可以看到，已有系统中包含三个主要角色：控制器、管理员、用户。每个控制器管辖的范围称为一个集群，控制器承担了集群内的发布订阅事务处理；管理员站在全局角度和所有控制器交互，负责全网链路计算和约束值的下发；用户是具体的使用者，通过调用接口发起请求。

已有系统在虚拟环境下运行稳定，采用控制器、管理员划分集群的方式，有较高的效率，但是与本系统需求相比，已有系统存在诸多弊端，最明显的是，本系统需要与SDN物理交换机连接，数据转发的情况无法通过虚拟软件获得，同时控制器也需要在物理拓扑中获取邻居集群的连接信息，这些都是已有系统架构无法实现的。同时，已有系统在某些方面也存在改进空间，如控制器功能过于复杂，路由计算、流表管理、消息监听这些都在控制器端进行，虽然满足了SDN网络的“控制-转发”分离的原则，但是当用户过多时，控制器压力太大。针对上述问题，本系统需要在已有系统架构的基础上进行改进。

1. **系统架构改进**

本文研究的主要内容包括发布订阅系统在SDN物理环境下的应用、多媒体传输机制、消息传输的质量保障方案等，结合实际情况，改进后的系统架构如下图所示。

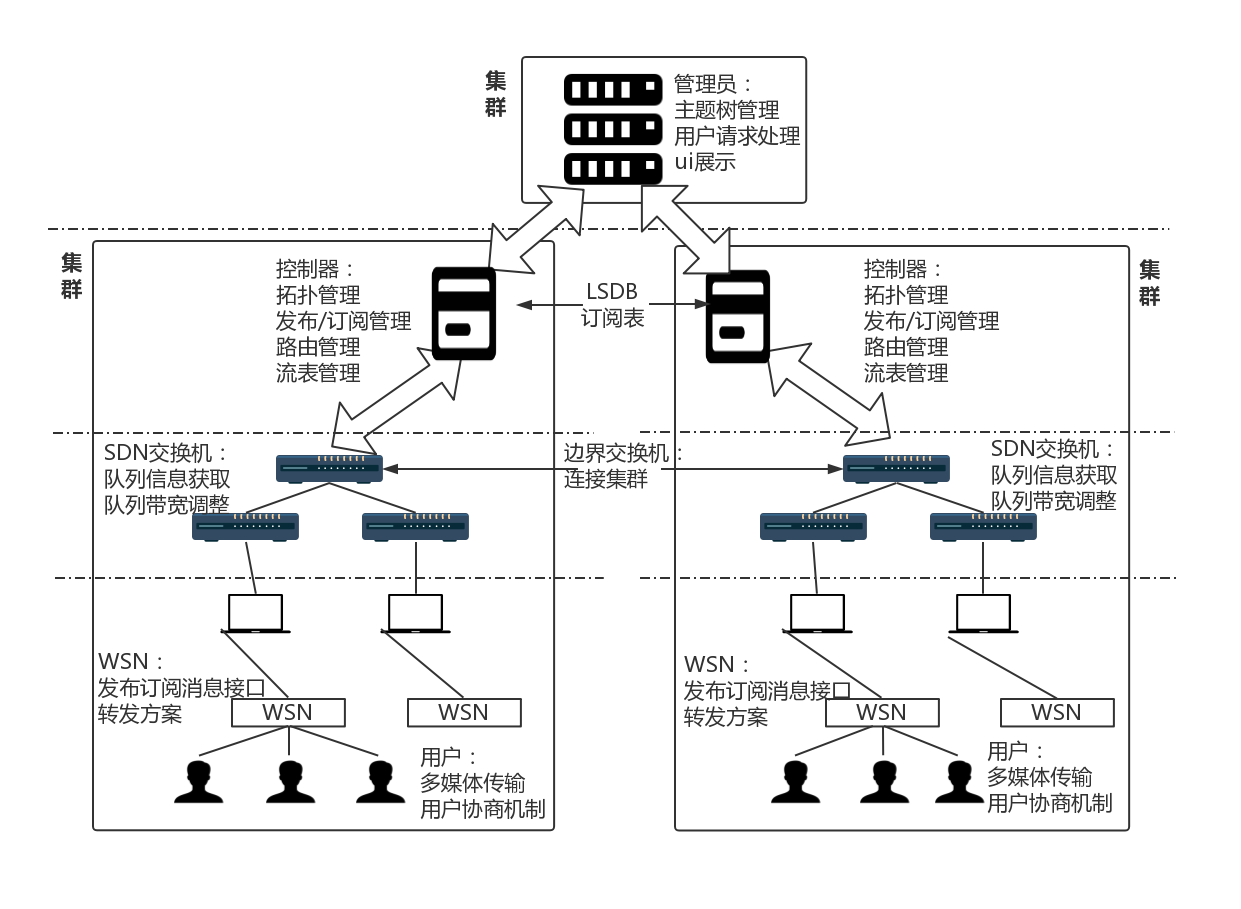
****

图4-2 改进后的系统架构图

从图中可以看出，改进后的系统吸收了已有架构的优点，通过划分集群的方式将复杂系统单元化，分别进行管理。集群控制器承担了集群内的发布订阅事务处理，由于在物理环境中，还需要具备拓扑管理、路由管理、流表管理的功能；管理员除了控制器职责功能外，还负责处理用户发起的时延、丢包率请求，以及主题树的管理和ui展示；SDN交换机提供队列转发功能，系统可以获取队列信息，并通过队列带宽调整提升系统性能；用户是系统的使用者，可以发起相应的请求，包括多媒体数据的发布订阅请求和时延带宽需求。

同时我们注意到，在用户与控制器之间还多了一个WSN层，它负责用户与控制器间发布订阅消息的接收，这是为了缓解控制器运行时的压力，将消息的接收和预处理过程下放到用户层面，这样在不影响系统功能的同时提升系统性能。同时WSN还负责将接收到的主题消息分发给指定用户，因此需要具备消息转发方案。

在改进后的系统架构中，存在集群控制器、管理员、WSN、用户等多个模块，并由SDN交换机进行连接，下面将分别介绍这几个模块的设计思路。

1. **控制器上发布订阅功能设计**

基于SDN的发布订阅系统其主要功能部署在集群控制器上，系统采用划分集群的方式进行管理，集群内部由交换机和主机构成，其中负责集群内部连接的交换机称为群内交换机，连接不同集群的交换机称为边界交换机，集群控制器和用户都部署在主机上。集群结构图如下所示。

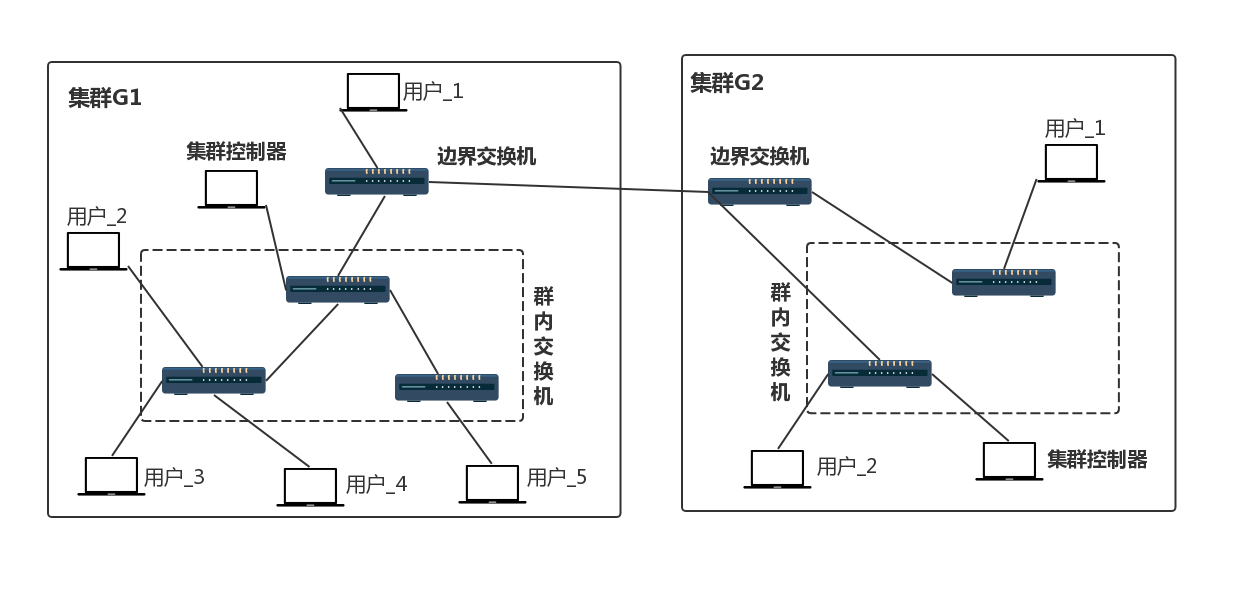


图4-3 集群结构图

集群控制器负责集群内的发布订阅处理，分为拓扑管理、发布订阅管理、路由管理、流表管理这几个子模块，拓扑管理模块使得集群控制器能够感知邻居集群的存在，用户的订阅请求保存在本地发布订阅表中，并与其他集群的发布订阅信息构成全局发布订阅表，路由管理模块结合全局订阅信息、拓扑连接情况，可以计算出主题转发路径，使得主题消息可以在不同集群间转发，到达目标集群后再广播通知订阅用户，流表管理模块负责将主题路径转换为匹配流表，并下发给交换机执行，这样就可以实现SDN“软件定义网络”的功能。集群控制器中发布订阅系统的功能图如下所示。

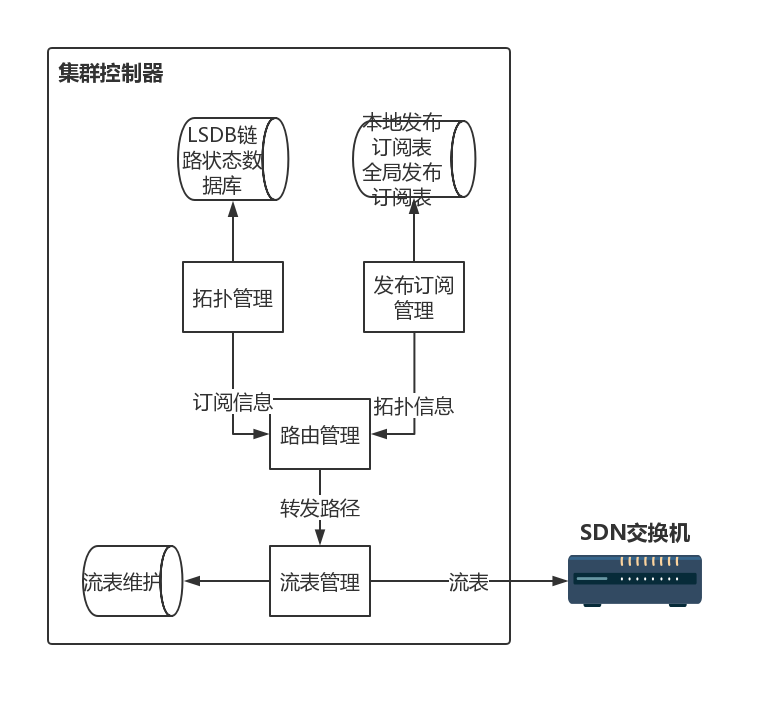


图4-4 发布订阅系统功能图

1. **拓扑管理**

拓扑是物理网络的连接情况，结合系统集群结构图，可以知道连接分为集群内的交换机与交换机连接、交换机与主机连接，以及集群间的边界交换机与边界交换机连接，因此拓扑管理模块分为集群内拓扑管理和集群间拓扑管理。拓扑管理的功能模块图如下所示。

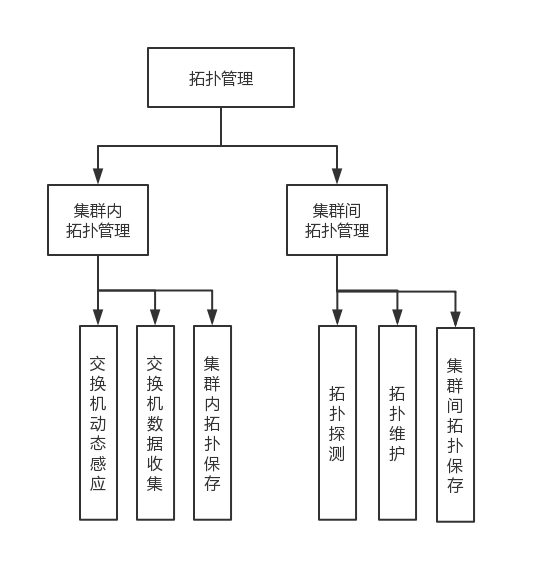


图4-5 拓扑管理功能模块图

集群内部由SDN交换机和主机构成，主机是信息接收的终端，数据只会在交换机之间进行转发，因此集群内的拓扑信息可以归结为SDN交换机的连接情况。SDN交换机具有动态感应的功能，当交换机的端口接入网线时该端口会点亮，同时SDN交换机的核心处理器会记录该端口的连接信息，因此集群控制器可以通过访问SDN交换机的方式获得交换机的连接情况，并将集群内的拓扑连接信息保存在集群内拓扑中。

集群间拓扑管理负责维护不同集群之间的连接关系，当新集群加入时必须具备感知功能，而由于网络的不确定性，这种连接关系可能因为集群控制器的故障、边界交换机连接的断开而失效，因此还需要具备维护功能，能够将失效集群从集群间的拓扑数据库中删除。本系统采用的是OSPF协议，集群启动后，会触发拓扑探测操作，即经过初始化后将本集群信息封装起来，向所有对外端口发送Hello消息进行探测，邻居集群收到消息后，经过分析处理，再将本集群的信息封装在ReHello消息中，反向发送给初始集群，初始集群收到解析后，发送FinalHello消息，并最终在两个集群间建立双向连接；拓扑维护的过程是在建立连接的集群之间发送LSA（链路状态）心跳消息，并将邻居集群发来的LSA消息保存在本地LSDB（链路状态数据库）中，若超时则认为该集群失效，将其从LSDB中删除。集群间的拓扑管理流程如下所示。

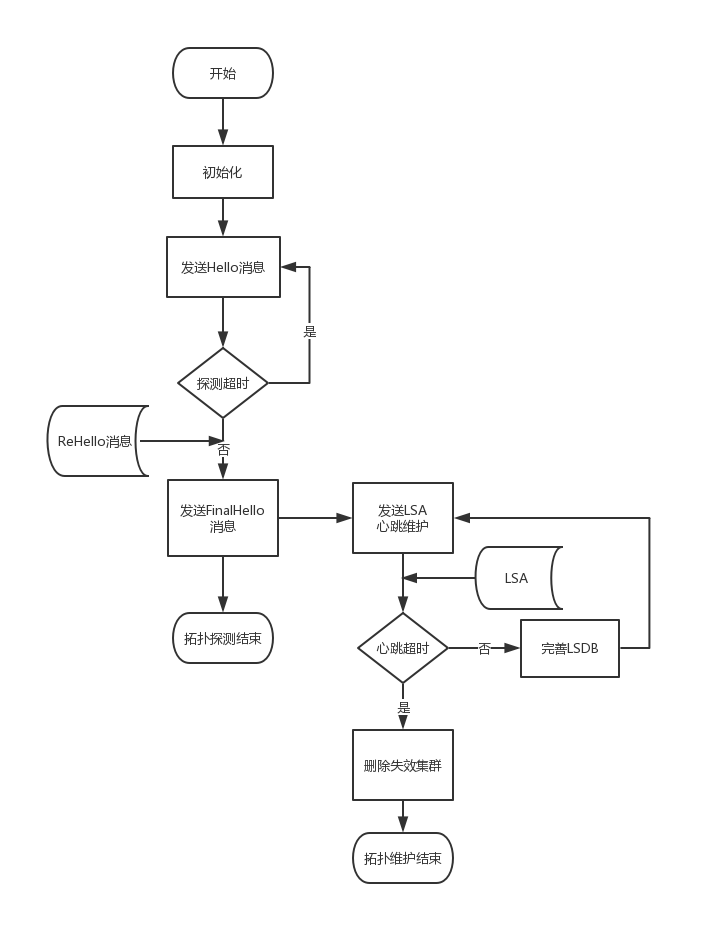


图4-6集群间拓扑管理流程图

集群间通过定时交换LSA的方式进行心跳维护，而LSA中也包含了本集群内的订阅信息、邻居集群信息等，方便集群控制器进行发布订阅管理和拓扑构建，收集到的所有LSA共同构成了LSDB，内存中使用Map散列表的形式进行存储，系统通过访问LSDB就可以得到各个集群的连接状态、集群内的订阅情况。拓扑管理模块需要对链路状态信息LSA、链路状态数据库LSDB进行定义，关键数据结构如下所示。

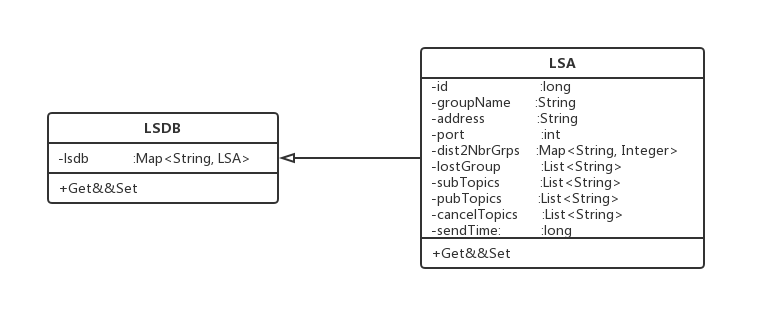


图4-7 LSA、LSDB类图

图中展示了拓扑管理的关键数据结构，下面介绍各字段含义。

表4-1 拓扑管理关键数据结构定义

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 类名 | 类功能 | 字段名 | 字段类型 | 字段功能 |
| LSA | 链路状态信息，包含本集群的邻居信息、发布订阅信息等，在相邻集群间负责心跳维护 | id | Long | LSA消息的序号，避免后发先至 |
| groupName | String | 集群名称 |
| address | String | 本集群控制器所在地址 |
| port | int | 本集群控制器的监听端口 |
| dest2NbrGrps | Map<String, Integer> | 与邻居集群的距离，key:邻居集群名，value:与邻居集群的距离 |
| lostGroup | List<String> | 丢失的邻居集群 |
| subTopics | List<String> | 本集群的订阅主题 |
| pubTopics | List<String> | 本集群的发布主题 |
| cancelTopics | List<String> | 本集群取消订阅的主题 |
| sendTime | long | LSA的方式时间，用于判断是否超时 |
| LSDB | 链路状态数据库，保存了收集到的所有LSA信息，为拓扑构建提供支持 | lsdb | Map<String, LSA> | 本集群收到的LSA消息，用于计算出全网拓扑连接情况 |

1. **发布订阅管理**

集群控制器的发布订阅管理，指的是维护用户的发布订阅信息，包括集群内部用户请求，以及集群间广播的LSA信息。集群内的发布订阅信息是本集群用户发起的服务请求信息，集群控制器在收到用户请求后分析判断，如果是发布、订阅请求，则将该主题保存在本地发布订阅表中，如果是取消发布、取消订阅请求，则需要在集群内将该主题删除，若果是时延带宽需求则不作处理，同时通过LSA心跳消息，定时向全网广播本集群的发布订阅信息；集群间的发布订阅管理负责记录从邻居集群发来的LSA消息，消息中包含了该集群内部发布订阅情况，系统需要将各个集群的发布订阅信息保存在全局发布订阅表中。发布订阅管理的流程图如下所示。

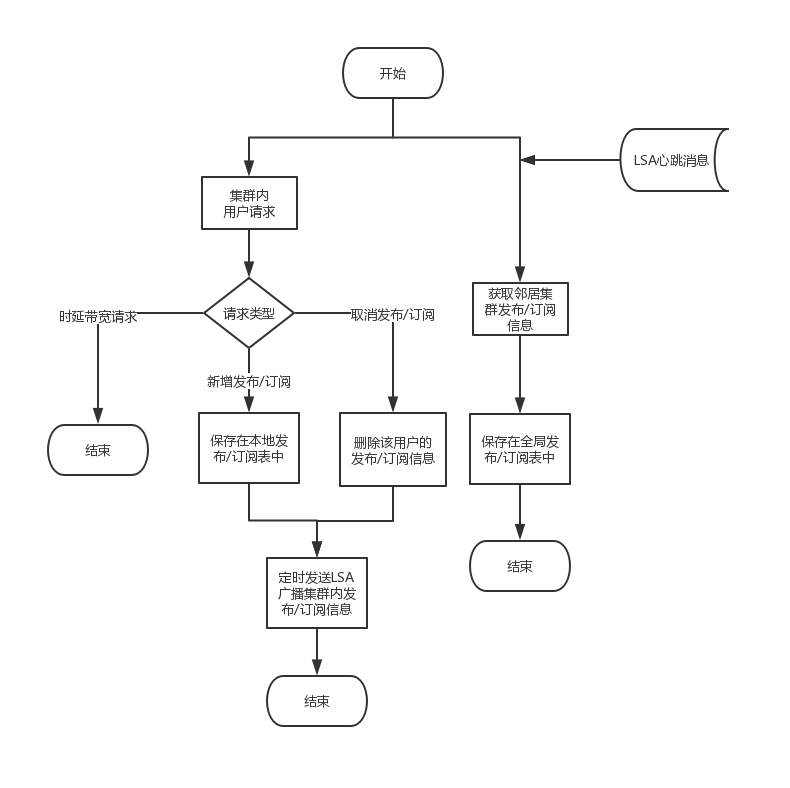


图4-8 发布订阅管理流程图

1. **路由管理**

在消息网络中，路由指的是消息传输的路径，在本系统中，路由管理模块需要计算出一条消息的转发路径。集群内部规模较小，一般为多台SDN交换机与主机直接连接，且连接情况较为固定，因此采用广播的方式进行数据的传输，也就不需要考虑路由的计算问题，而在集群间传输消息时需要计算出集群间的转发路径，使得消息能够在不同集群间传输。路由管理模块考虑的是集群间的转发路径，路由计算时根据消息种类的不同，分为管理路径的计算和主题路径的计算。

管理路径负责管理消息的传输，集群控制器接入到SDN网络后，能够通过管理路径与管理员交互，为了减小网络带宽损耗，各个集群控制器只需要计算出到达管理员的最短路径，结合上述定义，我们不难发现管理路径计算的关键在于维持当前路径的基础上，计算出新增集群与管理员集群的最短距离，本系统中采用的是Dijkstra最短路径算法，计算时将管理员节点设为源节点，并将剩余节点分为两组：unit\_1存储确定最短路径的节点，unit\_2保存未确定节点，每次从unit\_2中取出最短距离的节点node，添加进unit\_1中，并遍历node的所有邻居节点，若邻居节点经过node后与源节点的距离更短，则更新该距离，算法会重复上述过程，直至unit\_2中所有节点计算完成，最终返回unit\_1即为所有节点与源节点的最短距离。

上述计算过程中，使用到的节点、邻居距离信息都可以通过LSDB数据库获得，因此拓扑管理模块为管理路径计算提供了保障。Dijkstra算法的PAD图如下所示。

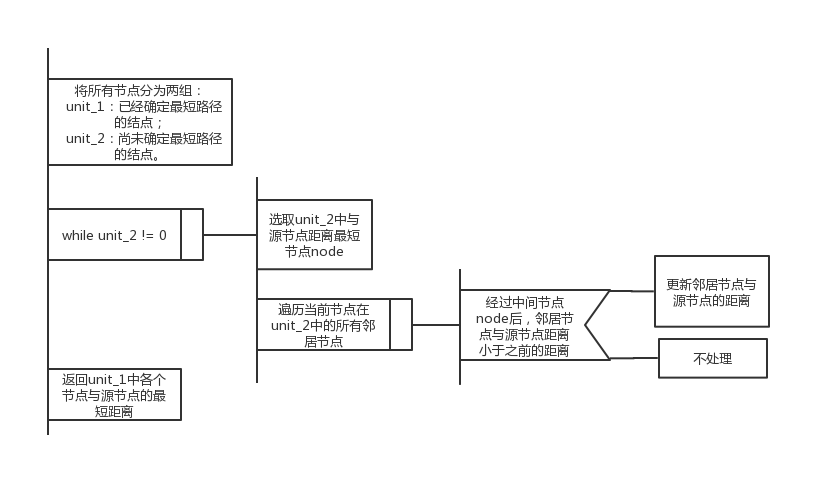


图4-9 Dijkstra算法PAD图

主题路径负责发布订阅消息的传输，消息从发布者经过多个集群转发后，能够到达订阅该主题的集群，在集群内广播后，最终通知给订阅该主题的用户。算法的输入是全网拓扑连接情况和各个集群的订阅信息，针对特定主题，不是所有集群都会存在发布、订阅情况，因此主题路径上包括订阅该主题的节点，以及部分中间节点，从上述需求中可以看到，主题路径的计算结果是多个节点构成的非连通图，Steiner算法用于计算多源最小非连通子图，这个结果与主题路径需求一致，因此系统中采用了Steiner算法来计算主题路径。Steiner算法基本步骤包括：

1. 遍历所有节点集合，构造完全连通图，得到各个节点间的距离
2. 根据选取的节点，计算出部分节点在所有节点中的联通子图
3. 分析联通子图中包含的中间节点，求解扩展子图的最小生成树
4. 删除最小生成树中度为1的节点，返回计算结果

上述过程中使用的所有节点连接情况、各个集群内的发布订阅信息都可以由LSDB数据库得到，因此拓扑管理模块和发布订阅管理模块为主题路径的计算提供了保障。Steiner算法的PAD图如下所示。

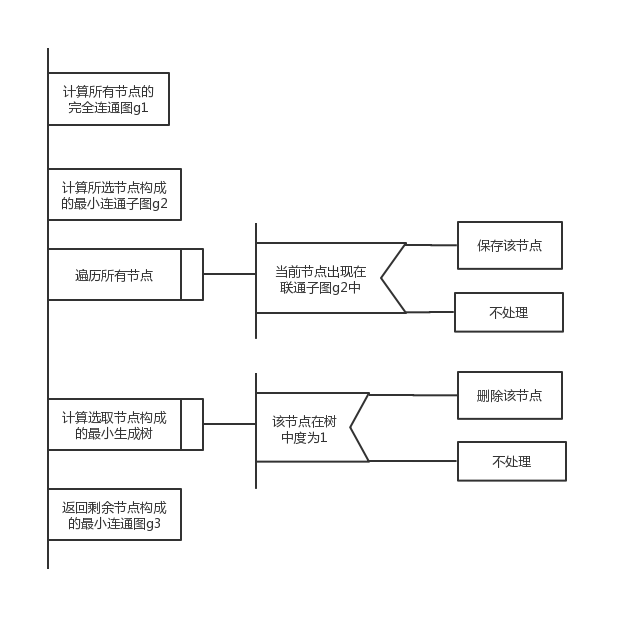


图4-10 Steiner算法PAD图

1. **流表管理**

流表管理模块是发布订阅系统与SDN网络结合的核心模块，交换机能够识别的是流表指令，因此路由模块生成的转发路径需要使用该模块转换为匹配流表，并下发给交换机执行，从而达到对交换机的“控制-转发”功能，同时系统还需要对下发的流表进行维护，当出现冲突时能够进行处理，流管管理分为流表生成、流表下发、流表维护等子模块，流表管理的功能模块图如下所示。

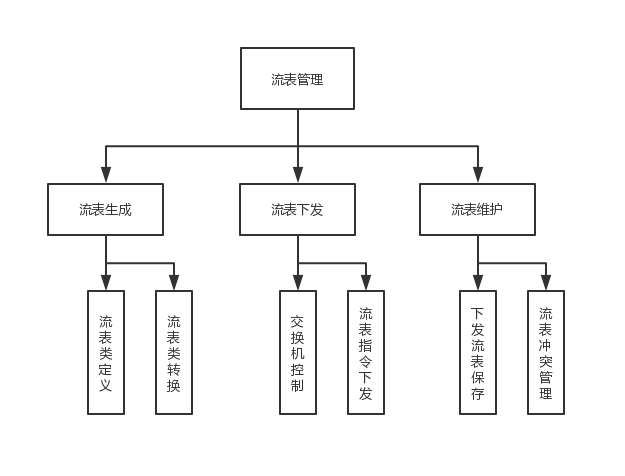


图4-11 流表管理功能模块图

流表管理首先需要对流表类有一个定义，程序中流表类的定义是参照SDN交换机流表项的格式，将各个字段抽象出来，转换时只需要从转发路径中取出目的地址、转发端口等参数填充进去，就可以得到相应的转发流表。

经过流表转换后我们需要将生成的流表下发到交换机，SDN交换机内部存在核心处理器，其中保存了交换机硬件匹配的所有指令，系统采用的方法是使用程序访问交换机处理器，这样就可以远程操控交换机，下发我们生成的流表，实现流表转发的功能。

下发的流表在交换机中执行匹配转发功能，如果对下发的流表不进行维护管理，会产生冲突现象，流表冲突如下图所示。

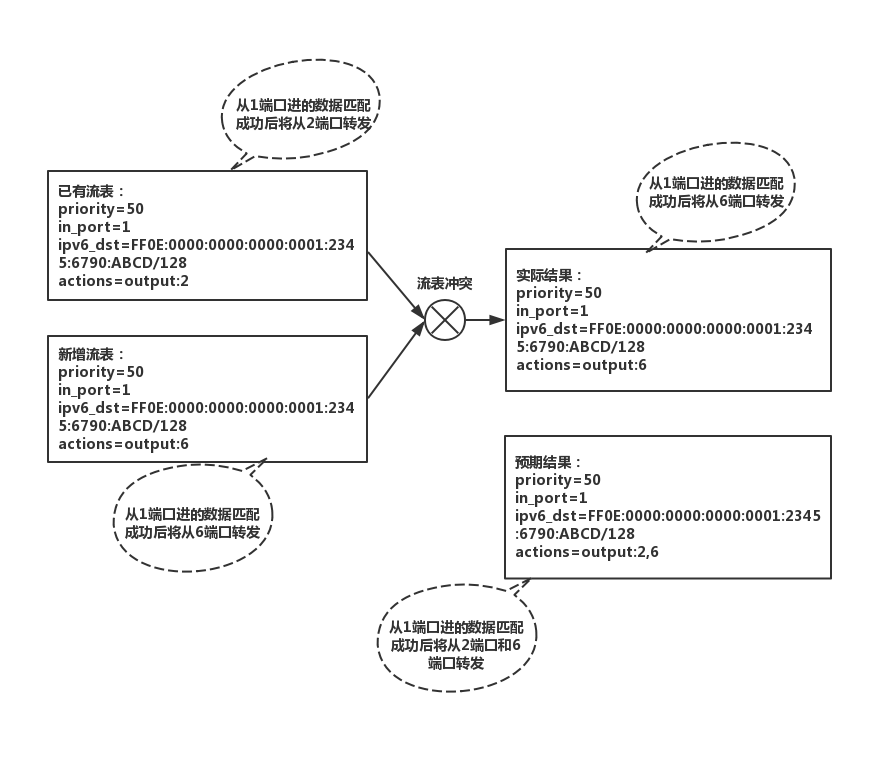


图4-12 流表冲突示意图

流表的转发地址、进端口号、优先级等字段一致时会产生冲突现象，后下发的流表将替换先下发的流表，我们预期的目标是将两条流表合并，实现多个端口转发的功能，但实际结果只会执行后下发的流表项，需要使用流表管理模块对冲突的流表进行合并处理。

1. **管理员功能设计**

管理员集群包含控制器集群的所有功能，整个网络中可以包含多个控制器集群，但只存在一个管理员集群，管理员模块还可以实现主题树管理、用户请求处理、ui展示等功能，管理员的功能模块图如下所示。

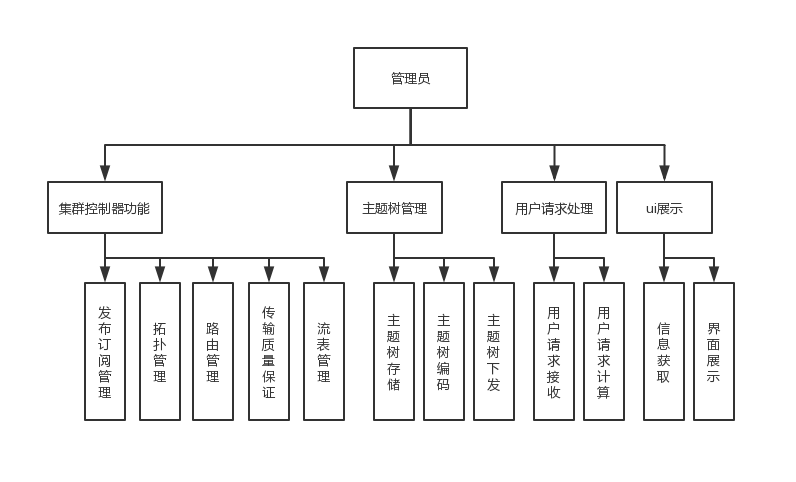


图4-13 管理员功能模块图

1. **主题树管理**

主题通过管理员进行增加、删除和修改，客户端订阅和发布相关消息时，首先会判断主题是否存在，如果该主题不存在，则无法进行发布、定义操作。主题树是多叉树，在父子节点之间存在直接关系，相邻的兄弟节点之间没有关联。主题树的管理包括主题树的存储、编码和下发，主题管理的流程图如下所示。

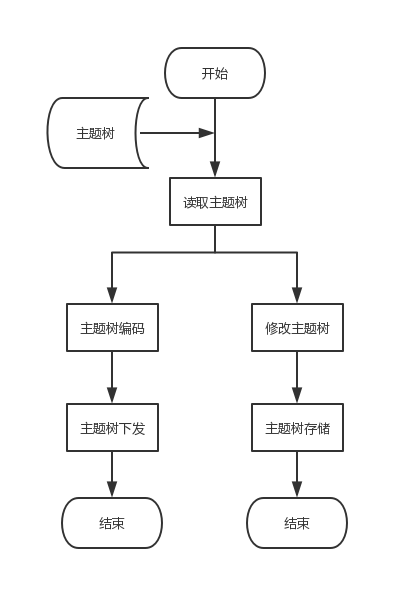


图4-14主题树管理流程图

管理员启动时会从XML文件中获取主题树信息，并在管理员ui界面中展示。主题树将通过管理路径下发给各个集群控制器，管理员通过ui界面或调用API接口修改主题树，修改后的内容将写回到XML文件中，并向集群控制器广播更新后的主题树信息。

主题除了消息分类的作用外，还具有地址编码的功能。消息在不同用户间采用IPv6组播的方式进行传输，在组播地址与主题间存在一一映射的关系，管理员能够将主题编码成组播地址，并确保编码地址的唯一性。

1. **用户请求处理**

用户请求处理模块负责接收用户针对特定主题的时延带宽需求，结合当前全网拓扑情况，计算出合理的带宽分配方案并反馈给用户。

在管理员处保存了各个集群的订阅信息，以及全网的拓扑连接情况，因此在处理用户请求时，首先需要计算出指定主题的转发路径，得到消息从发布者到订阅者经过的集群，由于采用划分集群的方式进行管理，各个集群的规模相近，因此在计算时不考虑集群个体的差异性，认为消息传输经过每个集群的时延相等，这样就可以使用时延需求除以该转发路径经过的集群个数，得到每个集群的时延要求，从而为每个集群下发新的时延约束结果。用户请求处理流程如下所示。

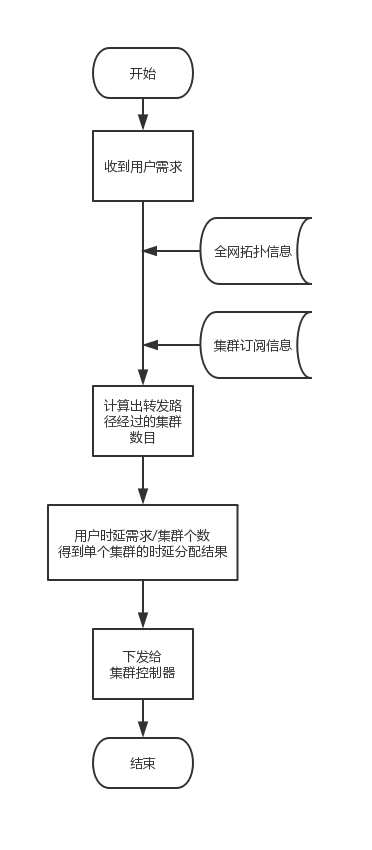


图4-15 用户请求处理流程图

1. **ui展示**

ui展示模块负责将管理员收集到的信息展示出来，同时提供界面操作方式，便于管理者下发集群配置信息。集群控制器会通过管理路径上报自己集群内部的运行信息，因此管理员ui界面需要展示各个集群的交换机运行状态、主机连接情况等，同时管理员集群还对主题内容进行管理，因此ui界面上需要列出主题树的相关内容。管理者可以通过界面对这些信息进行修改操作，如果是集群配置信息，相应的数据将发送给集群控制器，如果是主题树信息，修改的数据将写回到XML文件中。ui展示的交互图如下所示。

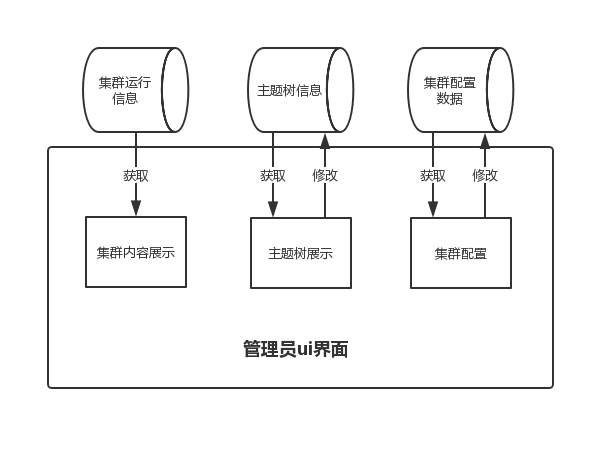


图4-16 管理员ui交互图

1. **WSN功能设计**

WSN部署在用户所在主机上，负责为用户提供服务调用接口，同时网络中传输的主题消息到达主机后由WSN统一收集，并根据数据转发方案分发给指定的用户。该模块需要完成消息接口、数据转发方案，WSN功能模块如下所示。

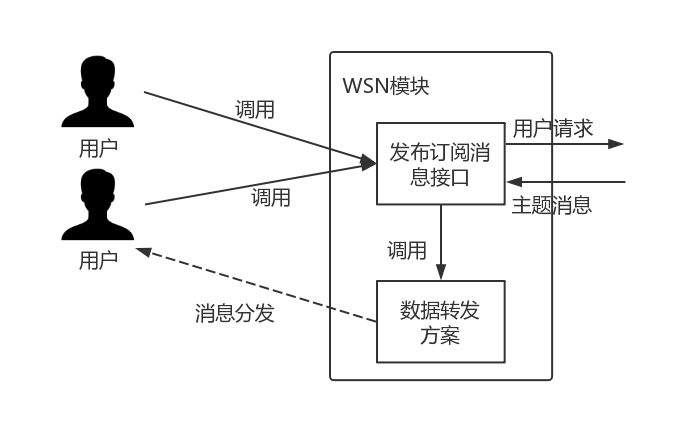


图4-17 WSN功能模块图

1. **消息接口**

WSN模块是中间层，用于连接用户和集群控制器，用户调用WSN的消息接口模块获得发布订阅服务，WSN设立的目的就是为了缓解集群控制器运行时压力，将发布订阅消息的注册、内容检测等过程下放到用户层面，这样在不影响系统功能的同时带来性能上的提升。

用户通过消息接口方式与WSN层进行通信，从而获得发布订阅服务，具体传输时使用了web service，关于该方式的定义在第二章中进行了介绍，WSN收到用户请求后首先进行内容分析，根据订阅请求、发布请求、时延带宽需求的不同分别进行处理，如果是订阅请求，会检验用户订阅主题是否在，存在则向集群控制器汇报用户新增订阅情况，并监听对应主题的内容，不存在就要通知用户订阅失败；如果是发布请求，WSN将计算出新的发布地址并反馈给用户，这样用户只需要向指定的发布地址推送消息就可以实现数据的传输；如果是时延带宽需求，WSN会判断该用户是否存在相应的订阅，存在且请求合理则向集群控制器上报用户需求。消息接口运行流程图如下所示。

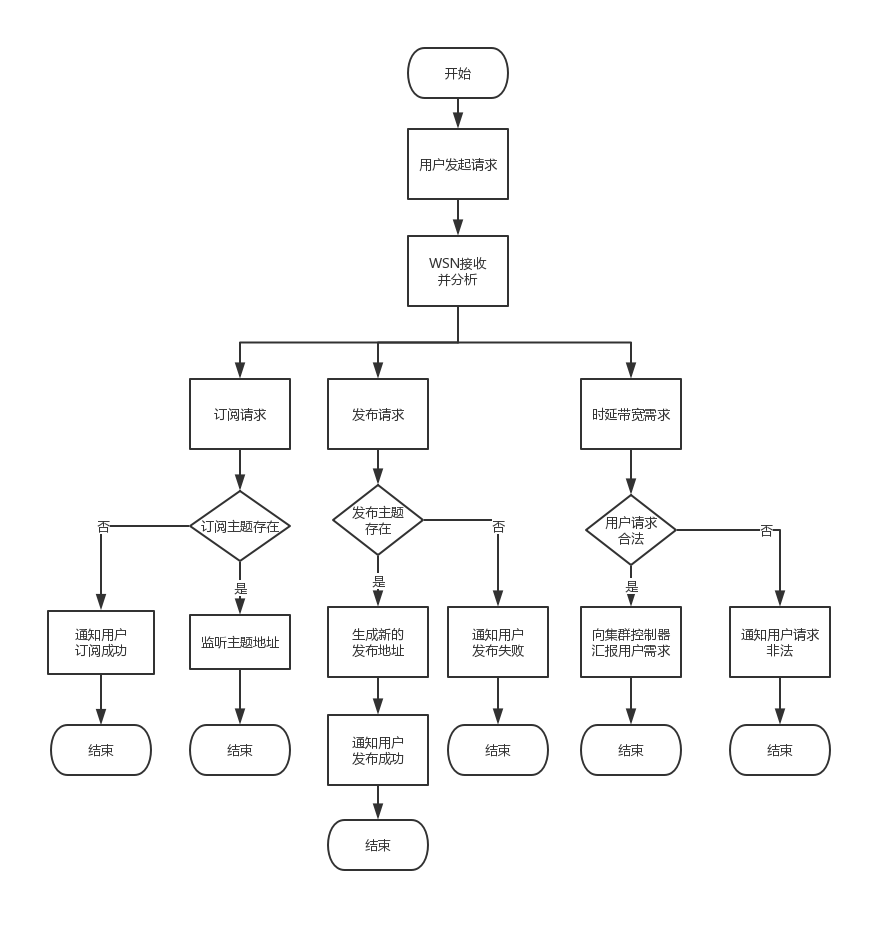


图4-18 消息接口运行流程图

1. **数据转发方案**

WSN负责将接收到的主题消息发送给实际订阅用户，真实网络环境下同一主机上存在多名用户，数据在转发过程中可能存在阻塞、丢失等情况，为了保证数据的稳定传输，WSN在数据接收、转发时需要提供传输保证，该模块的功能就是保障数据的稳定、高效传输，是系统整体质量保证方案的一环。

已有系统中，针对数据转发采取的是生产者-消费者模式，将消息的接收线程视为生产者，将处理线程视为消费者，当接收到订阅信息时，生产者将消息放在缓冲队列中，同时触发处理线程的处理功能，将特定主题的消息从队列中取出，并转发给订阅该主题的用户。这有两个好处，一方面，这是基于队列实现，使用队列缓存到来的消息，能够有效降低丢包率，另一方面，解决了生产者、消费者间处理速度的不同步问题。生产者-消费者的运行示意图如下所示。

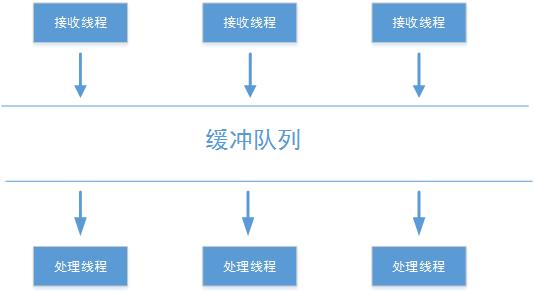


图4-19 生产者/消费者示意图

但是在实际使用过程中，我们发现该模式存在一些不足，一方面队列的大小有限，不可能一直缓存，另一方面，我们无法控制队列，不能对其中的消息采取负载均衡策略。因此需要对生产者-消费者模式进行改进，本系统使用了Reactor模式来解决上述问题。Reactor模式的运行示意图如下所示。

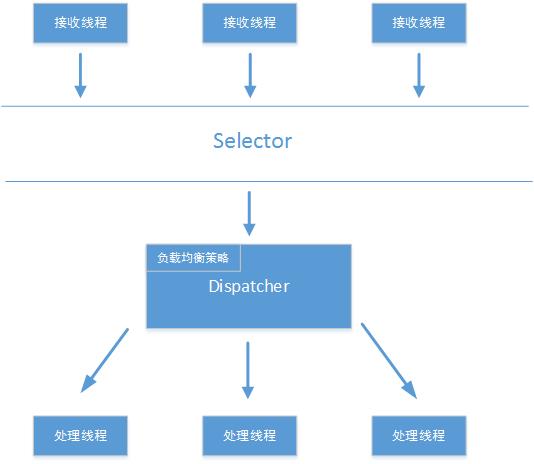


图4-20 Reactor模式示意图

消息到达WSN后首先存放在selector队列中，这就形成了一级缓存，同时采用事件驱动机制，触发分派器dispatcher从selector中获取数据，并结合当前用户订阅情况进行处理，这里由于我们可以自行控制分派过程，因此控制力度更细，在系统实现过程中还添加了相应的负载均衡策略，用于在主队列缓存数量过大时丢弃耗时长的任务，从而进一步提升系统性能。

1. **发布订阅系统的QoS保证方案设计**

本系统的核心功能之一是提供发布订阅系统的QoS保证方案，数据以数据包的形式在网络中传输，系统设计的重点是确保这些数据包的稳定、实时传输。交换机的每个端口包含多个队列，数据包就是在这些队列中进行转发，而SDN可以获得交换机各个端口中队列的运行时数据，包括队列进出字节数、丢失数据包个数等，结合这些数据就可以分析当前网络的拥塞情况，进一步使用队列带宽调整算法为发布订阅系统提供质量保证。系统质量保证方案的原理图如下所示。

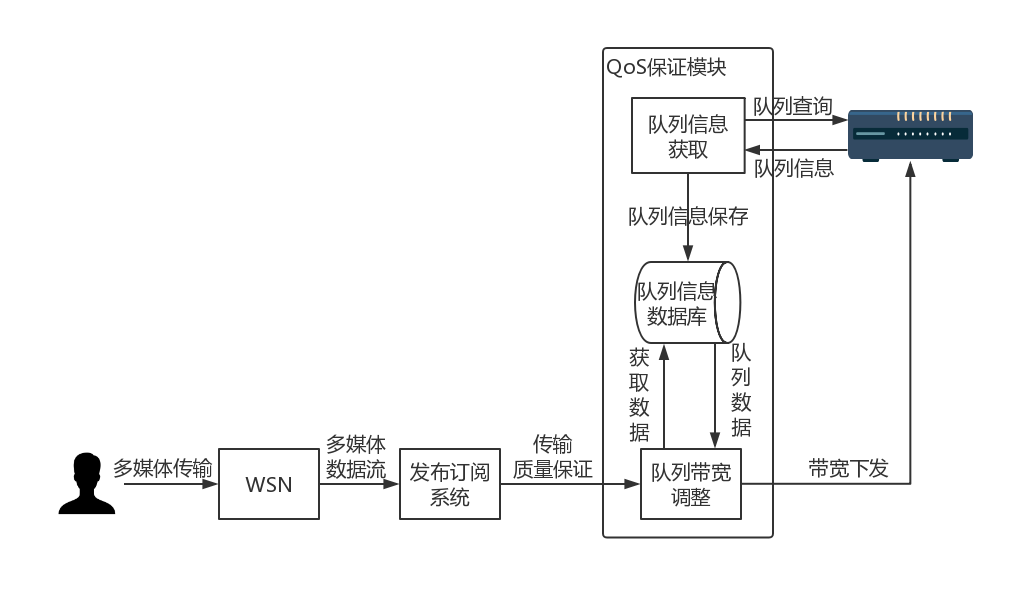


图4-21 质量保证方案原理图

1. **队列信息获取**

SDN交换机每个端口包含多个队列（最多存在8个队列），队列信息指的是交换机端口中每个队列的运行情况，包括进出队列字节数、传输的数据包个数等，SDN交换机提供了相应的查询指令方便我们获得队列数据，系统提供的质量保证方案就是结合交换机队列运行状况，动态调整各个队列的带宽大小，从而获得数据的稳定传输。系统需要使用Queue队列类存储队列信息，以及QueueMap保存交换机各个端口对应的所有队列数据。关键数据结构如下。

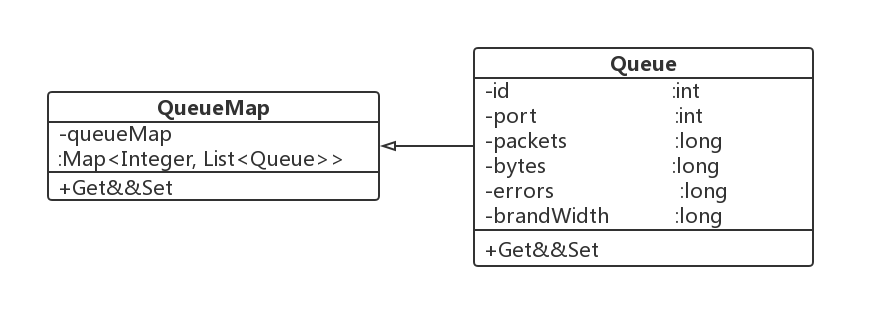


图4-22 队列类结构图

图中展示了队列类的消息格式，下面介绍各字段含义。

表4-2 队列类定义

|  |  |  |  |
| --- | --- | --- | --- |
| 类名 | 字段名称 | 字段类型 | 功能描述 |
| Queue | id | int | 队列id序号 |
| port | int | 交换机端口号 |
| packets | long | 队列传输数据包个数 |
| bytes | long | 队列传输字节数 |
| errors | long | 队列传输异常数据包个数 |
| brandwidth | long | 队列带宽大小 |
| QueueMap | queueMap | Map<Integer, List<Queue>> | 存储交换机各个端口中队列运行信息，key：交换机id，value：队列信息列表 |

1. **队列带宽调整算法**

队列信息获取模块将得到的队列数据保存在队列信息数据库中，这样我们可以对各个队列的运行情况有一定的了解，队列的进出字节数、丢失数据包个数等信息是网络拥塞情况的直接表现，队列调整算法的核心目的是降低各个队列拥塞情况的加权总和，即确保网络整体转发的稳定性。

队列调整算法首先获得队列信息，然后转换为队列拥塞情况，若不存在拥塞队列，则继续进行QoS管理，若存在拥塞队列，则进一步判断是否所有队列都发生拥塞，如果存在正常转发队列，则根据调整算法计算出合理的带宽分配值，并下发队列带宽分配方案，从而提升队列转发性能，如果所有队列都发生拥塞，就需要通知用户降低发包频率。队列带宽调整流程图如下所示。

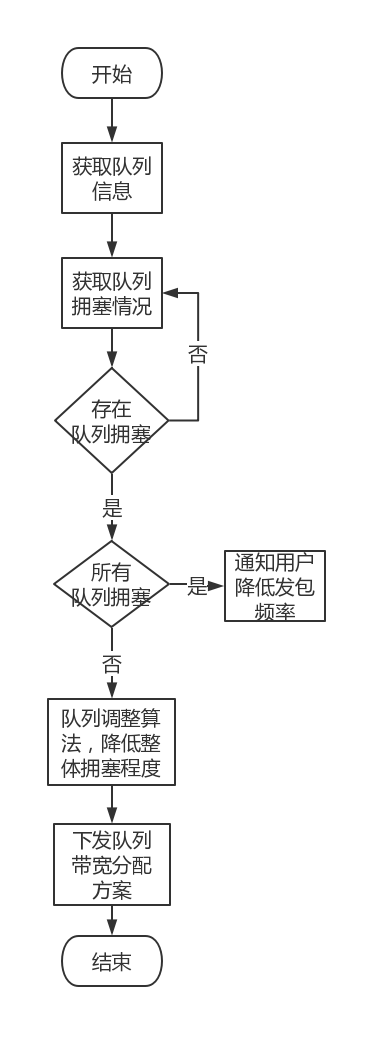


图4-23 队列调整流程图

1. **用户功能设计**

用户是本系统的实际使用者，通过发布订阅系统和SDN网络，用户可以实现多媒体数据的传输，同时系统也具备用户协商功能，用户可以主动发起时延带宽需求，针对特定主题提出时延、丢包率上的要求。用户模块的功能图如下所示。

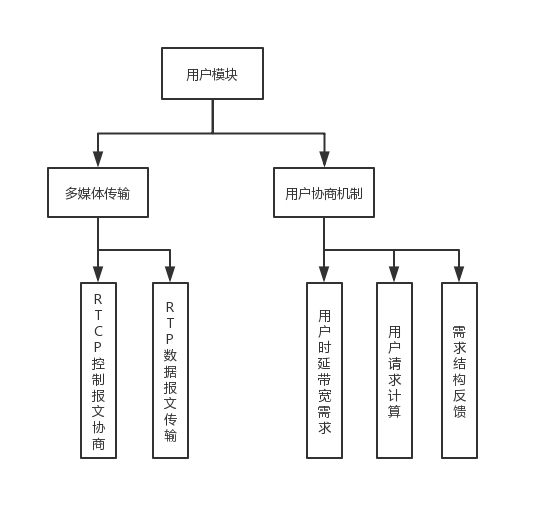


图4-24 用户功能模块图

1. **多媒体传输**

本系统的创新点之一就是丰富用户的传输功能，支持多媒体数据的传输，该模块设计的关键在于使用发布订阅系统承载多媒体传输功能。系统参考了RTP实时传输协议对数据进行封装，通信数据包括RTCP（实时传输控制消息）数据包，以及RTP（实时传输消息）数据包。

接收方首先订阅感兴趣的多媒体主题，发送方通过发布订阅系统获知该订阅，若自身满足服务需求，就发布能够提供该服务的信息，同时接收方需要在发送方的SessionMgr上注册成为listener，发送方获知订阅成功后发布该事件，接着双方需要通过RTCP报文协商视频传输的大小、格式、编码等协议信息，接收方订阅初始化的会话参数，发送方确认相关参数后同意开始RTP会话，至此RTP协商结束，发送方开始数据传输。发布订阅系统中的多媒体传输交互图如下所示。

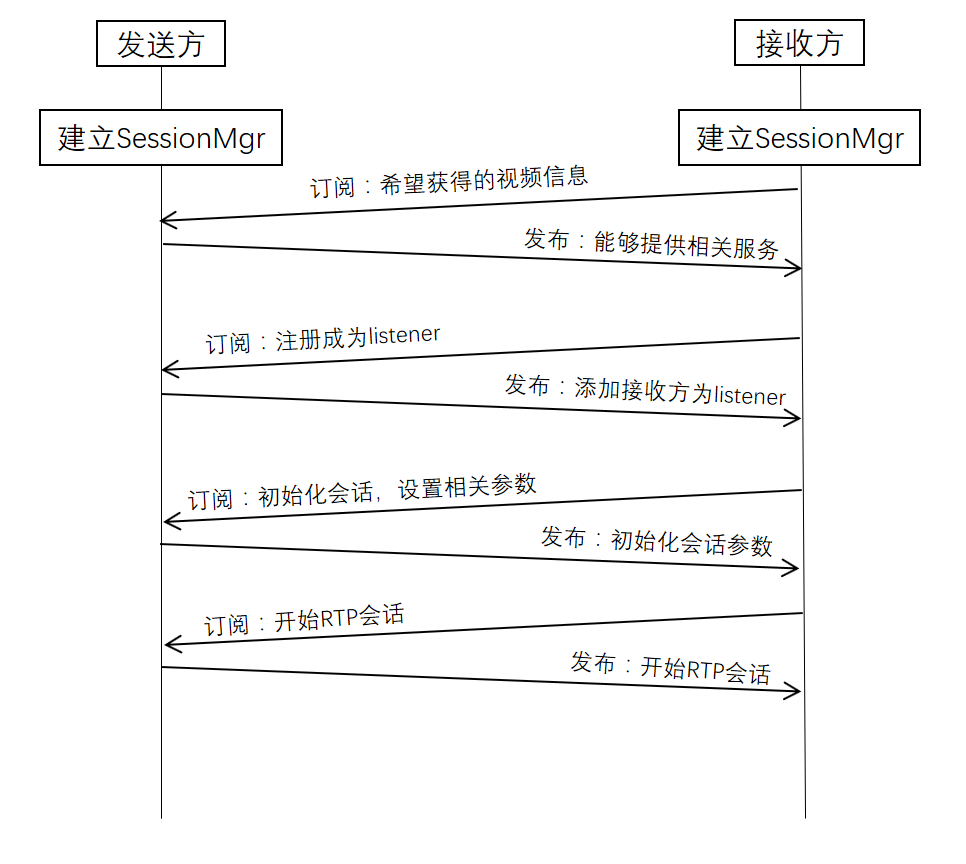


图4-25 发布订阅系统中多媒体传输交互图

1. **用户协商**

在实际应用场景下我们常常遇到这种情况：用户不满足于现有服务结果，会提出自己的需求。本系统部署于物理SDN网络，就应该考虑到用户的实际需求，用户协商模块保证了系统能够支持用户主动提出需求并给出相应的解决方案，用户协商机制是系统质量保证方案的重要环节。

用户通过消息接口向WSN发起时延、带宽请求，WSN收到后上报给集群控制器，集群控制器会以集群的身份向管理员汇报用户的需求，管理员收到后结合全网拓扑情况进行处理，处理结果将沿着管理路径反馈给集群控制器，若用户需求可以满足，则集群控制器执行带宽分配结果，若应需求无法满足，则将处理结果反馈给用户，由用户决定是否发起新的请求。用户协商的流程图如下所示。

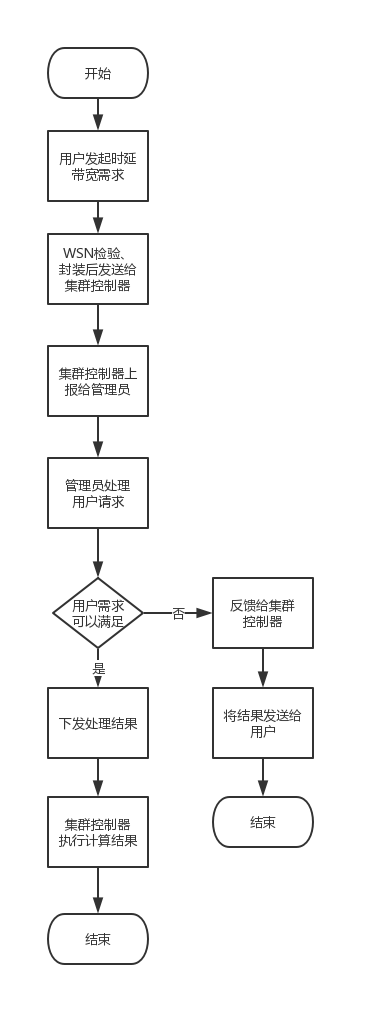


图4-26 用户协商流程图

1. **本章总结**

本章主要介绍了发布订阅系统在SDN网络中的架构设计，系统设计时划分为控制器上发布订阅功能设计、管理员功能设计、WSN功能设计、发布订阅系统的QoS保证方案设计、用户功能设计，关于它们的具体实现将在第五章进行介绍。

1. **系统详细设计与实现**

第四章介绍了系统的架构和模块设计，本章将进一步介绍基于SDN的发布订阅系统是如何实现的。

1. **系统架构实现**

系统通过参与角色划分为集群控制器、管理员、WSN、用户等模块，模块之间彼此独立，但是可以通过接口调用的方式相互关联，这样不论是前期编码还是后期维护都带来了便利。由于系统架构实现总图涉及模块较多，无法在一页内以完整形式展现，因此将管理员模块画在系统总图（一）中，将集群控制器画在系统总图（二）中，将WSN、用户模块画在系统总图（三）中，各个模块向外的线路表示与其他模块有数据交互，管理员模块在右边与集群控制器直接交互，集群控制器在右边与WSN交互，系统架构总图如下所示。

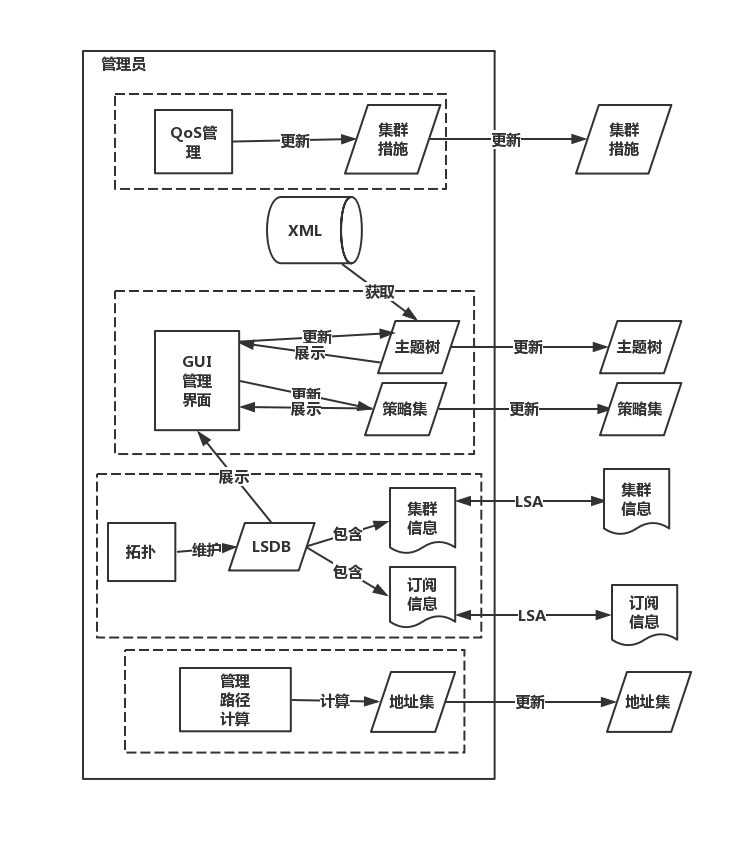


图5-1 系统架构总图（一）

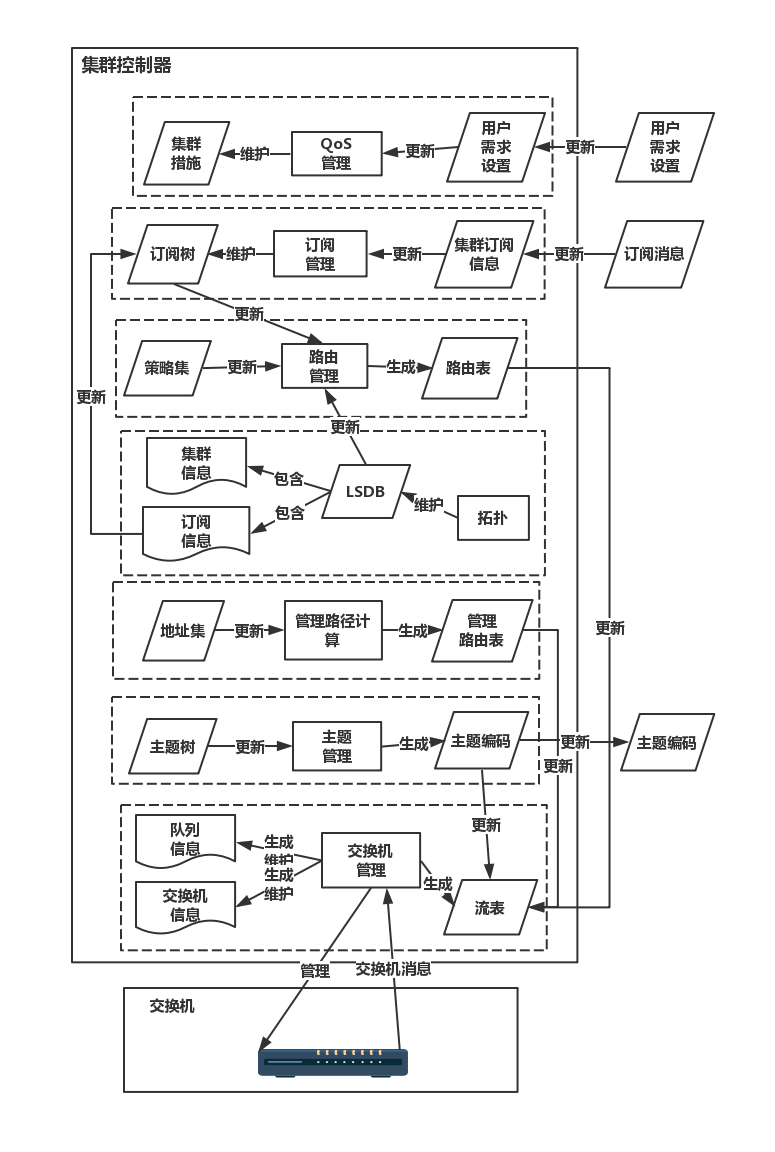


图5-2 系统架构总图（二）

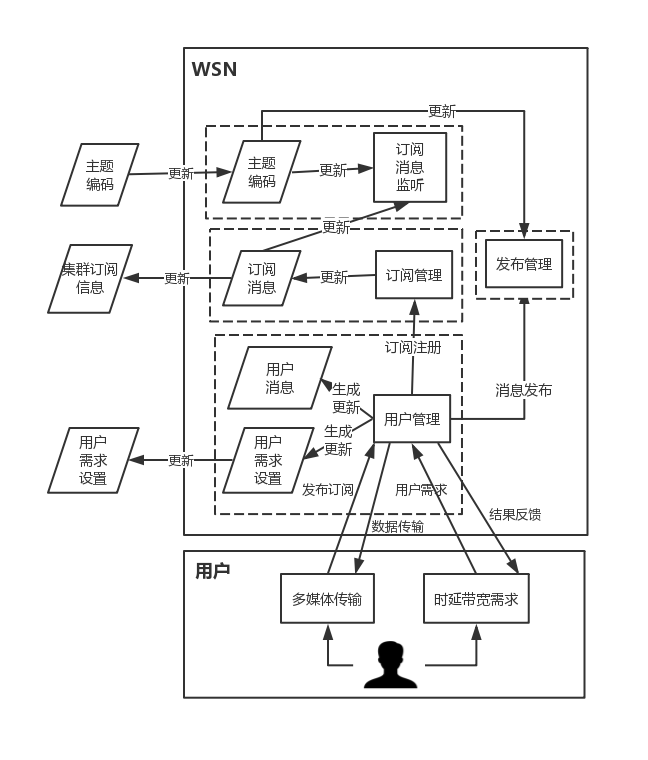


图5-3 系统架构总图（三）

1. **消息的定义与实现**

由于在各个模块间都存在消息交互，且消息的种类繁多，为了方便管理，本系统在构建之初就详细设计了消息的格式，除基本设备信息外，根据消息发送方角色的不同分为管理消息、系统消息和WSN消息，具体消息定义如下所示。

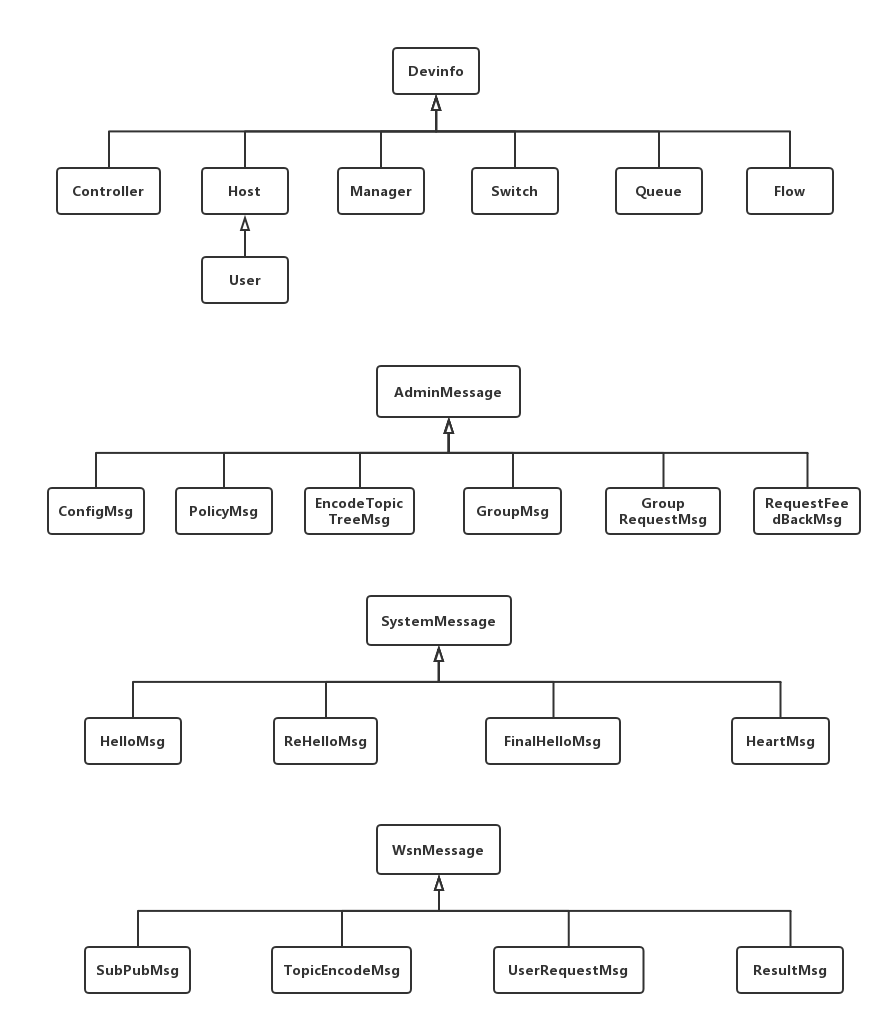


图5-4 消息定义图

图中列出了消息的分类和继承关系，详细内容见下表。

表5-1 消息定义及分类

|  |  |  |  |
| --- | --- | --- | --- |
| 消息分类 | 分类名称 | 消息名称 | 消息功能 |
| DevInfo | 设备消息 | Controller | 控制器信息，描述控制器所在集群 |
| Host | 主机信息，描述主机性能 |
| User | 用户信息，一台主机Host可以包含多个用户 |
| Manager | 管理员信息，描述管理员所在集群 |
| Switch | 交换机信息，描述交换机的基本配置 |
| Queue | 队列信息，包含从交换机收集到的队列数据 |
| Flow | 流表信息，表示一条流表项 |
| AdminMessage | 管理消息 | ConfigMsg | 管理员下发的配置信息 |
| PolicyMsg | 管理员下发的策略信息 |
| EncodeTopic  TreeMsg | 管理员下发给控制器的编码主题树 |
| GroupMsg | 控制器上报的集群信息 |
| GroupRequest  Msg | 控制器上报的集群内用户需求信息 |
| RequestFeed  BackMsg | 反馈信息，管理员下发的用户需求计算结果 |
| SystemMessage | 系统消息 | HelloMsg | Hello消息，第一次握手 |
| ReHelloMsg | ReHello消息，第二次握手 |
| FinalHelloMsg | FinalHello消息，第三次握手 |
| HeartMsg | 心跳消息，用于拓扑维护 |
| WsnMessage | WSN消息 | SubPubMsg | 用户发布、订阅信息 |
| TopicEncodeMsg | WSN收到的主题树编码信息 |
| UserRequestMsg | 用户时延、带宽请求信息 |
| ResultMsg | 用户需求反馈信息 |

系统中存在多条通信链路，在不同模块间传输了不同的消息，结合上述消息定义，可以得到主要的消息交互流程。

1. **拓扑消息**

拓扑消息是控制器在拓扑管理阶段交互的信息，以一次拓扑探测为例，集群G1、G2进行拓扑探测过程中，使用的是Hello、ReHello、FinalHello消息来达到三次握手的目的，拓扑维护阶段集群间交互的是HeartMsg心跳消息。消息交互流程见下图。

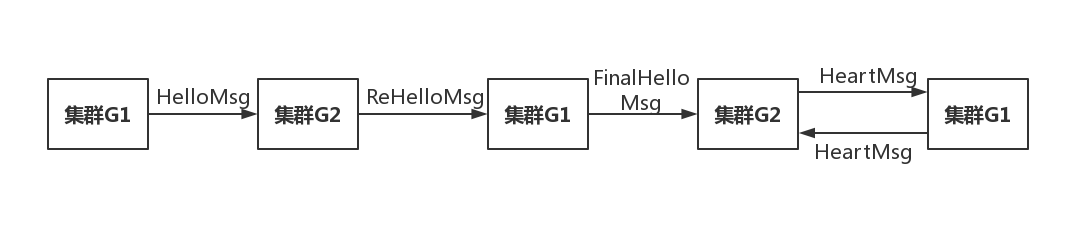


图5-5 拓扑消息交互图

1. **主题树消息**

主题树消息用于主题内容的下发，管理员将编码后的主题树封装为EncodeTopicTreeMsg消息下发给各个集群控制器，集群控制器分析处理后，再将包含主题编码的TopicEncodeMsg发送给WSN。消息交互流程见下图。

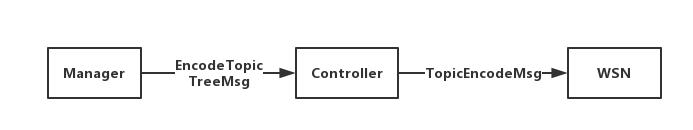


图5-6 主题树消息交互图

1. **发布订阅消息**

用户发起的请求都是由发布订阅消息进行传输，WSN收到消息后进行验证处理，将用户请求封装为SubPubMsg消息发送给集群控制器，不同集群间通过控制器的HeartMsg心跳消息广播集群内的发布订阅情况。消息交互流程见下图。

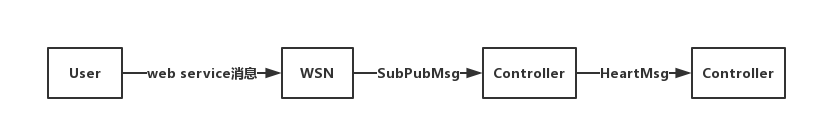


图5-7 发布订阅消息交互图

1. **用户请求消息**

本系统提供的性能保证方案之一就是用户可以自行发起时延带宽请求，WSN收到后经过验证处理，封装为UserRequestMsg消息上传给控制器，控制器收集本集群内的用户请求，以GroupRequestMsg的形式发送给管理员，经过分析计算后，相应的结果由管理员的RequestFeedBackMsg下发，进一步解析为ResultMsg消息发送给WSN，最终将处理结果告知用户。消息交互流程见下图。

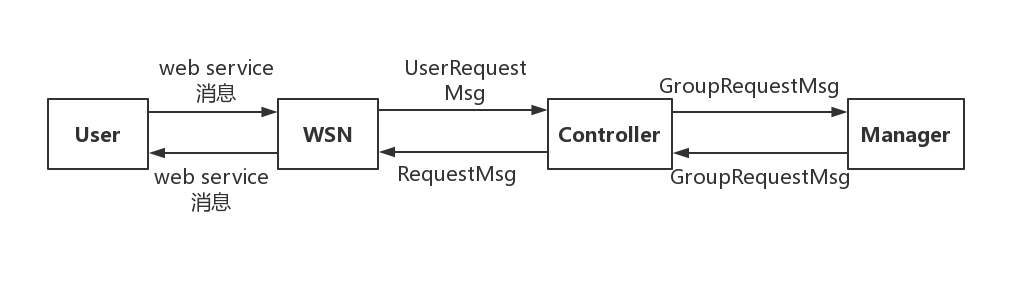


图5-8 用户请求消息交互图

1. **集群状态消息**

为了监测系统整体的运行状况，管理员需要收集各个集群的运行时信息，集群控制器定期将集群内的运行情况封装为GroupMsg消息发送给管理员。消息交互流程见下图。

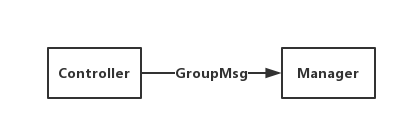


图5-9 集群消息交互图

1. **控制器上发布订阅功能实现**

集群控制器负责集群内的发布订阅事务，具体由拓扑管理、发布订阅管理、路由管理、流表管理共同完成，从系统架构总图（二）中可以看到集群控制器中各个子模块的交互关系。

1. **拓扑管理**

拓扑管理是系统的基本功能模块，能够帮助集群控制器在启动后动态感知周围的邻居集群，在本系统中采用的是OSPF协议，包括拓扑探测和拓扑维护过程，拓扑管理的时序图如下所示。

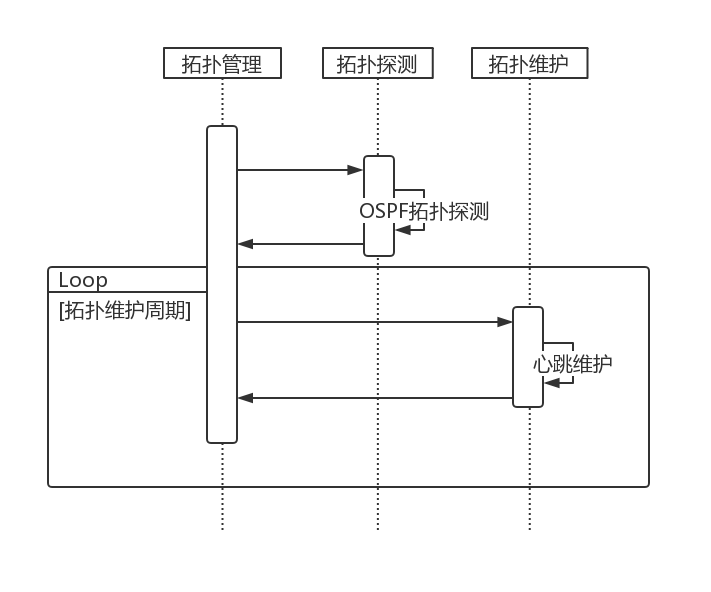


图5-10 拓扑管理时序图

1. 拓扑探测

拓扑探测是集群控制器动态获取邻居集群连接信息的过程，在OSPF的拓扑探测过程中传输的消息包括Hello、ReHello、FinalHello，这里以Hello消息为例，Hello消息类图如下所示。



图5-11 Hello消息类图

图中概括了Hello类的消息格式，下面介绍各字段含义。

表5-2 Hello消息类定义

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段类型 | 功能描述 |
| startGroup | String | 消息起始集群 |
| endGroup | String | 消息目的集群 |
| startBorderSwtId | String | 发送集群的边界交换机id |
| endBorderSwtId | String | 接收集群的边界交换机id |
| startOutPort | Int | 起始交换机的端口 |
| endOutPort | Int | 目的交换机的端口 |
| reHelloPeriod | long | ReHello消息间隔 |
| State | enum | 当前状态 |
| lsa | String | 本集群的lsa内容 |

ReHello、FinalHello消息的定义与Hello消息相似，这里就不重复描述了。在探测阶段初始集群向所有对外端口发送Hello消息，等待邻居集群的回复，若收到ReHello消息则说明邻居存在，初始集群将保存相关信息并返回FinalHello消息。拓扑探测处理的流程图如下所示。



图5-12 拓扑探测流程图

1. 拓扑维护

系统在拓扑探测阶段完成后，会使用维护线程通过定期发送心跳消息的方式来判断邻居集群是否失效，若收到心跳消息，则说明集群运行正常，并刷新该集群的存活时间，若消息超时或未收到心跳消息，则认为该集群出现故障无法继续使用，因此需要删除失效节点，并重新进行拓扑探测过程。拓扑维护流程图如下所示。

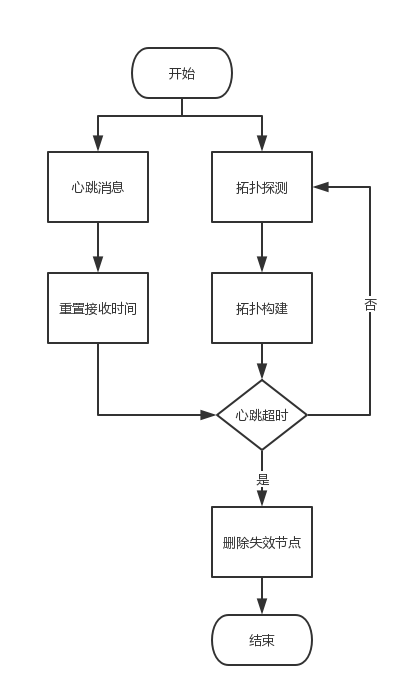


图5-13 拓扑维护流程图

1. **发布订阅管理**

集群控制器具备发布订阅管理功能，具体分为集群内发布订阅管理和集群间发布订阅管理，其中集群间发布订阅信息是通过LSA心跳消息得到的，系统需要进一步抽象为全局发布订阅信息。针对集群内的发布订阅信息，以及全局发布订阅信息，发布订阅管理模块分别使用本地发布订阅表LocalSubPub、全局发布订阅表GlobalSubPub进行存储，相关数据结构如下所示。

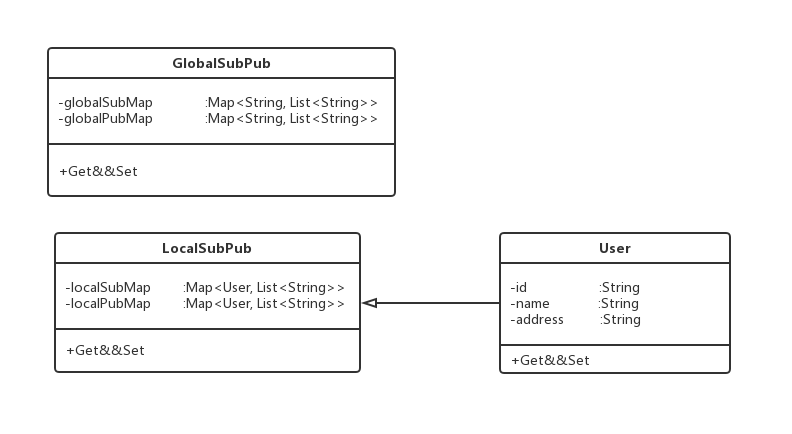


图5-14 发布订阅表类图

图中展示了发布订阅管理的关键数据结构，下面介绍各字段含义。

表5-3 发布订阅管理关键数据结构定义

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 类名 | 类功能 | 字段名 | 字段类型 | 字段功能 |
| User | 用户类，表示一个特定用户 | id | String | 用户序号，唯一标识一个用户 |
| name | String | 用户名，由用户自行定义 |
| address | String | 用户所在主机IP，用于确定用户地址 |
| LocalSubPub | 本地用户发布、订阅信息 | localSubMap | Map<User, List<String> | 本地用户订阅集合，key：用户，value:该用户订阅主题 |
| localPubMap | Map<User, List<String> | 本地用户发布集合，key：用户，value:该用户发布主题 |
| GlobalSubPub | 全网发布、订阅信息 | globalSubMap | Map<String, List<String> | 全网订阅集合，key：集群，value:该集群订阅主题 |
| globalPubMap | Map<String, List<String> | 全网发布集合，key：集群，value:该集群发布主题 |

发布订阅表使用Map散列表存储发布订阅信息，这样可以实现快速查询。本地发布订阅表的索引为User用户，这是因为集群内部存在多个用户，集群控制器需要详细记录每个用户的订阅信息，否则当某个用户取消订阅后，系统无法判断集群内是否存在其他用户订阅了该主题；全局发布订阅表只记录了某个集群内的发布订阅情况，这是因为全网订阅信息应用于主题路径的计算，在计算转发路径时，系统只需要判断该集群内是否存在针对特定主题的发布订阅情况，而不需要精确到某个用户，只要集群内存在该主题的发布订阅操作，就计算该集群的主题路径，因此采用集群名作为索引。

1. **路由管理**

路由管理模块的功能是计算出消息转发路径，确保发布订阅功能的实现，包括管理路径计算、主题路径计算等，下面分别进行介绍。

1. 管理路径计算

管理员通过管理路径能够与任意控制器进行通信，由于管理员身份的特殊性和唯一性，管理路径本质上是一棵以管理员为根节点的多播树，系统在管理路径的计算中使用了Dijkstra算法，当新增控制器节点时，只需要计算出新增节点与当前管理路径中的最短路径。Dijkstra算法是计算两点间最短距离的算法，算法以起始节点为中心，不断地向外扩展，直到扩展到所有节点，该算法具有简单、高效的特点，相关伪代码如下所示。

算法： Dijkstra 算法伪代码

输入： Graph 是一个连通图的矩阵表示

source 是计算的起始点

输出： source 在图中最小路径的权值以及对应路径

-----------------------------------------------------------------------------------------------

1：function Dijkstra(Graph, source):   
2： create vertex set Q  
3： //初始化各个点到源点的距离，设置为最大值  
4： for each vertex v in Graph:   
5： dis[v] ← INFINITY  
6： //将所有节点放在Q 集合中，标记为不可达状态  
7： add v to Q  
8： //从source 到source 的距离为0  
9： dis[source] ← 0  
10： //遍历所有不可达节点   
11： while Q is not empty:   
12： //选取Q 中与source 距离最短的节点  
13： u ← vertex in Q with min dis[u]   
14： //从Q 中移除该节点  
15： remove u from Q  
16： for each neighbor v of u:   
17： //遍历Q 的邻居节点  
18： alt ← dis[u] + length(u, v)   
19： //若经过邻居节点V 有更短距离，则更新该路径  
20： if alt < dis[v]:   
21： dis[v] ← alt  
22： pre[v] ← u  
23： //返回各个节点与source 的距离以及经过的路径  
24： 返回 dis[], pre[]

图5-15 Dijkstra算法伪代码图

算法中，输入参数包括图的连接情况Graph，以及需要求解的起始节点source，初始化时将与起始节点无直接关联的节点标记为不可达状态（距离设为最大值），依次选取与源点最近的点，若不可达节点经过该最短节点可以访问，则更新相应的距离值，这个步骤很关键，一方面，选取当前最近节点保证了计算结果符合要求，另一方面，更新相关距离，确保不可达节点可以经过该节点得以访问。重复上述过程，直至所有节点与源节点的距离都得到计算。

从上述过程可以看到，Dijkstra算法总是选择“最近”的结点加入到集合中，该算法使用的是贪心的策略，保证了结果的正确性。结合Dijkstra最短路径算法，管理路径具体计算流程如下图所示。

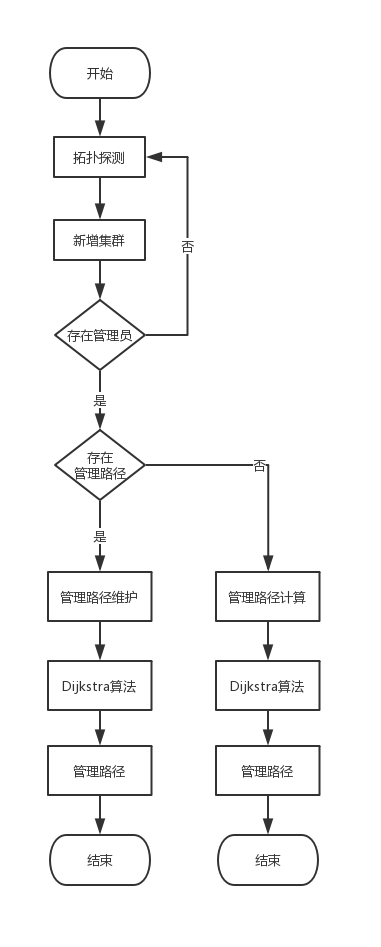


图5-16 管理路径计算流程图

1. 主题路径计算

主题路径应用于发布订阅消息的传输，在计算时需要结合全网拓扑，选取其中包含相同订阅主题的节点，计算出一条在这些节点间的最小生成树路径。它的计算结果对于传输的性能有着重要作用，保存一个旧有链路中订阅节点组成的主题路径没有意义，因此无论是全网链路的变化还是用户发布订阅信息的改变，都将触发主题路径的重新计算。

将指定集合中的所有节点连通，且代价总和最小的生成树称为最小斯坦纳树（Minimal Steiner Tree），这个计算结果与我们的需求一致，即在全网拓扑中求解多个节点的最小连通图，且Steiner算法具有快速收敛和高效的特点，因此本系统使用Steiner算法来实现主题路径的计算，相关伪代码如下所示。

算法： Steiner 算法伪代码

输入： allNodes 图中所有节点

select 需要计算的节点

临时变量：G：图中所有节点的集合，由allNodes 可以得到

D：所选节点的集合，由select 给予点表示

e： 全部边的集合

g： 最小生成树采用的边的集合

G2：最小生成树经过的节点集合

输出： select 节点在图中链路代价最小的连通图

图5-17 Steiner算法伪代码图（一）

1：function Steiner(Set<Node> allNodes, Set<Node> select):  
2： //step1：计算完全连通图  
3： create distance set dist  
4： for each vertex i in allNodes:   
5： for each vertex j in allNodes:  
6： for each vertex k in allNodes:  
7： //遍历所有节点，计算完全连通图  
8： dis[i][j] ← min(dist[i][j], dist[i][k]+dist[k][j])   
9： //step2： 求select 节点构成的最小生成树  
10： create edge set e  
11： create vertex set g  
12： create vertex set G2  
13： for each vertex i in allNodes:  
14： for each vertex j in allNodes:  
15： //如果该节点可以访问，则添加这条边  
16： if not dist[i][j] = INFINITY   
17： create edge start with i, end with j  
18： //在e 中添加这条边  
19： add edge to e[]   
20： //在g 中添加这个节点   
21： add vertex to g[]   
22： //调用kruskal 算法计算联通子图  
23： g ← kruskal(select, e)   
24： for edge ed in g:  
25： add edge ed to G2  
26： //step3：求扩展子图G2 的最小生成树  
27： create vertex set G3  
28： //调用kruskal 算法计算联通子图  
29： g2 ← kruskal(select, e)   
30： for edge ed in g2:  
31： add ed to G3  
32： //step4：删除度为1的顶点  
33： for vertex v in G3:  
34： //若当前节点的度为1，则删除该节点  
35： if degree(v) == 1:   
36： remove v from G3  
37： 返回 g2

图5-18 Steiner算法伪代码图（二）

系统中采用的是基于启发式算法的Steiner计算方式，输入参数为图中所有节点allNodes，以及需要计算的节点集合select，具体包括四个步骤：首先，遍历所有节点集合，构造完全连通图，这是为了保存各个节点间的距离，方便后续的计算；接着结合我们选取的节点，计算出联通子图，这是因为最终目的是为了算出这些节点构成代价最小的生成树，联通子图确保了节点间彼此的连通性；下一步需要求解扩展子图的最小生成树，我们得到包含所需节点的生成树，距离最终结果更进一步；最后，我们需要对这棵树进行“裁剪”，删除度为1的节点，这个步骤是为了将那些冗余、边界的无关节点删除，至此Steiner算法计算结束，我们得到了select选取节点的最小生成树。

Steiner算法比较复杂，需要首先计算出联通子图，然后根据这个联通子图计算出生成树，这样所有节点都包含在内，最后将那些无关节点删除，确保整个结果的性能最优。结合Steiner多源最小生成树算法，主题路径计算的流程图如下所示。

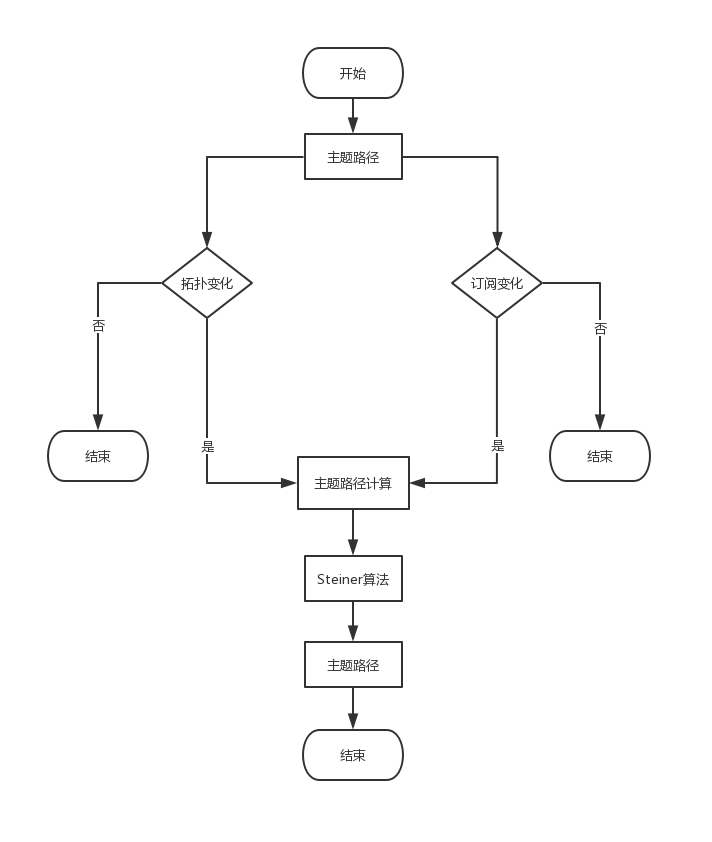


图5-19 主题路径计算流程图

1. **流表管理**

路由管理模块得到了管理路径和主题路径，但这是内存中的逻辑链路形式，需要进一步解析成流表并下发，才能充分发挥SDN软件定义网络的强大功能。流表管理模块包括流表的生成、流表的下发、流表的维护，模块整体的功能如下图所示。

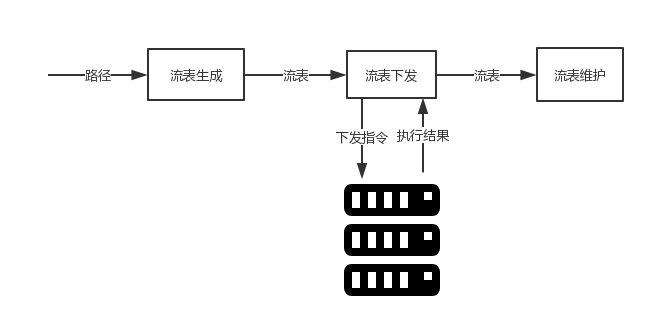


图5-20 流表管理功能图

1. 流表的生成

交换机识别的是操作指令，路由计算得到是拓扑链路，需要在两者之间进行转换，系统将交换机指令抽象出来，同时将转发路径封装为流表类的形式，这样在使用时调用不同的指令模板，填充相应的字段，如目的地址、转发端口等，可以生成SDN交换机识别的流表指令形式。流表的模板操作格式见下图。

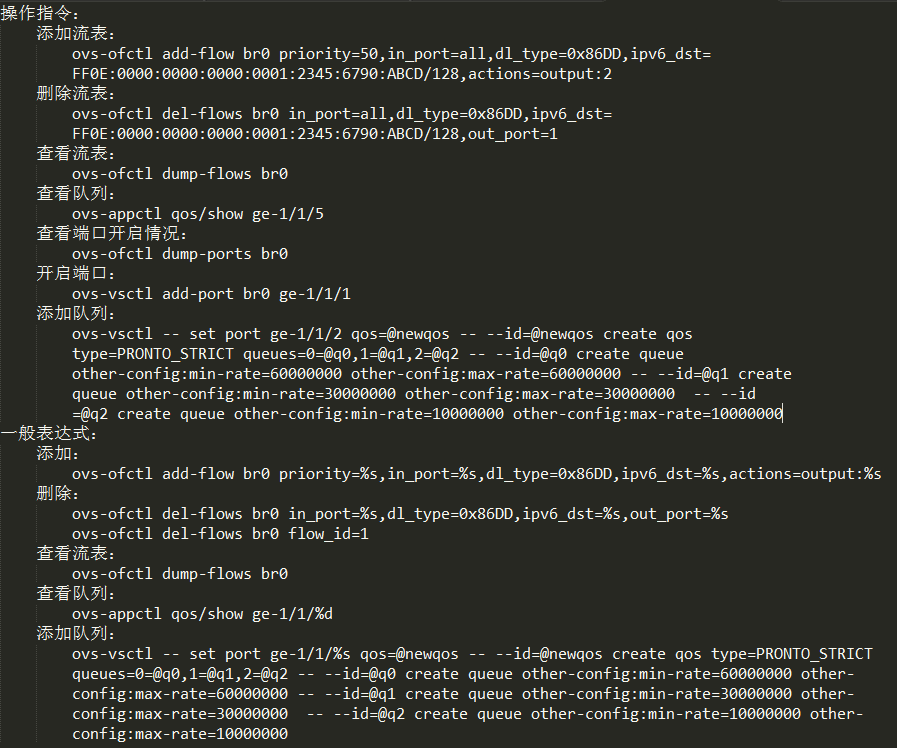


图5-21 流表指令操作模板

1. 流表的下发

经过流表生成模块，可以将转发路径转换为流表指令，此时系统需要与SDN交换机核心处理器交互，完成流表下发的功能。系统使用的是SSH连接方式，用网线将控制器所在主机和SDN交换机连接起来，配置两者在同一网段，通过指定交换机处理器的IP地址、用户名、密码，就可以使用JAVA程序远程访问SDN交换机的核心处理器，进而下发流表指令完成相应的需求。流表下发的核心代码见下图。



图5-22 流表下发核心代码

1. 流表的维护

实际匹配过程中流表存在重复、冲突的情况，重复的流表会影响交换机处理性能，冲突的流表会导致数据转发失败，因此在内存中需要有流表维护模块，对下发的流表进行维护，流表维护的相关类图如下所示。

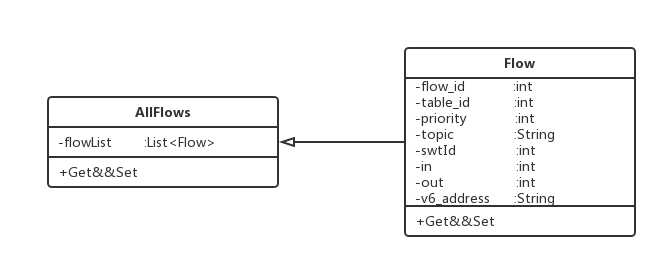


图5-23 流表维护类图

图中展示了流表类的格式，下面介绍各字段含义。

表5-4 流表消息类定义

|  |  |  |  |
| --- | --- | --- | --- |
| 类名 | 字段名称 | 字段类型 | 功能描述 |
| Flow | flow\_id | int | 流表项id序号 |
| table\_id | int | 流表id序号 |
| priority | int | 流表项的匹配优先级 |
| topic | String | 对应的主题 |
| swtId | int | 交换机id |
| in | int | 进端口号 |
| out | int | 出端口号 |
| v6\_address | String | 目标转发v6地址 |
| AllFlows | flowList | List<Flow> | 存储当前所有流表项 |

由于交换机会从优先级高的流表开始执行，匹配成功后直接转发，不再往下查询，因此内存中使用链表来定义AllFlow，从而在顺序遍历过程中达到优先级匹配的效果，当有新的流表项生成后，需要遍历已有流表，若存在冲突现象就将新增流表的出端口附加在已有流表的后边，达到多端口转发的目的。

1. **管理员功能实现**

管理员是所有集群的统一管理者，在集群控制器功能的基础上，还具有主题树管理、用户请求处理、ui展示的功能，从系统架构总图（一）中可以看到管理员中各个子模块的交互关系。

1. **主题树管理**

主题是发布订阅系统的纽带，发布者、订阅者通过指定相同的主题，实现消息的发送与接收，具体可以划分为主题树存储、主题树编码、主题树下发。主题管理模块可以实现主题的增删改查，这些修改后的数据也将写回到XML文件中，获取到的主题树通过编码子模块，转换为编码主题树信息，同时在管理模块内部存在周期性的主题通知机制，定期将编码主题树发送给各个集群控制器。主题管理时序图如下所示。

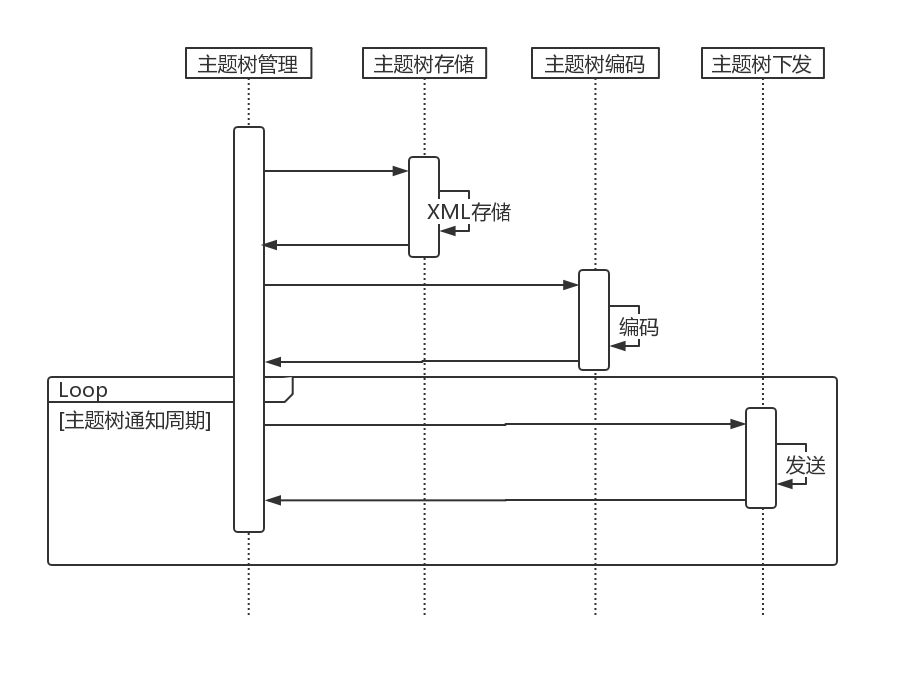


图5-24 主题树管理时序图

1. 主题树存储

主题树的定义与树形结构相似，因此我们可以将主题抽象为树形结构中的节点，若某个主题不存在子主题，即为叶子节点，同时在所有最上层主题外，抽象出虚拟的根节点，这样就形成了一棵以虚拟的“all”节点为根节点的主题树，本系统提供XML文件持久化存储方式，主题树的XML存储格式见下图。



图5-25 主题树存储内容

图中一共有个16主题，在虚拟“all”主题下包含“test1”、“test2”、“test3”等一级主题，各级主题递归定义，使用XML中存储的方式具有简洁性。

1. 主题树编码

主题树编码模块需要根据主题节点在树形结构中的相对路径和位置，完成主题到IPv6地址的转换，具体分为编码、末尾填充、组播地址封装这三个阶段，整体流程如下图所示。

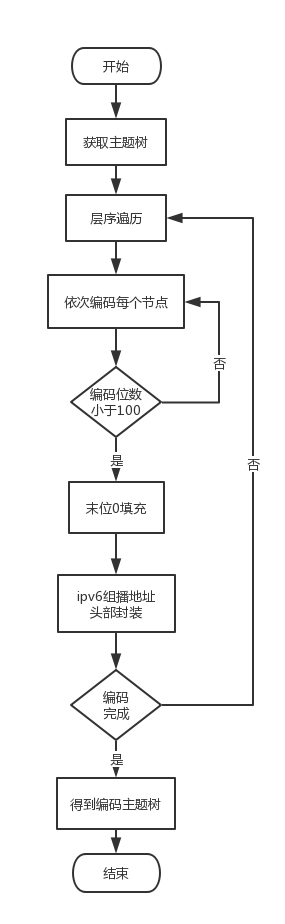


图5-26 主题树编码流程图

系统采取了IPv6 组播的形式进行消息传输，发送者向主题对应的IPv6 组播地址发送消息，接收者加入IPv6 组播的群组，接收相应的主题数据包。系统使用了128比特的IPv6 地址，并将其中100比特应用于主题树的编码。主题树编码的伪代码如下所示。

算法： 主题树编码伪代码  
输入： root 主题树根节点  
输出： encodeTopicTree 编码主题树

-----------------------------------------------------------------------------------------------  
1：function TopicTreeEncode(root):   
2： //创建编码主题树  
3： create EncodeTree encodeTopicTree   
4： create node set queue   
5： add root to queue   
6： //使用队列层序遍历  
7： while queue is not empty   
8： //获取队列中的第一个元素  
9： poll node current from queue   
10： create node list childList   
11： //获得所有子节点  
12： childList ← node.getChildList()  
13： for node child in childList:   
14： create encodeNode eNode   
15： add child to queue   
16： //结合主题节点的位置对当前主题进行编码  
17： encode ← Encode(child, childList)   
18： add encode to eNode   
19： //将编码主题节点添加到编码主题树中  
20： add eNode to encodeTopicTree   
21： //返回编码主题树结果  
22： 返回 encodeTopicTree

图5-27 主题树编码伪代码

主题树编码过程分为以下几步：采用层序遍历的方式获取主题树信息，将访问到的节点依次存储在队列中；对每一层进行编码，使用最少比特数表示；若当前节点编码位数小于100,，则进行末尾填充；最后，封装IPv6组播地址首部，得到需要的编码地址，并将结果保存在encodeTopicTree中，供程序使用。

从主题树编码的过程可以看到，这种编码方式既实现了主题的区分功能，不同主题编码结果不同，又保证了编码结果的高效，因为每一层都采用最少比特数表示，这样的代价最低，最终实现了主题树到编码主题树的转变。系统使用IPv6组播地址进行头部填充，相关字段如下所示。

表5-5 IPv6组播地址首部

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 总长度/bit | 默认值 | 含义 |
| fixed prefix | 8 | 11111111 | IPv6组播固定前缀 |
| flag | 4 | 0000 | 表示该组播地址是否“永久” |
| global scope | 4 | 1110 | 定义组播地址的范围，1110默认为全局范围 |
| event type | 2 | 无 | 事件类型 |
| topic length | 7 | 无 | 编码字符串长度，与具体主题节点有关 |
| queue No. | 3 | 000 | 数据包发送端口，默认为0号端口 |

同样，本系统还可以选择IPv4进行组播地址的编码，但由于IPv4长度有限，减去组播的固定前缀后，留给主题编码的部分只有20比特，远远小于IPv6的可编码长度，不适用于主题数目繁多的情况，因此本系统最终选择了编码长度更充裕的IPv6地址。

1. 主题树下发

本系统设计之初，想要将主题树存储在各个集群当中，这样可以减少系统中管理消息的传输，但是集群的分布式架构使得主题树的修改又成了大问题，甚至会产生集群间不同步的问题。综合考虑，本系统最终采取的是管理员集中管理的方式，管理员将主题树封装为主题树消息，通过管理员路径下发给各个控制器，这样虽然付出了一定的带宽代价，但是避免了分布式环境下，数据不一致的风险。主题树消息的类图如下所示。

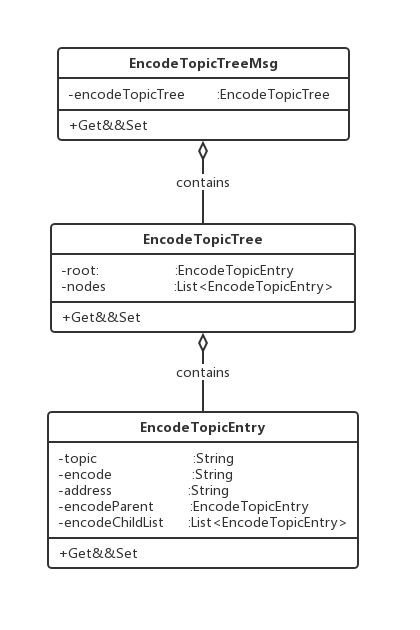


图5-28 主题树消息类图

图中概括了主题树消息、主题树及主题节点的内容格式，下面介绍各字段含义。

表5-6 主题树消息分类及定义

|  |  |  |  |
| --- | --- | --- | --- |
| 类名 | 字段名 | 字段类型 | 功能描述 |
| EncodeTopicEntry | topic | String | 主题名称 |
| encode | String | 编码结果 |
| address | String | 编码对应的IPv6地址 |
| encodeParent | EncodeTopicEntry | 父主题节点 |
| encodeChildList | List< EncodeTopicEntry> | 子主题节点列表 |
| EncodeTopicTree | root | EncodeTopicEntry | 主题树的根节点 |
| nodes | List< EncodeTopicEntry> | 主题树中所有节点列表 |
| EncodeTopicTreeMsg | encodeTopicTree | EncodeTopicTree | 编码主题树 |

1. **用户请求处理**

系统支持用户协商机制，即用户可以主动提出对时延带宽的需求，最终由管理员结合全网运行状态计算处理，这是系统质量保证方案的一部分。

管理员StartMgr启动后，会开启AdminMgr管理员监听，收到用户需求后将主题信息、时延需求等数据交由AdminProcess模块，计算出该主题的转发路径后将转发路径和用户时延需求作为参数，调用AdminService模块计算出各个集群的时延带宽分配情况，最终通过管理路径下发处理结果。用户请求处理的时序图如下所示。

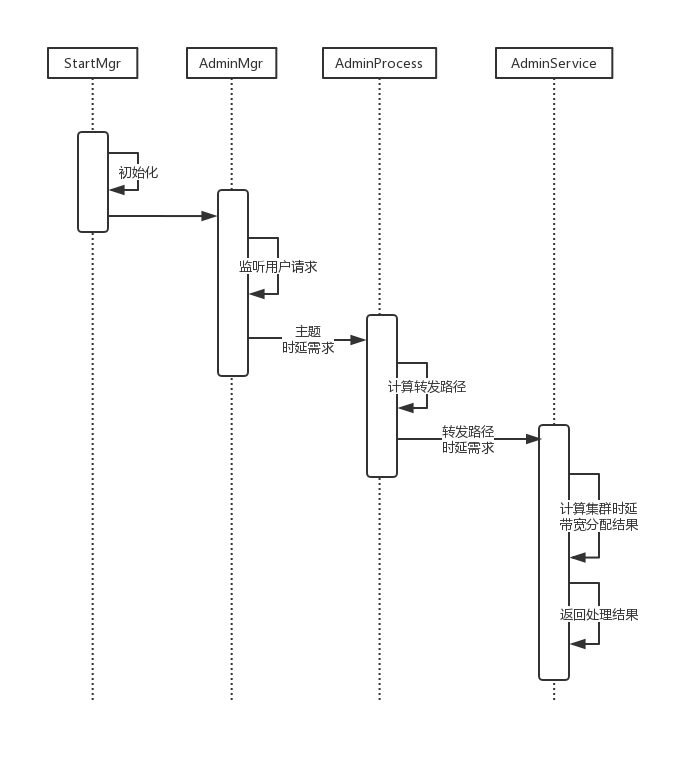


图5-29 用户请求处理时序图

1. **ui展示**

管理员模块为管理者提供了一个便于展示和操作的ui界面，如下图所示。



图5-30 管理员界面展示图

管理员ui界面需要展现集群基本信息，包括所有集群列表、各集群交换机状况、集群订阅信息、各集群配置信息，以及主题树、策略等内容，其中各集群的交换机信息是通过由集群控制器上报得到的；管理员可以在ui界面上进行操作，包括配置集群的失效阈值、扫描周期、发送周期等参数，相应数据将通过管理路径下发给各个集群控制器；主题树保存在XML中，管理员启动后将读取相关文件并在ui中展示，主题树展示图如下所示。

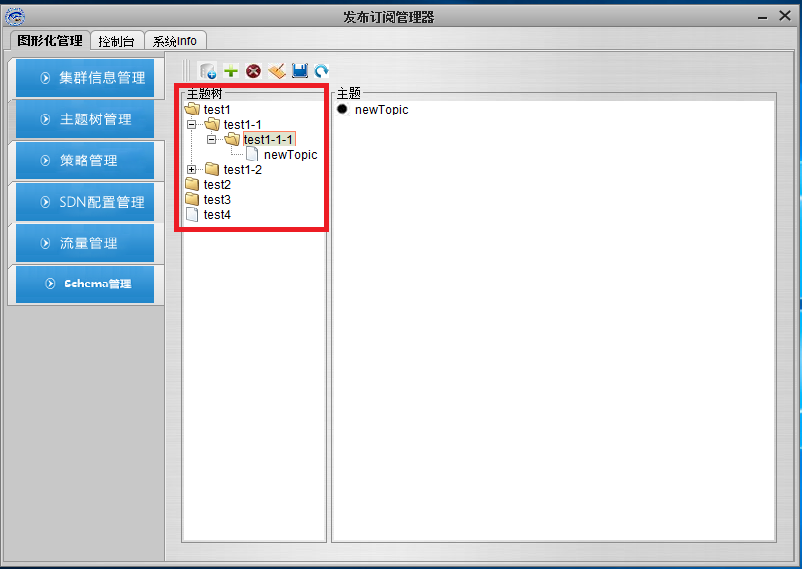


图5-31 主题树管理展示图

1. **WSN功能实现**

WSN为用户提供消息接口功能，这样用户可以调用系统的发布订阅服务，同时实现了与用户的双向通信，WSN可以将收到的主题消息发送给指定的用户；WSN存在数据转发方案，当网络中的主题消息到来时，能够使用该方案向用户发送消息，保证数据的稳定传输。

1. **消息接口**

消息接口支持用户发起服务调用，同时能够将收到的主题消息发送给用户，即在WSN与用户之间形成稳定的双向通信。以一次通信流程为例，WSN暴露服务地址（http://192.168.10.101:9010/wsn-core），用户可以通过消息接口进行服务调用，本质是将自己的IP、端口信息封装进SOAP协议中，使用HTTP方法将数据传送给服务端，这样服务端通过监听并解析，就可以得到用户端的IP地址、监听端口等信息，相当于获知了用户在网络中的位置，接着向用户发送处理结果，从而形成WSN与用户之间的双向通信。WSN与用户的交互如下图所示。

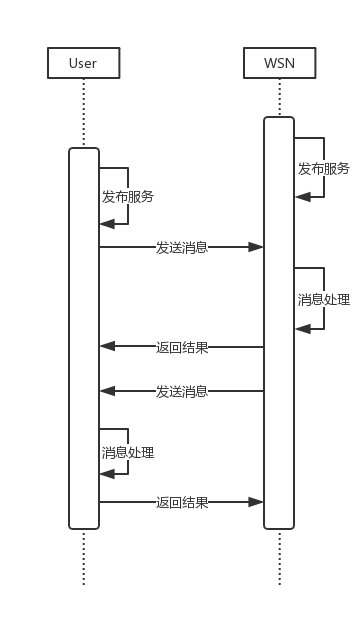


图5-32 WSN交互图

1. **数据转发方案**

WSN负责将主题消息发送给用户，这是数据传输的最后一步，在WSN中需要使用数据转发方案来保障用户数据的稳定转发。系统使用了Reactor模式来实现数据转发方案，Reactor模式基于事件驱动，遵循注册—调用机制，使用时用户需要定义handler，然后将其注册到dispatcher分派器中，接收到的消息首先保存在selector阻塞队列中，进一步抽象出任务task的形式进行调度，分派器在任务获取间隔不停地访问selector获取任务，并调用注册到dispatcher上的用户处理逻辑完成后续的消息处理。整个流程可以视为用户消息的到达“触发”了具体的处理逻辑，由于分派过程是自行定义的，我们可以添加过滤机制、负载均衡策略等，因此Reactor模式具有很好的扩展性。Reactor模式下的数据转发时序图如下所示。

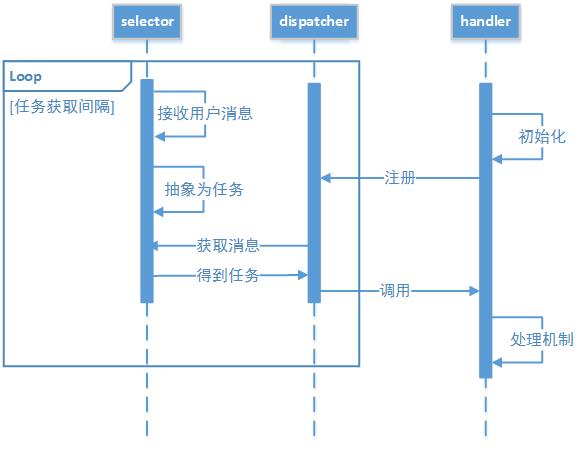


图5-33 Reactor模式时序图

1. **发布订阅系统的QoS保证方案实现**

系统通过获取队列信息可以得到转发队列的拥塞情况，结合队列带宽调整算法计算出整体最优解，从而为发布订阅系统提供质量保证。

1. **队列信息获取**

在流表管理模块我们使用SSH方式远程连接SDN交换机的核心处理器，从而达到流表下发的目的，队列信息可以通过相同的方式获取到，在建立与交换机的连接后，通过下发查询语句就能得到交换机指定端口的队列运行数据，最终保存在Queue队列类中。关键代码如下所示。

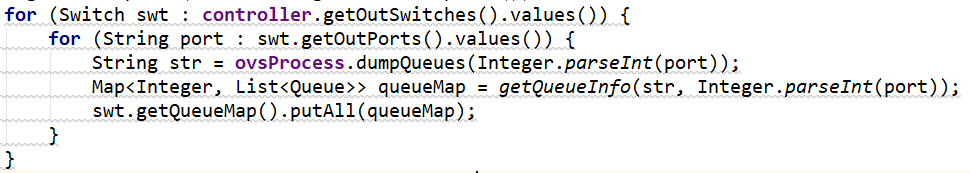


图5-34 队列信息获取代码

1. **队列带宽调整算法**

系统根据队列信息可以得到队列拥塞程度，然后依据排队时延的大小动态分配带宽资源，从而使得队列整体时延最低，队列带宽调整算法为系统提供了QoS保证。由于系统与SDN交换机建立连接需要耗费一定的时间，因此系统在OvsProcess连接初始化后需要保存这个连接，然后启动队列管理类QueueMgr，进一步调用队列调整算法QueueAdjust进行队列带宽调整，上述步骤被设置为定时任务以应对复杂的网络环境，由RefreshQueue周期性获取队列信息并进行调整。队列调整算法的时序图如下所示。

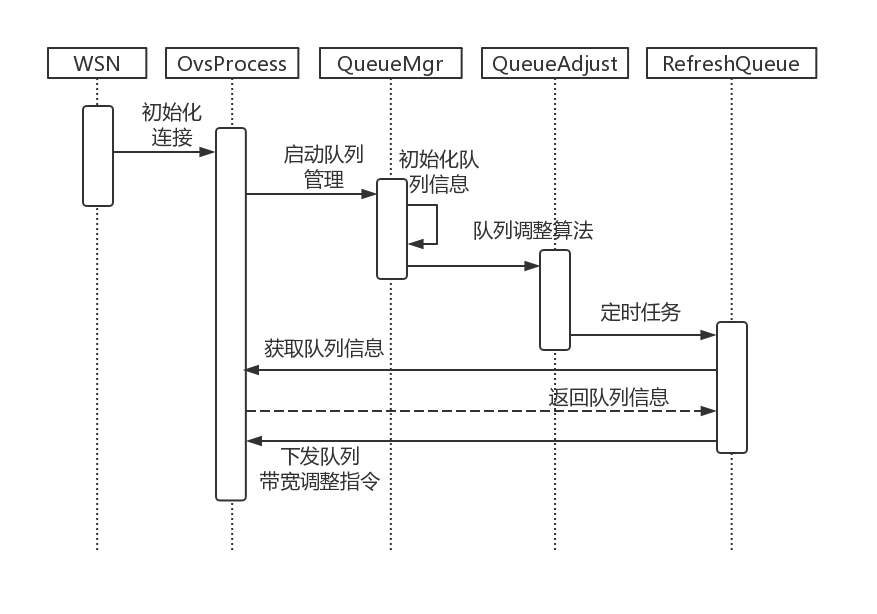


图5-35 队列调整时序图

1. **用户功能实现**

用户是系统的使用者，通过发布订阅系统和SDN网络，用户可以进行多媒体传输和需求协商。

1. **多媒体传输**

基于SDN的发布订阅系统建立后，用户可以在该系统上进行多媒体数据的传输，传输过程遵循的是RTP实时传输协议，关于该协议的基本概念已经进行了介绍，这里不再赘述。具体消息类图如下所示。

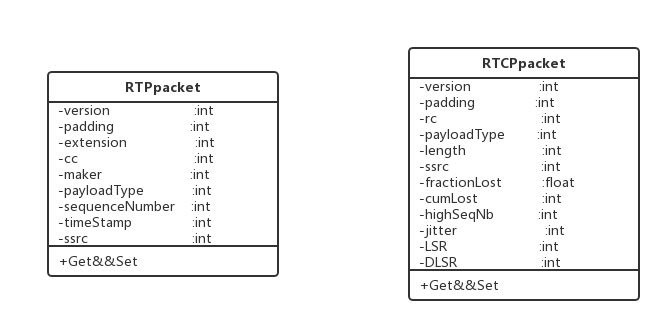


图5-36 多媒体传输类图

图中概括了RTP、RTCP的消息格式，下面介绍各字段含义。

表5-7 RTP、RTCP消息类定义

|  |  |  |  |
| --- | --- | --- | --- |
| 类名 | 字段名 | 字段类型 | 功能描述 |
| RTPPacket | version | int | RTP协议版本号，默认为2 |
| Padding | int | 填充标志 |
| Extension | int | 拓展标志 |
| Cc | int | CSRC计数器 |
| Maker | int | 帧标记位 |
| payloadType | int | 有效载荷类型 |
| sequenceNumber | int | 序列号 |
| timestamp | int | 时间戳 |
| ssrc | int | 同步信号源 |
| RTCPPacket | version | int | RTP协议版本号，默认为2 |
| padding | int | 填充标志 |
| rc | int | 接收报告计数器 |
| payloadType | int | 有效载荷类型 |
| length | int | 数据包长度 |
| ssrc | int | 同步源标识 |
| fractionLost | float | 丢失部分 |
| cumLost | int | 丢失包累计数量 |
| highSeqNb | int | 收到已扩展的最高系列号 |
| jitter | int | 间隔抖动 |
| LSR | int | 最后SR标识 |
| DLSR | int | 来自最后一个SR来的延迟 |

用户端发起视频传输请求，首先需要向接收方发送RTCP控制信息，包括数据流格式、RTP版本信息等，在收发双方确认了传输协议后才能进一步传输数据。数据获取模块以字节流的方式获取视频数据，经过填充数据包头部、校验位后，封装为RTP数据包，这样就可以调用发送模块将视频数据发送到接收方处，接收方经过反向解析就能得到视频数据。具体传输时序图如下所示。

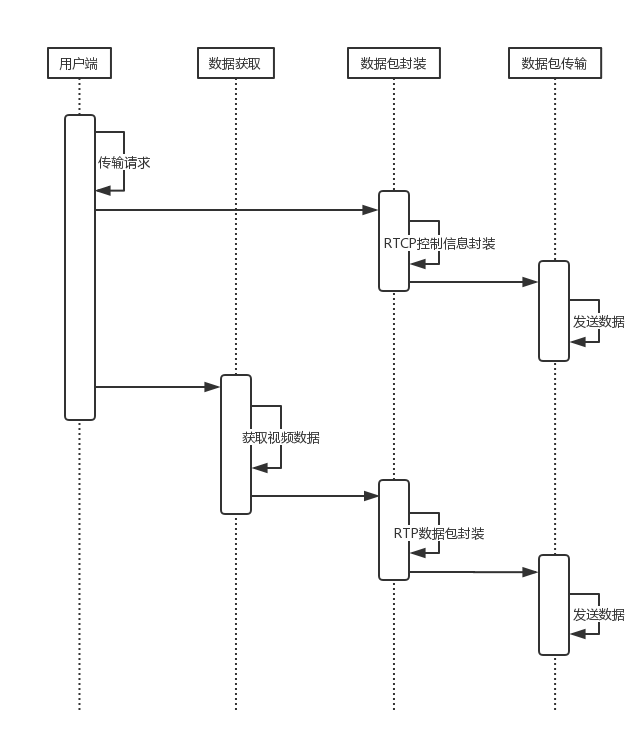


图5-37 多媒体传输时序图

1. **用户协商**

用户协商是系统质量保证方案的一部分，实现时首先需要对协商过程中出现的各种消息进行定义，具体消息类图如下所示。

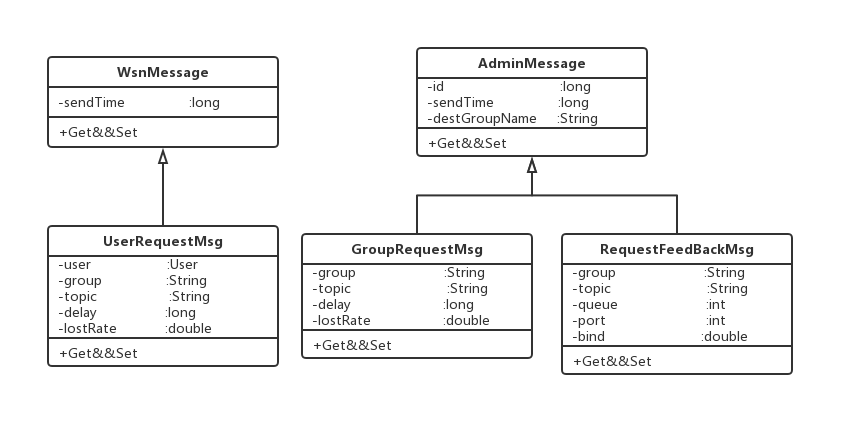


图5-38 用户协商消息类图

图中概括了用户协商流程中涉及的消息的组成，包括继承自WsnMessage（WSN层消息）的UserRequestMsg（用户请求消息），以及继承自AdminMessage（管理层消息）的GroupRequestMsg（集群请求消息）和RequestFeedBackMsg（请求反馈消息），下面介绍各字段含义。

表5-8 用户协商消息分类及定义

|  |  |  |  |
| --- | --- | --- | --- |
| 类名 | 字段名 | 字段类型 | 功能描述 |
| UserRequestMsg | user | User | 发起请求的用户信息 |
| group | String | 集群名称 |
| topic | String | 主题名称 |
| delay | long | 时延需求 |
| lostRate | double | 丢包率需求 |
| GroupRequestMsg | group | String | 集群名称 |
| topic | String | 主题名称 |
| delay | long | 时延需求 |
| lostRate | double | 丢包率需求 |
| RequestFeedBackMsg | group | String | 集群名称 |
| topic | String | 主题名称 |
| queue | int | 队列号 |
| port | int | 端口号 |
| bind | double | 带宽分配大小 |

用户发起协商请求，请求消息到达WSN模块后封装为UserRequestMsg消息格式发送给控制器，控制器收集并分析后，封装为GroupRequestMsg消息格式发送给管理员，在管理员处结合全网拓扑情况对用户请求进行处理，并将处理结果沿着管理路径下发给控制器，控制器收到RequestFeedBackMsg后将处理结果转换为流表并下发，用户协商过程至此完成。用户协商时序图如下所示。

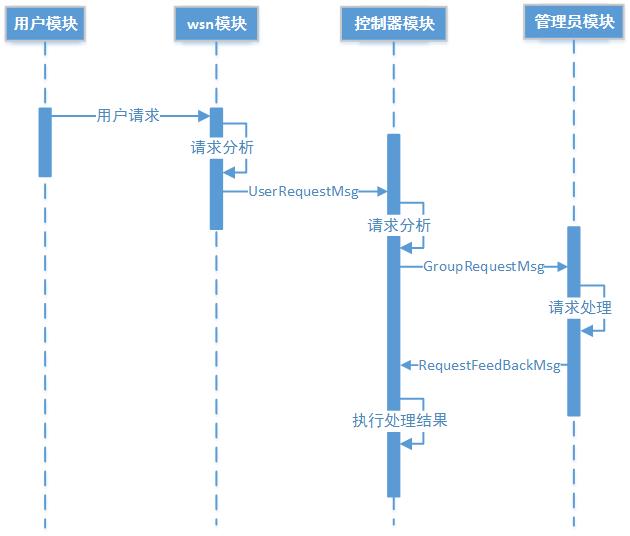


图5-39 用户协商时序图

1. **日志功能实现**

传统工程使用命令行打印的方式实现系统消息的展示，在发布订阅系统中，不论是消息的数量还是类别都更为复杂，需要一套新的消息输出机制。系统采用了log4j框架，将消息按照重要性划分为不同的级别，常用的包括trace、debug、info、warn、error等。日志系统的通知机制如下图所示。



图5-40 日志通知机制

系统提供了多级通知机制，普通消息trace、debug会打印在日志文件中，警告消息info、warn输出在命令行窗口中，系统错误消息error除基本展示外，还提供了邮箱提示功能。根据系统消息的不同级别区分处理，既提供直观的命令行输出，也具备日志文件存储、邮件提醒功能，这样给系统消息展示和代码维护也带来了便利。

1. **本章总结**

本章主要对系统核心模块的实现进行了详细阐述，包括SDN发布订阅系统架构实现、消息的定义与实现、控制器上发布订阅功能实现、管理员功能实现、WSN功能实现、发布订阅系统的QoS保证方案实现、用户功能实现、日志功能实现。第六章将通过系统测试和验证证明本系统的可靠性，以及质量保障方案的有效性。

2. **系统测试**

本章为系统测试，测试分为两部分：功能测试和性能测试。功能测试从系统的功能模块出发，测试各个模块运行的稳定性，同时验证模块间的协作处理逻辑是否正确，具体包括验证系统的SDN控制器功能、发布订阅功能、多媒体传输功能。性能测试从系统的可靠性出发，主要测试系统整体性能指标，判断是否符合预期设想。

1. **测试环境**

本系统部署于SDN物理网络中，使用发布订阅系统实现多媒体的传输，因此测试环境分为硬件环境和软件环境两个部分。

1. **硬件环境**

硬件测试环境包括SDN交换机设备、控制器及用户所在主机设备，具体参数如下表所示。

表6-1 交换机参数

|  |  |
| --- | --- |
| 参数名称 | 参数值 |
| 交换机型号 | Pica-p3290 |
| 支持OpenFlow版本 | OpenFlow1.3、1.4 |
| 支持OVS版本 | 2.3 |
| 网口数量 | 48 |
| 端口队列数量 | 最多支持8个优先级队列 |
| 支持IPV6地址 | 支持IPV6地址匹配 |
| 多级流表 | 支持 |

表6-2 主机参数

|  |  |
| --- | --- |
| 参数名称 | 参数值 |
| 主机型号 | 联想启天 B4360-B015 台式电脑 |
| 处理器 | 英特尔 Pentium(奔腾 ) G2030 @ 3.00GHz双核 |
| 内存大小 | 4GB |
| 硬盘大小 | 希捷500GB |
| 网卡 | 瑞昱 RTL8168/8111/8112 |

1. **软件环境**

由于控制器、管理员、WSN、用户端的代码由JAVA程序编写，因此相关的软件参数如下表所示。

表6-3 软件参数

|  |  |
| --- | --- |
| 参数名称 | 参数值 |
| 系统环境 | Windows 10 |
| JAVA版本 | JDK1.8.0\_131 |
| JAR包管理环境 | Maven依赖 |

1. **功能测试**

功能测试负责验证系统是否稳定运行，以及各模块能否完成相应的功能，本系统涉及的功能测试包括SDN控制器下属功能，以及发布订阅系统的全部功能，测试环境中的SDN交换机连接情况如下图所示。

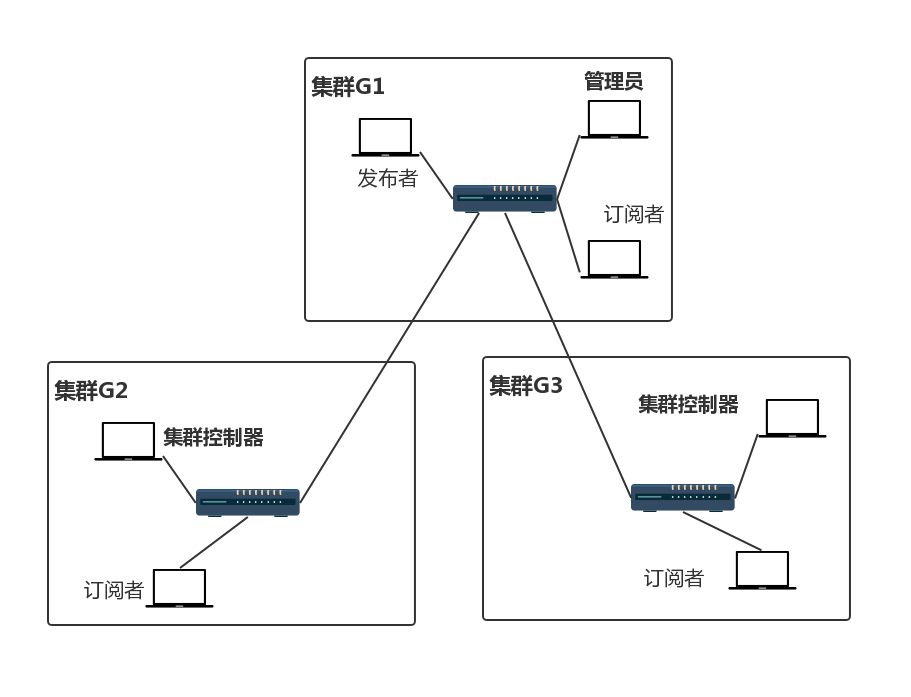


图6-1 功能测试拓扑

图中展示了真实的测试环境，集群G1上启动管理员程序，集群G2、G3上分别启动控制器程序，并通过网线连接到G1集群，同时在各个集群都存在相应的发布和订阅节点。下面详细说明功能测试的情况。

1. **拓扑功能**

系统需要具备拓扑功能，从而进行邻居的探测、全网链路的感知，涉及的功能模块如下图所示。

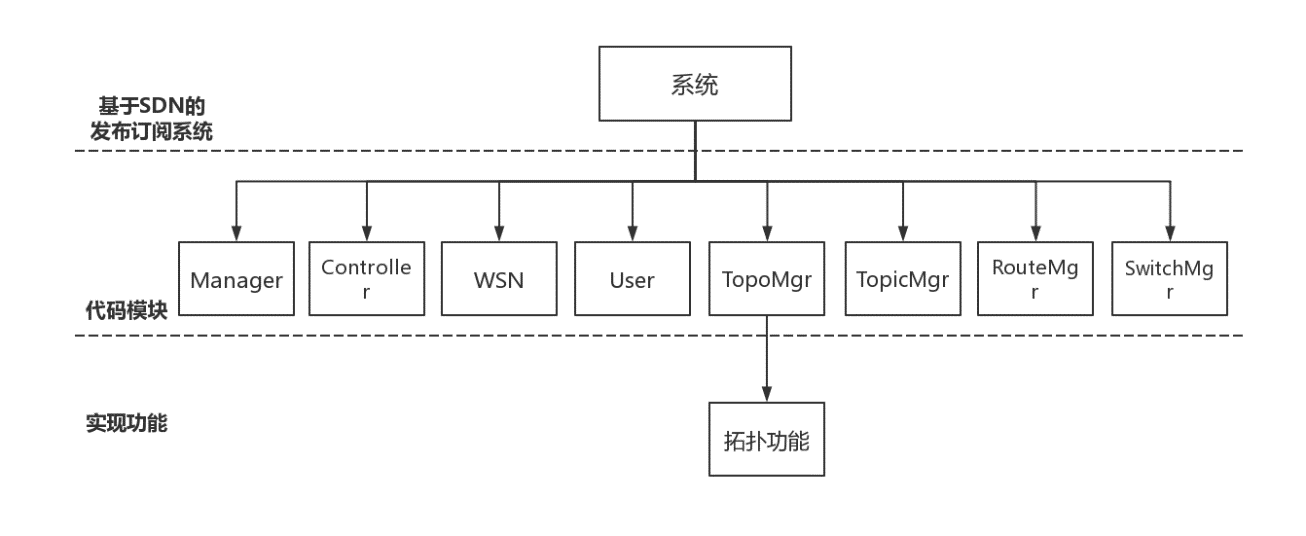


图6-2 拓扑功能涉及的代码模块

从图中可以看到，拓扑功能主要由TopoMgr拓扑管理模块完成，下面详细介绍拓扑功能的测试情况。

1. 拓扑构建

系统通过拓扑构建可以探测到邻居集群的存在，并保存邻居集群的相关信息，具体测试过程见下表。

表6-4 拓扑构建测试样例

|  |  |
| --- | --- |
| 测试项目 | 拓扑构建 |
| 测试目的 | 验证当新增邻居集群时，会根据ospf协议进行三次握手，从而建立邻居关系 |
| 测试步骤 | 1. 启动集群G1  2. 启动集群G2 |
| 预期结果 | 集群间进行了三次握手，最终建立邻居关系 |
| 实际结果 | 与预期相同 |
| 结果分析 | 拓扑模块采用的是ospf协议，控制器会定时向对外端口发送Hello消息，收到Hello消息的集群建立单向连接关系，并反向发送ReHello消息，初始集群收到后建立双向连接关系，至此，拓扑构建过程结束 |

拓扑构建的测试结果如下所示。

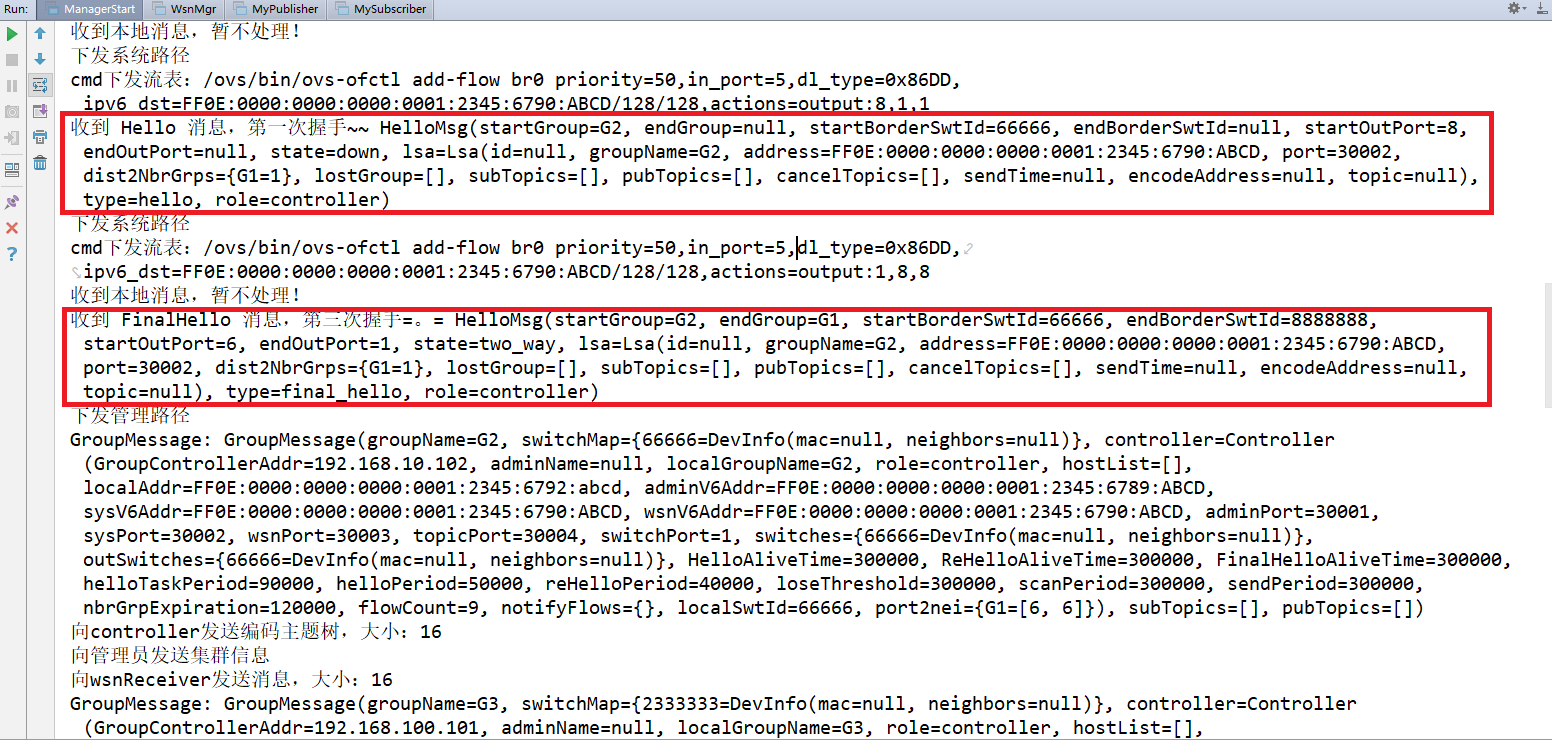


图6-3 拓扑构建结果（一）

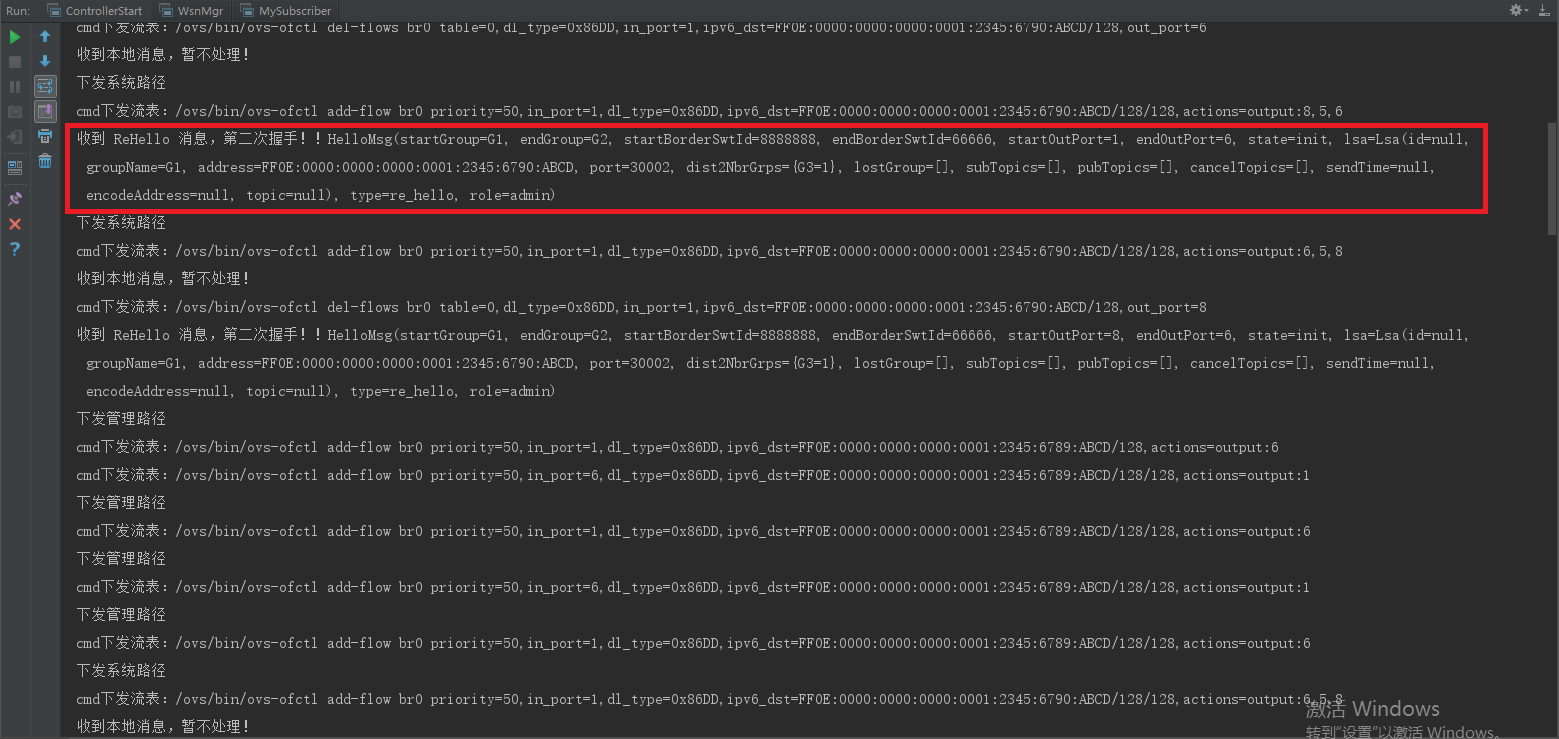


图6-4 拓扑构建结果（二）

1. 拓扑维护

集群简历连接关系后，还需要通过心跳消息、LSA消息进行拓扑维护，具体测试过程见下表。

表6-5 拓扑维护测试样例

|  |  |
| --- | --- |
| 测试项目 | 拓扑维护 |
| 测试目的 | 验证邻居集群可以通过lsa消息共享拓扑信息，从而完善全局拓扑 |
| 测试步骤 | 1. 启动集群G1  2. 启动集群G2  3. 启动集群G3 |
| 预期结果 | 1. 集群G1、G2间由于直接相邻，经过三次握手后建立双向连接关系  2. 集群G1、G3间由于直接相邻，经过三次握手后建立双向连接关系  3. 集群G1、G2间交换lsa消息，G2获知了G3集群的存在；  同样，集群G1、G3间交换lsa消息，G3获知了G1集群的存在 |
| 实际结果 | 与预期相同 |
| 结果分析 | 拓扑模块采用的是ospf协议，邻居集群间定期交换lsa信息，lsa构建出的lsdb保存了全局统一的拓扑，从而各个集群才能计算出相同的全网拓扑信息 |

拓扑维护的测试结果见下图。



图6-5 拓扑维护结果

1. **控制器功能**

系统需要具备控制器功能，一方面能够下发相应的流表，控制交换机的匹配情况，另一方面能够通过收集端口流量信息得到当前交换机的运行情况。涉及的功能模块见下图。

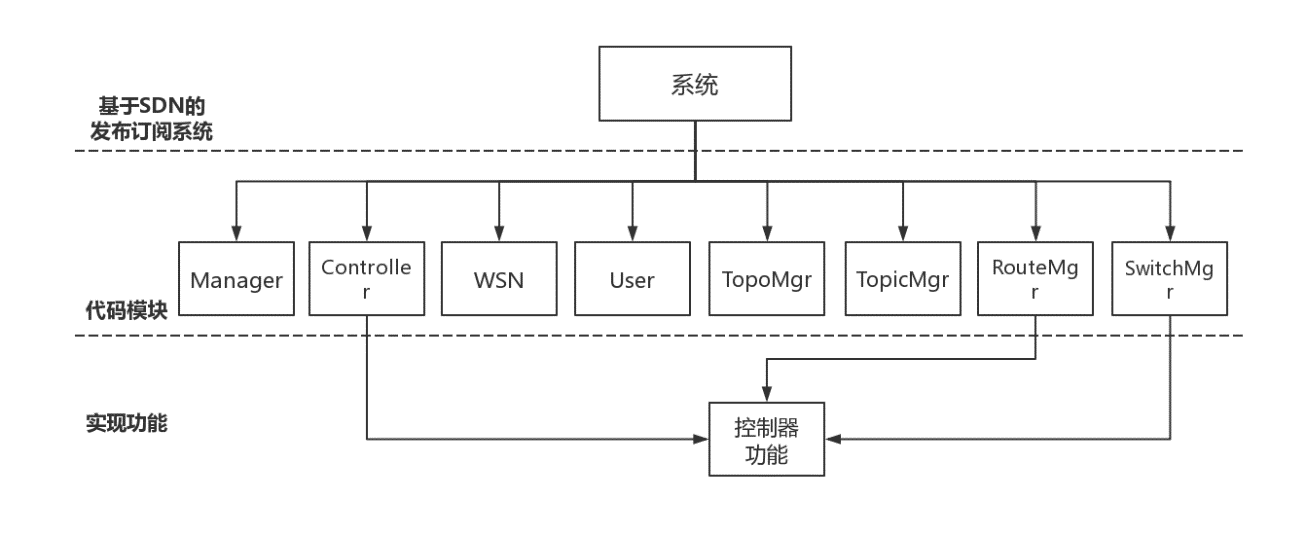


图6-6 控制器功能涉及的代码模块

图中列出的是控制器功能需要的代码模块，包括Controller控制模块、RouteMgr路由管理模块、SwitchMgr交换机管理模块，下面详细叙述控制器功能的测试情况。

1. 队列信息收集

系统能够获取到交换机的运行情况，包括端口流量、队列信息等，具体测试过程见下表。

表6-6 队列信息收集测试样例

|  |  |
| --- | --- |
| 测试项目 | 队列信息收集 |
| 测试目的 | 验证通过下发相应查询指令，可以获取交换机对外端口的队列信息 |
| 测试步骤 | 1. 通过ssh连接交换机系统  2. 遍历对外端口，下发查询队列语句  3. 查看获取的队列信息 |
| 预期结果 | 解析返回的查询结果，获得队列的进出字节数、速率等信息 |
| 实际结果 | 与预期相同，获取到了队列运行情况 |
| 结果分析 | 通过访问SDN交换机的核心处理器，系统能够获取到交换机的队列信息 |

队列信息收集的测试结果见下图。

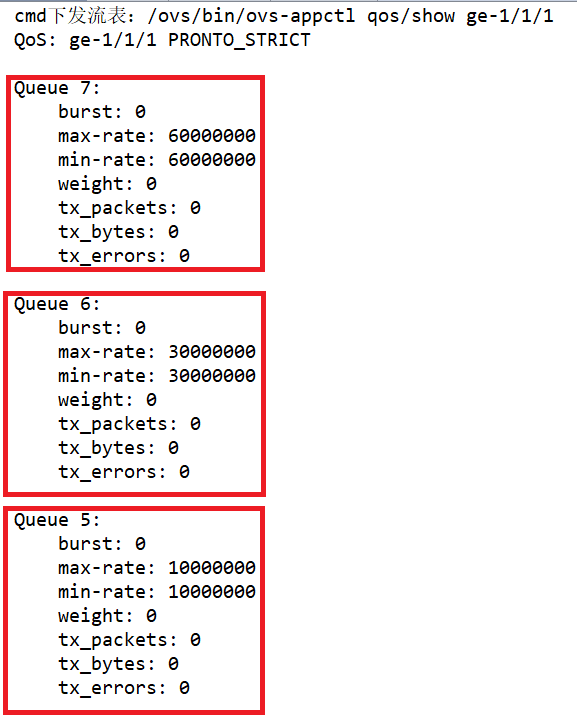


图6-7 队列信息收集结果

1. 流表生成

系统结合拓扑连接情况、用户订阅信息计算出转发路径，这条逻辑链路还需要转换为相应的流表，具体测试过程见下表。

表6-7 流表生成测试样例

|  |  |
| --- | --- |
| 测试项目 | 流表生成 |
| 测试目的 | 验证计算出的（管理、主题）路径，可以转换为SDN交换机识别的流表形式 |
| 测试步骤 | 1. 启动管理员  2. 启动控制器节点  3. 新增发布、订阅节点，触发主题路径的计算  4. 调用流表管理模块，转换为对应的流表 |
| 预期结果 | 计算出的主题路径是在集群G1、G3间针对主题“test1”的通路，具体表现为：  1. IPv6地址为“FF0E:0100:0000:0000:0000:0000:0000:0000”的消息可以从G1集群交换机的5、8端口进出；  2. IPv6地址为“FF0E:0100:0000:0000:0000:0000:0000:0000”的消息可以从G3集群交换机的2、3端口进出 |
| 实际结果 | 与预期相同，生成了对应主题的流表 |
| 结果分析 | 计算出相应路径后，根据保存的拓扑信息，可以得到具体端口进出情况，从而封装IPv6地址、交换机进出端口、优先级，转换为SDN交换机识别的流表项 |

1. 流表下发

生成的流表还需要进一步下发给SDN交换机，具体测试过程见下表。

表6-8 流表下发测试样例

|  |  |
| --- | --- |
| 测试项目 | 流表下发 |
| 测试目的 | 验证转换后的流表项，可以下发各指定交换机 |
| 测试步骤 | 1. 使用ssh方式远程连接交换机  2. 完成流表生成模块  3. 查看交换机流表信息 |
| 预期结果 | ssh远程连接上了指定的交换机，生成的流表项在交换机的流表中查找成功 |
| 实际结果 | 与预期相同，交换机中存在流表内容 |
| 结果分析 | 通过指定交换机IP、端口、用户名、密码，可以远程操作交换机的cmd窗口，进一步能够下发流表项，从而达到流表下发的目的 |

流表下发的测试结果如图所示。

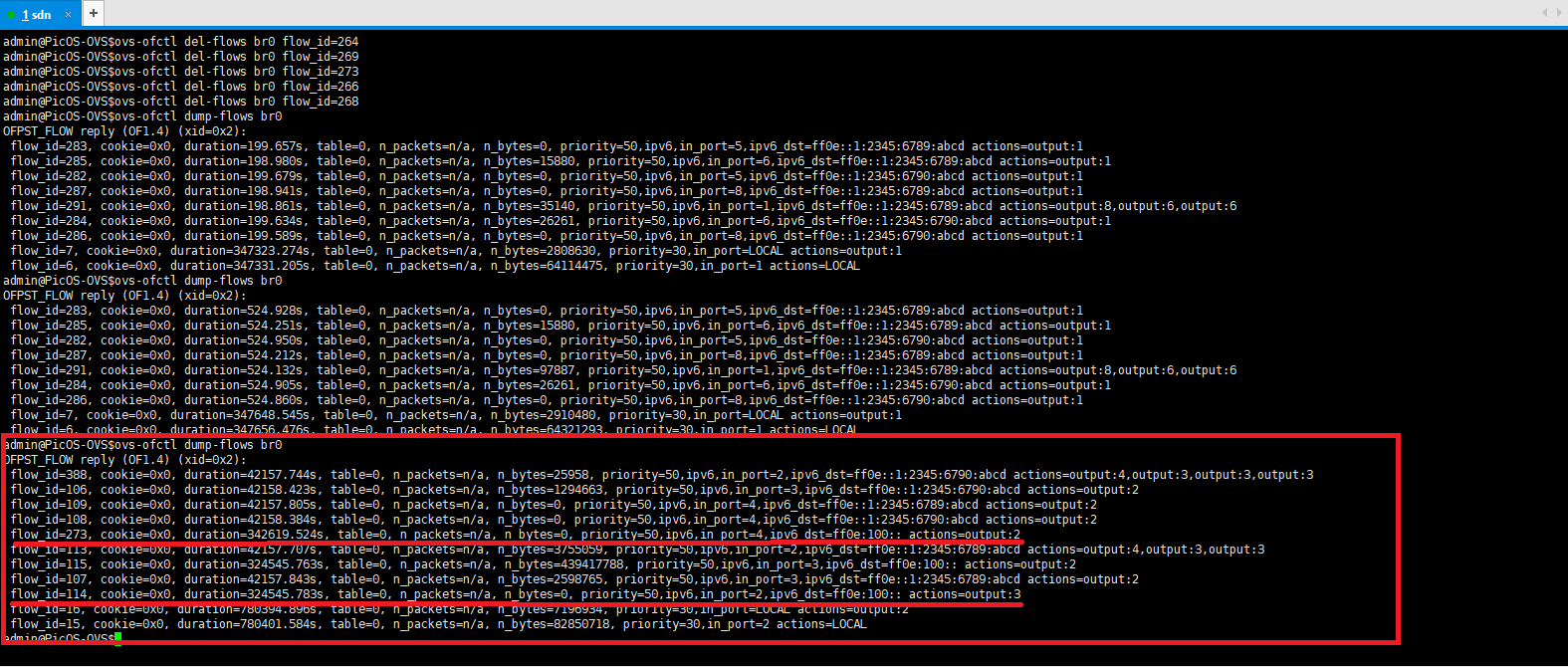


图6-8 流表下发测试结果

1. 流表维护

生成的流表以Flow类的形式存储在内存中，系统还需要具备相应的维护功能，用于判断流表是否重复或冲突，具体测试过程见下表。

表6-9 流表维护测试样例

|  |  |
| --- | --- |
| 测试项目 | 流表维护 |
| 测试目的 | 验证本集群内对下发的流表进行了维护，当有新的流表项加入时会自动保存 |
| 测试步骤 | 1. 完成流表下发测试模块  2. 继续生成流表并下发  3. 查看集群内的流表信息 |
| 预期结果 | 集群内对流表项进行了统一管理，整理为流表信息 |
| 实际结果 | 与预期相同，内存中保存了所有的流表内容 |
| 结果分析 | 由于交换机只识别单一流表项，因此需要对具有相同IPv6转发地址、进端口、优先级的流表项进行合并整合，这就需要在程序中维护一份和交换机相同的流表信息 |

流表维护的测试结果见下图。

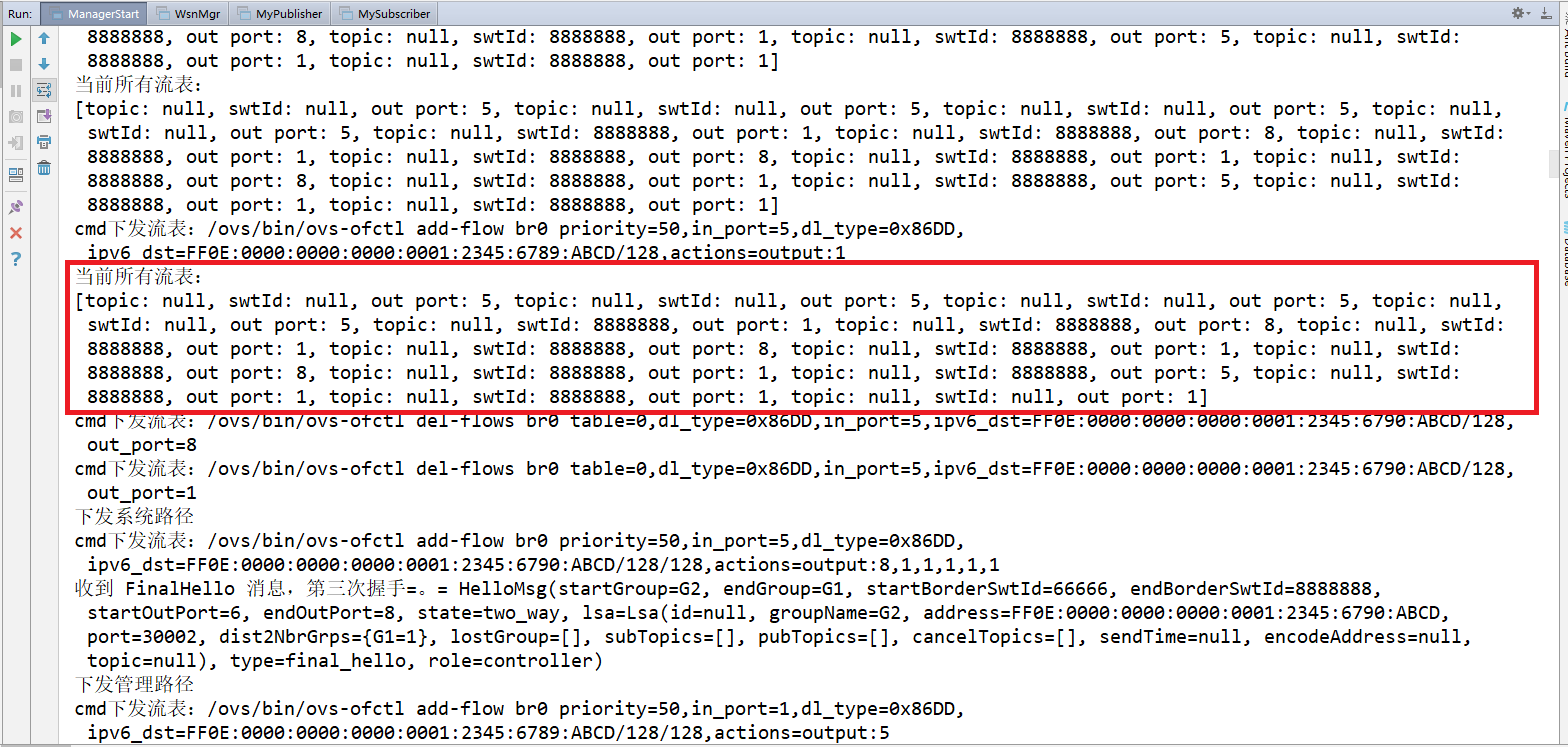


图6-9 流表维护测试结果

1. **主题树管理功能**

管理员处存在ui展示界面，能够方便管理员查看当前全网运行情况，同时也提供接口供管理员操作。涉及的功能模块如下图所示。



图6-10 主题树功能涉及的代码模块

图中，主题树管理功能主要由代码模块中的Manager管理员模块和TopicMgr主题管理模式实现，下面详细介绍主题树功能的测试情况。

1. 管理员ui界面展示并修改主题树

管理员ui界面可以展示当前主题树的详细信息，同时提供操作接口，管理员可以在ui界面上对主题树进行修改操作，具体测试过程见下表。

表6-10 主题树修改测试样例

|  |  |
| --- | --- |
| 测试项目 | 主题树修改 |
| 测试目的 | 验证针对主题的修改能够生效，并写回到XML文件中 |
| 测试步骤 | 1. 启动管理员模块，调用主题树修改接口  2. 在主题“test1-1-1”下添加“新建节点”主题  3. 重新读取主题树，并查看输出内容 |
| 预期结果 | 主题“test1”的内容被修改，重新打印的主题信息为修改后的主题树 |
| 实际结果 | 与预期相同，新增主题节点成功 |
| 结果分析 | 主题树保存在XML文件中，每次读取时采用dom4j方式，得到主题树句柄，遍历主题节点并保存在内存中；若在ui界面产生修改，则动态写回文件中，这样保证主题数据的一致性 |

主题树修改结果见下图。

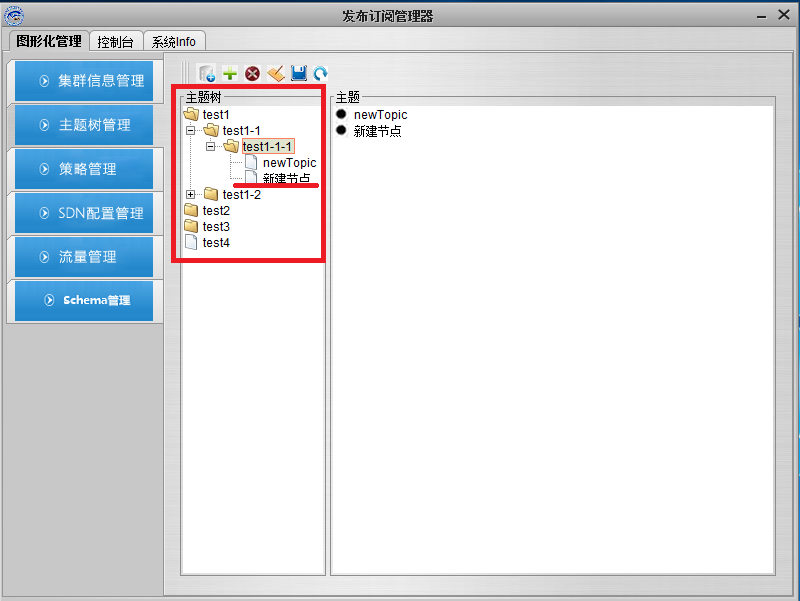


图6-11 主题树修改结果

1. 主题树下发

管理员模块在获取主题树后，需要进行编码并将编码主题信息下发给各个集群控制器，具体测试过程见下表。

表6-11 主题树下发测试样例

|  |  |
| --- | --- |
| 测试项目 | 主题树下发 |
| 测试目的 | 验证当启动控制器模块后，可以接收并保存来自管理员的主题树信息 |
| 测试步骤 | 1. 启动管理员模块  2. 启动控制器模块  3. 管理员通过管理路径下发编码主题树消息 |
| 预期结果 | 在启动控制器模块后，收到管理员下发的主题树信息 |
| 实际结果 | 与预期相同，控制器收到来自管理员的编码主题树消息 |
| 结果分析 | 管理员模块与控制器之间采取的是IPv6组播的方式，由于监听了相同的组播地址，因此可以接收到管理员下发的主题树信息 |

接收到的编码主题树信息见下图。

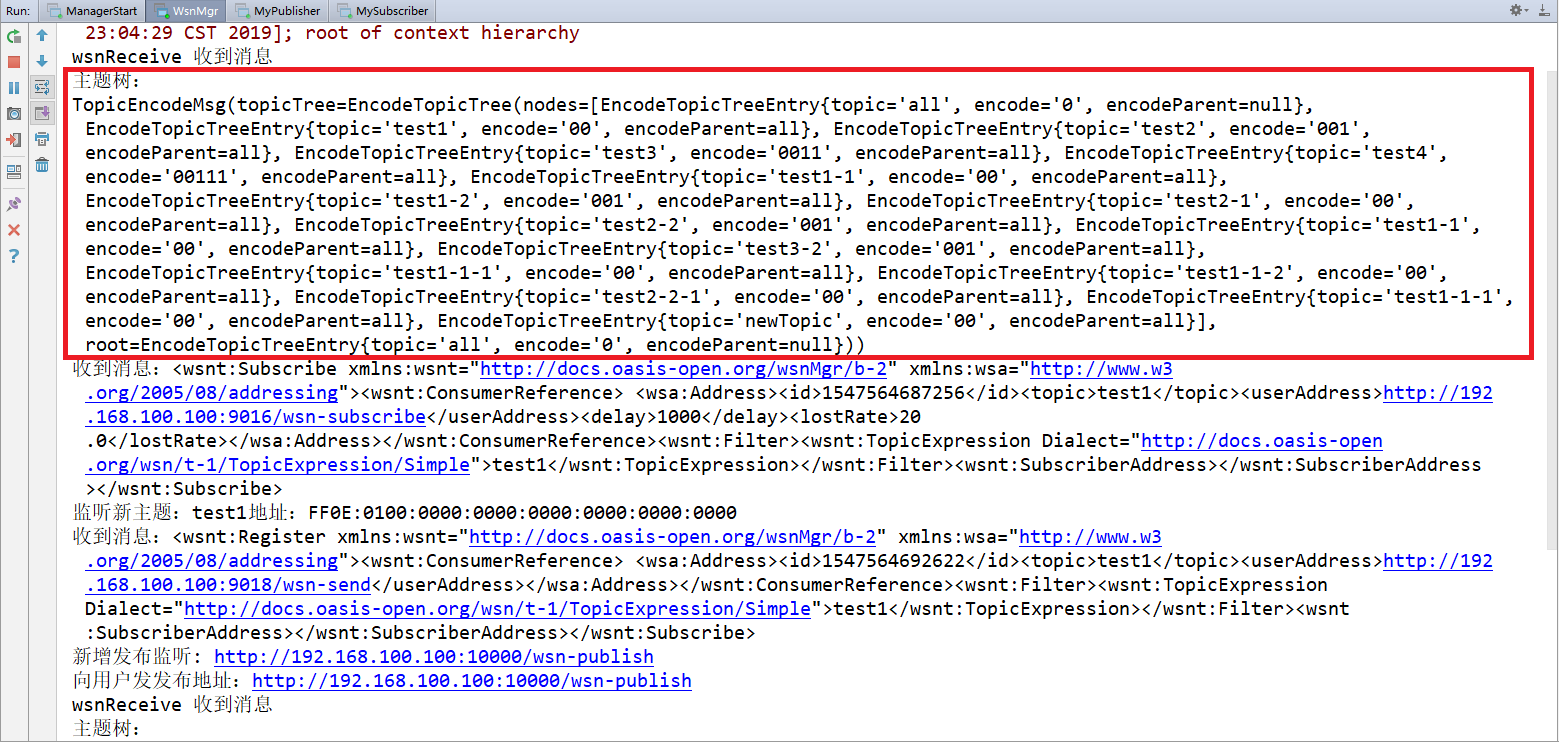


图6-12 主题树下发结果

1. **发布订阅功能**

系统需要支持用户发起发布、订阅请求，具体包括用户新增订阅、新增发布等，涉及的功能模块如下图所示。

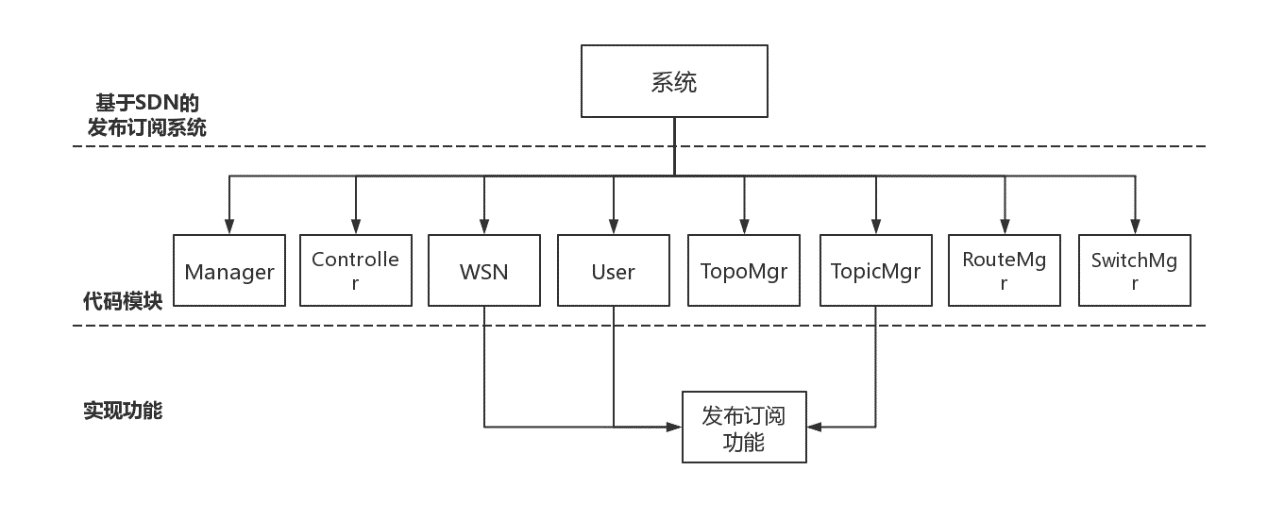


图6-13 发布、订阅功能涉及的代码模块

发布、订阅功能需要由WSN模块、User模块、TopicMgr主体管理模块共同协作完成。用户向WSN层发起请求，相应的主题必须存在且编码地址有效才视为一次合理请求。下面详细介绍。

1. 订阅请求

用户发起订阅请求，消息到达WSN层后，需要解析出主题对应的编码地址，同时WSN层需要开启主题监听，等待消息的传输。具体测试过程见下表。

表6-12 订阅请求测试样例

|  |  |
| --- | --- |
| 测试项目 | 订阅请求 |
| 测试目的 | 验证当WSN模块启动后，可以接收到下属用户的订阅请求 |
| 测试步骤 | 1. 启动WSN模块  2. 启动订阅用户，触发订阅请求 |
| 预期结果 | WSN收到用户的订阅请求，并监听主题对应的IPv6编码地址 |
| 实际结果 | 与预期相同 |
| 结果分析 | WSN模块与用户之间采取的是ws传输方式，因此WSN需要预先发布服务，用户向指定的服务地址发送请求，这样WSN模块在收到用户请求后根据字段分类处理，若为订阅请求，则在WSN层开启主题对应地址的监听 |

新增订阅的测试结果如下图所示。

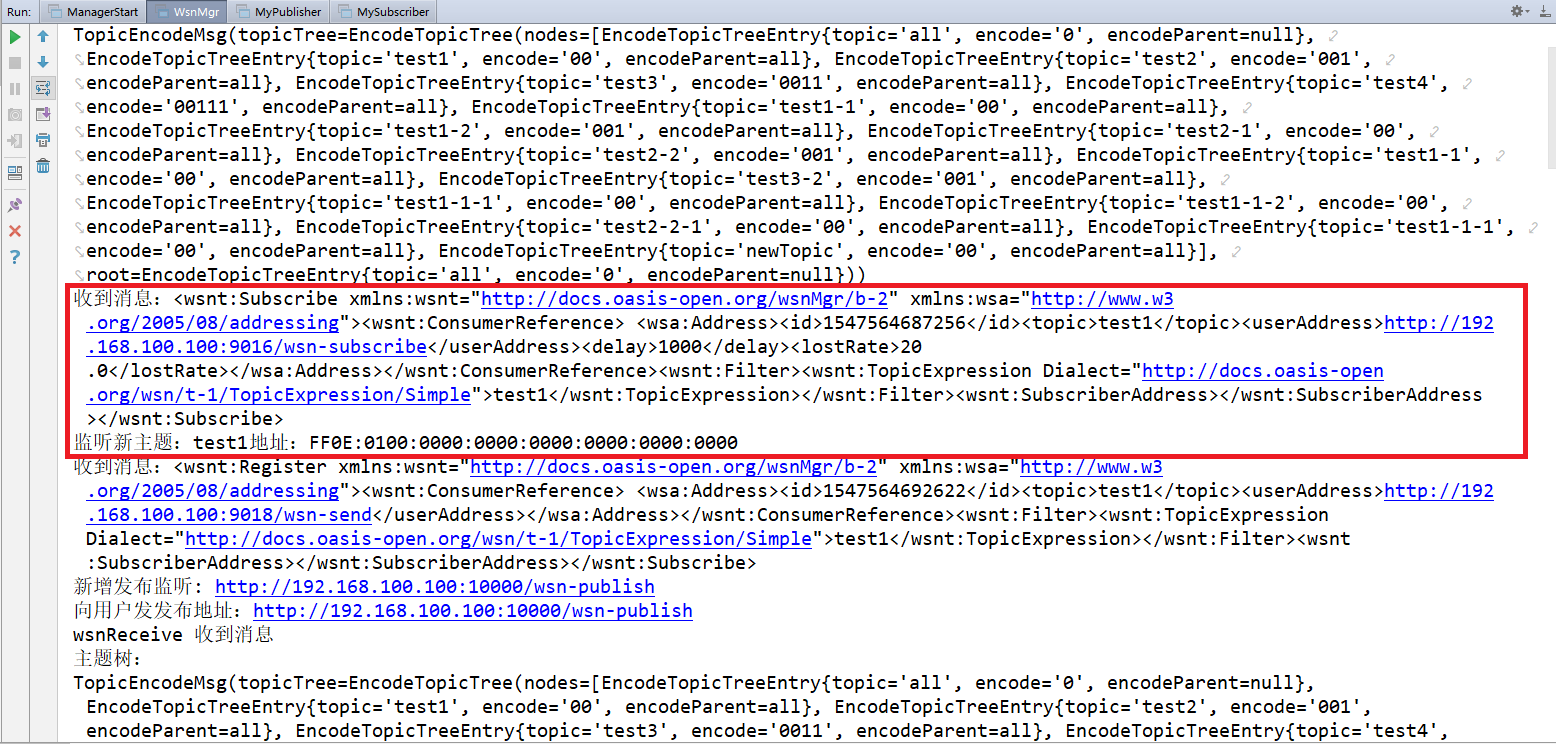


图6-14 新增订阅测试结果

1. 发布请求

用户发起发布请求，WSN收到后需要计算出一个可用的发布地址，并将结果返回给用户，这样用户直接向该地址推送消息即可，具体测试过程见下表。

表6-13 发布请求测试样例

|  |  |
| --- | --- |
| 测试项目 | 发布请求 |
| 测试目的 | 验证当WSN模块启动后，可以处理下属用户的发布请求 |
| 测试步骤 | 1. 启动WSN模块  2. 启动发布用户，触发发布请求 |
| 预期结果 | WSN收到用户的发布请求，生成发布地址并返回给用户 |
| 实际结果 | 与预期相同，WSN监听新的发布地址，等待用户的消息 |
| 结果分析 | WSN模块与用户之间采取的是ws传输方式，因此WSN需要预先发布服务，用户向指定的服务地址发送请求，这样WSN模块在收到用户请求后根据字段分类处理，若为发布请求，则生成新的发布地址给用户 |

新增发布请求的测试结果见下图。



图6-15 新增发布测试结果

1. **用户协商功能**

本系统的质量保证方案之一就是提供了灵活的用户协商机制，用户能够主动发起对时延、带宽的请求，交由控制器上报给管理员，管理员处结合全局链路运行状况分析计算，并将结果反馈给用户。涉及的模块见下图。

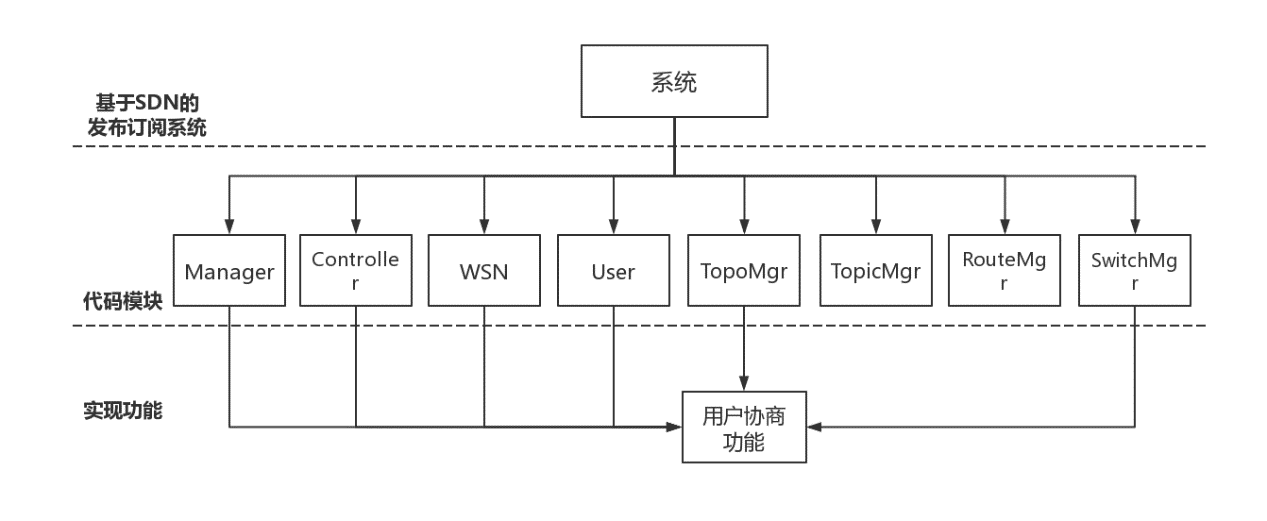


图6-16 用户协商功能涉及的代码模块

从图中可以看到，用户协商功能需要由Manager管理员、Controller控制器、WSN、User、TopoMgr拓扑管理、SwitchMgr交换机管理代码模块共同协作完成，具体测试过程见下表。

表6-14 用户协商测试样例

|  |  |
| --- | --- |
| 测试项目 | 用户协商 |
| 测试目的 | 验证系统能够支持用户协商机制 |
| 测试步骤 | 1. 完成上述所有测试模块  2. 开启订阅节点  3. 订阅用户发起时延请求，等待系统的处理 |
| 预期结果 | 管理员处收到了用户的时延需求，经过分析处理，将结果反馈给用户 |
| 实际结果 | 用户收到了管理员的处理结果 |
| 结果分析 | 本系统提供了完整的发布订阅功能，用户的请求经过控制器上报给管理员，处理完成后将结果反馈给用户，各模块相互合作共同完成 |

1. **多媒体传输功能**

当SDN系统、发布订阅系统稳定运行后，系统还需要支持多媒体传输，能够实现视频流的接收、播放功能。涉及的模块见下图。

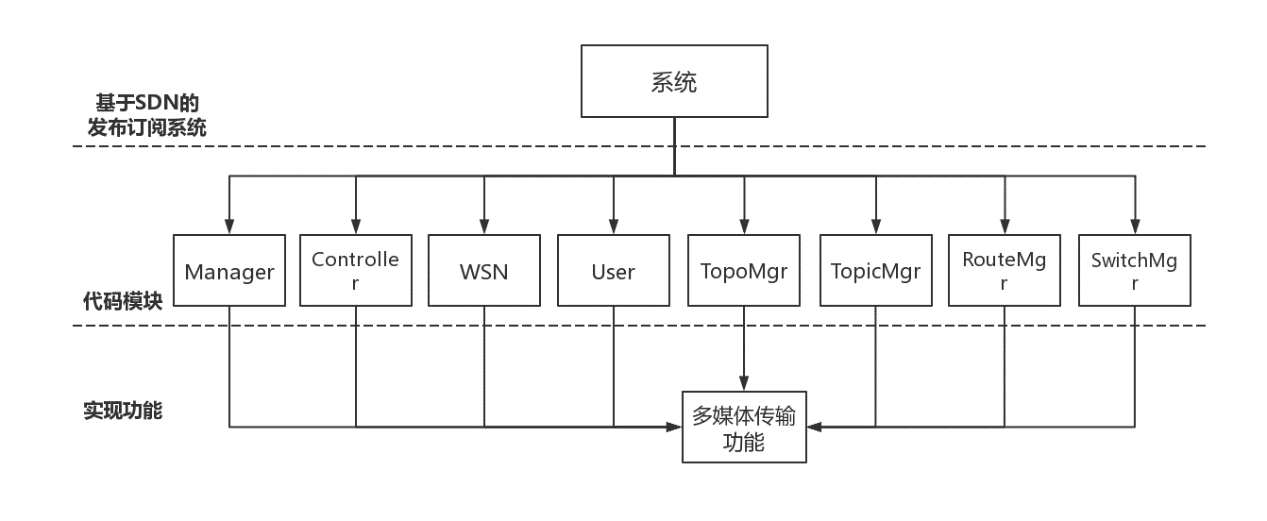


图6-17 多媒体传输涉及的代码模块

从图中可以看到，多媒体传输功能需要由所有的代码模块共同协作完成，具体测试过程见下表。

表6-15 多媒体传输测试样例

|  |  |
| --- | --- |
| 测试项目 | 多媒体传输 |
| 测试目的 | 验证系统能够支持多媒体数据的传输 |
| 测试步骤 | 1. 完成上述所有测试模块  2. 分别开启发布、订阅节点  3. 在发布方开启视频的传输 |
| 预期结果 | 系统可以支持多媒体的传输，在订阅方能够进行视频媒体的播放 |
| 实际结果 | 与预期相同，接收方正常播放订阅的多媒体视频 |
| 结果分析 | 本系统建立在SDN交换机上，并且具备发布订阅完整的系统功能，各模块相互协作，订阅者与发布者通过主题关联，视频数据使用RTP协议封装，经过路由达到目的节点，最终解析并完成播放 |

多媒体传输的测试结果见下图。

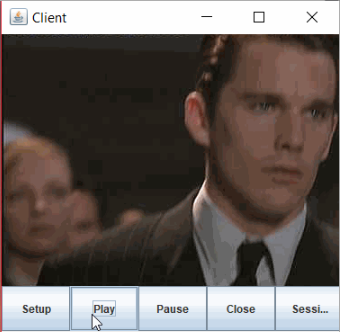


图6-18 多媒体传输测试结果

1. **功能测试小结**

经过上述功能模块的测试，可以看到，系统设计的各个子模块运行稳定，且相互之间通过协作，最终实现了多媒体数据的传输。

1. **性能测试**

系统的各个功能模块实现了功能性要求，但在实际应用过程中，还存在着性能上的需求。因此，在完成功能测试后，还需要进行性能测试，下图是具体的测试环境。

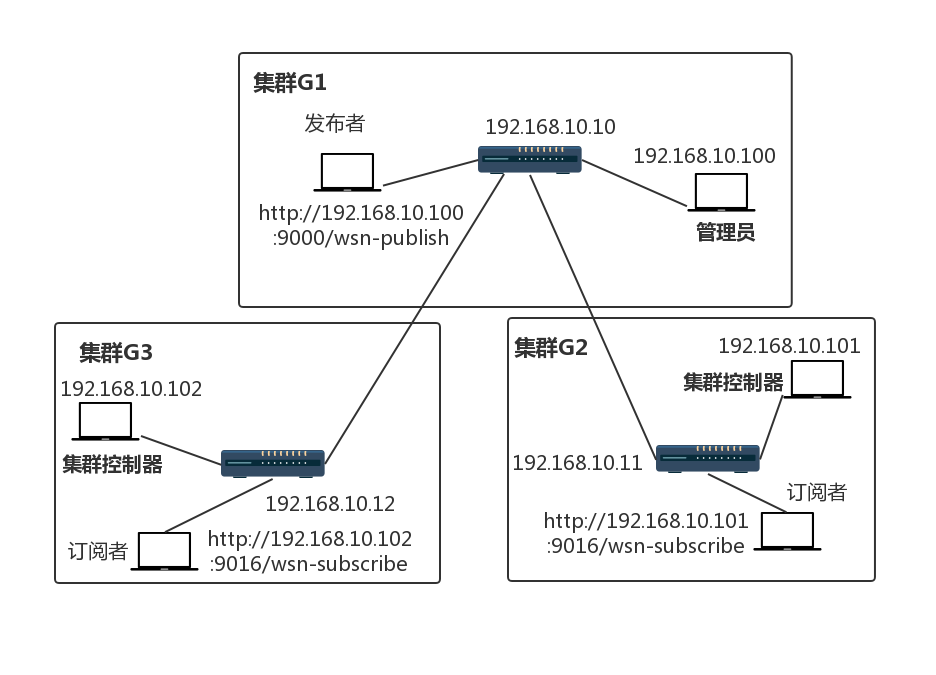


图6-19 性能测试拓扑

由第五章系统详细设计可知，传输模块存在三种方式：单线程、生产者-消费者、Reactor模式，其中单线程的传输性能见下图。

图6-20 单线程传输模式下丢包率结果图

图6-21 单线程模式下时延结果图

从图中可以看到，随着发包数量的增长，单线程模式下的时延平缓增长，并稳定在150毫秒左右，这是单线程传输的优势，但是丢包率却维持在一个很高的水准，在大规模传输条件下，丢包率达到90%，这对于用户来说是绝对不能接受的，因此单线程传输在多媒体、大数据的传输条件下无法使用，性能测试阶段就不再进行与单线程传输模式的比较。

在本系统的传输模块中，使用的是Reactor模式来替代生产者-消费者模式，本章将通过传输过程中的丢包率、时延，比较生产者-消费者与使用Reactor模式的系统性能情况。

1. **丢包率测试**

丢包率是发布订阅系统重要的指标之一，丢包率过大时将严重影响用户的服务体验，丢包率的影响因素包括发包数量、数据包大小、发包频率等，下面分别讨论。

1. 不同发包数量下，两种传输模式的丢包率情况

表6-16 不同发包数量对丢包率的影响

|  |  |
| --- | --- |
| 测试项目 | 不同发包数量下，两种传输模式的丢包率情况 |
| 测试目的 | 测试系统在不同发包总量下，使用Reactor模式调整前后的丢包率情况 |
| 测试环境 | 物理连接情况如上图所示  数据包大小：256B  发送方式：连续发送 |
| 测试用例 | 1. G1启动管理员集群，G2、G3启动控制器集群 2. G1新增发布节点，G2、G3新增订阅节点 3. 使用发布节点向订阅节点持续发送数据包，分别发送500、1000、1500、2000、2500、3000、3500、4000、4500、5000个数据包 4. 查看不同发包总量下，不同转发策略的丢包率情况 |
| 预期结果 | 1. 随着发包数量的增大，生产者-消费者方式、使用Reactor质量保障方案的丢包率会变化，但是稳定在某一个范围内 2. 生产者-消费者传输模式下丢包率大于Reactor模式 |

实际运行结果见下图。

图6-22不同发包数量下丢包率变化图（256B，连续发送）

从图中可以看到，随着发包数量的增大，两种模式的丢包率缓慢增加，说明发包总数对系统性能有影响，且发包数量越多，系统的丢包率越大；生产者-消费者方式的丢包率稳定在16%左右，而Reactor模式的丢包率在8%附近变化，说明使用了Reactor传输模式，丢包率明显降低，这是因为生产者-消费者方式传输时处理程序被动地接收任务，而Reactor模式可以控制任务分派过程，当消息量过大时丢弃耗时长的任务，从而使得系统性能获得提升。

1. 数据包大小对丢包率的影响

表6-17 数据包大小对丢包率的影响

|  |  |
| --- | --- |
| 测试项目 | 数据包大小对丢包率的影响 |
| 测试目的 | 测试系统发送不同数据包时，使用Reactor模式调整前后的性能情况 |
| 测试环境 | 物理连接情况如上图所示  发送方式：连续发送 |
| 测试用例 | 1、分别启动管理员集群、控制器集群  2、新增发布订阅节点  3、使用发布节点向订阅节点持续发送数据包，数据包大小分别为256B、512B、1024B  4、查看不同的数据包大小下，不同转发策略的丢包率情况 |
| 预期结果 | 1. 随着数据包内容的增大，生产者-消费者方式、使用Reactor质量保障方案的丢包率会变化，但是稳定在某一个范围内   2、生产者-消费者传输模式下丢包率大于Reactor模式 |

实际运行结果见下图。

图6-23不同数据包大小下丢包率变化图（256B，连续发送）

图6-24不同数据包大小下丢包率变化图（512B，连续发送）

图6-25不同数据包大小下丢包率变化图（1024B，连续发送）

从图中可以看到，随着数据包内容的增大，两种模式下的丢包率都有一定程度的上升，这说明数据包大小是对系统性能有影响，且数据包增大，系统的丢包率也会增大；不同数据包大小的丢包率基本稳定在一个范围，说明系统运行基本稳定；生产者-消费者的丢包率大于Reactor模式，证明了系统性能确实进行了提升。

1. 发包方式对丢包率的影响

表6-18 发包方式对丢包率的影响

|  |  |
| --- | --- |
| 测试项目 | 发包方式对丢包率的影响 |
| 测试目的 | 测试系统使用不同发包方式时，使用Reactor模式调整前后的性能情况 |
| 测试环境 | 物理连接情况如上图所示  数据包大小：256B |
| 测试用例 | 1、分别启动管理员集群、控制器集群  2、新增发布订阅节点  3、使用发布节点向订阅节点发送数据包，分别采用连续发送和每发送一秒休眠一秒的方式  4、查看不同的发包方式下，不同转发策略的丢包率情况 |
| 预期结果 | 1、采用休眠方式的丢包率都会降低  2、Reactor模式的丢包率低于生产者-消费者 |

实际运行结果见下图。

图6-26不同发布方式下丢包率变化图（256B，连续发送）

图6-27 不同发布方式下丢包率变化图（256B，间隔发送）

从图中可以看到，采用隔秒休眠的发送方式，丢包率会明显降低，这说明不同的发包方式对系统性能有影响，采用间隔发包的方式能够降低丢包率；两种发包方式下Reactor模式的丢包率都低于生产者-消费者，说明系统性能有了提升。

1. **时延测试**

由于传输的是多媒体视频数据，因此用户对于时延有一定的要求，时延过高就无法满足实时传输的需求，下面结合测试样例，比较不同传输保障方案的性能。

1. 发包数量对时延的影响

表6-19 发包数量对时延的影响

|  |  |
| --- | --- |
| 测试项目 | 不同发包数量下，两种传输模式的时延情况 |
| 测试目的 | 测试系统在不同发包总量下，使用Reactor模式调整前后的时延情况 |
| 测试环境 | 物理连接情况如上图所示  数据包大小：256B  发送方式：连续发送 |
| 测试用例 | 1. G1启动管理员集群，G2、G3启动控制器集群 2. G1新增发布节点，G2、G3新增订阅节点 3. 使用发布节点向订阅节点持续发送数据包，分别发送500、1000、1500、2000、2500、3000、3500、4000、4500、5000个数据包 4. 查看不同发包总量下，不同转发策略的时延情况 |
| 预期结果 | 1. 随着发包数量的增大，生产者-消费者方式、使用Reactor质量保障方案的时延都会平稳增加 2. 生产者-消费者传输模式下时延大于Reactor模式 |

实际运行结果见下图。

图6-28 不同发包数量下时延变化图（256B，连续发送）

从图中可以看到，随着发包数量的增大，两种模式的时延基本稳定，并逐渐增加，说明发包总数对系统有影响，且发包数量越大系统的时延越大；使用了Reactor传输模式，时延明显降低，这是因为生产者-消费者方式定时从缓冲队列中接收任务，而Reactor模式基于事件驱动，当消息到来时，分派器主动获取并调用相应的处理机制，速度更快，从而使得系统性能获得提升。

1. 数据包大小对时延的影响

表6-20 数据包大小对时延的影响

|  |  |
| --- | --- |
| 测试项目 | 数据包大小对时延的影响 |
| 测试目的 | 测试系统发送不同大小的数据包时，使用Reactor模式调整前后的性能情况 |
| 测试环境 | 物理连接情况如上图所示  发送方式：连续发送 |
| 测试用例 | 1、分别启动管理员集群、控制器集群  2、新增发布订阅节点  3、使用发布节点向订阅节点持续发送数据包，数据包大小分别为256B、512B、1024B  4、查看不同的数据包大小下，不同转发策略的时延情况 |
| 预期结果 | 1. 随着数据包内容的增大，生产者-消费者方式、使用Reactor质量保障方案的时延也会增加   2、生产者-消费者传输模式下时延大于Reactor模式 |

实际运行结果见下图。

图6-29 不同数据包大小下时延变化图（256B，连续发送）

图6-30 不同数据包大小下时延变化图（512B，连续发送）

图6-31 不同数据包大小下时延变化图（1024B，连续发送）

从图中可以看到，随着数据包内容的增大，两种模式下的时延都有一定程度的上升，这说明数据包大小是对系统性能有影响，且数据包增大，系统的时延也会增大；不同数据包大小下，生产者-消费者的时延都大于Reactor模式，证明了系统性能确实进行了提升。

1. 发包方式对时延的影响

表6-21 发包方式对时延的影响

|  |  |
| --- | --- |
| 1. 测试项目 | 发包方式对时延的影响 |
| 测试目的 | 测试系统使用不同发包方式时，使用Reactor模式调整前后的性能情况 |
| 测试环境 | 物理连接情况如上图所示  数据包大小：256B |
| 测试用例 | 1、分别启动管理员集群、控制器集群  2、新增发布订阅节点  3、使用发布节点向订阅节点发送数据包，分别采用连续发送和每发送一秒休眠一秒的方式  4、查看不同的发包方式下，不同转发策略的时延情况 |
| 预期结果 | 1、采用休眠方式的时延都会降低  2、Reactor模式的时延低于生产者-消费者 |

实际运行结果见下图。

图6-32 不同发包方式下时延变化图（256B，连续发送）

图6-33 不同发包方式下时延变化图（256B，间隔发送）

从图中可以看到，采用隔秒休眠的发送方式，时延会明显降低，Reactor模式下降低了一半左右，这说明不同的发包方式对系统性能有影响，且发送间隔一段时间能够降低传输时延；两种发包方式下Reactor模式的时延都低于生产者-消费者，说明系统性能有了提升。

1. **性能测试小结**

经过比较测试，可以知道发包数量对于时延、丢包率是正相关，即随着发包总数的增加，时延、丢包率也会平稳上升；数据包大小对时延、丢包率正相关，即数据包内容越大，时延、丢包率越大；发包方式对系统也有影响，采用间隔发包的方式能够降低丢包率和时延。

单线程传输的优势在于低时延，但是它在正常传输环境下的丢包率过大，甚至会达到90%，这对于任何一个追求稳定、高效的系统来说都是不能接受的；相比于生产者-消费者模式，Reactor模式在不同的测试条件下，时延、丢包率都有了明显提升；因此，综合看来Reactor模式在不影响传输效率的情况下，有效降低了传输过程中的丢包率和时延，使得系统的性能有了显著提升。

1. **本章总结**

本章对基于SDN网络的发布订阅系统中，功能和性能上进行了测试，包括拓扑功能、控制器功能、主题树功能、发布订阅功能、多媒体传输功能、QoS保障等，用实际数据和测试结果验证了本系统的可靠性，以及质量保障方案的可行性。

1. **总结与展望**

本章为工作总结和对未来的展望，主要对目前已经完成的内容进行总结性陈述，并对文章中的不足进行反思，同时提出自己对未来工作的设计思路，希望对今后的研究工作有所帮助。

1. **工作总结**

本文涉及的研究内容分为功能性的SDN发布订阅架构设计、多媒体传输，以及性能上的保障，涉及的知识点颇多。从2017年12月开题，到目前过了两年半的时间，期间除了两个月的实习期外，我都在努力完成本系统的功能，经过努力，目前功能性要求全部满足，性能上也提供了相应的保障。在开题阶段，提出了SDN网络与发布订阅系统相结合的要求，这与以往系统中的虚拟环境、开源控制器方式大为不同，需要开发出一套新的适合实验室研究环境的控制器系统。刚开始时千头万绪，无从下手，后来是在导师的帮助下，调整了研究思路，不着急于编码实现，而是先理清楚功能模块的划分及其之间的交互关系，同时从消息定义和分类出发，分别细化各个交互流程中包含的消息类别，这对于把握一个复杂系统，并深入实现提供了很大的帮助。随后，通过学习RTP实时视频传输协议，我进一步实现了多媒体传输功能。在系统研究后期，结合Reactor模式的事件驱动机制和用户协商机制也确保了系统整体的可靠性。经过相关的功能测试和性能测试，上述成果都得到了合理的验证。

当然，由于系统较为复杂，本系统还存在着或多或少的问题，比如管理员针对用户请求的分析处理，目前采用的算法只是单纯计算转发经过的集群数量，这其实是不太合理的，应该需要结合全网流量、拓扑连接关系重新计算带宽分配结果；此外，本系统对交换机的控制力度不够，不能实时监控每个端口的队列运行情况，今后的研究工作还需要进一步深入。

1. **工作展望**

本系统实现了SDN物理环境与发布订阅系统的结合，同时在该系统上能够进行多媒体传输，并提供多种质量保障方案，虽然从功能和性能角度都是一个比较成熟的系统，但是仍然还有一些方面值得进一步研究，主要包括：

1. 管理员计算

当前系统中，用户提出的时延、带宽需求在管理员处计算，依据的是时延总需求除以该主题转发路径经过的节点数，这对于网络实时环境来说误差太大，需要提出一种新型的需求计算方法，能够结合当前各集群的运行情况，动态地进行调整。

1. 消息持久化

目前的发布订阅系统并未在持久化上深入研究，订阅用户只有持续在线才能顺序接收对应的主题消息，若用户中途取消订阅或系统故障，则消息也会中断处理，这大大降低了用户的使用体验。如果系统引入消息持久化机制，利用缓存在WSN层将消息存储一段时间，那么即使用户掉线或丢失，也可以在下次恢复订阅时接收到所有消息，同时还能保证消息的顺序性，这有助于提升系统的性能。

1. 主题的分裂和聚合

主题是发布订阅系统连接的纽带，当前系统中，各级主题相互独立，但其实主题间是存在继承关系的，订阅了父主题则可以接收到所有子主题的消息，若不加处理，则用户会收到大量冗余信息。引入主题的分裂和聚合，可以有效缓解链路压力，进一步提升发布订阅系统的性能。

1. 大规模集群的部署

由于SDN交换机设备昂贵，且不同厂家的功能有较大差异，因此本系统部署的环境只基于3台pica型号的交换机，这在测试过程中难以开展多集群测试，今后需要多调研其他厂家的产品，综合起来扩大规模，提供更加具有说服力的数据和结论。

基于SDN的发布订阅系统以其良好的业务适应模式和优越的性能，正受到越来越多的关注，相信在今后的研究中会得到更好地开发。

**参考文献**

[1] 王丽君, 刘永强, 张健 . 基于OpenFlow的未来互联网试验技术研究的未来互联网试验技术研究[J]. 电信 网技术, 2011, 8(6): 1-4.

[2] Ng E. Maestro: A system for scalable openflow control[J]. Rice University, 2010. [3] 何国锋. OpenFlow在下一代数据中心网络的应用研究 [J]. 互联网天地互联网天地 , 2013, 3(3): 71-74.

[4] Canini M, Venzano D, Perešíni P, et al. A NICE way to test OpenFlow applications[A]. // Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)[C]. 2012: 127-140.

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

[6] Zhou L, Rodrigues J J P C. Service-oriented middleware for smart grid: PrincIPle, infrastructure, and application[J]. IEEE Communications Magazine, 2013, 51(1): 84-89.

[7] 张瑾，丁爱萍，马良. 度约束欧式Steiner最小树问题及其求解[D]. 上海：上海理工大学，2008

[8] 温鹏. 基于WS-Notification的高性能、高可靠性发布订阅系统[D]. 北京：北京邮电大学，2014

[9] 王双锦. 基于OpenFlow的发布订阅系统中拓扑和路由子系统[D]. 北京：北京邮电大学，2014

[10] 肖丹. 基于WSN发布订阅系统的实时可靠分发方案的设计与研究[D]. 北京：北京邮电大学， 2014

[11] 臧亚强. 基于SDN网络的发布订阅模式统一消息中间件核心设计与实现[D]. 北京：北京邮电大学，2015

[12]牛琳琳，章洋. 基于SDN的发布订阅系统拓扑维护及数据转发的研究与实现[D]. 北京邮电大学，2016

[13] 刘昌威. 基于SDN网络的发布订阅中间件路由计算模块的性能优化[D]. 北京：北京邮电大学，2017

[14] 韩波. SDN网络中发布订阅系统性能优化方案的研究与设计[D]. 北京：北京邮电大学，2018

[15] Pfaff B, Pettit J, Koponen T, et al. The design and implementation of OpenvSwitch[A]. // 12th USENIX symposium on networked systems design and implementation (NSDI 15)[C]. 2015: 117-130.

[16] Huang T, Yan S, Yang F, et al. Building SDN-Based Agricultural Vehicular Sensor Networks Based on Extended OpenvSwitch[J]. Sensors, 2016, 16(1): 108. [17] Pan H Y, Wang S Y. Optimizing the SDN control-plane performance of the OpenvSwitch software switch[A]. // 2015 IEEE Symposium on Computers and Communication (ISCC)[C]. IEEE, 2015: 403-408.

[18] Vahdat A, Clark D, Rexford J. A Purpose-built Global Network: Google's Move to SDN[J]. Queue, 2015, 13(8): 100.

[19] Lara A, Kolasani A, Ramamurthy B. Network innovation using Openflow: A survey[J]. IEEE Communications Surveys & Tutorials, 2014, 16(1): 493-512. [20] Kobayashi M, Seetharaman S, Parulkar G, et al. Maturing of OpenFlow and Software-defined Networking through deployments[J]. Computer Networks, 2014, 61: 151-175.

**攻读学位期间发表的学术论文目录**

[1]张冠群，章洋. 基于SDN发布订阅系统中多媒体传输质量保证方案的设计与实现. 北京：中国科技论文在线 [2018-12-21]. http://www.paper.edu.cn/releasepaper/content/201812-95.