1. **绪论**
   1. **研究背景**

随着互联网的不断发展，网络带宽不再成为限制数据传输的瓶颈，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，存储转发的传输模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，无法为多媒体传输质量提供可靠保障，因此，急需一种稳定、高效的新型网络传输模式，能够提升用户的服务体验。

SDN（Software Defined Networking）是一种新型的网络架构，与传统网络匹配--转发模式相比，它将网络交换机、路由器中的控制层与转发层剥离开来，网络层设备仅负责流量的转发，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流到达交换机时，传统网络的处理方案是使用默认规则（视硬件而定），但每个交换机不会对全网拓扑有完整的认知，因此采取的默认方法有很大几率是错误的，浪费了链路资源，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至目的交换机，从而提高转发的准确率。传统网络中，对流量的控制和转发都依赖于网络设备实现，且设备中集成了与业务特性紧耦合的操作系统和专用硬件，这些操作系统和专用硬件都是各个厂家自己开发和设计的，在实际应用中很难人为的控制其转发的行为，在灵活性等方面有着很大的不便。在SDN网络中，网络设备只负责单纯的数据转发，可以采用通用的硬件；而原来负责控制的操作系统将提炼为独立的网络操作系统，负责对不同业务特性进行适配，而且网络操作系统和业务特性以及硬件设备之间的通信都可以通过编程实现。

发布-订阅（publish–subscribe）是一种消息传播模式，消息的发送者（发布者）不会将消息直接发送给特定的接收者（订阅者），而是将发布的消息按主题分类，把带有主题的消息传送给网络节点，而无需对订阅者（如果有的话）有所了解。同样的，订阅者可以表达对一个或多个类别的兴趣，只接收感兴趣的消息，无需对发布者（如果有的话）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，消息的生产者与消费者实现了脱离，可以更好地完成消息的交互。这种发布者和订阅者的解耦可以允许更好的可扩放性和更为动态的网络拓扑。基于SDN的发布/订阅系统是一个软硬件结合、以软件为主的网络应用系统，其优点在于借助SDN网络高度集中式管理、可动态改变、可编程的特点，解决传统网络中无法保证的数据时效性、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，最终构建出一个可控且可靠的统一消息中间件网络。

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，同时因为SDN交换机的造价昂贵，生产厂家只有华为、思科这些大型公司，因此不少高校、研究机构采用虚拟化技术，比如Open vSwitch、Mininet等，这在一方面带来了极大地便利性和可行性，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暴露无遗，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，本课题基于真实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新的路径，为多媒体数据的传输增强时延、抖动上的保证；使用发布订阅系统模拟真实网络环境下用户间的消息传输行为，从而适应动态的网络需求。本课题在原有系统的队列调整、流量管理的基础上，将原有功能模块迁移并采用全新的设计理念，为多媒体数据传输、路由算法、消息接口等需求提供支持，同时，编码实现多媒体数据传输，使用Reactor模式和用户协商机制提供传输质量上的保证。

* 1. **研究内容**

本文的主要内容是SDN发布/订阅系统中多媒体传输质量保证方案，系统在物理SDN交换机上部署、运行、测试，避免了虚拟化技术的弊端，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效率上的提升。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，实现视频流端到端的传输；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**

已有系统采取的是模拟环境下，使用OpenDayLight控制器软件监测虚拟网桥、虚拟交换机，这对于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并非完全一致，遇到的问题也各有不同，因此，本系统的重点目标是开发一款控制器系统，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、主题管理、消息接口、路由计算、流表管理这几个方面。

拓扑管理负责新集群加入时的拓扑探测，以及后续的拓扑维护，本系统采用的是OSPF协议，邻居集群间会通过三次握手的方式进行感知，并将邻居信息保存在本地LSDB数据库中，这样经过信息交互，所有集群都将获得全网拓扑信息，便于各个集群后续的独立计算。拓扑又分为集群内和集群间拓扑，系统将网络划分为若干集群，每个集群由一个集群控制器进行统一管理，这样既可以减轻单个控制器的业务压力，也可以保证每个节点的路由层只需给部分交换机下发流表项，这样就可以有效地减少交换机和控制器之间的交互，从而提高发布/订阅管理系统整体的稳定性。

主题管理的功能是通过编码的方式，快速区分不同主题，并能够在主题内容与ipv6地址间形成一一映射关系，方便消息的转发。基于 SDN 网络的发布/订阅系统的主题与流表以及路由绑定，因此主题会与一条自定义匹配项绑定，通过这条匹配项，加上 OpenFlow 交换机支持的流表项精确匹配，节点可以对消息进行更准确的转发，同时节点也可以根据这条匹配项来对流表进行管理。

消息接口方面，是发布/订阅节点与用户之间的消息交互接口，用户通过这个接口实现发布、订阅的功能，同时也可以提出时延、带宽上的需求，节点最终计算结果也是通过相应接口反馈给用户，本系统采用的是web service 方式，面向接口实现，便于拓展与维护。

路由计算方面，分为管理路径的计算和主题路径的计算，结合拓扑管理模块生成的全网拓扑，以及消息接口模块提供的用户订阅关系，调用相关算法计算出转发一条路径。

流表管理是发布订阅系统与SDN物理环境结合的重要一环，吸收了SDN网络的控制、转发分离的特性和基于发布/订阅系统主题匹配机制的优点，将其结合在一起形成了“集中管理，分布计算”的体系结构，通过指定转发端口、目的ip地址，系统将消息转发路径转换为具体的流表，并通过ssh的方式下发给控制器，这样当消息到来时SDN交换机直接匹配转发，最终可以实现数据在物理链路上的高效传输。

上述功能是系统的基本组成架构，各个模块需要相互协作，拓扑管理用于邻居集群的发现，构建出全局链路连接情况，主题管理是发布/订阅的核心功能，用户间通过主题进行联系，消息接口是用户与系统交互的方式，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，最终为多媒体的传输提供保障。

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字节流，这并不符合真实网络环境中用户的需求，在实际传输过程中，音频、视频等多媒体文件正逐渐成为互联网中信息交互的主体，同时用户对于实时性也有进一步的要求，实时传输视频流也是重要的一环。

RTP协议是由IETF开发的实时传输协议，可以在面向连接或无连接的下层协议上工作，通常和UDP协议一起使用；RTP定义了两种报文：RTP报文和RTCP报文，RTP报文用于传送媒体数据（如音频和视频），RTCP协议用于传送控制信息，以实现协议控制功能，这是一种基于接受者反馈的网络传输QoS监测机制，在RTCP的接收报告中包含了当前网络传输QoS有关信息，如报文丢失率、平均时延等，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，提供了多队列传输、路由计算等功能，可以在不同队列中分别传输RTCP、RTP信息，同时根据RTCP中得到的反馈，重新计算当前网络环境中发布/订阅者间相应的路径，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质量。

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**

已有的发布订阅系统中，针对消息的发布大多采用单线程直接转发，或是线程池并发处理，这在数据量较小时运行良好，但经过测试，随着发布消息的增大，这两种方式都存在相应的弊端：单线程转发的优势在于低时延，线性处理，但是在面对高压力的情况下丢包率迅速增大；多线程采用并发的方式能够在一定程度上缓解丢包率过大的问题，但是一方面多线程的处理机制不可避免的增加了转发时延，另一方面，多线程的新建、销毁也给机器带来了负担。

本系统在经过讨论设计后提出了两种质量保障方案：Reactor模式和用户协商机制。Reactor模式的核心思想是使用阻塞队列缓存消息，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，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。Reactor模式是在已有方法上的提升，综合考虑了丢包率、转发时延等因素，对系统整体性能有一定提升。同时，系统还提供了用户协商机制，用户通过发起针对丢包率、时延的请求，由控制器上报给管理员，管理员维护了全网拓扑和各个集群的运行情况，经过分析处理，可以为用户的请求计算出合理的结果，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，由用户自行决定是重新请求还是接受当前处理结果。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QoS保障功能。

这两种质量保障方案中，Reactor模式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，用户协商机制是用户视角的主动请求，更加具有有效性和说服力，这两种方式在实际的测试中，系统性能都有了显著提升。

* 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，说明了本文涉及的相关技术和系统知识，包括软件定义网络、发布订阅系统、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、基本流程和涉及的算法；

第五章是系统详细设计与实现，详细阐述了各模块的具体实现方式，消息的定义、方法调用、核心算法等；

第六章是系统测试与验证，该章节介绍了测试环境、测试方案、测试结果，并得出具体的测试结论，证明了本系统的可靠性，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。

2. **相关技术概述**

本文的研究内容是SDN网络中，结合发布订阅系统，实现多媒体传输质量保障方案，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，下面将对这些内容进行简要介绍，以便于读者对本文涉及的技术背景有基本的了解。

1. **软件定义网络技术**

软件定义网络技术是一个复杂的概念，从广义上讲，软件定义网络，指的是具备可编程特性、控制与转发分离的网络架构；而在狭义范围内，软件定义网络单指基于OpenFlow协议的南向接口网络。本文将从如下几个方面进行介绍。

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OpenFlow通过将网络设备的控制面与数据面分离开来，从而实现了网络流量的灵活控制，使网络作为管道变得更加智能，为核心网络及应用的创新提供了良好的平台。SDN的架构如图2-1所示。

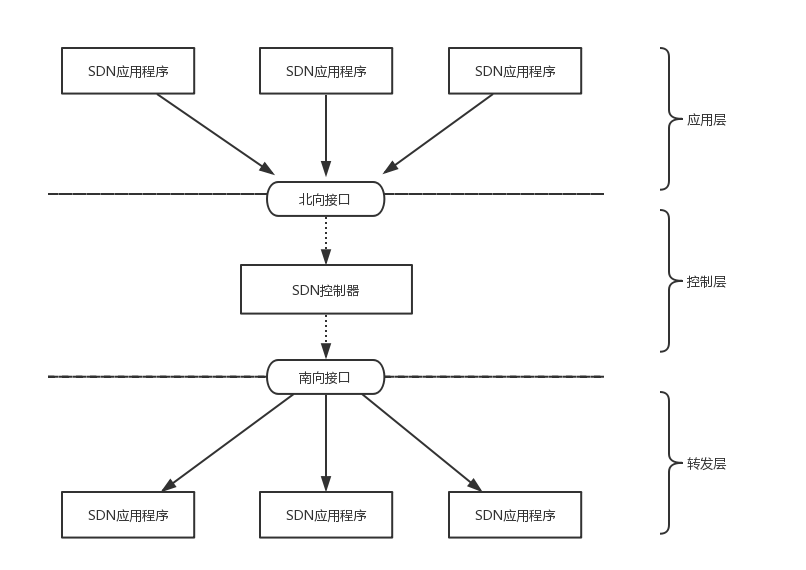


图2-1 SDN架构图

图中SDN的核心架构分为三层，最上面的是应用层，主要负责为用户提供SDN应用程序和业务需求；中间为控制层，在该层实现了拓扑管理、路由计算以及数据资源的控制；最底层是转发层，也称为基础设备层或数据层，负责基于流表匹配的数据转发和设备状态反馈。这里，我们将应用层于控制层之间的接口称为北向接口，主要提供控制层到应用层的API（应用程序编程接口）；将控制层和转发层之间称为南向接口，负责定义控制和转发之间的协议规范，目前比较通用的是OpenFlow协议。

1. **OpenFlow协议**

经过上面的描述，我们可以知道，OpenFlow协议与SDN是两个不同的概念，SDN是一种架构设计理念，而OpenFlow是一种协议规范，规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。

OpenFlow的最初概念是从2008年在斯坦福大学开始，到2009年12月，OpenFlow交换规范1.0版发布，自成立以来，OpenFlow一直由开放网络基金会（ONF）管理，ONF是一个致力于开放标准和SDN应用的用户主导型组织。由于OpenFlow已经发展成为较为成熟的协议，因而在众多的南向接口协议中脱颖而出，成为SDN通信协议的事实标准。OpenFlow的设计思路是在SDN交换机中维护自身的转发流表，关于流表的概念将在下面叙述。消息到达交换机后，首先进行包头域和流表项的匹配，如果匹配成功，则执行相应的操作，如转发、丢弃等，若匹配失败，则通过安全信道转发至交换机进行后续的处理。流表的生成、下发及维护都需要由控制器提供，关于SDN控制器的概念将在下面给出。

OpenFlow自从2008年提出到现在，经过了多次的版本更迭，由1.0版本发展到了1.5版本，随着版本的更新，它的功能也进一步的完善，在移动网络、广域网、数据中心网络等领域的应用越来越广泛，为SDN网络的发展提供了良好的支持。

1. **SDN控制器**

作为控制中心，SDN控制器在软件定义网络中担任着重要角色。网络拓扑的管理、路由的计算、流表的下发与维护都需要由控制器完成，控制器通过南向接口与交换机进行交互，从而获得SDN的控制权；同时在北向接口，提供API接口供开发人员自己使用，丰富系统功能。

目前，市面上存在多款开源的控制器软件，比如OpenDayLight、Floodlight、Ryu、ONOS等，已有的发布/订阅系统采用的就是OpenDayLight软件，其作为一款java语言编写的开源控制器，支持多种设备和不同的平台，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物理环境下无法使用，而SDN交换机与仿真环境也大为不同，因此，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，最终实现了发布/订阅与SDN控制器功能的结合。

1. **SDN交换机**

在SDN架构图中，列出的底层SDN设备指的就是SDN交换机，也称为OpenFlow交换机。从功能上来说，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，同时能够自我收集端口流量信息，这就是所谓的“可编程网络”。

1. **流表**

流表，基本的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。与传统路由层的路由表相比，流表中的各个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一个重要体现。除此之外，流表还能够整合多个网络层次的配置信息，提供更加丰富全面的匹配规则。流表是由流表项组成的，流表项是数据包匹配的最小单位，其组成结构如图2-2所示。

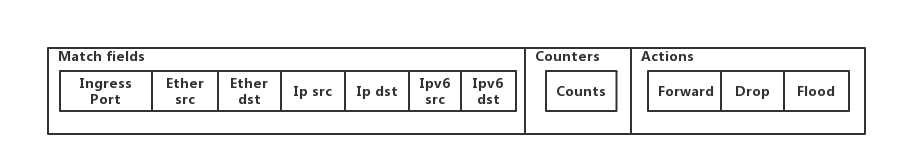


图2-2 流表项结构图

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 | 域作用 | 字段名 | 功能描述 |
| 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 |

流表由多种多样的流表项组成，正是由于其丰富的匹配项和动作，使得SDN交换机能够提供传统路由表基本功能之外的操作，因而SDN网络的使用变得更加灵活、高效。

1. **基于SDN的发布/订阅系统**

发布/订阅系统是一种基于发布/订阅模式的分布式消息传输系统，消息作为系统通信的基本单位，在系统传输过程中担任着信使的角色。每个消息都含有标记位，这是对消息类别的分类，我们称之为主题（topic），发布、订阅同一主题的用户才可以进行交互，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，中间的转发过程由发布/订阅代理节点（broker）提供。多个订阅者可以同时订阅同一个主题，而一旦主题发生变化，所有的订阅者都将收到系统通知。发布/订阅的消息架构如图2-3所示。

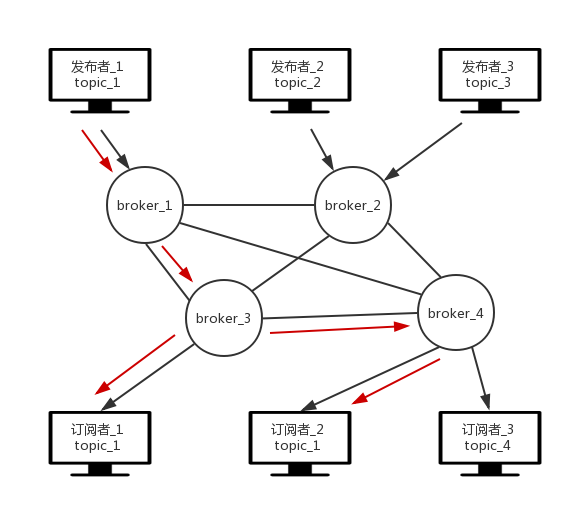


图2-3 发布/订阅消息架构图

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，这样当发布者\_1发布相应的主题消息时，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，相应的转发路径为图中红色箭头部分。

经过分析，SDN架构设计、发布订阅架构组成都是三段式结构，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，是将发布/订阅系统部署在SDN网络环境中，利用SDN流表匹配机制实现消息的转发，同时结合SDN控制器的特点，完成发布/订阅集中管理的功能。简单来说，SDN网络为发布/订阅提供了传输支持，而发布/订阅系统是在SDN网络上的具体应用，两者间的关系如图2-4所示。

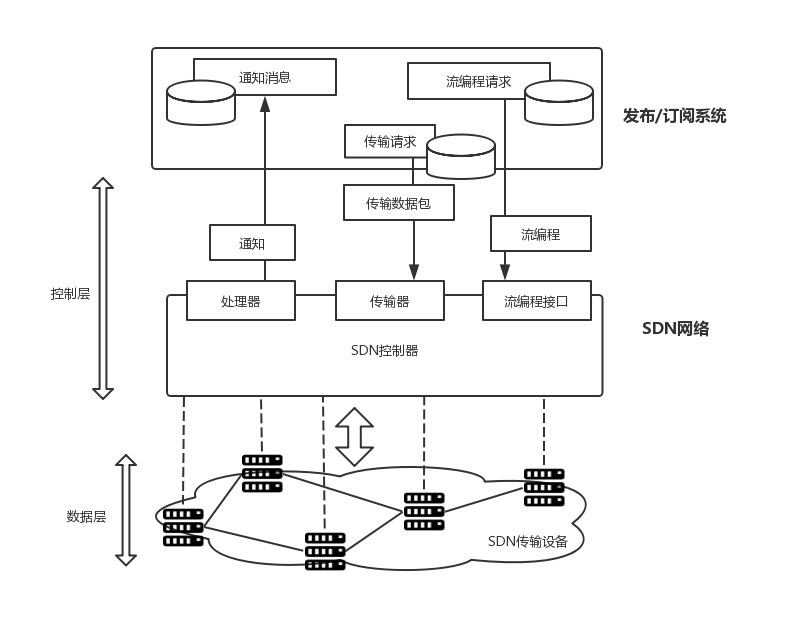


图2-4 发布/订阅和SDN关系图

发布/订阅系统作为一款消息中间件，具体众多优点。例如，使用该系统后，网络中的参与者能够以低耦合的方式实现消息交互，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，发布者发布消息时只需要向特定的发布地址传输即可，这对于大规模分布式环境有着显著的性能提升效果。此外，发布订阅系统还具有可扩展性好、支持多点订阅、匿名性好等特点，能够充分应对复杂的业务环境。

发布/订阅系统和SDN网络的结合，在充分发挥SDN性能的同时，采纳发布/订阅的消息解耦机制，针对不同的用户提供个性化QoS保障，具有非常广阔的发展前景。

1. **web service**

Web Service是一种跨编程语言和跨操作系统平台的远程调用技术，Web Service通过HTTP协议发送请求和接收结果时，发送的请求内容和结果内容都采用XML格式封装，并增加了一些特定的HTTP消息头部，以说明 HTTP消息的内容格式，这些特定的HTTP消息头和XML内容格式就是SOAP协议。SOAP提供了标准的RPC方法来调用Web Service，简单来说：SOAP协议 = HTTP协议 + XML数据格式,系统在使用http发送之前需要进行协议的封装，在收到消息后也需要进行相应的解析。

1. **OSPF协议**

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，作用为使得网络间路由器互相发现、交换信息并最终生成路由表。它是一个内部网关协议(Interior Gateway Protocol，简称IGP），用于在单一自治系统（autonomous system,AS）内决策路由。基本的交互流程如图2-5所示。

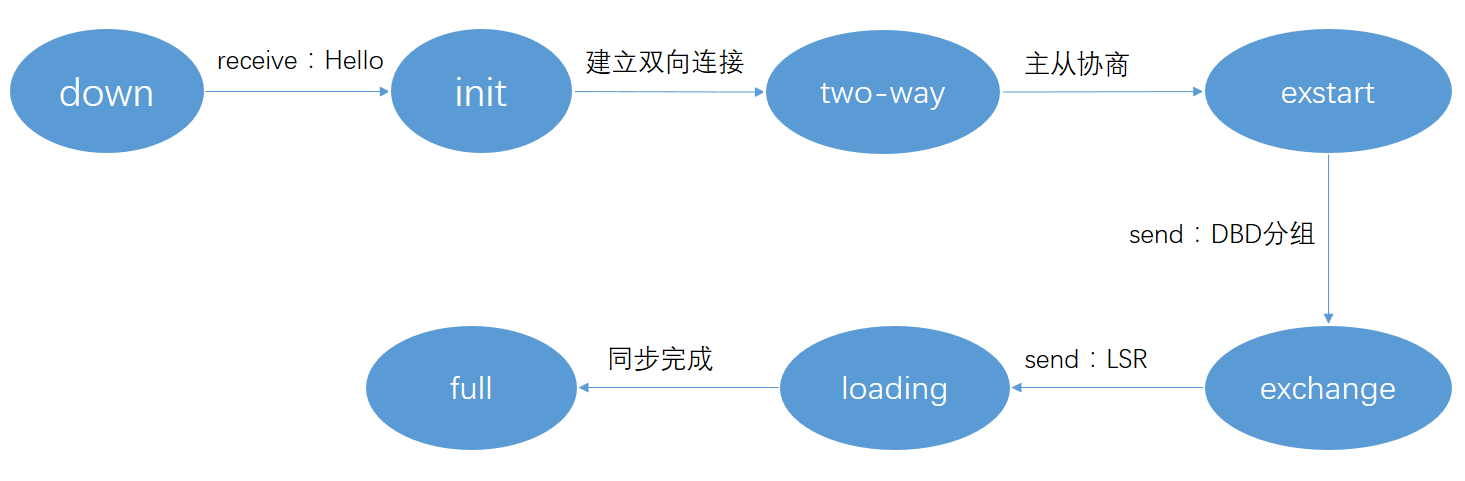


图2-5 OSPF交互流程图

图中列出了OSPF交互流程中可能存在的几种状态，包括：Down：新建状态，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init状态； Init：初始化状态，新建路由收到网络中的Hello消息，此前发送的Hello消息被接收，邻居路由将新建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，然后进入two-way状态；Two-way：双向连接状态，新建路由器收到回复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exstart状态；Exstart：信息交换初始状态，在这个状态下，相邻路由器之间建立Master/Slave关系，然后双方交换DBD分组，进入exchange状态；Exchange：信息交换状态，本地路由器和邻居交换一个或多个DBD分组（也叫DDP），DBD包含有关LSDB中LSA条目的摘要信息，然后交换机通过LSR请求，进入loading状态；Loading：信息加载状态，收到DBD后，将收到的信息同LSDB中的信息进行比较。如果DBD中有更新的链路状态条目，则向对方发送一个LSR，用于请求新的LSA，然后更新完成后进入full状态；Full：完全邻接状态，邻接的链路状态数据库同步完成。

在本系统中，OSPF协议应用于拓扑探测阶段，基本的操作流程为：发Hello报文——建立邻接关系——形成链路状态数据库——SPF算法——形成路由表。通过OSPF拓扑探测，最终各个集群控制器维护了全网链路连接状态。

1. **RTP协议**

RTP协议是由IETF开发的实时传输协议，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；RTP定义了两种报文：RTP报文和RTCP报文，RTP报文用于传送媒体数据（如音频和视频），RTCP协议用于传送控制信息，以实现协议控制功能，这是一种基于接受者反馈的网络传输QoS监测机制，在RTCP的接收报告中包含了当前网络传输中，与QoS有关信息，如报文丢失率、平均时延等，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，提供了多队列传输、路由计算等功能，可以在不同队列中分别传输RTCP、RTP信息，同时根据RTCP中得到的反馈，重新计算当前网络环境中发布/订阅者间相应的路径，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质量。RTP包格式如图2-6所示。

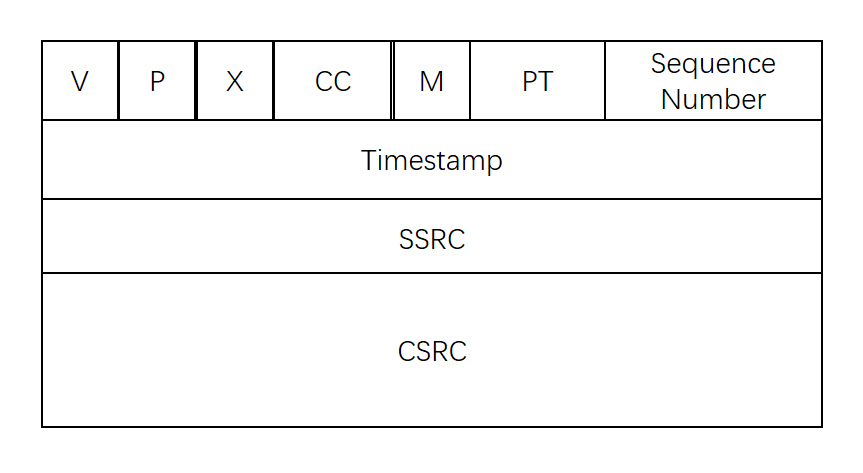


图2-6 RTP数据包格式图

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 |

从表中可以看到RTP数据包的基本格式，在发布/订阅系统中，结合发布、订阅原语，将发布者—订阅者间的会话建立流程如图2-7所示。

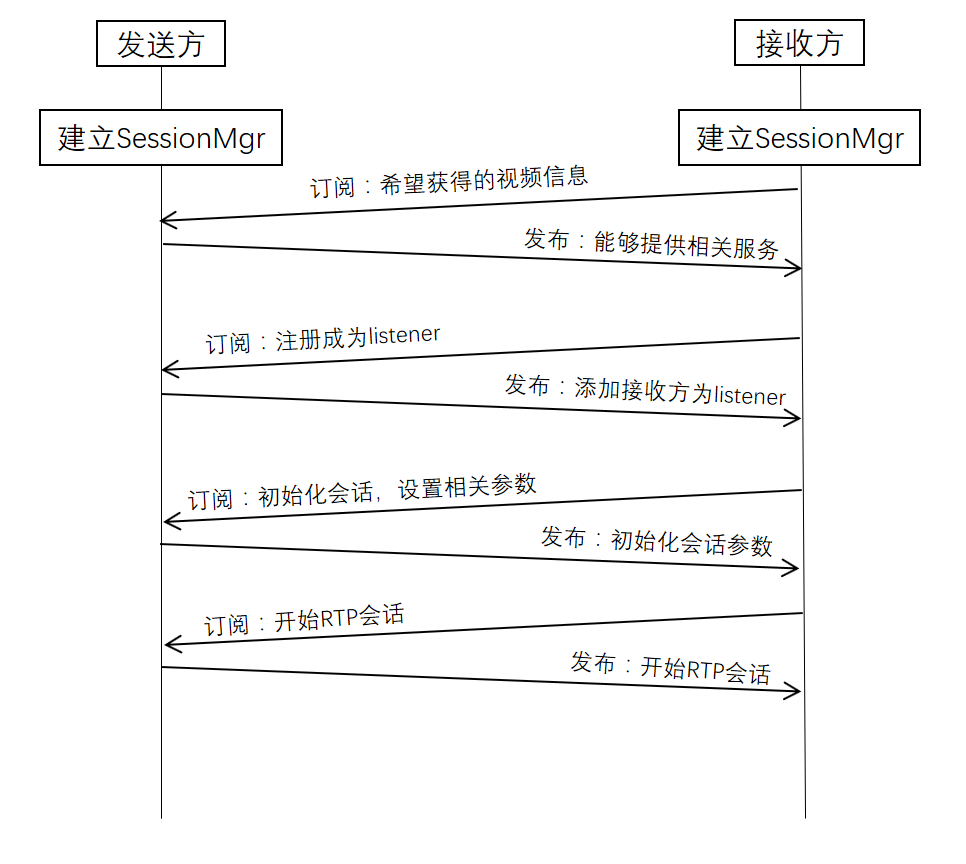


图2-7 发布/订阅系统中RTP协议交互图

图中接收方首先订阅希望获得的视频信息，发送方通过发布订阅系统获知该订阅，且自身满足服务需求，则发布方发布出能够提供相关服务的信息；接收方需要注册在发布者的SessionMgr上成为listener，订阅相应事件，发送方获知订阅后添加成功，并发布该事件；接收方需要协商视频传输的大小、格式、编码、事件戳等具体协议信息，发送方发布初始化的会话参数；发送方确认相关参数后，同意开始RTP会话，发送方发布开始会话事件。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，而系统会将主题编码，作为流表的匹配项下发至交换机，这样RTP数据包就可以在网络中直接传输。

1. **Reactor模式**

Reactor设计模式是事件驱动编程的一种实现方式，用于处理多个客户端同时向服务端请求服务的场景。每种服务在服务端可能由多个方法组成，reactor会解耦并发请求的服务并分发给对应的事件处理器来处理，适用于高并发、大数据量的情况。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。Reactor模式的处理机制如图2-8所示。

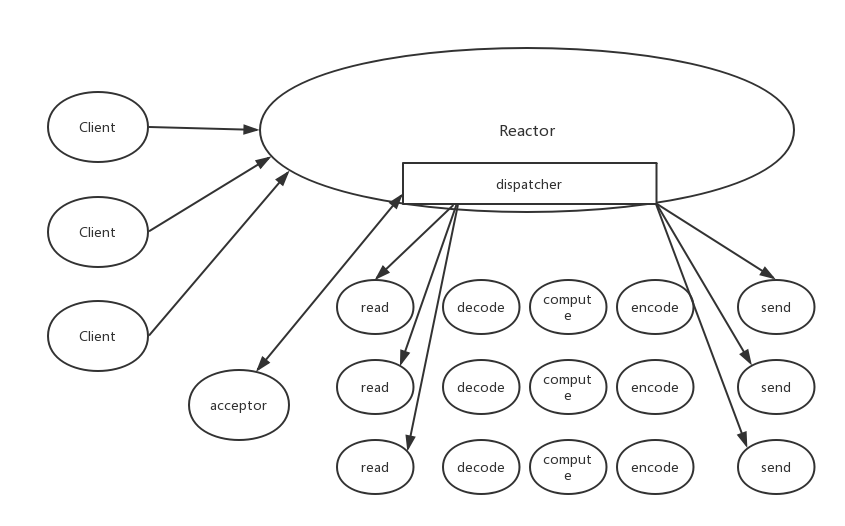


图2-8 Reactor模式处理机制

它的核心思想是当消息到来时，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。经过测试，在发布/订阅系统中，Reactor模式极大地提升了系统的处理性能。

1. **本章总结**

本章主要对本文涉及的相关背景知识进行了简单的介绍，包括软件定义网络技术、发布/订阅系统、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，基于SDN的发布订阅系统架构、多媒体传输是本系统的实现目标，QoS性能保障是本系统的优化目标，详细内容将在后续章节中说明。

2. **需求分析**

已有的系统是建立在虚拟环境、开源控制器下的发布/订阅版本，而本系统设立之初，需要达到的目的是构建一套基于SDN物理交换机上的发布/订阅系统，因此，基本的需求包含两方面：SDN控制器需求和发布/订阅需求。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，达到SDN“控制-转发”的目的，该需求包括拓扑管理、路由计算、流表管理；同时系统实现发布/订阅功能，能够在用户间构建关联，满足用户的发布、订阅请求，以及时延需求等，具体包括主题管理、消息接口、多媒体传输、用户协商机制。系统整体用例如图3-1所示。

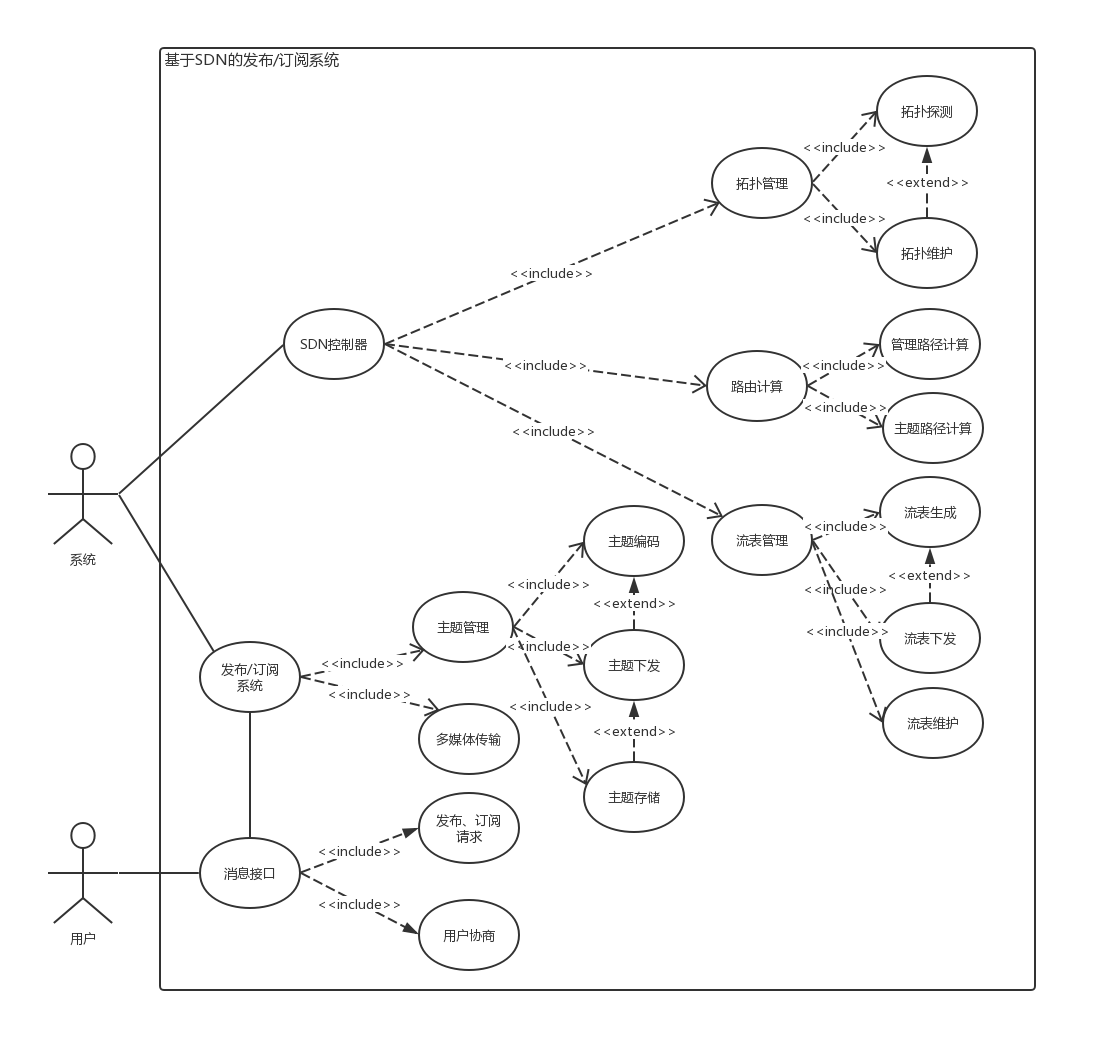


图3-1 系统整体用例图

从图中可以看到，系统的需求是由系统的功能组成的，分为SDN控制器和发布/订阅系统这两个部分，用户通过消息接口调用系统的发布/订阅功能，完成相应的请求。下面进行具体阐述。

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发现、对发布/订阅消息调用路由算法计算出转发路径、获取交换机运行状态等，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，这里讨论的是集群间拓扑的探测和维护，即通过拓扑管理，集群控制器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。

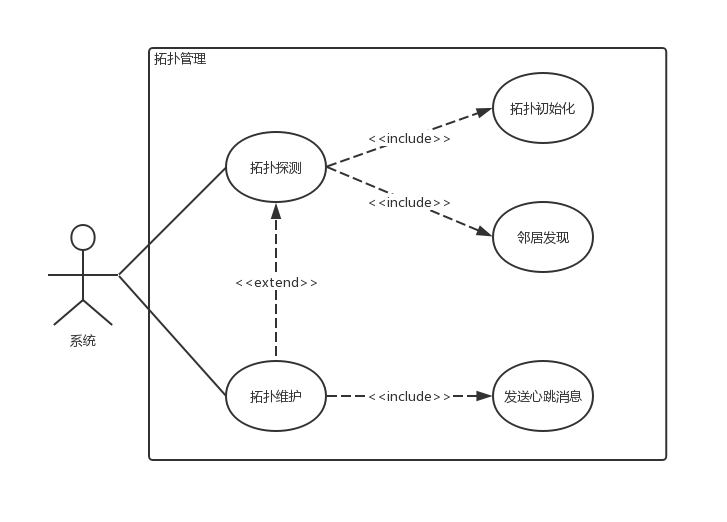


图3-2 拓扑管理用例图

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. 若心跳消息超时或未回复，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，这相当于知道了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，系统提供了管理路径和主题路径的计算。路由计算的用例图如下所示。

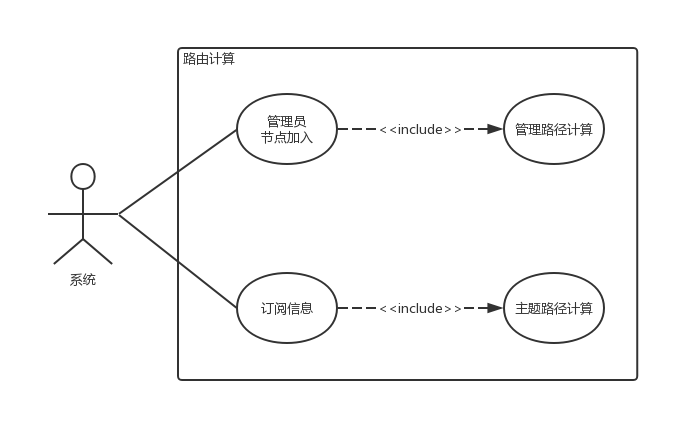


图3-3 路由计算用例图

管理路径计算，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，本系统运行环境为物理SDN网络，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。该模块应该完成包括流表生成、流表下发、流表维护的功能。流表管理的用例图如下所示。

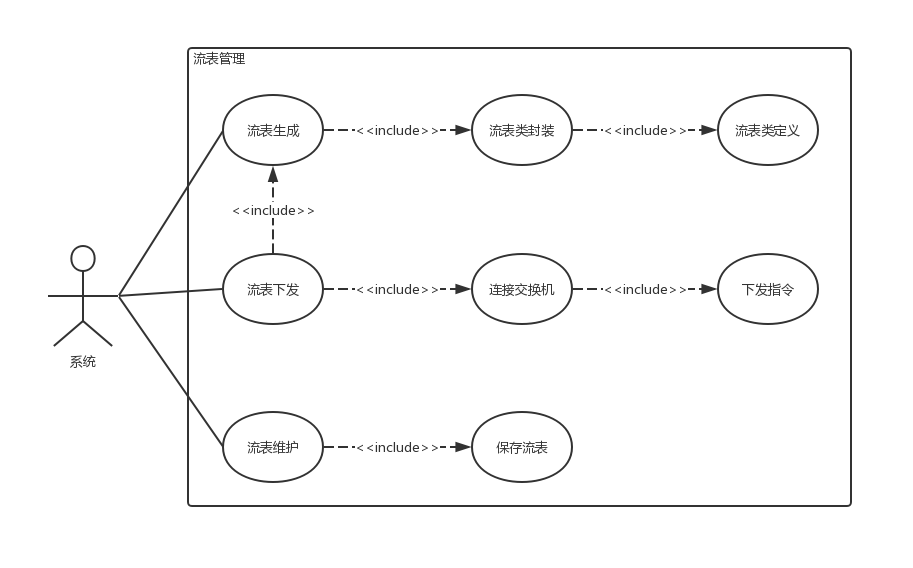


图3-4 流表管理用例图

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，需要系统能够控制SDN交换机，将生成的流表指令下发，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/订阅需求**

根据本文的研究内容，发布/订阅需求包括主题管理、消息接口、多媒体传输和用户协商，这些都是发布/订阅系统必不可少的功能。用户调用消息接口进行发布、订阅注册，不同用户间通过主题管理联系在一起，发布的消息经过SDN网络进行传输，最终达到多媒体传输和用户协商的目的。下面分别进行叙述。

1. **主题管理需求**

主题是发布/订阅系统连接的纽带，不同用户间通过相同的主题进行匹配，若主题不同，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。

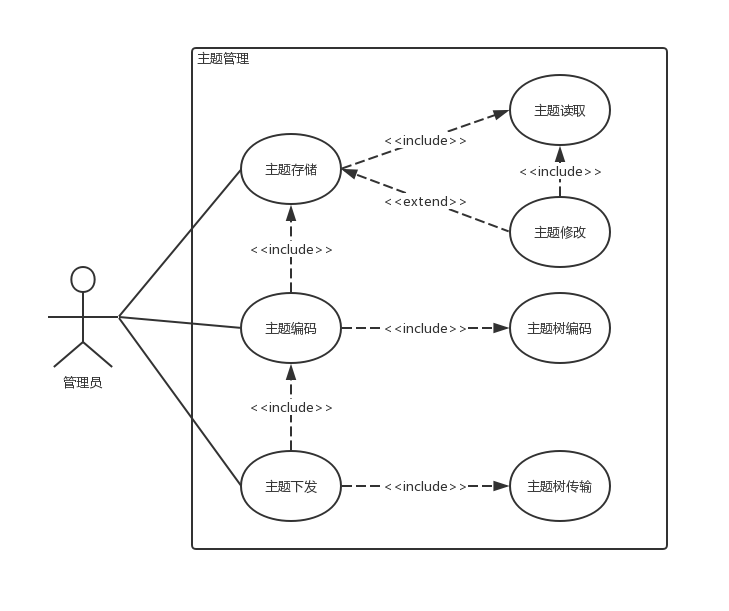


图3-5 主题管理用例图

主题树存储，管理员可以获取主题树存储信息，如果对主题树进行修改，就将修改后的主题树内容写回到持久化文件中，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内容读取到内存中 2. 管理员对主题树进行修改 3. 系统将修改后的主题树内容写回到持久化文件中，便于再次读取 |
| 后置条件 | 修改后的主题树写回到持久化文件中，管理员能够再次获取 |

主题树编码，通过相应的编码方式，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 | 系统将主题树内容读取到内存中 |
| 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，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信息 |

* + 1. **消息接口需求**

本系统为发布者、订阅者提供的服务调用接口，是基于SOAP协议的web service方法，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。

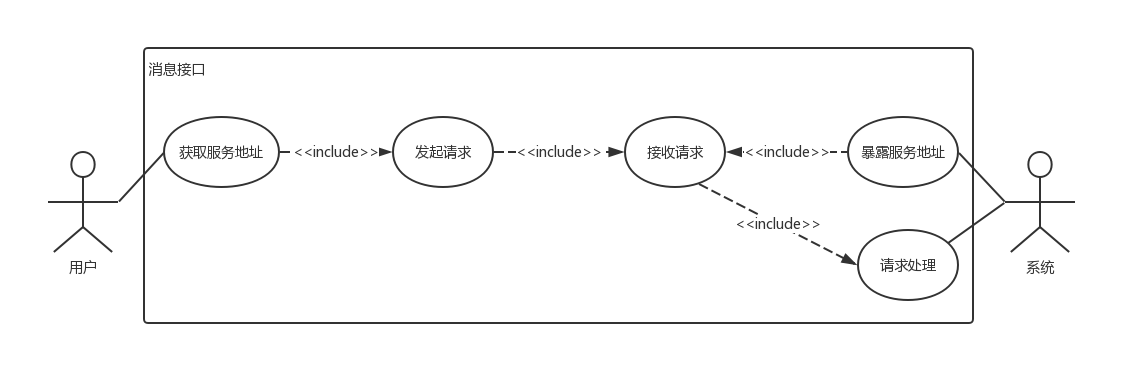


图3-6 消息接口用例图

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. **多媒体传输需求**

本系统的创新点之一就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即可。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. 发布者调用系统进行传输 4. 订阅者获得视频流数据 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. **用户协商机制**

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，由管理员计算完成后下发处理策略，若管理员经过计算后可以满足，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该请求无法满足，用户可以决定接受当前结果或重新发起请求。具体用例图如下所示。

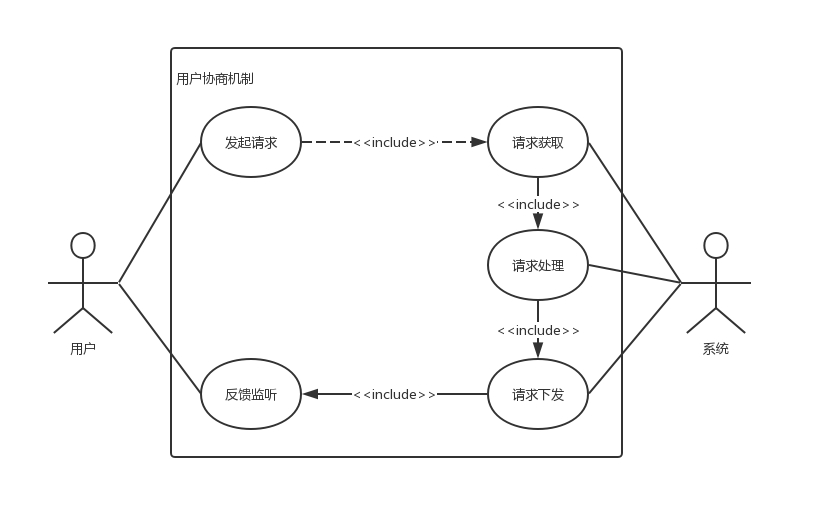


图3-8 用户协商用机制例图

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**

本章主要对本系统涉及的相关需求进行描述，分为SDN控制器需求和发布/订阅系统需求。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、多媒体传输需求、用户协商机制等。关于各个需求的解决方案将在第四章系统概要设计中给出。

2. **系统概要设计**

第三章详述了实现本系统所需完成的需求和目标，本章将对系统各模块的设计思路进行详细阐述，说明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、各模块划分及作用有一定了解。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，而在已有系统中，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。

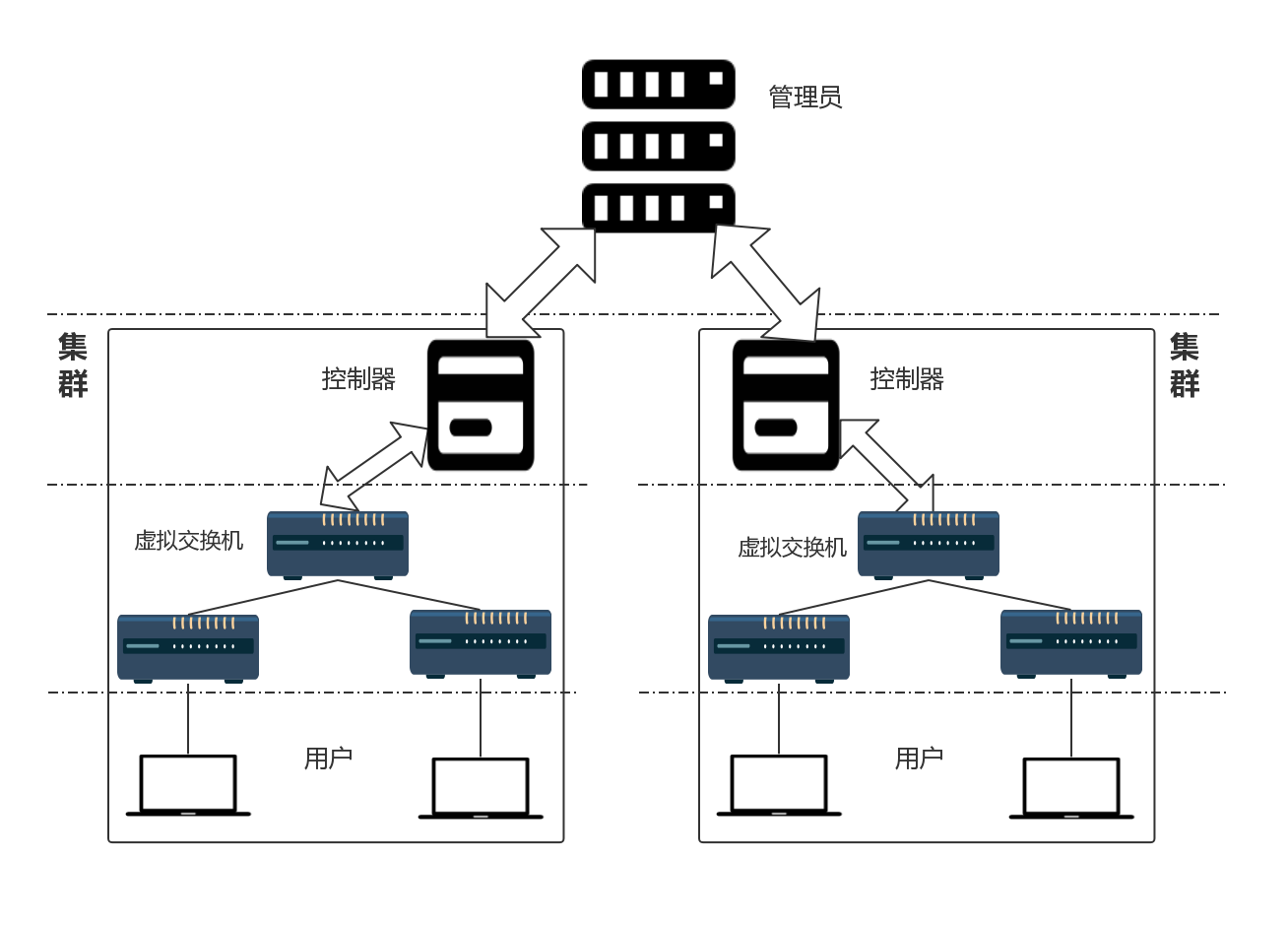


图4-1 已有系统架构图

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ODL虚拟软件获得，同时控制器也需要在物理拓扑中获取邻居的连接信息，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。

结合上述问题，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QoS保障方案等，改进后的系统架构如下图所示。

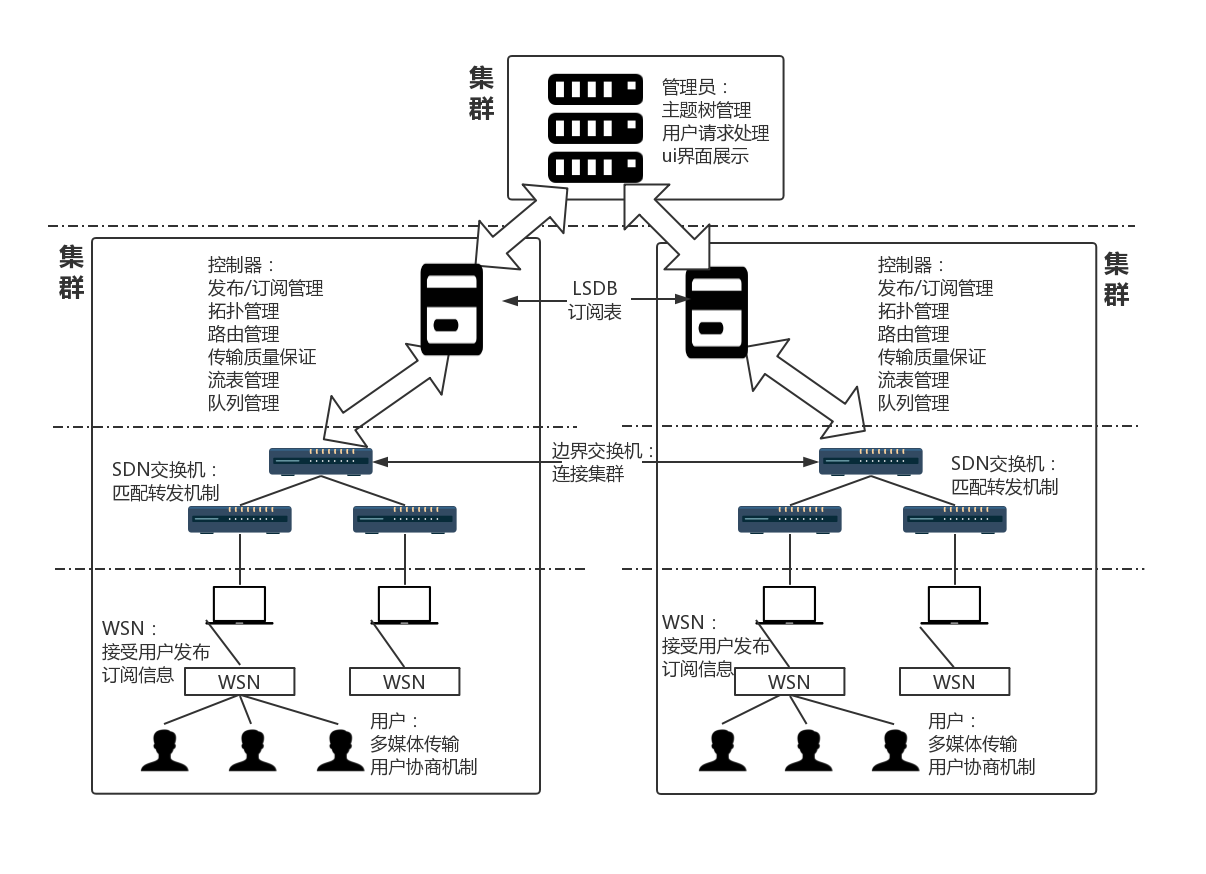
****

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，由于是在物理环境中，因此还需要具备拓扑管理、路由管理、流表管理、队列管理的功能；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，以及主题树的维护与下发；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，负责用户与控制器间的消息校验和主题监听，这是为了缓解控制器运行时压力，将消息的预处理、过滤、主题消息的监听等环节下放到用户层面，这样在不影响系统功能的同时提升系统性能。一台电脑上只能存在一个WSN层，用于接收用户请求、主题消息的监听，但是可以存在多个注册用户，这在另一个层面上加强了系统的拓展性。

在系统架构中，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. **集群控制器模块设计**

集群控制器是我们开发的java程序，运行于主机之上，集群由SDN交换机、集群控制器及其所在主机、用户及其所在主机构成，彼此之间通过网线连接，不同的集群之间通过交换机连接，如果一台交换机只用于连接本集群的主机和交换机，那么就称为群内交换机，而连接了不同集群的交换机称为边界交换机。一个集群间的连接示意图如下所示。

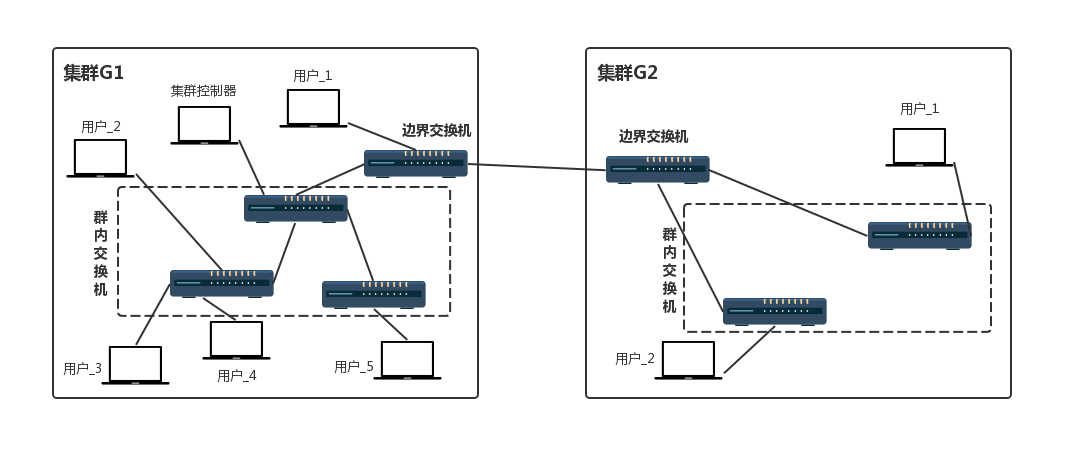


图4-3 集群连接示意图

集群控制器负责集群内的发布/订阅处理，分为发布/订阅管理、拓扑管理、路由管理、流表管理、队列管理这几个子模块。其中发布/订阅管理、拓扑管理、路由管理共同完成了发布/订阅系统的基本功能，用户的订阅请求保存在本地发布/订阅表，并与其他集群的发布/订阅信息构成全局发布/订阅表，拓扑管理模块使得集群控制器能够感知邻居集群的存在，并保存邻居集群的LSA，进而获得集群分布、连接情况，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网络的“软件定义网络”功能，同时集群控制器还需要通过队列管理模块收集集群内交换机的运行情况，并上报给管理员。集群控制器中各个子模块的功能图如下所示。

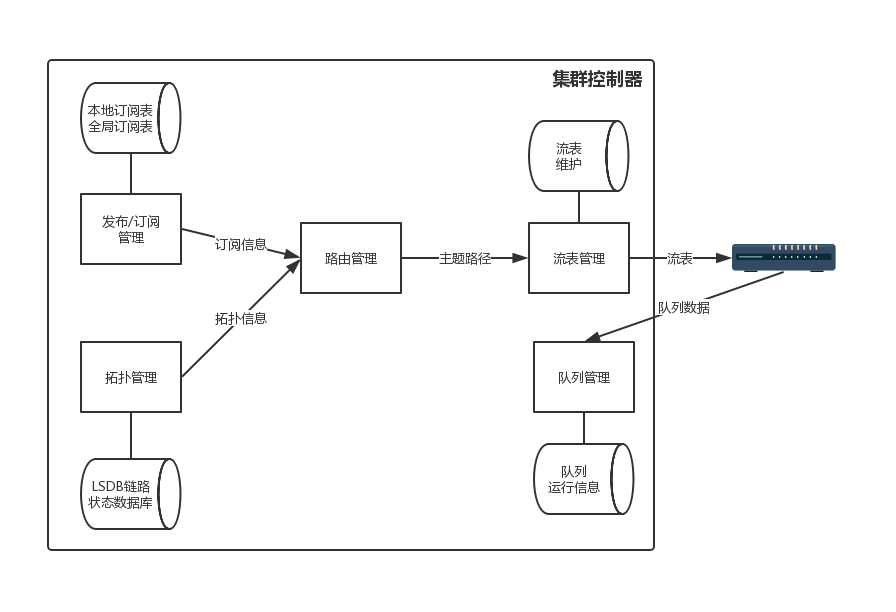


图4-4 集群控制器子模块功能图

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，包括集群内部收集到的用户请求，以及集群间广播的LSA信息（将在拓扑管理中介绍）。集群内的发布/订阅信息是本集群用户通过调用WSN的消息接口，发起的服务请求信息，包括用户名、订阅主题、订阅时间等内容，集群控制器在收到WSN发来的用户请求后，经过分析，如果用户是发布、订阅请求，则将该主题保存在本地缓存中，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，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/订阅信息分别保存，便于主题路径计算时使用。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。

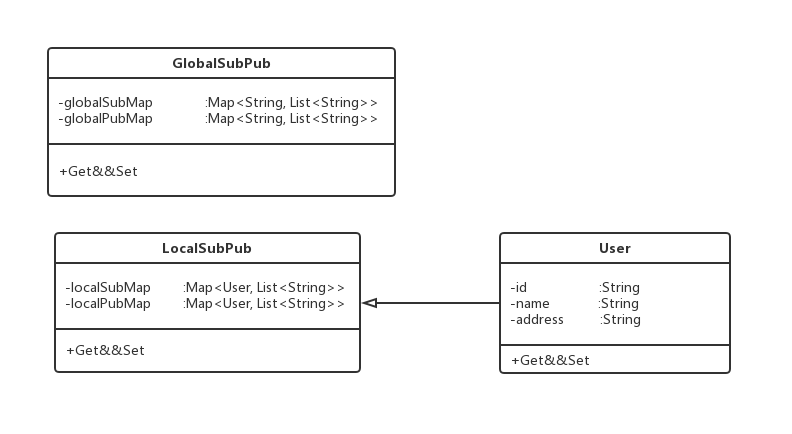


图4-5 发布/订阅表类图

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5-5 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 |

从表中可以看到，使用Map散列表存储发布/订阅信息，这是为了快速查询。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时，系统无法判断当前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，就需要计算该集群的主题路径，因此采用集群名作为索引。

1. **拓扑管理**

拓扑管理是物理网络的连接情况，结合图4-3集群连接示意图，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。

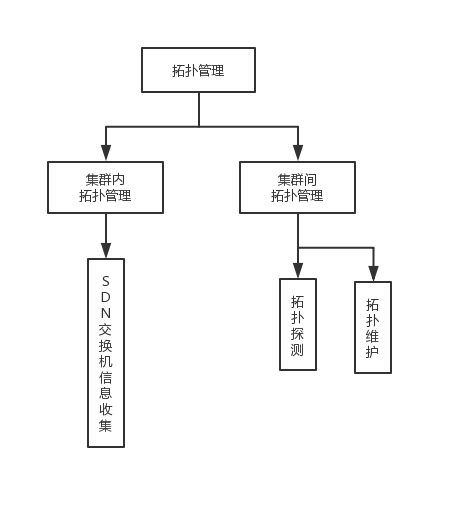


图4-6 拓扑管理功能模块图

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，交换机的连接拓扑会对数据的转发路径产生影响，因此集群内的拓扑信息可以归结为SDN交换机的连接情况，而SDN交换机具有动态感应的功能，当某个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，进一步得到集群内的拓扑连接信息。

集群间拓扑管理负责维护不同集群之间的连接关系，当新集群加入时就必须具备相应的感知功能，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中删除即可。集群间的拓扑管理流程如下所示。

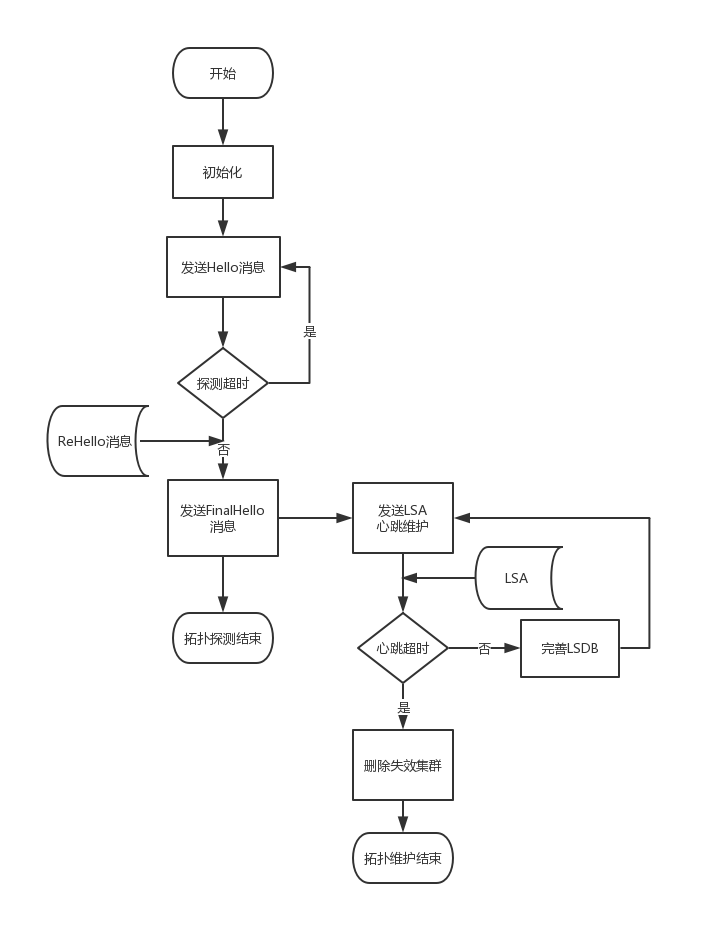


图4-7拓扑管理流程图

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、分布情况。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。

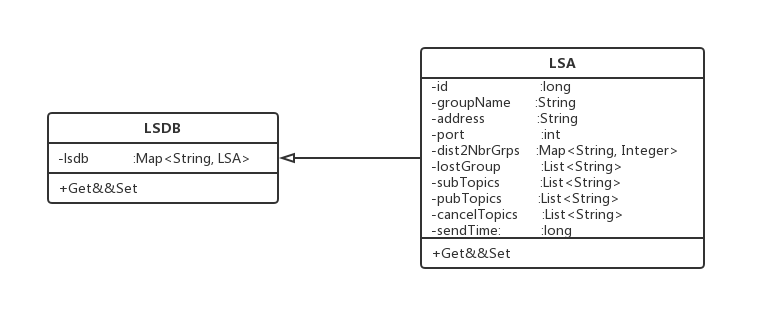


图4-8 链路状态信息类图

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5-5 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 | 本集群名 |
| 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 |

拓扑管理模块使得集群控制器能够通过访问SDN交换机的方式获得集群内拓扑，以及OSPF三次握手的方式获得集群间拓扑，保障了系统的稳定运行，解决了新系统在物理环境下的链路探测和维护问题。

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，由于集群拓扑的动态性与复杂性，消息可能经过一个或多个中间集群，中间集群没有针对该主题的发布/订阅操作，也就不会接收、转发该消息，因此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，各个集群控制器接入到拓扑中后，能够通过管理路径与管理员交互，包括管理员编码主题树的下发和控制器集群运行状态的上报，由于管理员角色的特殊性，全网拓扑中有且只有一个管理员角色，为了减小网络带宽损耗，各个集群控制器之间只需要存在到达管理员的链路，因此管理路径中不存在环形结构，是一棵以管理员为根节点的树形结构。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路径的最短距离，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，使用到的管理员节点、所有节点、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。

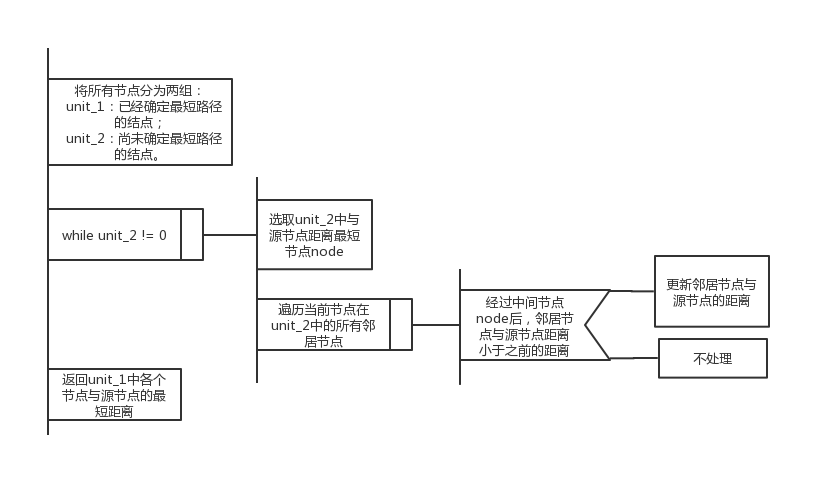


图4-9 Dijkstra算法PAD图

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、各个集群的订阅信息，由于针对特定主题，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。

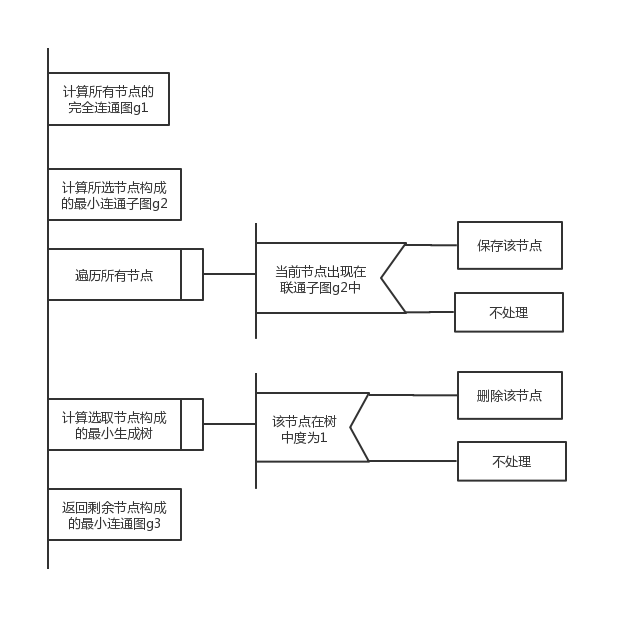


图4-10 Steiner算法PAD图

1. **传输质量保证**

一套完善的系统具有功能和性能上的需求，路由模块计算出了相应的转发路径，但是真实网络环境下，数据在链路转发过程中可能存在阻塞、丢失等情况，为了保证多媒体数据的稳定传输，系统在数据接收、转发时需要提供传输方案，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，针对数据转发方案采取的是生产者-消费者模式，将订阅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、消费者的处理速度不同步问题。生产者-消费者的运行示意图如下所示。

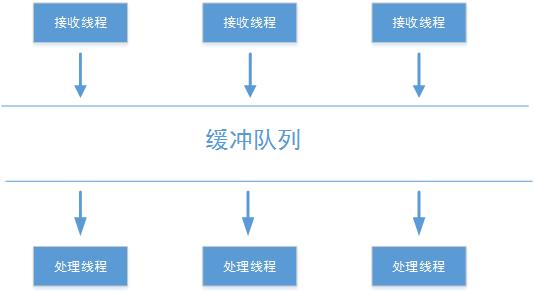


图4-12 生产者/消费者示意图

但是在实际使用过程中，我们发现虽然存在队列，但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。

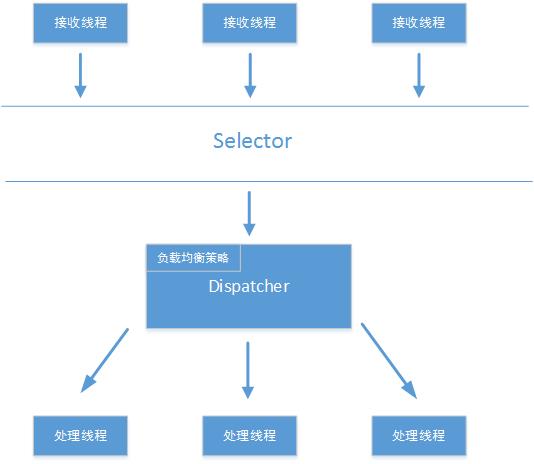


图4-13 Reactor模式示意图

从图中可以看到，消息到来时，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、性能更好，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，实际运行结果也证明了Reactor模式的可靠性。

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，经过路由模块生成的转发路径，需要经过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，因此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。

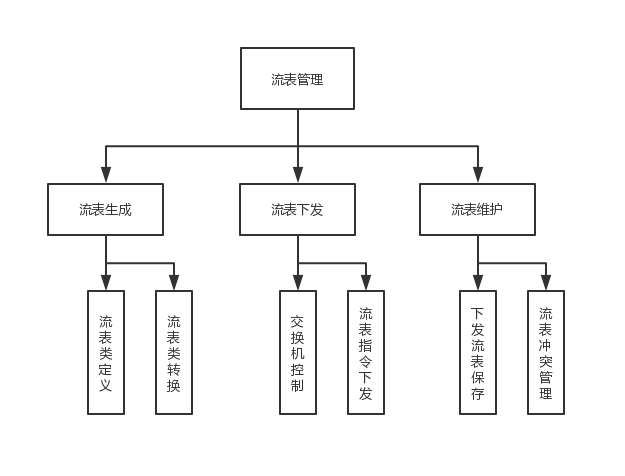


图4-11 流表管理功能模块图

流表管理首先需要对流表类有一个定义，系统中的流表的定义如图4-13中Flow类所示，可以看到，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。

经过转换，我们得到了内存中的流表，管理模块还需要将该流表下发到交换机处，实现真正的匹配转发机制。SDN交换机内部有核心处理器，保存了交换机硬件匹配的所有指令，因此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，实现控制交换机的功能。

下发的流表在交换机中执行匹配转发功能，但是经过测试，我们发现SDN交换机中，当流表项的转发地址、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，流表冲突现象如下图所示。

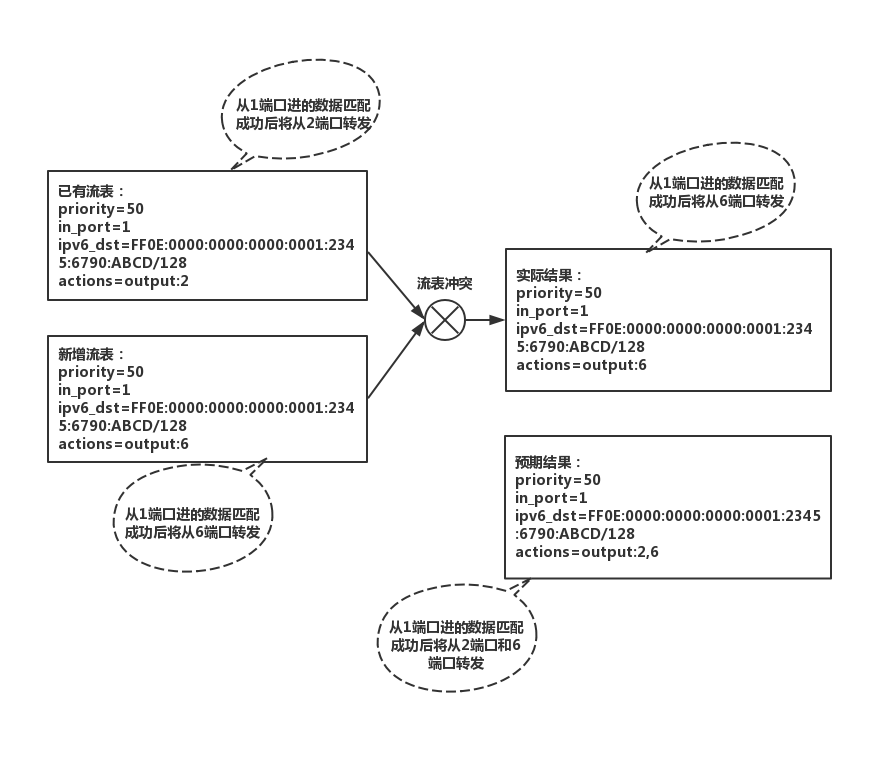


图4-12 流表冲突示意图

内存中需要对下发流表进行维护，系统使用流表类Flow定义流表项，使用AllFlows保存下发的所有流表，相关数据结构如下所示。

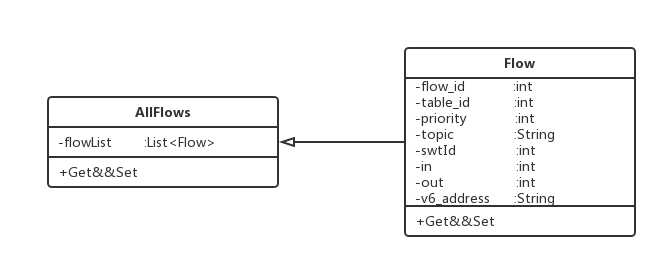


图4-13 流表类图

图中展示了流表类的消息格式，下面介绍各字段含义。

表5-5 流表类定义

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |

由于流表在交换机中会从优先级高的流表项开始执行，匹配成功后直接转发，不再往下查询，因此内存中定义AllFlow时使用了链表的数据结构，这样程序中使用顺序遍历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，将新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. **管理员模块设计**

管理员模块是我们开发的程序，基本功能和集群控制器相似，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，通过管理员模块，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。

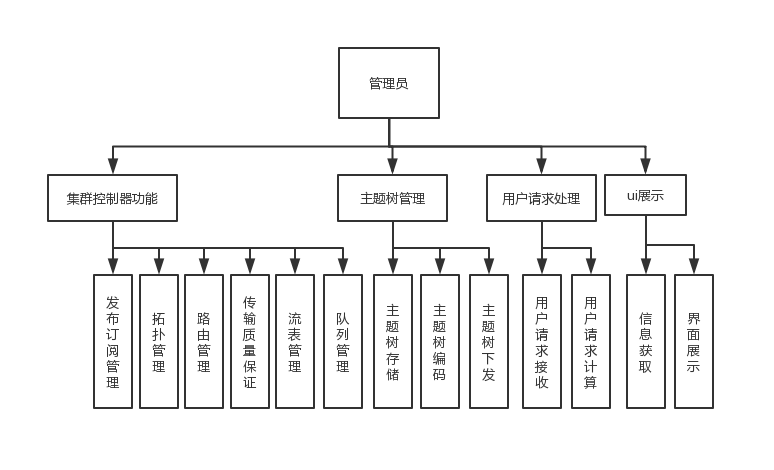


图4-15 管理员功能模块图

1. **主题树管理**

发布/订阅系统依靠主题进行不同发布者、订阅者之间的关联，主题的定义多种多样，不同主题间存在某种联系，比如用户订阅的“球类运动”主题与“足球”主题间就存在父子包含关系，我们将主题定义为：一个主题可以拥有一个或多个子主题，同时也可以存在父主题，若订阅了父主题则默认获得所有子主题的订阅内容。可以看到，主题的定义与树形结构相似。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，首先会判断该主题是否存在，如果主题不存在则无法订阅和发布。主题树是任意多叉树，只有父子节点之间有直接关系，兄弟节点之间没有任何关联。主题树的管理包括主题树的存储、编码和下发。

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。主题树信息将通过管理路径下发给各个集群控制器，关于管理路径的计算在4.2.3路由管理模块中已经进行了介绍，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。

系统中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，我们需要保证组播地址的唯一性，即在组播地址与主题间存在一一映射的关系，控制器能够将用户订阅的主题编码成组播地址，并监听该地址等待消息的到来。主题管理的流程图如下所示。

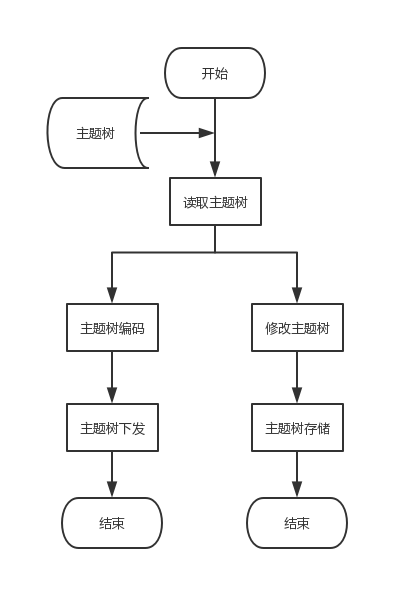
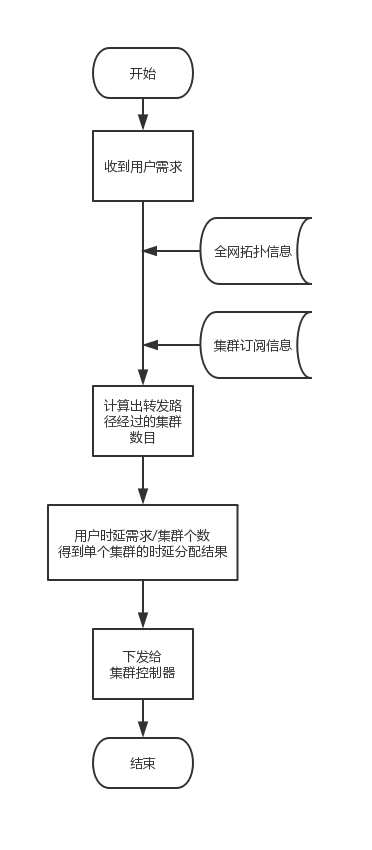


图4-16主题树管理流程图

1. **用户请求处理**

该模块的功能是收集用户的请求信息，计算出合理的带宽分配方案，并反馈给用户，关于用户协商机制的通信过程，将在4.5.2小节中进行介绍，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，计算出一个解决方案。

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，因此在处理时，首先需要计算出该主题的转发路径，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，这样就可以将用户的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带宽分配结果。用户请求处理流程如下所示。



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，使得管理员对集群运行状态有直观的感受，同时提供页面操作接口，便于管理者下发相应的配置信息。通过集群控制器模块的设计，我们知道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。

ui界面的详细实现将在第五章中进行介绍，这里只是简单说明该模块需要展示的内容。

1. **WSN模块设计**

WSN模块是中间层，用于连接用户和集群控制器，用户发来的请求由WSN接收，并进行基本的内容检验，合法的请求将由WSN发送给集群控制器，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，否则通知用户订阅失败；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，请求合理则向集群控制器上报用户需求。WSN运行流程图如下所示。

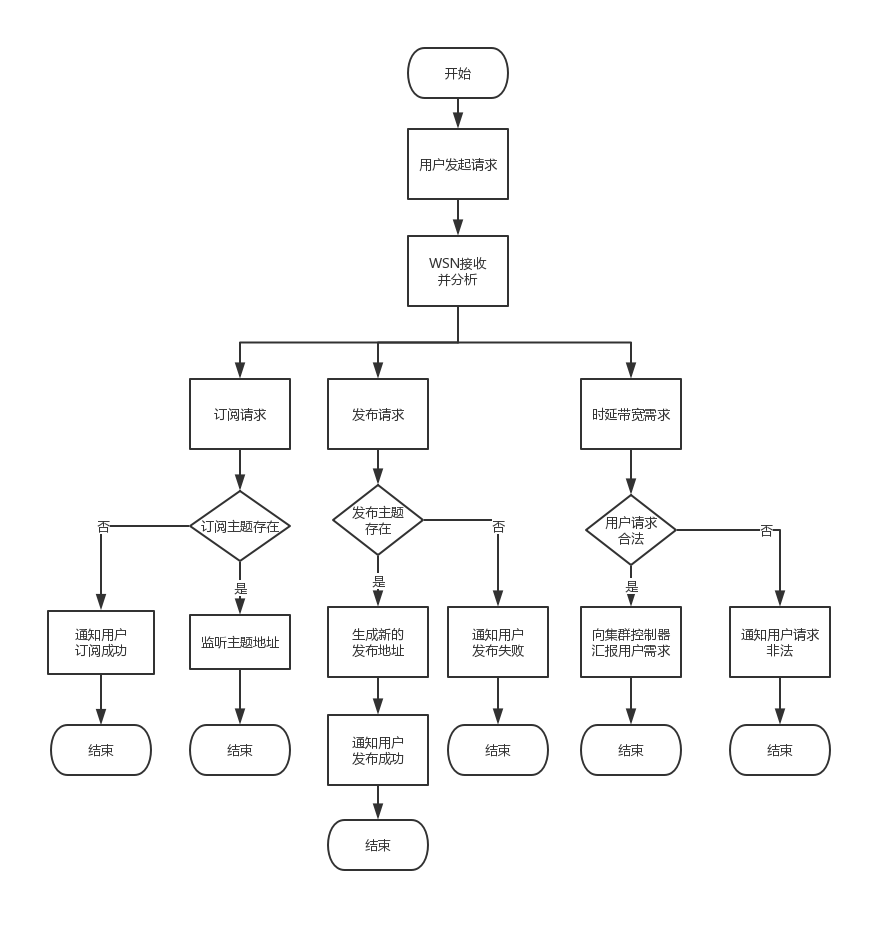


图4-18 WSN流程图

1. **用户模块设计**

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。

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，对视频数据进行封装，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。

系统先发送RTCP控制信息，双方协商协议版本、视频格式等内容，然后再通过RTP数据包传输视频数据。RTP数据包定义了充足的字段来维护消息的稳定传输，

1. **用户协商机制**

系统提供的传输质量保证模块，能够在传输过程中提供数据的稳定传输，但是在具体应用场景下，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，用户协商机制和传输质量保证共同组成了本系统的质量保证方案。

本系统提出的用户协商机制更加具有现实意义，即用户通过消息接口，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，这个过程在4.4WSN模块中进行了介绍，管理员收到后结合全网拓扑情况进行处理，相应的计算使用了4.3.2用户请求处理模块，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。

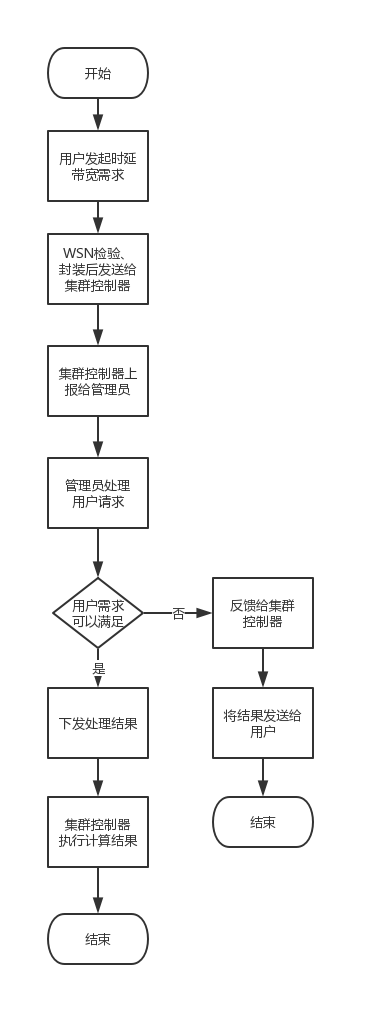


图4-19 用户协商流程图

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，系统根据参与角色的不同划分为集群控制器、管理员、WSN、用户，这里只是简要介绍了各个模块的设计思路和它们在系统中的运行流程，关于它们的具体实现将在第五章进行介绍。

2. **系统详细设计与实现**

第四章介绍了本系统的简要设计，根据各个模块的设计原理和工作流程，我们对系统的整体结构有了一定的了解。本章将进一步介绍基于SDN网络的发布/订阅系统是如何实现的。

1. **消息分类与定义**

在本系统中，由于存在拓扑、主题、发布订阅等多个消息交互模块，因此消息的种类繁多，为了方便管理，本系统在构建之初就详细设计了各种消息的格式，除基本设备信息外，根据消息发送方角色的不同划分为管理消息、系统消息和WSN消息，具体消息分类见下图。

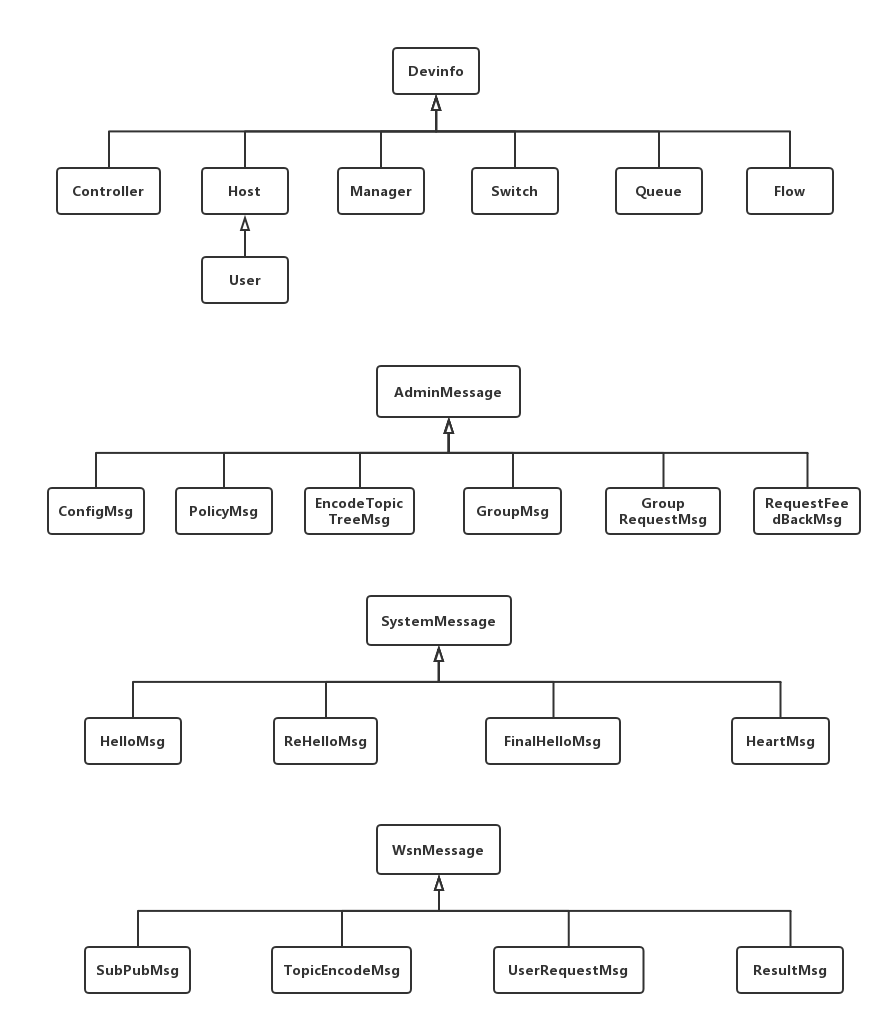


图5-6 消息分类图

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信息 |
| 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，结合上述消息分类，可以给出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。

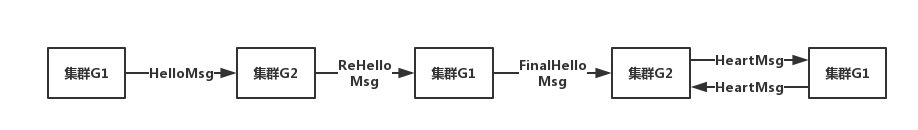


图5-7 拓扑消息交互图

1. **主题树消息**

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

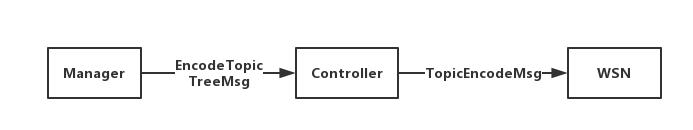


图5-8 主题树消息交互图

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消息发动给控制器，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。

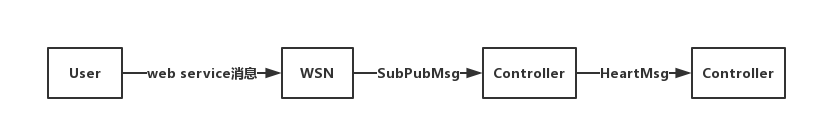


图5-9 发布/订阅消息交互图

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层，最终将处理结果告知用户。消息交互流程见下图。

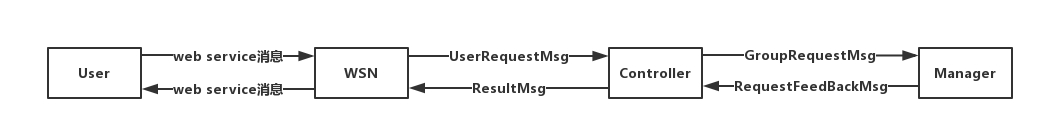


图5-10 用户请求消息交互图

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，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。

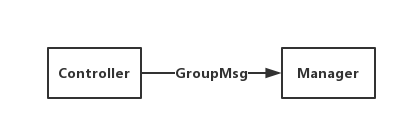


图5-11 集群消息交互图

1. **系统架构实现**

在定义了系统消息的分类，以及主要消息交互流程后，就可以开始对系统进行详细设计，采用划分模块的方式，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。此外本系统还具备了日志管理功能，为系统的可靠性提供保障。由于系统结构总图涉及模块较多，无法以完整形式展现，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，各个模块向外数据线路表示与其他模块有信息交互，具体如下所示。

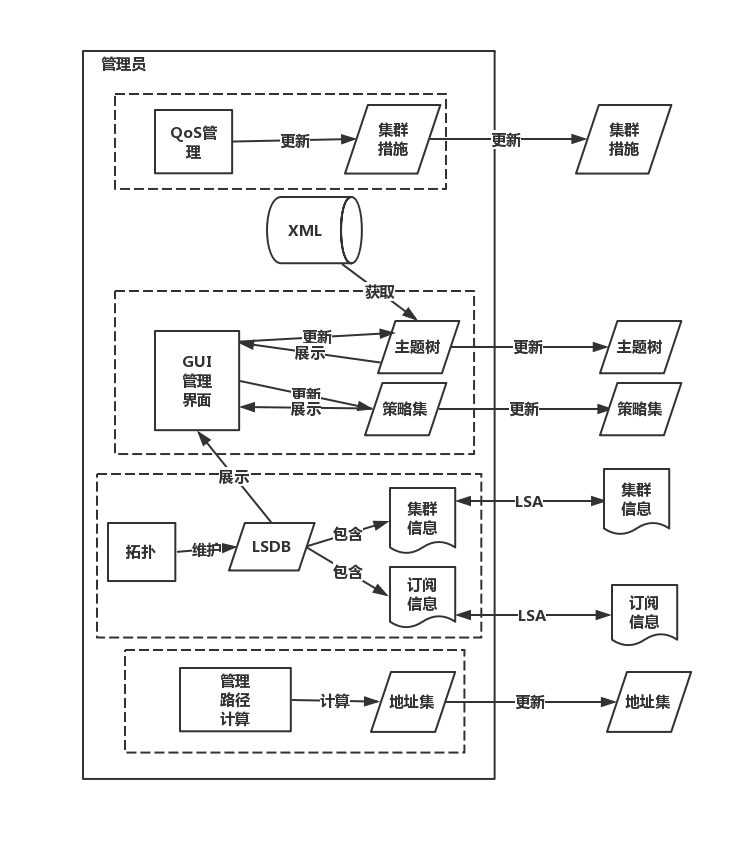


图5-11 系统架构总图（一）

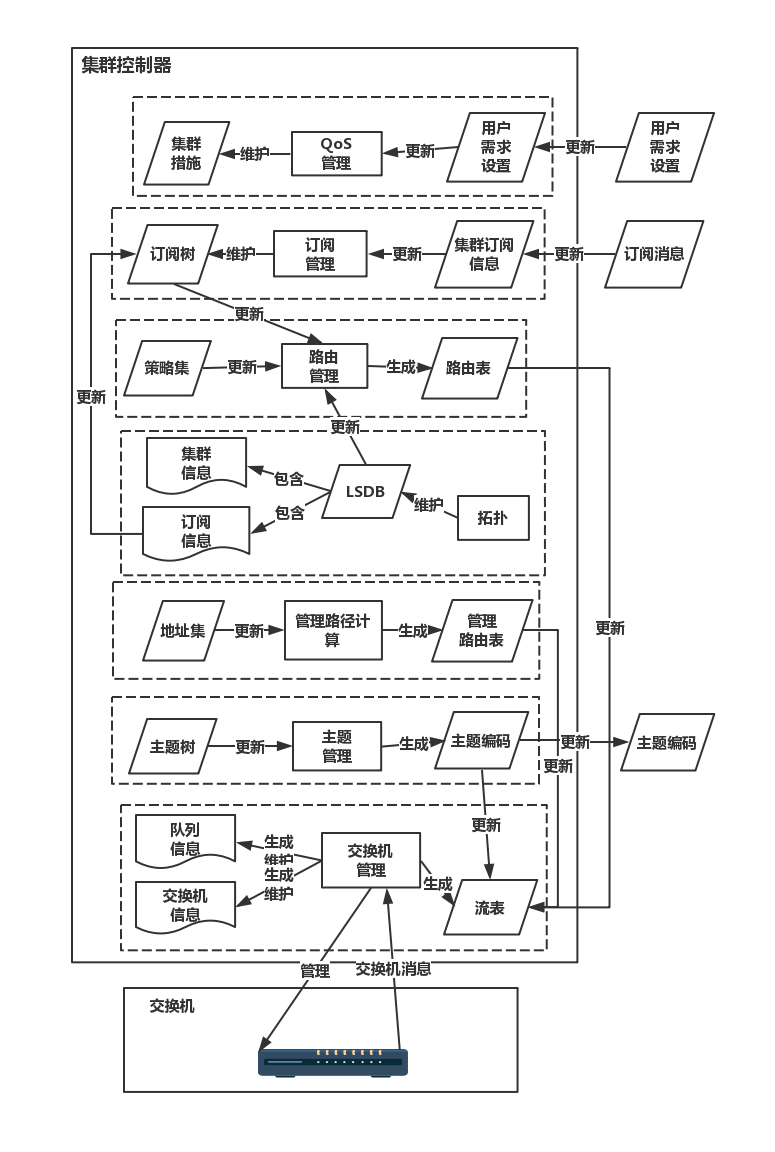


图5-11 系统架构总图（二）

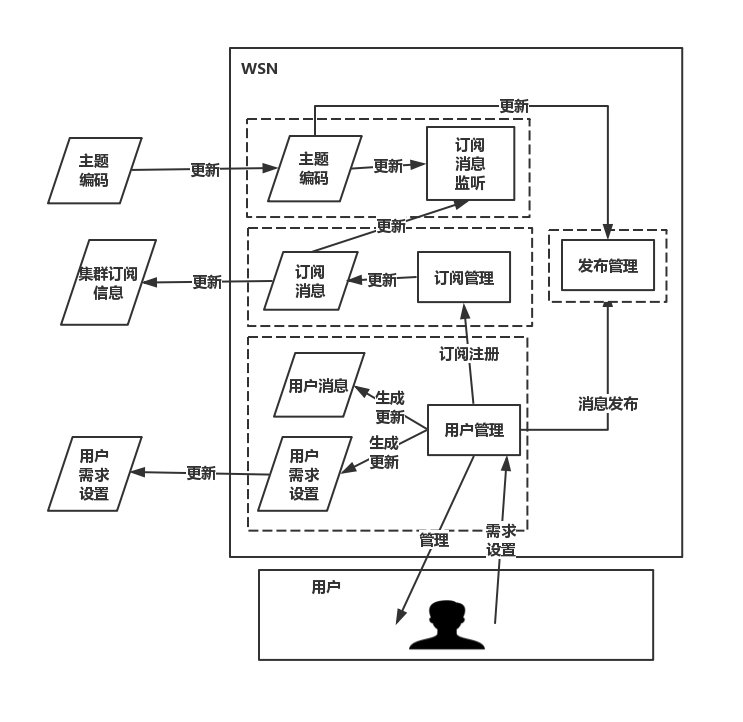


图5-12 系统架构总图（三）

1. **集群控制器模块实现**

集群控制器负责集群内的发布/订阅消息，从系统架构总图（二）中可以看到，该模块。

1. **发布/订阅管理**
2. **拓扑管理**

拓扑管理是系统的基本功能模块，相当于系统的“眼睛”，能够使得系统在启动后动态感知周围的邻居集群，对全网拓扑的构建、后续路由计算都具有重要作用，因此需要设计出完善的功能。在本系统中采用的是OSPF协议，关于该协议的基本概念在上面已经进行了阐述，这里介绍的是系统中具体实现过程，包括拓扑探测、拓扑维护，具体子模块的时序图如下所示。

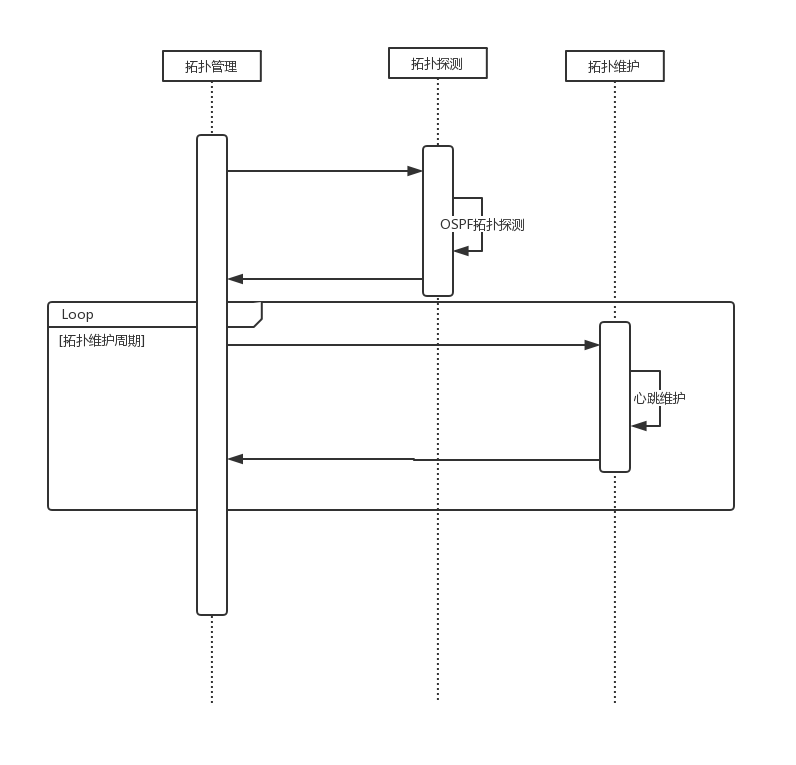


图5-20 拓扑管理时序图

1. 拓扑探测

拓扑探测是控制器动态获取邻居集群信息的过程，通过消息交互的方式提供本集群信息，同时保存邻居集群的运行状态。我们首先给出消息类的定义，在OSPF的拓扑探测过程中，消息包括Hello、ReHello、FinalHello这三种，本质上三个类的属性相似，这里以Hello消息为例，Hello消息类图如下所示。



图5-21 Hello消息类图

图中概括了Hello消息的消息格式，下面介绍各字段含义。

表5-4 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。在探测阶段初始化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，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5-22 拓扑探测流程图

1. 拓扑维护

拓扑探测负责与邻居集群交互，并构建连接关系，但这个连接并非永久性的，不论是邻居集群的宕机，还是网络连接的故障，都会导致邻居集群在逻辑上的失效，因此拓扑模块还需要后续的维护过程。

本系统在正常的拓扑构建阶段完成后，会有专门的维护线程定期发送心跳消息，来判断邻居集群是否失效，若收到心跳消息，则刷新该集群的有效时间；若消息超时或未收到心跳消息，则认为该集群已经失效或出现重大故障，无法继续使用，因此需要删除失效节点，重新进行拓扑感知。拓扑维护具体流程见下图。

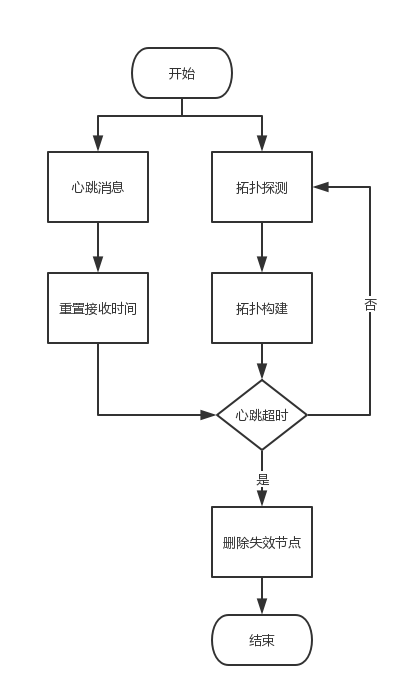


图5-23 拓扑维护流程图

1. **路由管理**

拓扑管理模块获得的是“静态”的链路连接信息，消息接口能够知道某些链路节点存在消息交互的需求，凭借这两个信息无法得到消息传输链路，还需要结合路由计算模块，构建这些参与节点间的转发路径，具体包括管理路径的计算、主题路径的计算等，下面分别进行介绍。

1. 管理路径计算

管理路径是为了保障管理员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，且管理路径上不存在环形结构，因此管理路径本质上是一棵以管理员为根节点的树形结构，本系统在计算完成后保存了相关结果，这样当后续节点加入时，只需要计算与当前管理路径中的最短路径。

Dijkstra算法是最通用的计算最短路径的算法，算法以起始点为中心向外层层扩展，直到扩展到终点为止，具有简单、高效的特点，因此本系统使用Dijkstra最短路径算法来实现管理路径的计算，相关伪代码如下所示。

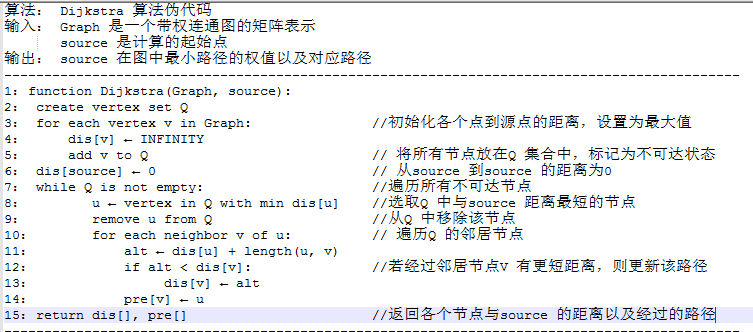


图5-24 Dijkstra算法伪代码图

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位不可达状态（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达到，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。

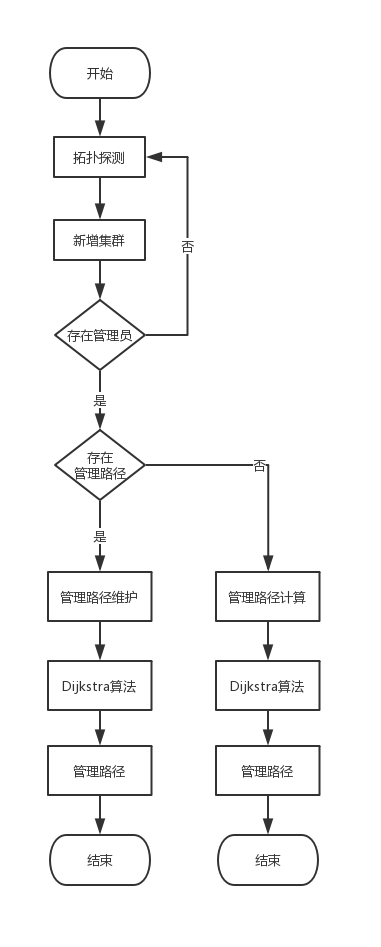


图5-25 管理路径计算流程图

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。

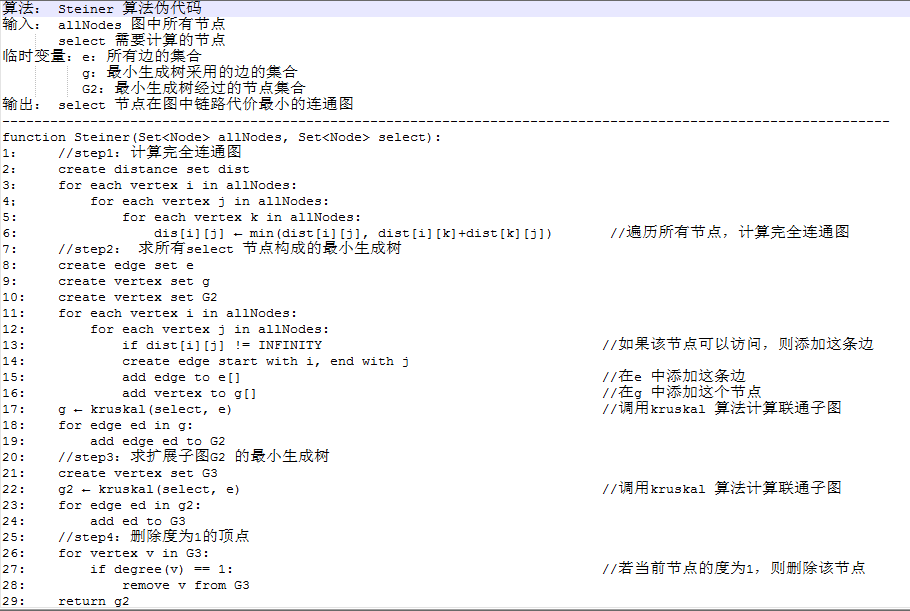


图5-26 Steiner算法伪代码图

本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；下面需要求解扩展子图的最小生成树，经过这一步，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。

从算法过程中可以看到，Steiner算法比较复杂，需要首先计算出联通子图，确保结果的正确性，同时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。

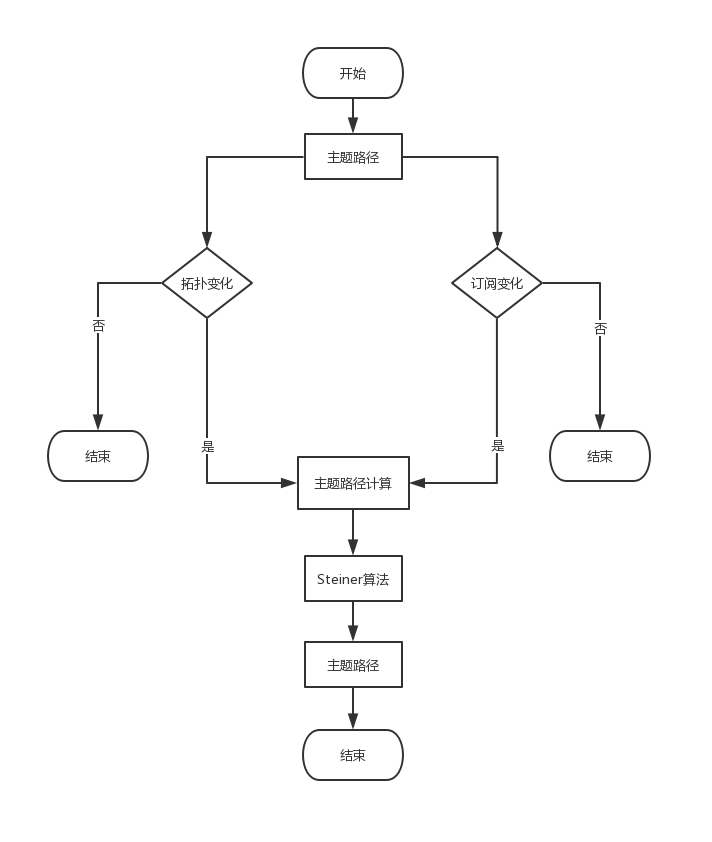


图5-27 主题路径计算流程图

1. **传输质量保证**

已有系统中，程序使用生产者-消费者模式提供可靠性保障，关于生产者-消费者的基本概念已经进行了介绍，这是一种协调收发双方处理速率的方法，但是还不够完善，基于时间驱动方式不适用于高并发、大数据量的情况，这在测试模块中将通过具体的测试样例进行说明。

Reactor模式是基于事件驱动，遵循注册—调用机制，在具体实现时，用户自定义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，并进一步抽象出任务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，由于分派过程可以自行定义，比如添加过滤机制、负载均衡策略等，因此Reactor模式具有更高的性能和扩展性。具体时序图如下所示。

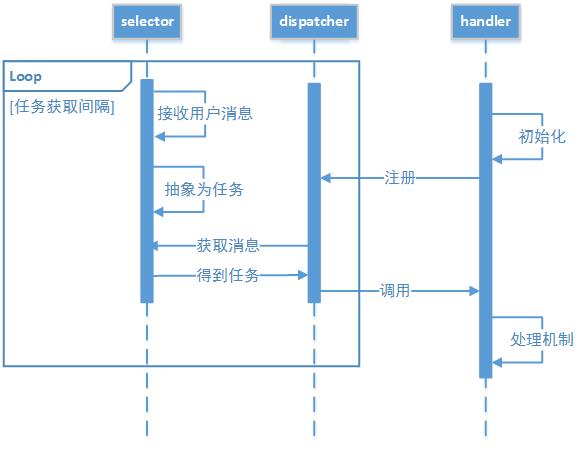


图5-34 Reactor模式时序图

1. **流表管理**

经过路由算法模块的计算后，生成了管理路径、主题路径的结果，但这是内存中的逻辑链路形式，还需要进一步解析成流表，这样才能结合SDN物理交换机，充分发挥SDN软件定义网络的强大功能。该部分具体划分为流表的生成、流表的下发、流表的维护，模块整体的功能如下图所示。

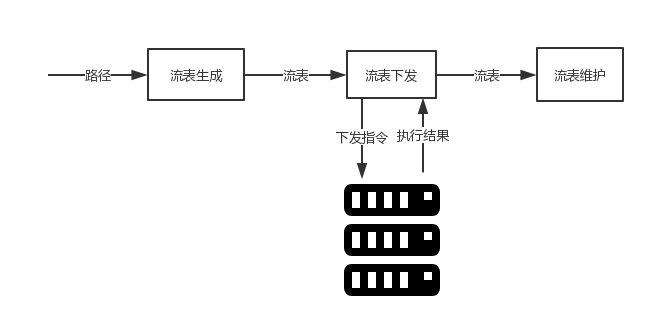


图5-28 流表管理功能图

上图展示了流表管理的几个步骤：经过路由模块计算出的路径结果调用流表生成方法，转换为流表的形式，进一步可以下发给交换机，流表下发模块将执行成功的流表交由维护模块集中管理。下面对各个模块分别进行介绍。

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的结果，因此就需要在两者之间进行转换，最直观的思路就是将交换机指令抽象成模板，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，这样的设计便于开发者重复使用，以及后期的维护。流表的一般操作格式见下图。

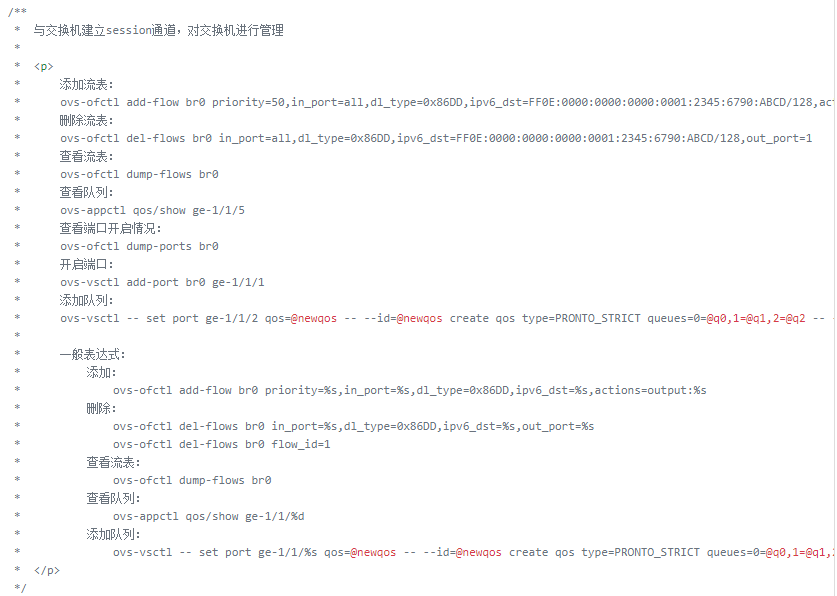


图5-29 OVS操作指令

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。在系统代码编写中，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交换机处理器，进而下发流表指令，完成相应的需求。流表下发具体执行结果见下图。

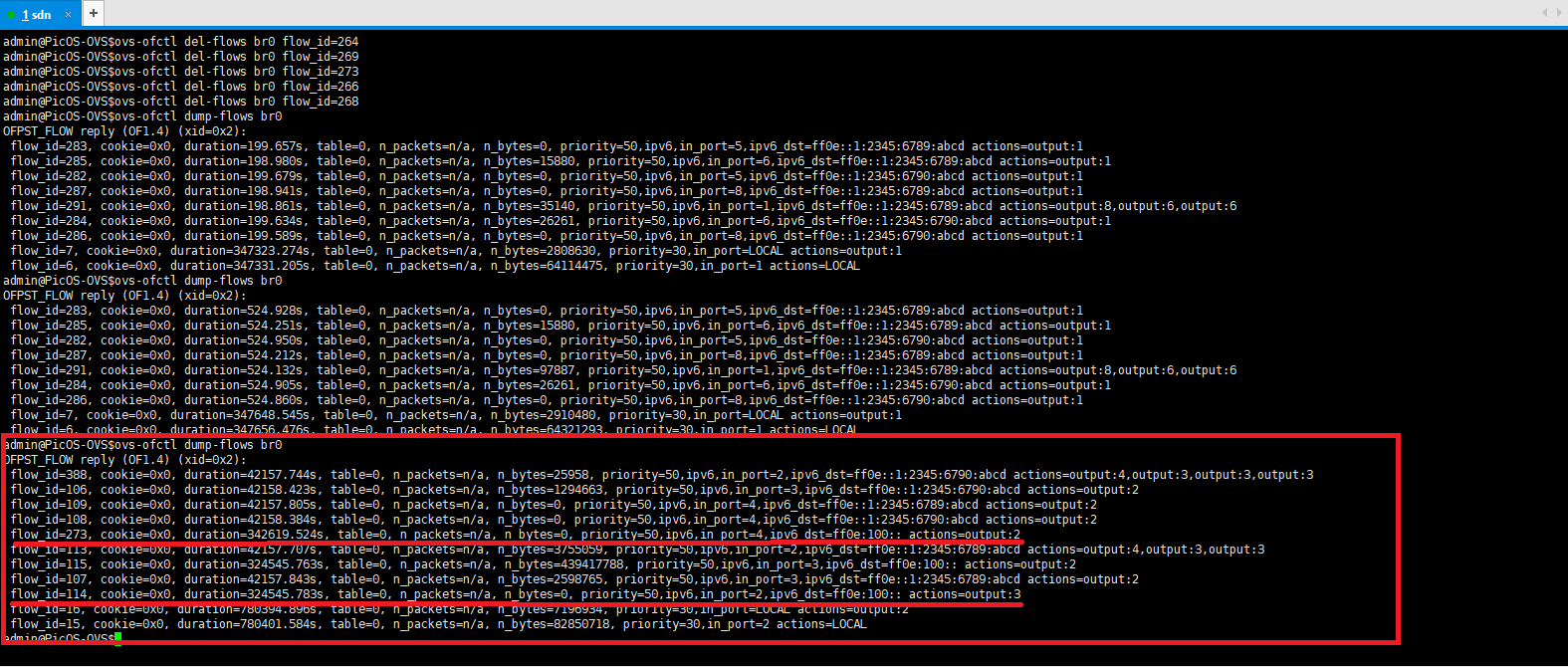


图5-30 流表下发执行结果

1. 流表的维护

流表项在交换机中通过下发模块可以执行，但是实际操作过程中，流表可能存在重复、冲突等情况，重复的流表下发会影响交换机处理性能，极端情况下会产生冲突流表，即某些流表项的目的转发地址、出端口号、优先级等字段一致，区别在于进端口号的不同，而本系统采用的SDN交换机在面对冲突流表时会使用后者替代先下发的流表，这与我们真实需求相反，因此在内存中也需要有流表维护模块，对生成、下发的所有流表进行维护，流表维护的相关类图如下所示。

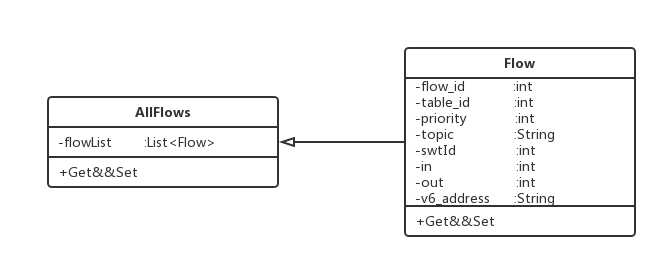


图5-31 流表维护类图

图中展示了流表类的消息格式，下面介绍各字段含义。

表5-5 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 |

可以看到，在定义了基本的流表项（Flow）类后，内存中使用List链表的形式存储所有的流表，这样当生成新的流表时，可以在当前所有流表中判断是否存在重复或者发生流表冲突，进一步通过流表的合并操作解决冲突。

1. **管理员模块实现**
2. **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，实现消息的传输与接收。主题管理模块具体可以划分为主题的存储、主题树的编码、主题树的下发，在设计时，系统划分多个子模块是为了便于编码实现以及后续的维护。

主题管理模块调用接口，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，获取到的主题树信息通过编码子模块，得到编码后的主题树信息，同时在管理模块内部存在周期性的主题通知机制，定期将编码主题树内容发送给各个控制器。主题管理时序图如下所示。

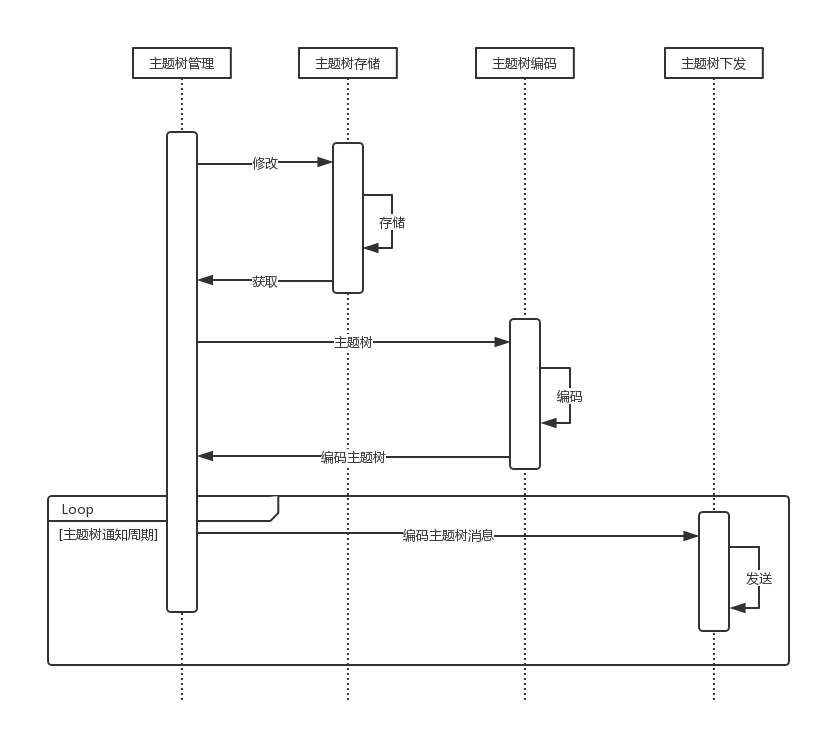


图5-13 主题树管理时序图

1. 主题存储

主题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，这样就形成了一颗以虚拟的“all”节点为根节点的主题树。经过分析，主题的存储转换为树形结构的存储，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5-14 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完成主题到ipv6地址的转换，这里依据的是主题节点在树形结构中的相对路径和位置，具体划分为编码、末尾0填充、组播地址头部封装这三个阶段，整体流程如下图所示。

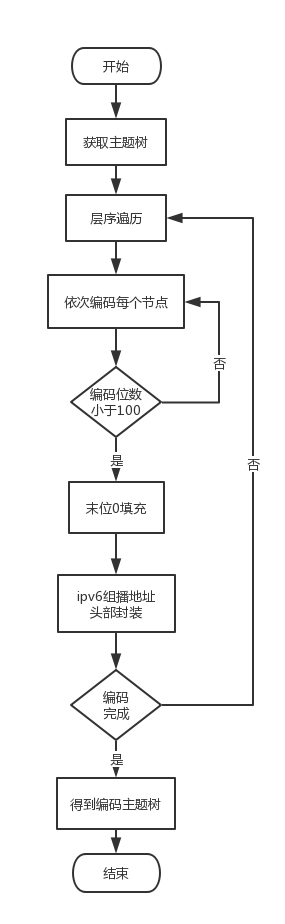


图5-15 主题树编码流程图

本系统采取了IPv6 组播的形式进行消息传输，发送者向相应主题对应的IPv6 地址进行IPv6 组播，接收者加入IPv6 组播的群组，接收发来的相应主题数据包。针对128bit 的IPv6 地址，系统将其中100bit 的空间预留给了主题树的编码。主题树编码的伪代码如下所示。

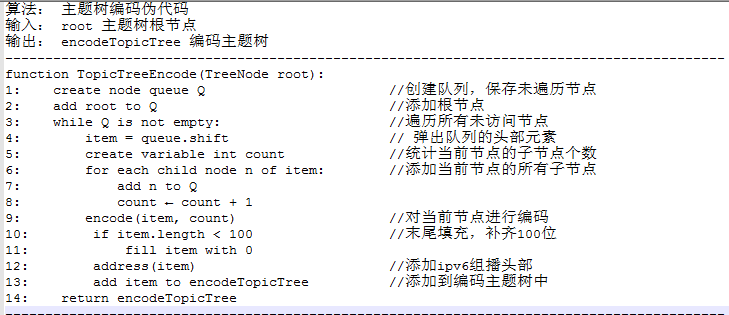


图5-16 主题树编码伪代码

图中可以看到，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bit位表示，这样的代价最低，最终实现了主题树到编码主题树的转变。其中，系统使用ipv6组播地址进行头部填充，相关字段如下所示。

表5-2 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，但由于IPv4地址的长度有限，只有32bit，减去组播的固定前缀，留给主题编码的部分就很少了，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主题树消息的传输，减轻传输负担，但是分布式集群的架构，使得主题树的修改又成了大问题，甚至会产生集群间内容不同步的问题。综合考虑，本系统最终采取的是管理员集中管理。

在读取主题树内容后，封装为主题树消息格式，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。传输的主题树消息类定义如下所示。

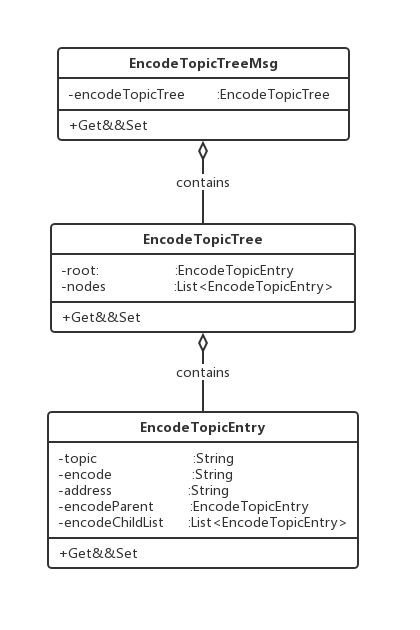


图5-17 主题树消息类图

图中概括了主题树消息的组成、主题树及主题节点的消息格式，下面介绍各字段含义。

表5-3 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> | 主题树所有节点列表 |
| EncodeTopicTreeMsg | encodeTopicTree | EncodeTopicTree | 编码主题树 |

定义了消息格式后，就可以将xml文件中获取的主题消息编码，并下发给控制器。主题树下发的时序图如下所示。

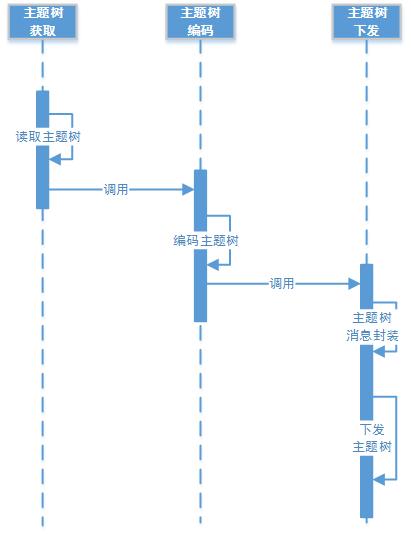


图5-18 主题树下发时序图

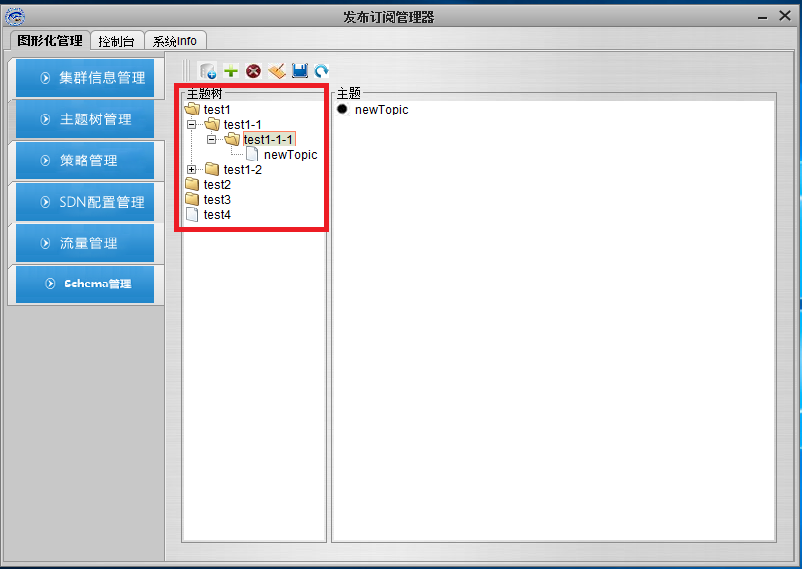
1. **用户请求处理**
2. **ui展示**

管理员模块为管理者提供了一个便于操作、展示的ui界面，如下图所示。



图4-17 管理员界面设计图

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，相关内容在4.2.5队列管理中进行了介绍。同时管理员可以在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，再由集群控制器自行更改。主题树内容保存在xml中，管理员启动后将读取相关文件并在下图中展示。



1. **WSN模块实现**

用户与WSN层之间是通过消息接口进行通信，以一次通信流程为例，开始时，服务端暴露服务地址（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，接着向用户指定地址发送处理结果，从而形成WSN与用户之间的双向通信。经过消息接口模块，用户与WSN之间能够完成消息的交互。消息接口的时序图如下所示。

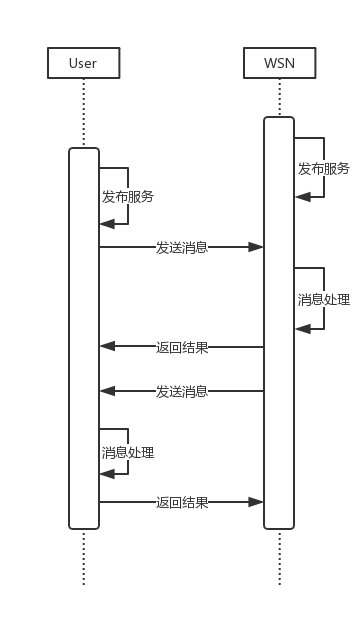


图5-19 消息接口时序图

1. **用户模块实现**
2. **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。

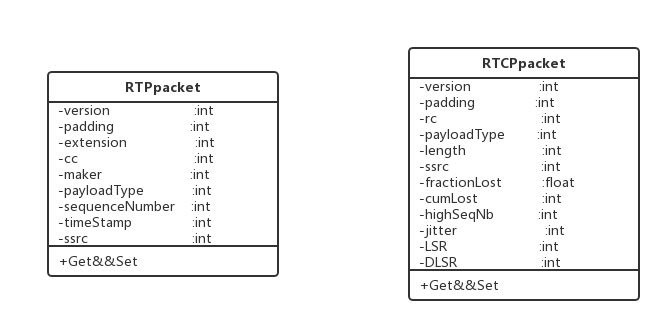


图5-32 多媒体传输类图

图中概括了RTP、RTCP的消息格式，下面介绍各字段含义。

表5-6 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 | 拓展位标志 |
| 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 |

在定义了消息格式后，就需要结合协议规范，实现多媒体传输功能了。本系统中，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。数据获取模块通过字节流的方式从硬盘上获取视频内容，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传输到接收方处，接收方经过反向解析就能得到视频数据信息。具体传输时序图如下所示。

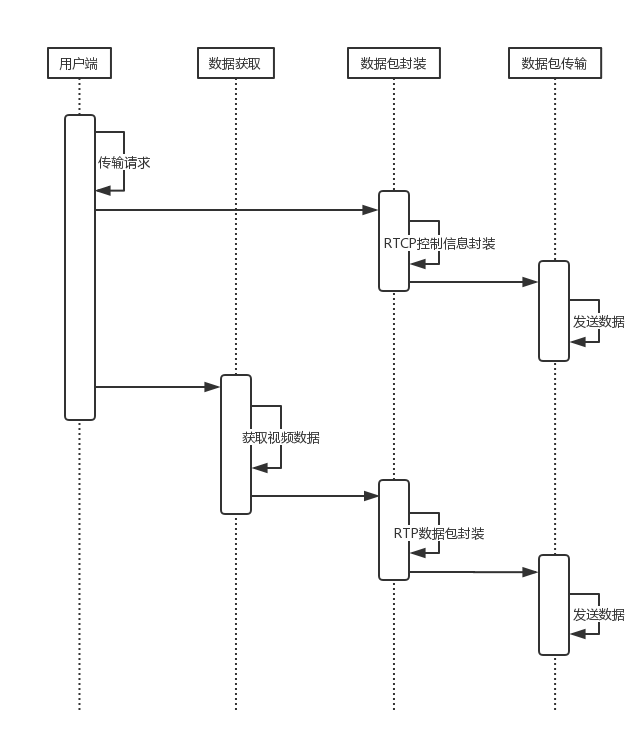


图5-33 多媒体传输时序图

1. **用户协商机制**

支持用户与系统间的需求协商，是本系统的创新点之一，在实际使用过程中，用户的确会主动提出针对时延、带宽的请求，这具有现实意义。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。

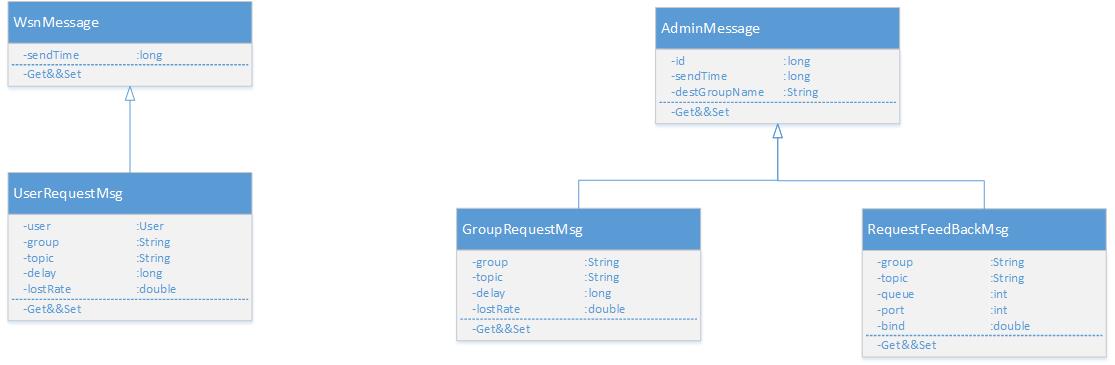


图5-35 用户协商消息类图

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5-7 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 |

在定义了用户协商流程中涉及到的消息格式后，我们可以对协商流程进行抽象。首先由用户模块发起时延、带宽请求，请求消息到达wsn模块后，提取相关参数，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。

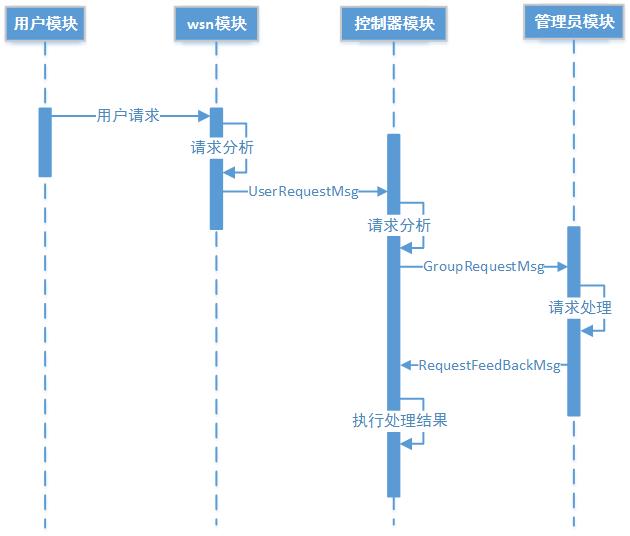


图5-36 用户协商时序图

1. **日志系统实现**

已有系统中，针对数据输出提示采用的是命令行窗口打印的方式，在普通程序中可以达到预期目的，但在发布/订阅系统中，不论是信息展示的数量还是类别都更为复杂，常常会因为系统打印内容过多而找不到关键信息，因此需要一套新的输出提示机制。本系统采取的是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，重要性逐级递增。日志系统的通知机制如下图所示。



图5-37 日志通知机制

从图中可以看到，本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**

本章主要对系统中各个核心模块的设计进行了详细阐述，包括参与角色、消息的分类和定义、SDN发布/订阅系统架构设计、多媒体传输、QoS保障方案、日志系统等，第六章将进行系统测试和验证，证明本系统的可靠性，以及QoS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，验证各个模块运行的稳定性，同时测试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各模块是否稳定运行，以及模块间的交互是否顺利，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。

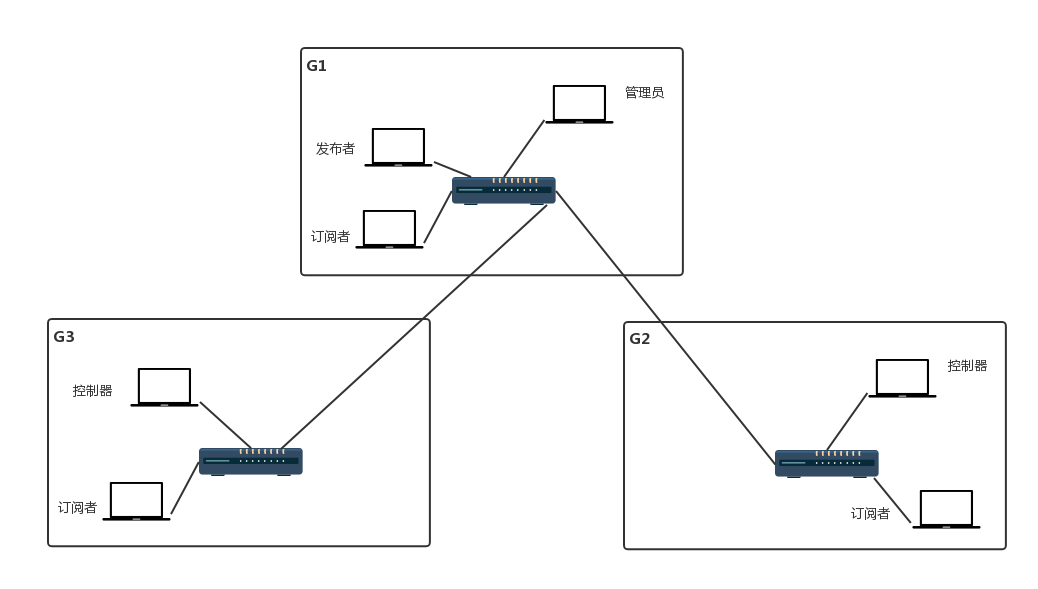


图6-1 功能测试拓扑

图上展示出了真实的测试环境，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。

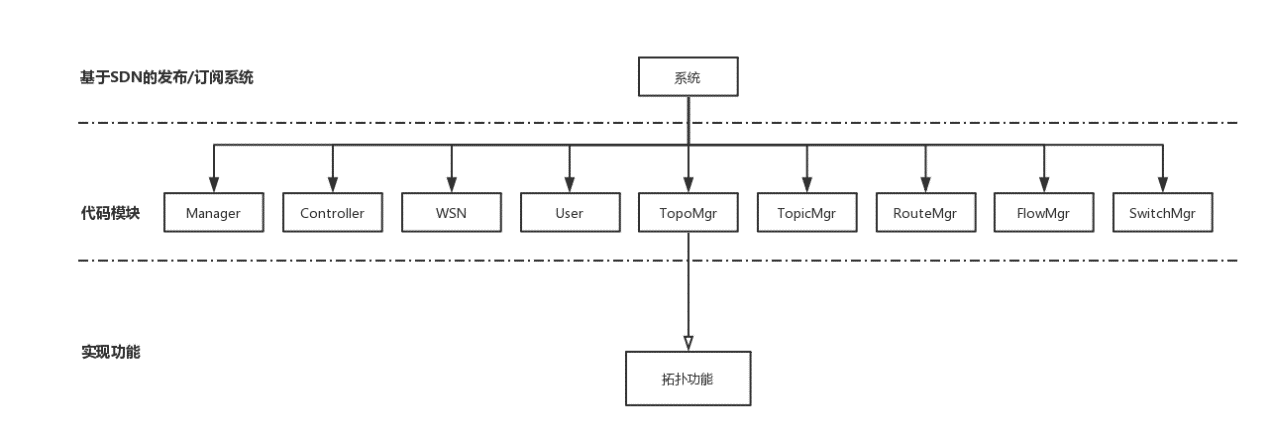


图6-2 拓扑功能涉及的代码模块

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，具体测试样例见下表。

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。

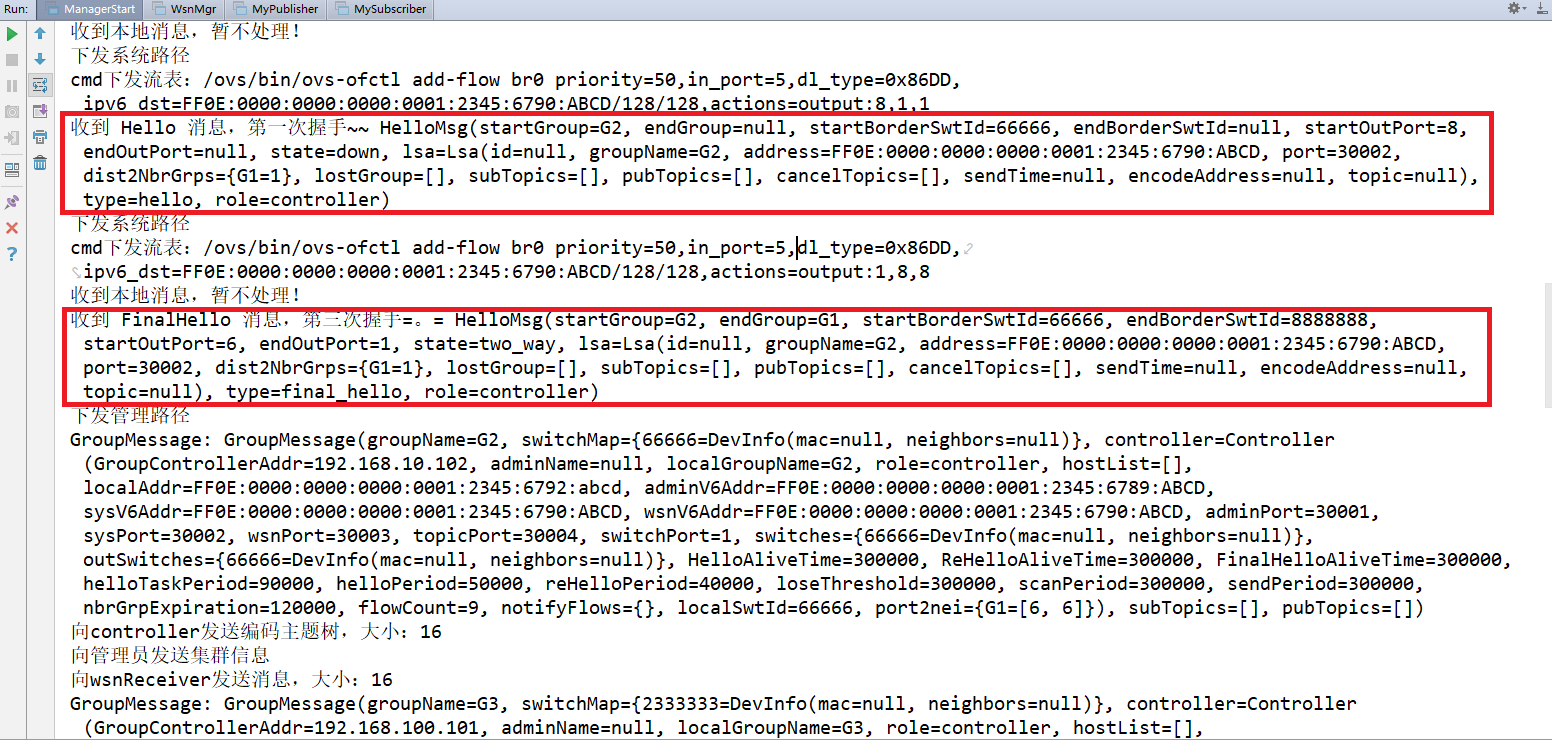


图6-3 拓扑构建结果（一）

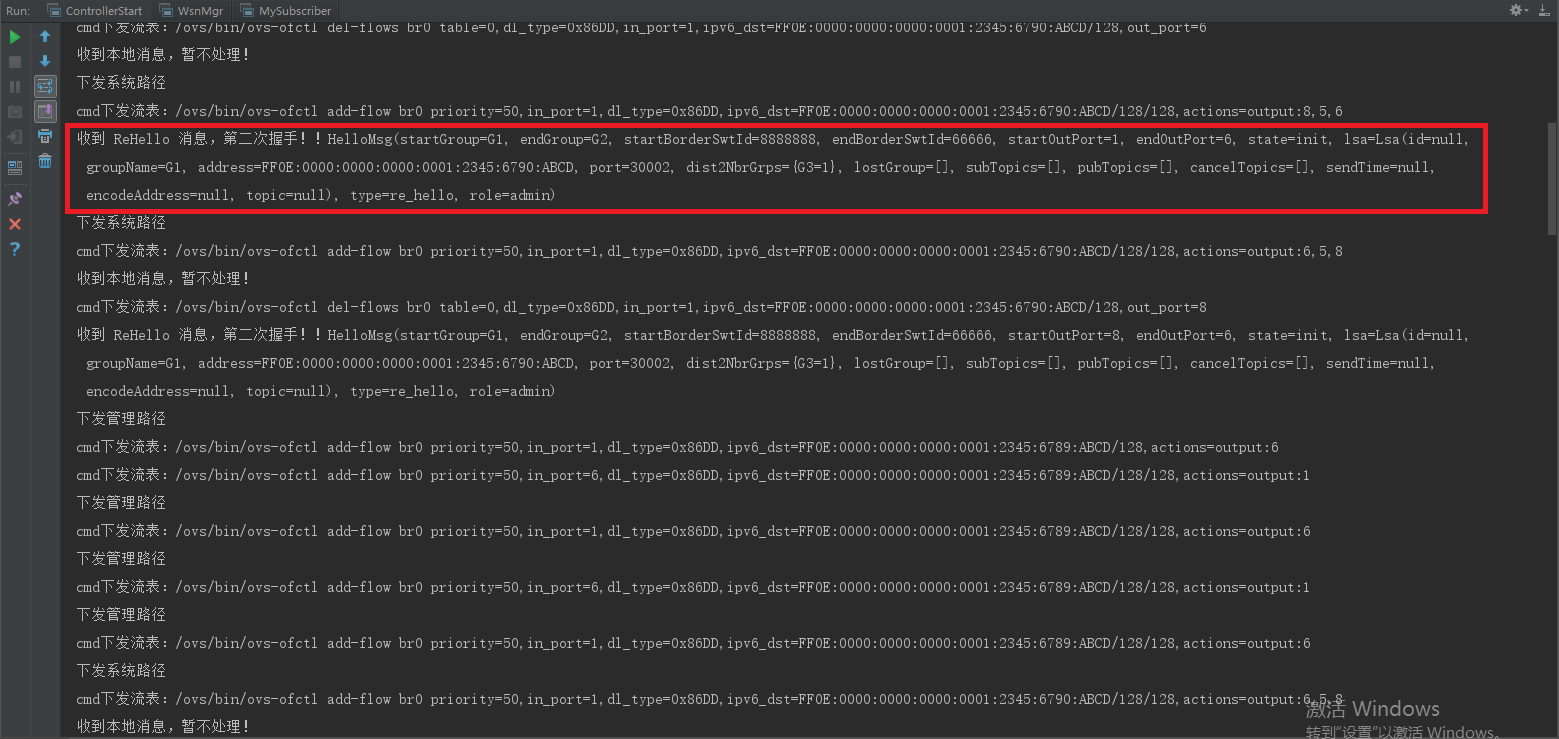


图6-4 拓扑构建结果（二）

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，测试样例如下所示。

表6-4 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，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。

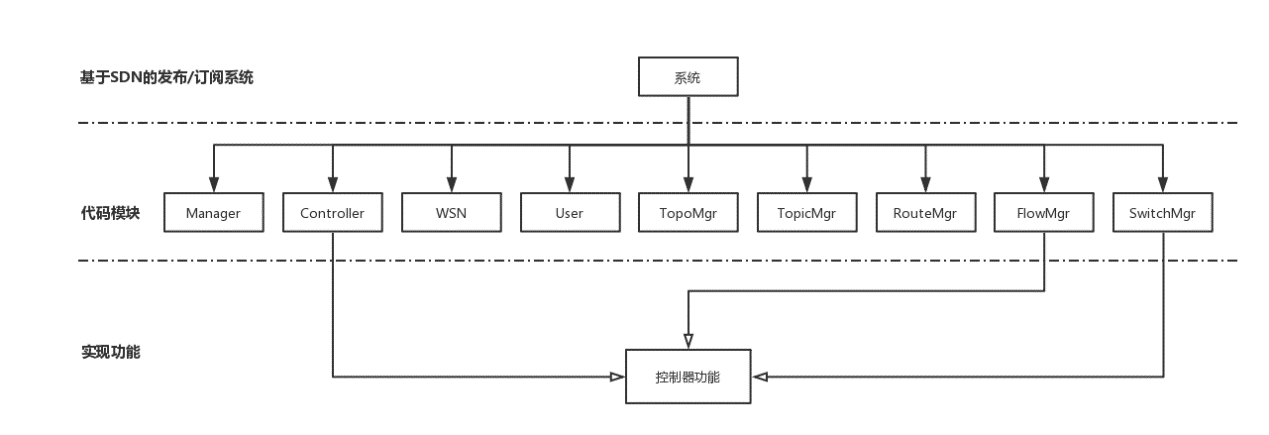


图6-6 控制器功能涉及的代码模块

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、FlowMgr流表管理模块、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，测试样例见下表。

表6-5 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。

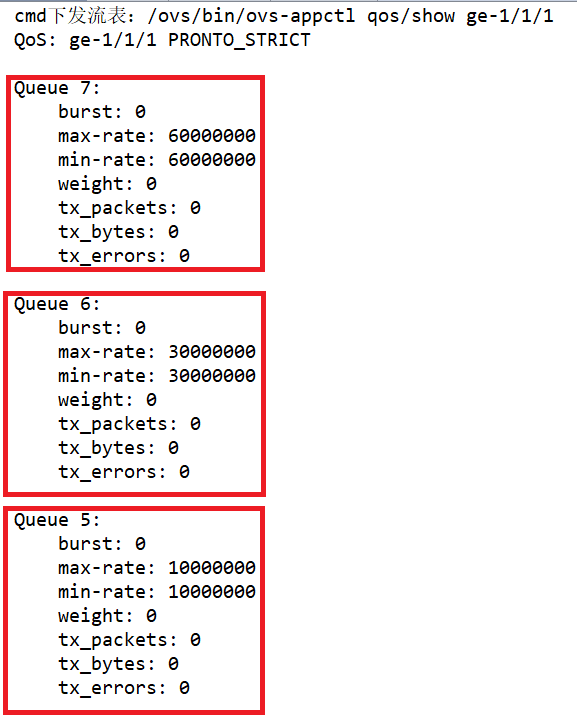


图6-7 队列信息收集结果

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。具体测试情况见下表。

表6-6 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，具体测试样例见下表。

表6-7 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。

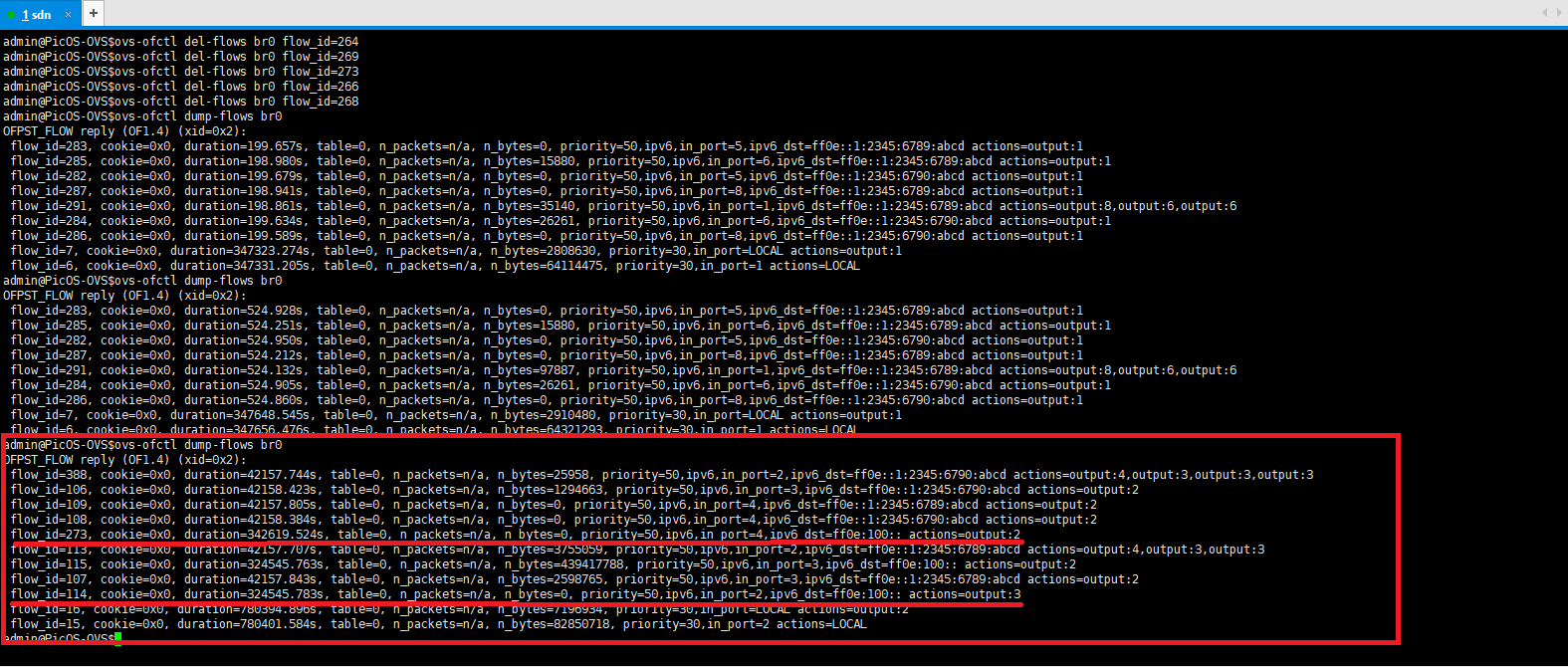


图6-8 流表下发测试结果

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。测试样例见下表。

表6-8 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。

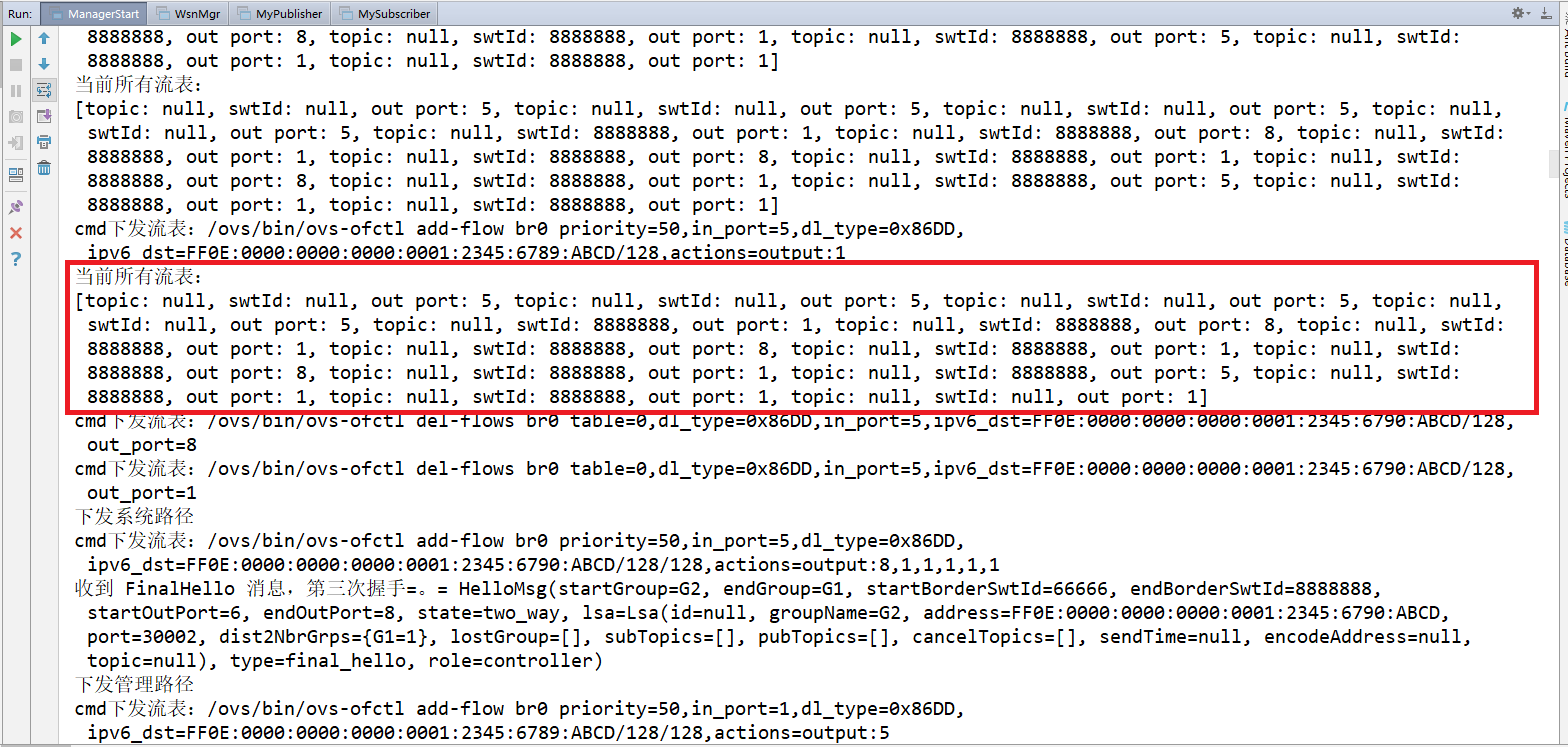


图6-9 流表维护测试结果

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。

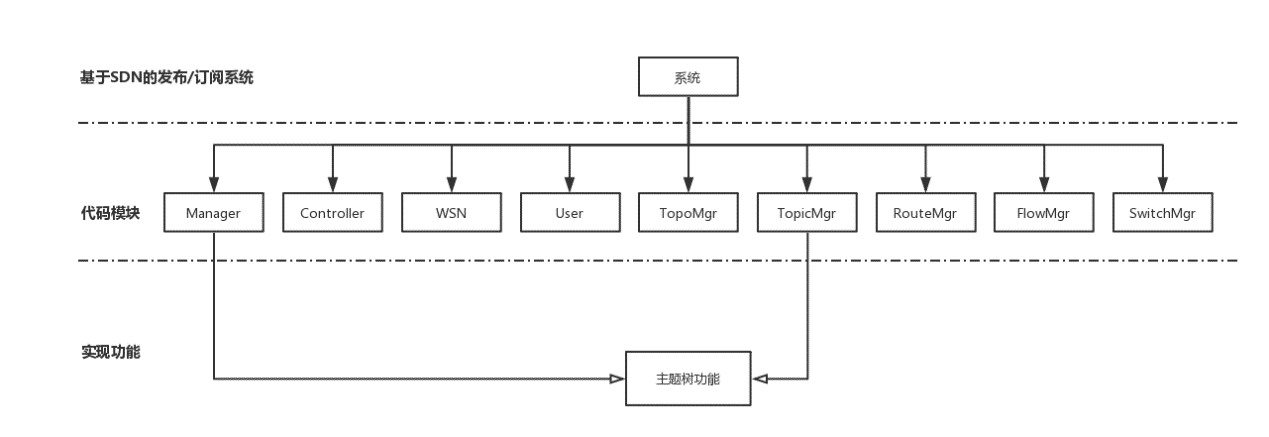


图6-10 主题树功能涉及的代码模块

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所有信息，同时提供操作接口，管理员可以在ui界面上对主题树信息进行增删改查操作，具体测试样例见下表。

表6-9 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，并写回到持久化文件中 |
| 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。

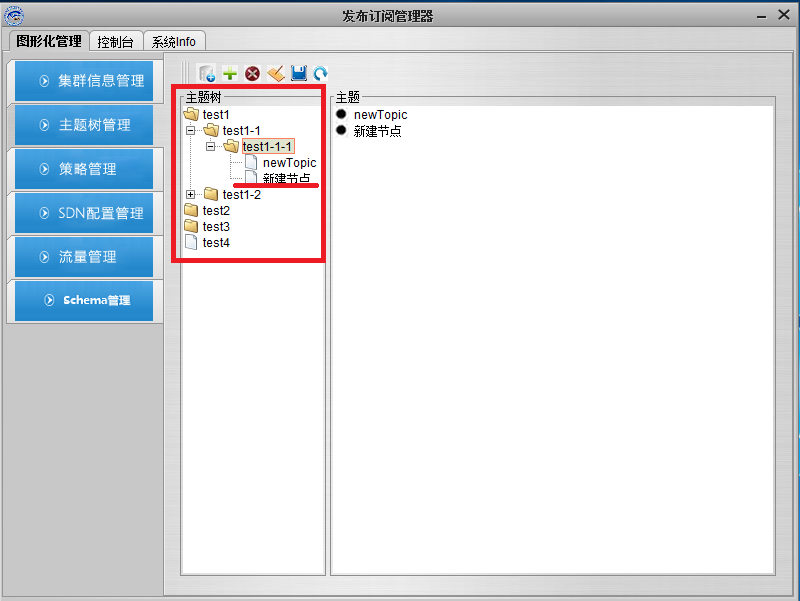


图6-11 主题树修改结果

1. 主题树下发

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

表6-10 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。

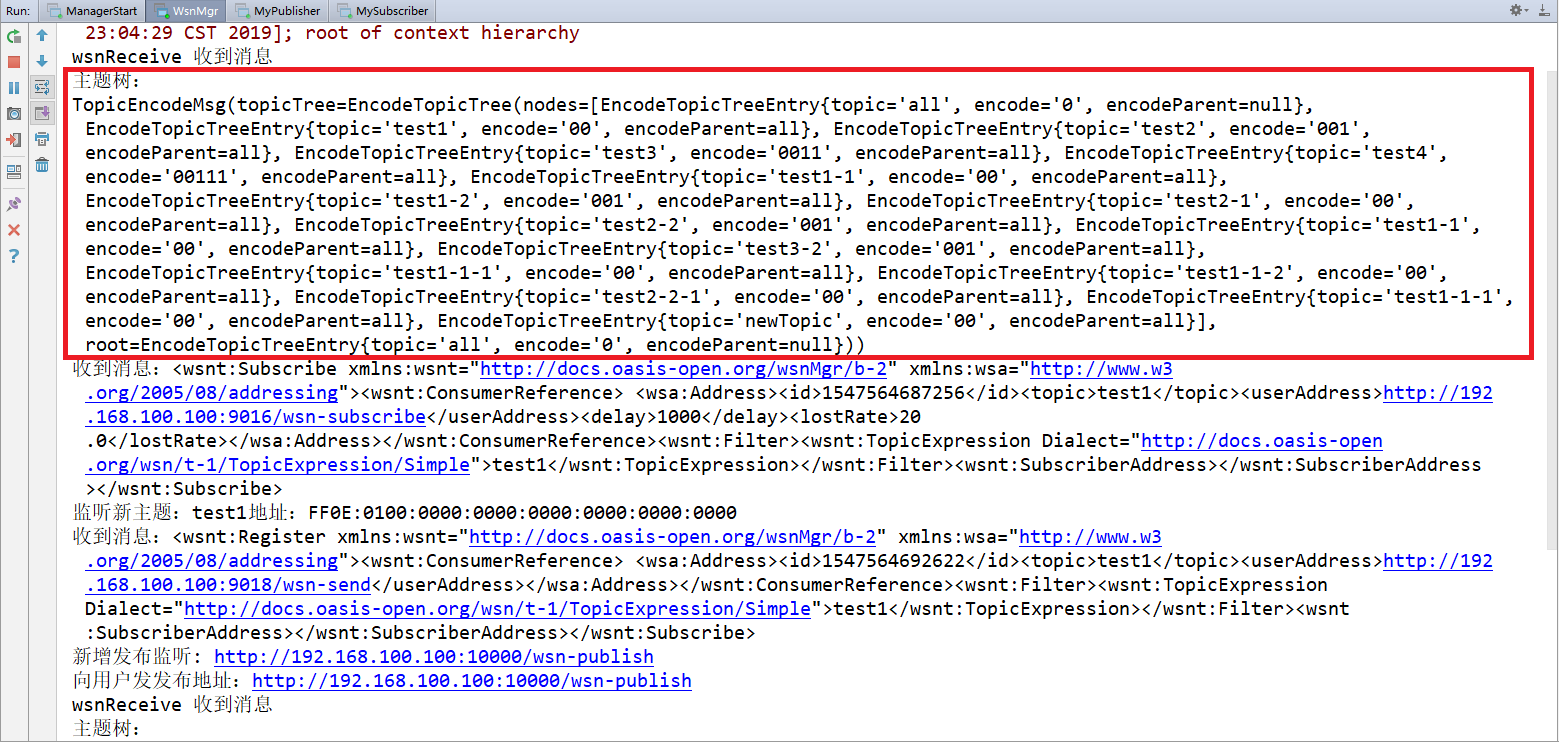


图6-12 主题树下发结果

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。

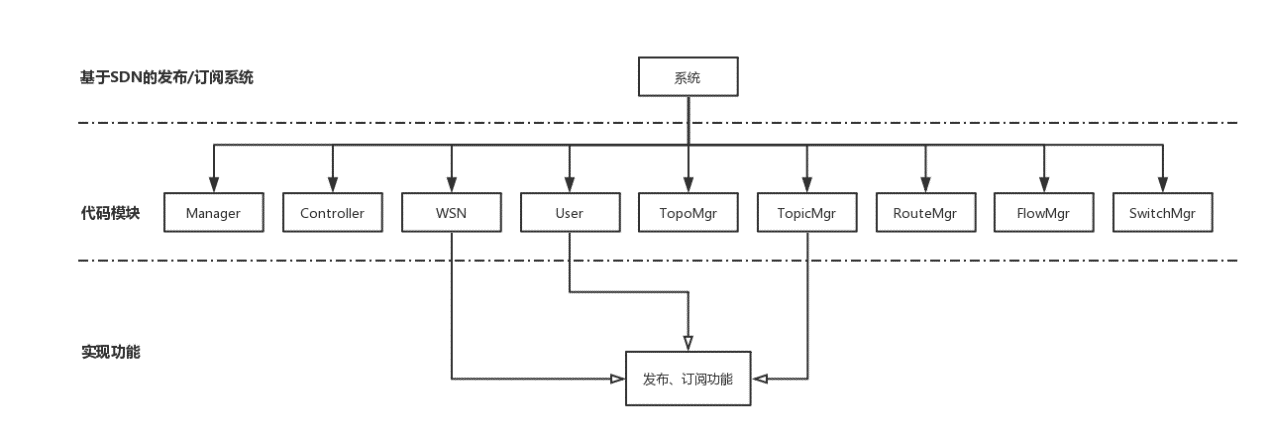


图6-13 发布、订阅功能涉及的代码模块

图中展现的是发布、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。具体测试样例见下表。

表6-11 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。

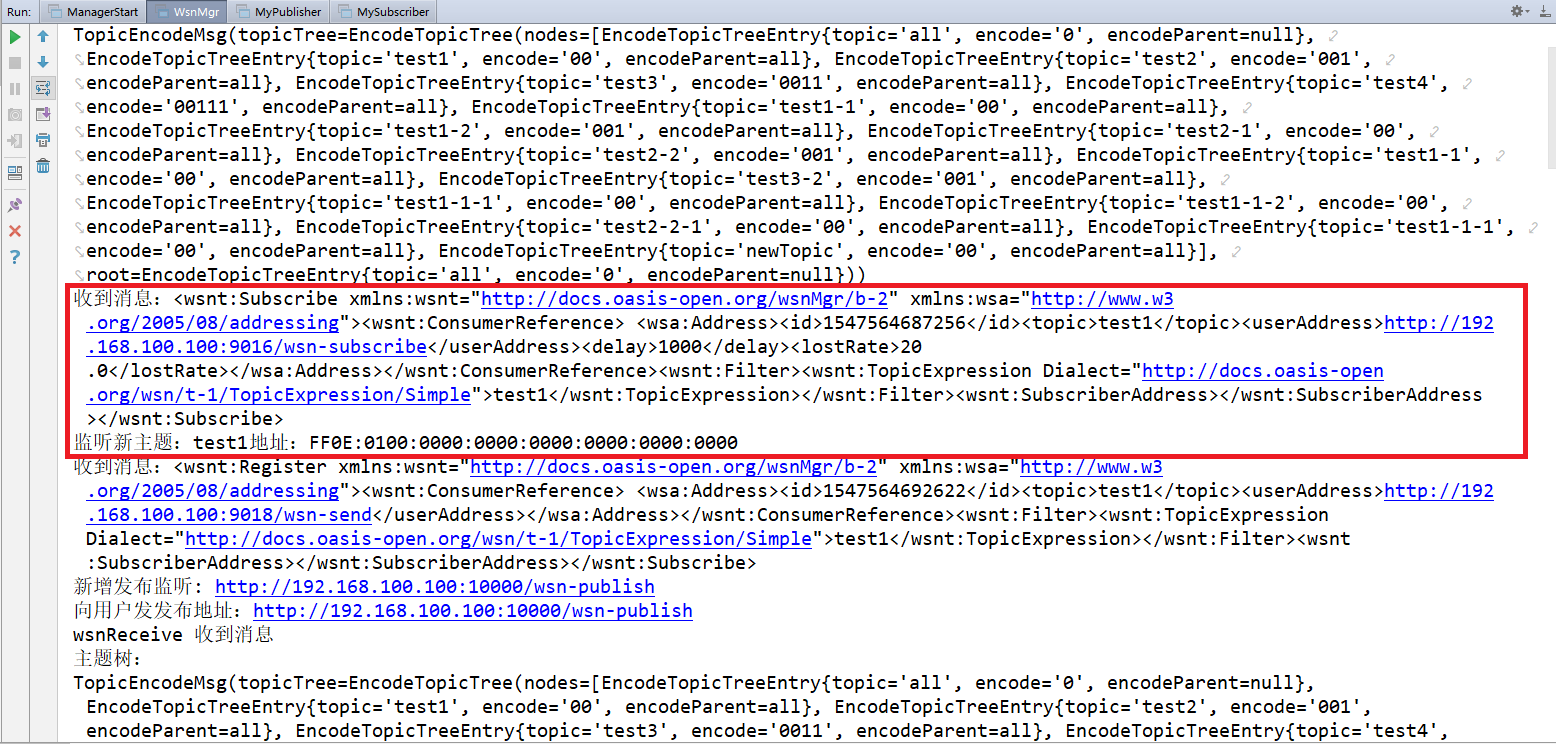


图6-14 新增订阅测试结果

1. 发布请求

用户发起发布请求，wsn层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。测试样例见下表。

表6-12 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。

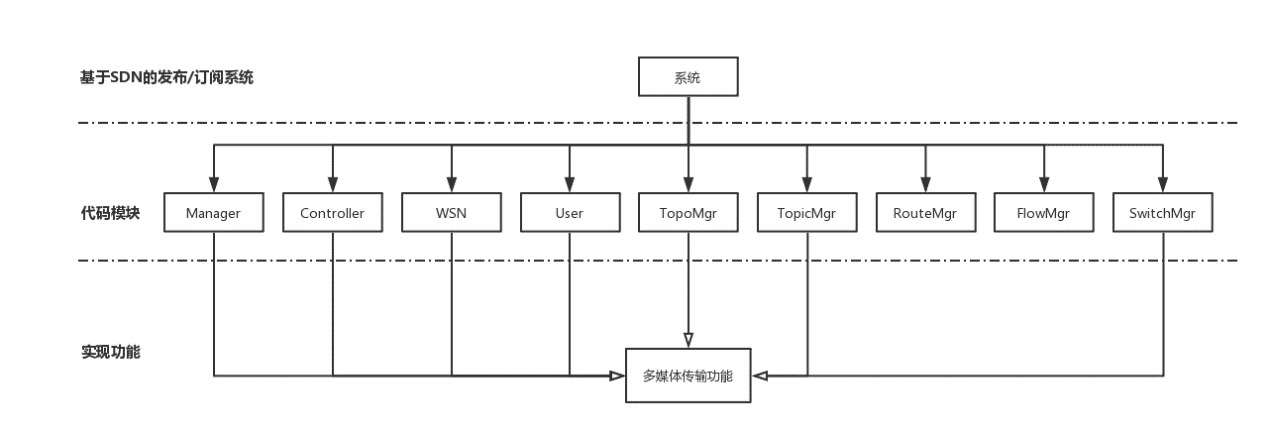


图6-16 用户协商功能涉及的代码模块

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，具体的测试样例如下表所示。

表6-13 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，具体的测试样例如下表所示。

表6-14 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，视频流数据根据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。

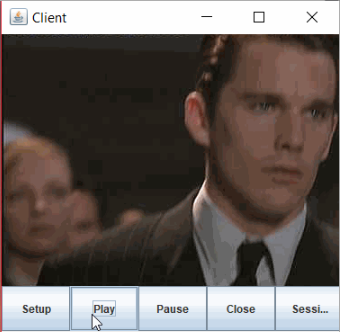


图6-18 多媒体传输测试结果

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。

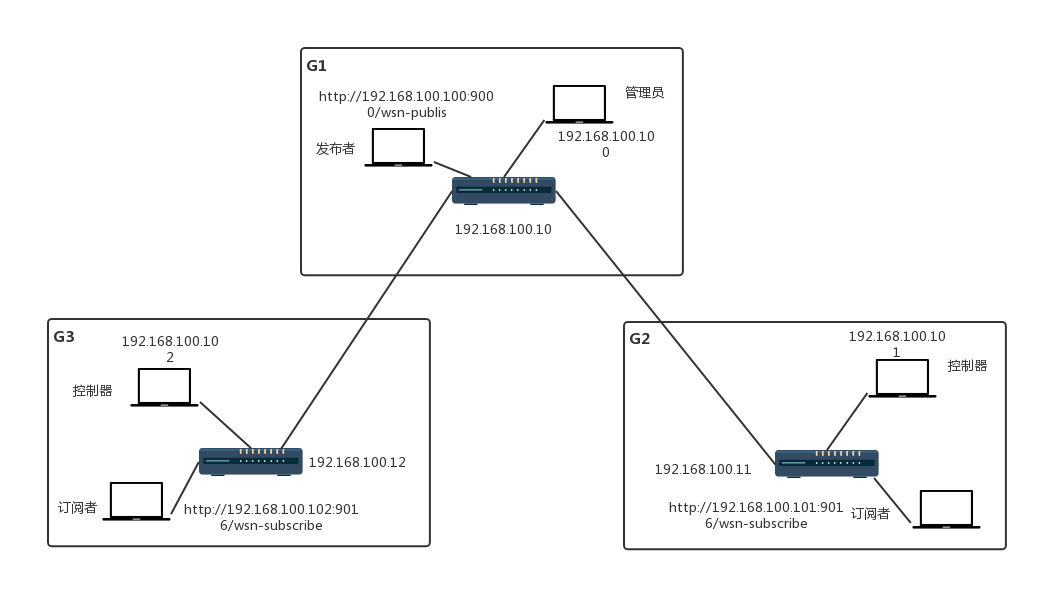


图6-19 性能测试拓扑

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6-15 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6-16 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6-17 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6-18 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6-19 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6-20 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。