目录

[第一章 绪论 2](#_Toc535351619)

[1.1 研究背景 2](#_Toc535351620)

[1.2 研究内容 2](#_Toc535351621)

[1.3 论文组织 2](#_Toc535351622)

[1.4 本章总结 2](#_Toc535351623)

[第二章 相关技术概述 2](#_Toc535351624)

[2.1 SDN 2](#_Toc535351625)

[2.2 发布订阅系统 2](#_Toc535351626)

[2.3 OSPF 2](#_Toc535351627)

[2.4 多媒体传输技术 2](#_Toc535351628)

[2.5 本章总结 2](#_Toc535351629)

[第三章 需求分析 2](#_Toc535351630)

[第四章 系统概要设计 2](#_Toc535351631)

[4.1 系统架构 2](#_Toc535351632)

[4.2 本章总结 2](#_Toc535351633)

[第五章 系统详细设计与实现 2](#_Toc535351634)

[5.1 工程概览 2](#_Toc535351635)

[5.2 参与角色 2](#_Toc535351636)

[5.5.1 控制器 2](#_Toc535351637)

[5.5.2 管理员 2](#_Toc535351638)

[5.5.3 发布者 2](#_Toc535351639)

[5.5.4 订阅者 2](#_Toc535351640)

[5.3 功能模块划分及实现 2](#_Toc535351641)

[5.5.1 管理员ui模块 2](#_Toc535351642)

[5.5.2 队列模块 2](#_Toc535351643)

[5.5.3 路由模块 2](#_Toc535351644)

[5.5.4 交换机模块 2](#_Toc535351645)

[5.5.5 主题树模块 2](#_Toc535351646)

[5.5.6 拓扑模块 2](#_Toc535351647)

[5.5.7 wsn模块 2](#_Toc535351648)

[5.4 Qos保证 2](#_Toc535351649)

[5.5.1 Reactor模式 2](#_Toc535351650)

[5.5.2 负载均衡策略 2](#_Toc535351651)

[5.5.3 RTP保障策略 2](#_Toc535351652)

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，因而无法为多媒体传输质量提供保障，急需一种高效、分类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，与传统网络转发相比，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流传至控制器，由控制器计算相关路径再将流表下发至交换机，从而提高转发的准确率。

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。

本课题基于SDN系统下的发布/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、多端口注册等需求提供支持，同时，编码实现多媒体数据传输，使用队列调整策略和路由算法提供传输质量上的保证。

* 1. 研究内容

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，构建一个可控且可靠的统一消息中间件网络。

本课题借助实现了OpenFlow协议的交换机，搭建了一个基于SDN的发布/订阅系统，目前，该系统已经实现了基本的转发功能，在队列调整、流量管理上也获得了初步成功，但是这些功能模块还停留在实验阶段，测试环境也仅仅局限在传输文本文件，要想真正实现软件控制网络进而发挥SDN网络的优异性，还需要在真实的网络环境中运行。除此之外，对比之前基于传统网络的发布/订阅系统仍有许多功能需要添加和完善。本课题在充分调研现有项目和阅读大量相关文献和已发表论文之后提出了以下三个研究内容：

1. SDN发布订阅系统的架构设计

现有的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，便于各个集群的独立计算。分为集群内和集群间拓扑维护，将网络划分为若干集群，每个集群由一个集群控制器进行管理，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，交换机获取拓扑通过控制器实现。

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/订阅节点消息交互的接口，用户通过这个接口实现数据的交互，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，基于 SDN 网络的发布/订阅吸收了基于传统 IP 网络的发布/订阅系统和基于SDN 网络的发布/订阅系统的优点，将其结合在一起形成了“集中管理，分布计算”的体系结构，将根据 Steiner 树计算路由的功能下发到每个节点中，只在流表下发时才统一通过集群控制器进行；

流表管理是发布订阅系统与SDN物理环境结合的重要一环，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，这样就可以实现数据在物理链路上的直接传输。

上述功能在目前的实验环境下已经能很好的运行，但实验内容多为传输文本文件时的时延、丢包率分析，本课题研究的多媒体传输质量的保证还未曾涉及。此外，原本的队列调整算法还处于试验阶段，在调整的策略上还存在着滞后、效果不明显等问题，本课题需要结合多媒体数据的特点与实际传输情况，优化已有的队列调整策略，结合路由算法，为多媒体数据的传输提供质量上的保障。

1. 多媒体传输

在基于SDN的发布订阅系统中， 现有研究的内容还是传输文本字节流，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，用户需要的往往是更大的文件信息，更进一步，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。

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，然后在网络中传输。

1. 质量保证方案

2.1 Reactor模式

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。

本系统提出了另一种基于Reactor模式的转发方案，该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。

2.2 用户协商机制

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。

本系统在提升QoS保障方面，除了改进系统自身处理方式外，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，在管理员处维护了全网拓扑和各个集群的运行情况，经过分析处理，将用户需求的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，则将处理结果返回给用户，由用户自行决定是重新请求还是接受当前结果。

相比系统处理而言，由用户发起的请求更加具有真实性，这大大完善了发布订阅系统的QoS保障功能。

* 1. 论文组织

本文主要提出SDN发布订阅系统的架构，以及多媒体传输的质量保障方案，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，以及各自的需求，核心参与角色包括控制器、管理员、用户；

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

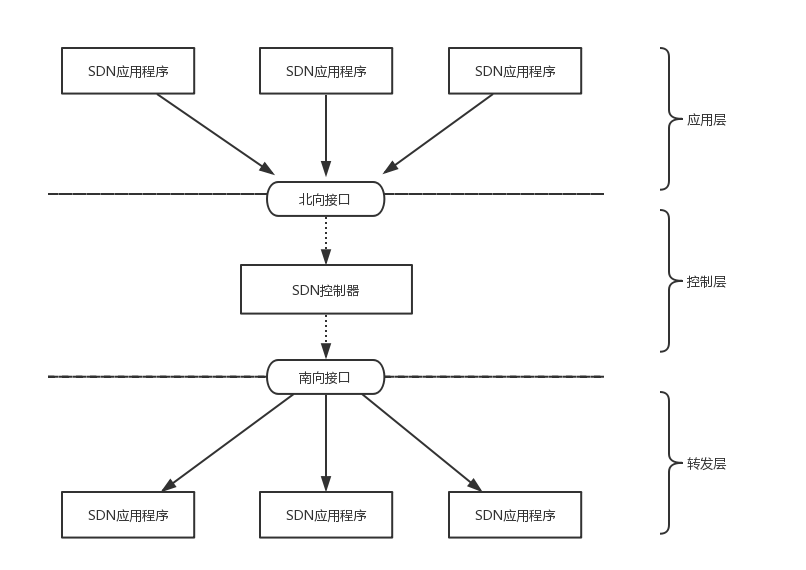
本章是绪论，说明了本文的研究背景和主要研究内容，介绍了关于SDN软件定义网络的相关知识，介绍了发布订阅系统的概念和作用，同时对本文的研究内容进行了简要阐述，包括SDN发布订阅系统的架构设计、多媒体传输方式、质量保障方案的设计等，最后对论文的组织结构进行了概括，方便读者了解文章的结构和写作目的。

1. 相关技术概述
2. 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的架构可由下图描述。



SDN的核心架构可以分为三层，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。

OF的最初概念始于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唾液出，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，已经经过多次版本更迭，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权软件，比如OpenDayLight、Floodlight、Ryu、ONOS等，现有的发布订阅系统采用的就是OpenDayLight，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，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，SDN交换机与仿真环境也大为不同，因此，在开源控制器的设计思路基础上，经过架构设计、编码实现，最终完成了本系统，实现了发布订阅与SDN控制器功能的结合。

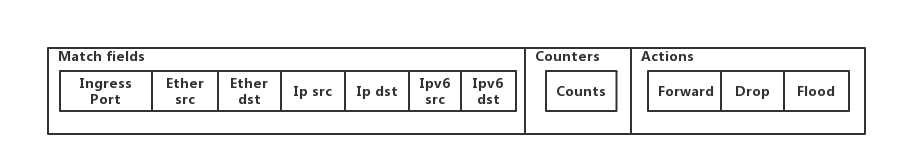
1. SDN交换机

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、传输过程中的数据包信息等。

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，基本功能是对数据包进行匹配、操作，为数据提供转发依据。与传统路由层的路由表相比，流表中的各个流表项是有控制器生成并下发的，这也是SDN网络中，控制层于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，从而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，其组成结构如下图所示。



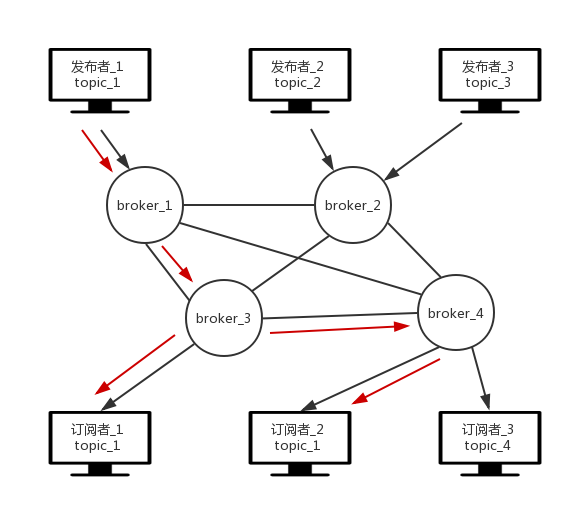
上图列出了流表项的基本构成，各字段作用如下表所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 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。

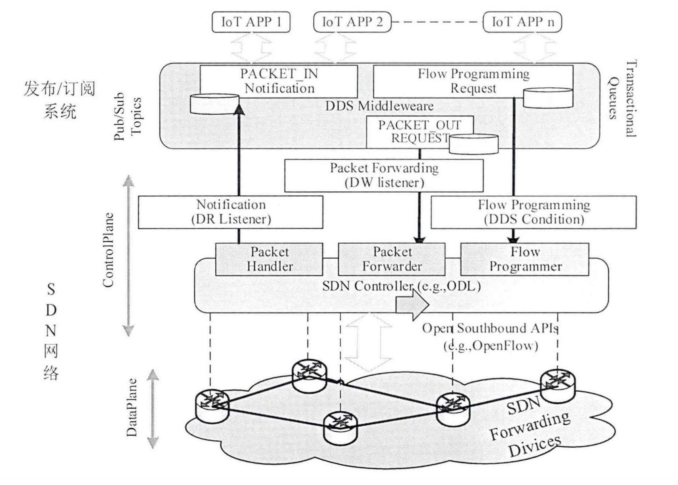
1. 基于SDN的发布订阅系统

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下图所示。



从图中可以看到，发布者\_1、订阅者\_1、订阅者\_2通过相同的主题topic\_1构建联系，这样当发布者\_1发布相应的消息时，经过节点的路由转发功能，最终订阅者\_1、订阅者\_2都将收到通知消息，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，两者间的关系如下图所示。



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，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，针对不同的用意提供个性化QoS保障，具有非常广阔的发展前景。

1. OSPF协议
2. 基本概念

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。

1. 实现原理

1. 每台路由器通过使用Hello报文与它的邻居之间建立邻接关系

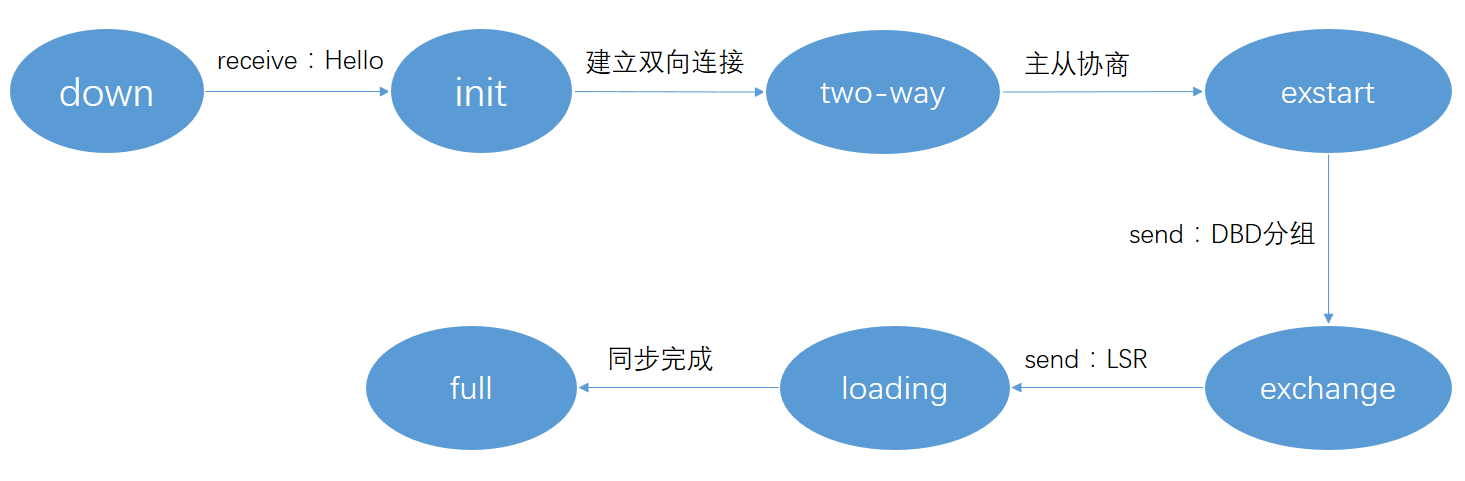
2. 每台路由器向每个邻居发送链路状态通告(LSA),有时叫链路状态报文(LSP). 每个邻居在收到LSP之后要依次向它的邻居转发这些LSP(泛洪)

3. 每台路由器要在数据库中保存一份它所收到的LSA的备份，所有路由器的数据库应该相同

4. 依照拓扑数据库每台路由器使用Dijkstra算法（SPF算法）计算出到每个网络的最短路径，并将结果输出到路由选择表中

OSPF的简化原理：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。

1. 交互流程



图中列出了OSPF交互流程中可能存在的集中状态，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消息被接受，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

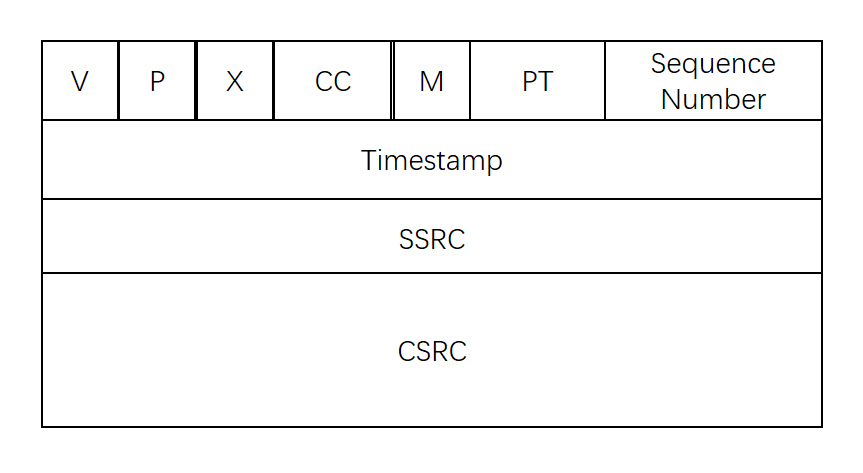
Exchange：信息交换状态；本地路由器和邻居交换一个或多个DBD分组（也叫DDP），DBD包含有关LSDB中LSA条目的摘要信息，然后交换机通过LSR请求，进入loading状态

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

1. RTP协议
2. 基本概念

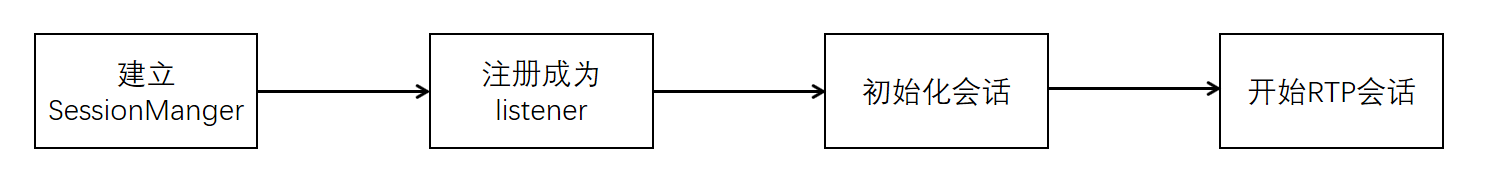
RTP协议是由IETF开发的实时传输协议，可以在面向连接或无连接的下层协议上工作，通常和UDP协议一起使用；RTP定义了两种报文：RTP报文和RTCP报文，RTP报文用于传送媒体数据（如音频和视频），RTCP协议用于传送控制信息，以实现协议控制功能，这是一种基于接受者反馈的网络传输QoS监测机制，在RTCP的接收报告中包含了当前网络传输QoS有关信息，如报文丢失率、平均时延等，发送者可以通过这些信息监测和评价网络传输QoS状况，并采取适当的策略实施同步。而基于SDN的发布订阅系统，提供了多队列传输、路由计算等功能，可以在不同队列中分别传输RTCP、RTP信息，同时根据RTCP中得到的反馈，重新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。RTP包格式如图2所示。



其中比较重要的几个域及其意义为：1.CC：表示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；2.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；3.sequence number：序列号，用来为接收方提供探测数据丢失的方法；4.timestamp：时间戳，记录了负载中第一个字节的采样时间。

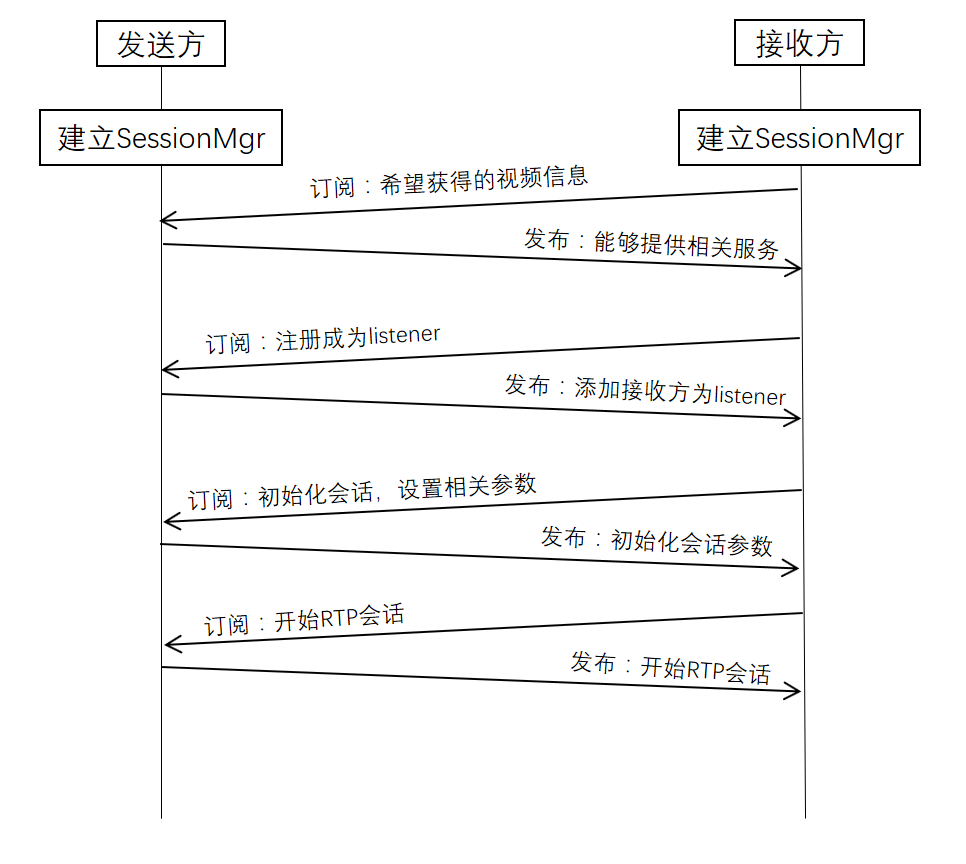
1. 实现原理

其中，建立会话流程细化为建立会话管理机制，注册成为listener，初始化会话和开始会话几个步骤，具体流程如图4所示。



1. 交互流程

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表示如下。

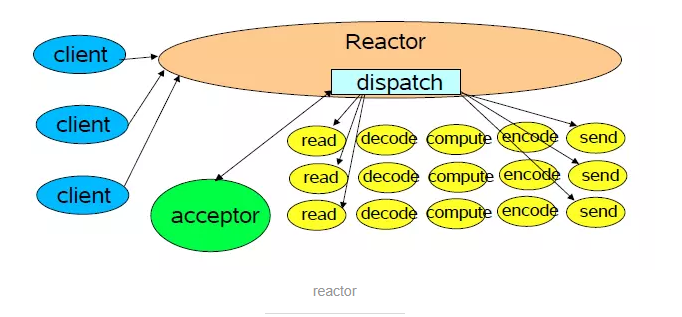


接收方首先订阅希望获得的视频信息，发送方通过发布订阅系统获知该订阅，且自身满足服务需求，则发布能够提供相关服务的信息；接收方需要注册在发布者的SessionMgr上成为listener，订阅相应事件，发送方获知订阅后添加成功，并发布该事件；接收方需要协商视频传输的大小、格式、编码、事件戳等具体协议信息，发送方发布初始化的会话参数；发送方确认相关参数后，同意开始RTP会话，发送方发布开始会话事件。在实际传输过程中，接收方订阅相应主题，发送方发布匹配的主题，而系统会将主题编码，作为流表的匹配项下发至

交换机，然后在网络中传输。

1. Reactor模式

reactor设计模式是事件驱动编程的一种实现方式，处理多个客户端并发的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。目前，许多流行的开源框架都用到了reactor模式，如：netty、node.js等，包括java的nio。



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，并触发分派器取出消息进行分发处理，在发布订阅系统中，适用于高并发、大数据量的情况，经过测试，Reactor模式极大地提升了系统的QoS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保障，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QoS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。

1. 需求分析

本文所实现的基于服务容器的服务语义重构系统的目的是通过读取虚拟化

平台中运行的物联网Java 服务内存，还原出Java 服务的函数栈帧，分析函数调

用轨迹和函数调用参数，构造内存事件为物联网运行时验证系统提供支持。本系

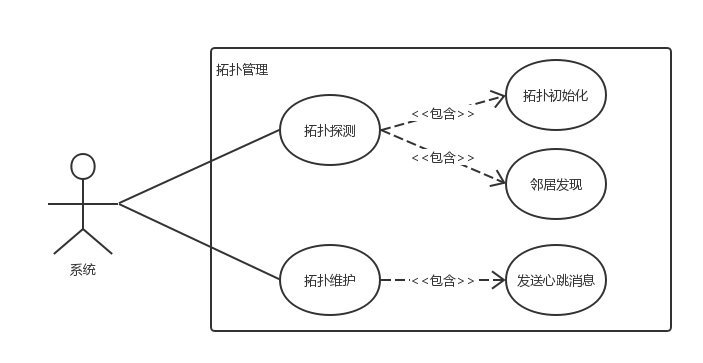
统通过界面操作和显示不仅可以配置要监控的服务，而且可以实时查看函数调用

和内存事件相关信息。根据本系统所要实现的功能和物联网运行环境对本系统的

要求，确定以下功能性需求和非功能性需求。

1. 功能需求
2. 拓扑管理

拓扑管理是本系统的最基础功能，通过拓扑管理，集群控制器能够感知邻居集群的存在，进一步构建全网链路信息。为实现拓扑管理需求，需要完成拓扑探测、拓扑维护功能，拓扑管理的用例图如下所示。



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，用例详细情况见下表。

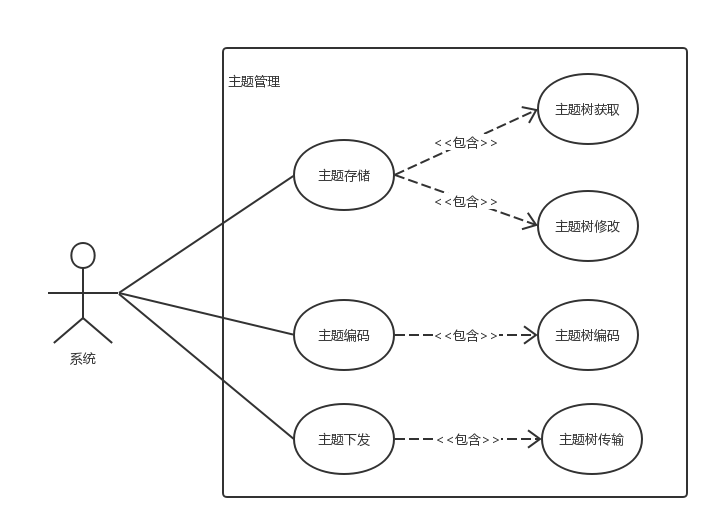
|  |  |
| --- | --- |
| 参与者 | 系统 |
| 用例描述 | 拓扑探测 |
| 前置条件 | 系统启动，获取本集群配置信息 |
| 详细描述 | 1、根据配置信息，获取集群名、集群交换机地址等信息  2、下发初始化流表  3、向所有对外端口发送探测消息 |
| 后置条件 | 系统得到邻居集群的消息反馈，拓扑探测完成 |

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，防止因故障导致的系统失效，用例详细情况见下表。

|  |  |
| --- | --- |
| 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. 将邻居集群的详细信息保存在内存中 2. 定时向邻居集群发送心跳消息 3. 若超时未回复，则认为邻居已经失效，将该集群删除 |
| 后置条件 | 系统维护了全网拓扑，并将失效集群删除 |

1. 主题管理

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来进行匹配，因此系统中需要对主题进行管理。为实现主题管理需求，需完成主题树存储、主题树编码、主题树下发等功能，主题管理的用例图如下所示。



主题树存储，需要通过接口获取主题树，如果主题树发生了修改，就将修改后的主题树内容写回持久化文件中，用例详细情况见下表。

|  |  |
| --- | --- |
| 参与者 | 系统 |
| 用例描述 | 主题树存储 |
| 前置条件 | 系统启动，调用主题树获取接口 |
| 详细描述 | 1. 用户通过接口将主题树内容读取到内存中 2. 对主题树进行修改 3. 将修改后的主题树内容写回到持久化文件中 |
| 后置条件 | 修改后，内存、持久化存储中都进行了更新 |

主题树编码，通过相应的编码方式，将每个主题树节点编码成ipv6地址的形式，用例详细情况见下表。

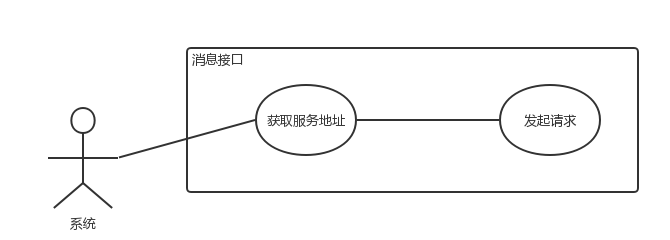
|  |  |
| --- | --- |
| 参与者 | 系统 |
| 用例描述 | 主题树编码 |
| 前置条件 | 系统将主题树内容读取到内存中 |
| 详细描述 | 1. 调用编码方式，将每个节点编码成为ipv6地址 2. 将每个编码主题连接在一起，封装为编码主题树 |
| 后置条件 | 经过编码，系统获得编码主题树 |

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。

|  |  |
| --- | --- |
| 参与者 | 系统 |
| 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信息 |

1. 消息接口

本系统为发布者、订阅者提供的服务调用接口，是基于SOAP的web service方法，这是一种XML对象访问协议。为了提高转发校历，系统将服务抽象为一种资源，向用户暴露服务接口，从而提升系统的性能和可扩展性。消息接口的用例图如下所示。



用户通过接口，向系统暴露出的服务地址发起服务调用请求，这样就能获取到系统提供的服务，用例详细情况见下表。

|  |  |
| --- | --- |
| 参与者 | 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. 用户获取服务地址 2. 向服务地址发起服务调用请求 |
| 后置条件 | 用户可以得到系统提供的服务 |

1. 路由计算

经过拓扑管理，可以得到全网拓扑信息，但是还需要进一步转换为路由信息，针对管理路径和主题转发路径，系统提供了不同的策略和算法。路由计算的用例图如下所示。

|  |  |
| --- | --- |
| 参与者 | 系统 |
| 用例描述 |  |
| 前置条件 |  |
| 详细描述 |  |
| 后置条件 |  |

1. 流表管理

|  |  |
| --- | --- |
| 参与者 | 系统 |
| 用例描述 |  |
| 前置条件 |  |
| 详细描述 |  |
| 后置条件 |  |

|  |  |
| --- | --- |
| 参与者 | 系统 |
| 用例描述 |  |
| 前置条件 |  |
| 详细描述 |  |
| 后置条件 |  |

|  |  |
| --- | --- |
| 参与者 | 系统 |
| 用例描述 |  |
| 前置条件 |  |
| 详细描述 |  |
| 后置条件 |  |

1. 多媒体传输

|  |  |
| --- | --- |
| 参与者 | 系统 |
| 用例描述 |  |
| 前置条件 |  |
| 详细描述 |  |
| 后置条件 |  |

1. 性能需求
2. 系统性能保障

|  |  |
| --- | --- |
| 参与者 | 系统 |
| 用例描述 |  |
| 前置条件 |  |
| 详细描述 |  |
| 后置条件 |  |

1. 用户协商机制

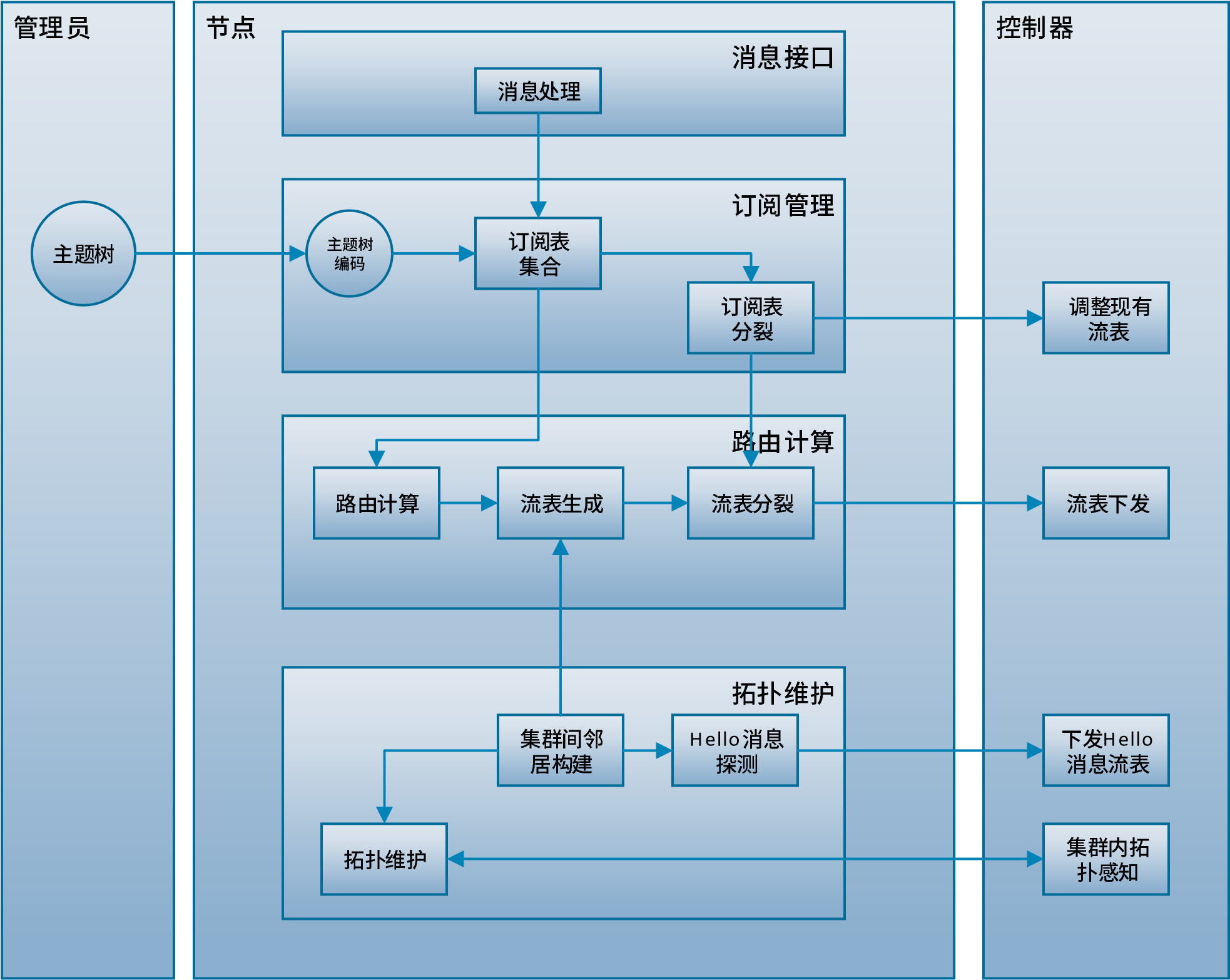
|  |  |
| --- | --- |
| 参与者 | 系统 |
| 用例描述 |  |
| 前置条件 |  |
| 详细描述 |  |
| 后置条件 |  |

1. 系统概要设计

系统架构是系统功能的直观体现，通过架构分析，读者可以对系统的核心模块、消息分类、各模块具体作用有一定了解。本章将对系统各模块的设计思路进行详细阐述。

1. 系统架构

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，本系统主要由控制器、节点、管理员三个部分组成。下面就开始对整个系统的架构设计进行介绍。系统架构图如下所示。



从架构图中可以看出，系统包含三个主要角色：管理员、节点、控制器。每个控制器管辖的范围称为集群，集群内部包含多台SDN交换机，控制器承担了集群内的发布订阅事务处理，包括拓扑探测、路由计算、流表下发与维护等；管理员在控制器职责功能外，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；节点层负责联系用户与控制器，一台电脑上只有一个wsn，但可以有多个用户的存在。

以一次用户发布订阅过程为例，当有新的订阅节点接入SDN网络中后，首先需要通过拓扑维护模块进行拓扑感知，从而获知全网拓扑结构，之后可以通过消息接口发起订阅操作，订阅信息由订阅模块统一管理；发布操作的初始化过程相似，订阅管理模块在更新发布信息后，会调用路由模块计算发布节点和订阅节点之间的组播树路径，并下发相应流表，这样发布消息时，数据沿既定路径直接传输即可。

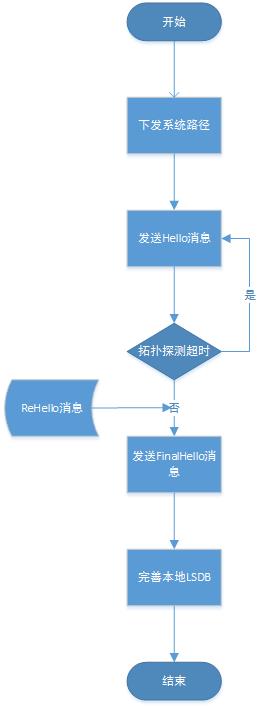
1. 基于SDN的发布订阅系统架构设计

本系统的研究重点是发布订阅系统与物理环境下的SDN交换机的结合，已有的研究成果大多是模拟环境下，使用开源控制器opendaylight测试发布订阅系统的性能，本文需要自行开发控制器、管理员系统，并结合物理交换机进行验证，相比原方法更加真实可信。下面针对发布订阅系统与SDN交换机的架构设计展开论述。

1. 拓扑管理

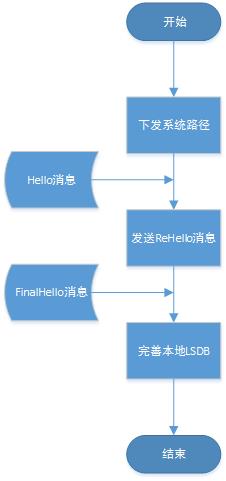
已有的发布订阅系统中，由于虚拟控制器的存在，用户通过OpenDayLight提供的API，可以方便的获得全网的拓扑连接情况，但在物理环境中，一个新集群的加入无法主动感知，因此需要我们指定一种拓扑探测、维护的方式，本系统采取的是开放最短路径优先（OSPF）协议。

关于该协议的基本概念、状态转移等内容在第二章中已经进行了介绍，这里就不再赘述。在具体实现时，分为时间驱动和事件驱动，时间驱动的拓扑探测流程图如下所示。



当集群控制器开始启动后，会触发定时的拓扑感知操作，即通过下发系统路径，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，当邻居集群收到消息后，经过分析处理，再将邻居集群的信息封装在ReHello消息中，反向传输给初始集群，最终在两个集群间建立双向连接，完善了本地LSDB数据库；若拓扑感知时间超过探测阈值，则无需处理。上述流程在系统中设置为定时任务。

系统还存在另一种拓扑探测方式，基于事件驱动的拓扑探测，流程图如下所示。

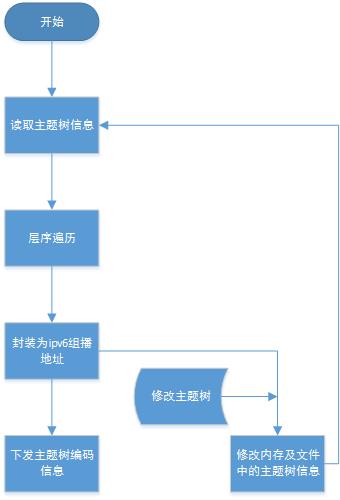


当集群控制器稳定运行后，若全网拓扑新增集群，则新增集群控制器会向所有对外端口发送Hello消息进行探测，此时集群收到Hello消息后，知道存在邻居集群，就会反向发送ReHello消息，通知邻居集群自身的存在，从而得到FinalHello消息，最终三次握手完成后，完成本地LSDB数据库。

基于时间和事件驱动的拓扑探测方式，使得集群控制器能够快速发现邻居，同时允许全网动态添加新的集群，保障了系统的稳定运行。

1. 主题树管理

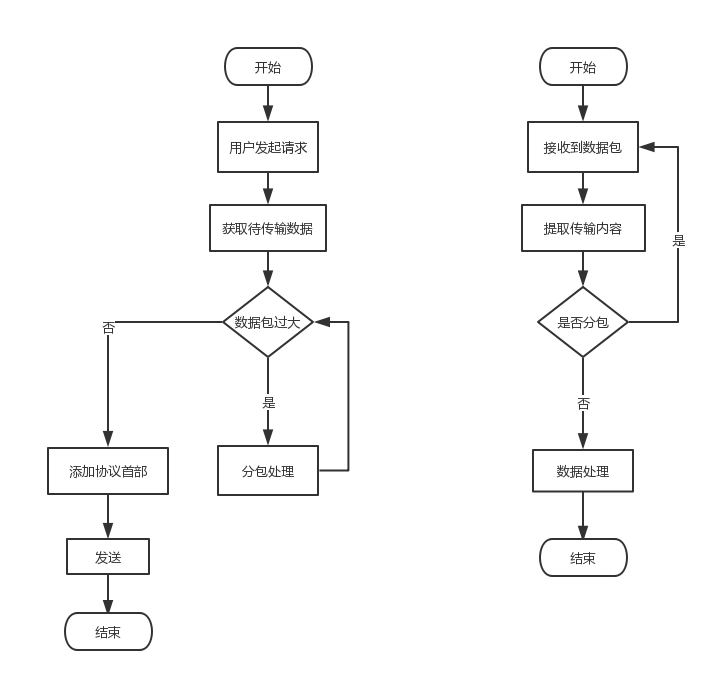
发布订阅系统中采用主题分类的方式匹配用户订阅、发布操作，一方面，管理员处采用ldap轻量目录访问协议存储主题信息，当发现新的控制器集群加入时下发管理的主题树信息；另一方面，基于 SDN 网络的新系统采取了 IPv6 组播的形式进行消息传输，将数据包头中的 ipv6\_dst 字段作为自定义的匹配项。发送者向相应主题对应的 IPv6 地址进行 IPv6 组播，接收者加入 IPv6 组播的群组，接收发来的相应主题数据包。这样可以实现主题到地址的转换。主题树在系统中的编码流程图如下所示。



管理员处可以主动获取主题树信息，并通过相应的编码方式转换为ipv6组播地址，若管理员通过ui界面或调用API接口手动修改主题树，则需要将修改后的主题树内容写回到持久化文件中，并能够再次调用主题树编码模块进行编码，将主题树编码信息下发给控制器。具体编码实现方式见第五章相关部分。

1. 消息接口

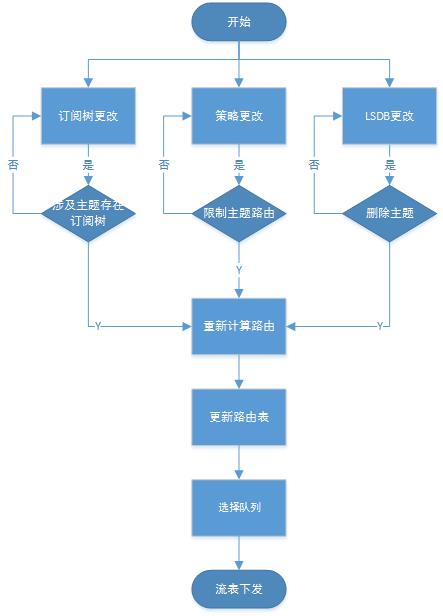
用户与WSN层之间采用的是web service传输协议。WebService是一种跨编程语言和跨操作系统平台的远程调用技术，WebService通过HTTP协议发送请求和接收结果时，发送的请求内容和结果内容都采用XML格式封装，并增加了一些特定的HTTP消息头，以说明 HTTP消息的内容格式，这些特定的HTTP消息头和XML内容格式就是SOAP协议。SOAP提供了标准的RPC方法来调用Web Service，简单来说：SOAP协议 = HTTP协议 + XML数据格式，在使用http发送之前需要进行协议的封装。用户与WSN之间具体交互流程图如下所示。



由上图分析可知，发送方获取待传输数据后，首先需要判断数据包大小，若超过传输限制，则需要进行分包处理，直至每个数据包都满足传输要求，接着需要使用协议首部进行封装，然后调用传输模块发送给系统；接收方通过监听接收地址，可以得到相关消息，经过协议解析，提取传输的内容，此时需要判断该数据包是否经过分包处理，若后续还有数据包，则重复上述接收处理流程，直至完整的数据包接收完成，再进行下一步处理。

1. 路由算法

当系统进行拓扑探测后，各个集群对周围邻居都有了LSDB保存相关邻接信息。此时集群内发起发布或订阅请求，各个集群需要结合当前全网拓扑情况分别进行计算，求解出针对特定主题的订阅树。这里使用的是路由算法，包括Dijkstra最短路径算法和Steiner多元最小生成树算法。路由算法的调用流程图如下所示。



控制器启动后，需要接受来自管理员的主题树编码内容，才可以将用户的发布订阅信息转换为拓扑节点格式，触发路由计算的条件包含订阅树修改、策略更改、LSDB更改等，这三个条件分别表示：订阅树修改，用户的订阅信息发生改变，即集群内的发布订阅关系更改，显然广播该消息并重新计算路由；策略更改，管理员处下发了关于转发的限制条件，针对特定主题，某些集群不能用作转发节点，因此需要重新计算路由；LSDB更改，此时全网拓扑发生变化，根据Dijkstra算法计算出的最短路径可能改变，因此也需要重新计算路由。

本系统采用的路由算法包括Dijkstra最短路径算法和Steiner多元最小生成树算法，在实际使过程中需要满足稳定性和可靠性，这在测试模块进行了相应的描述。

1. 流表管理

经过路由计算，可以求解出针对特定主题的订阅树，同时结合全网拓扑，也就能得到转发路径，由于拓扑部分采用的是OSPF协议，邻居集群间维护的LSDB完全一致，因此各个集群单独计算出的路径结果相同，此时通过判断当前集群是否在路径中，可以根据进出端口、组播地址封装出对应的流表。

该部分除计算出流表外，还负责获取交换机数据，维护交换机、队列信息，并根据路由、主题编码，下发对应的流表，具体流程图如下所示。

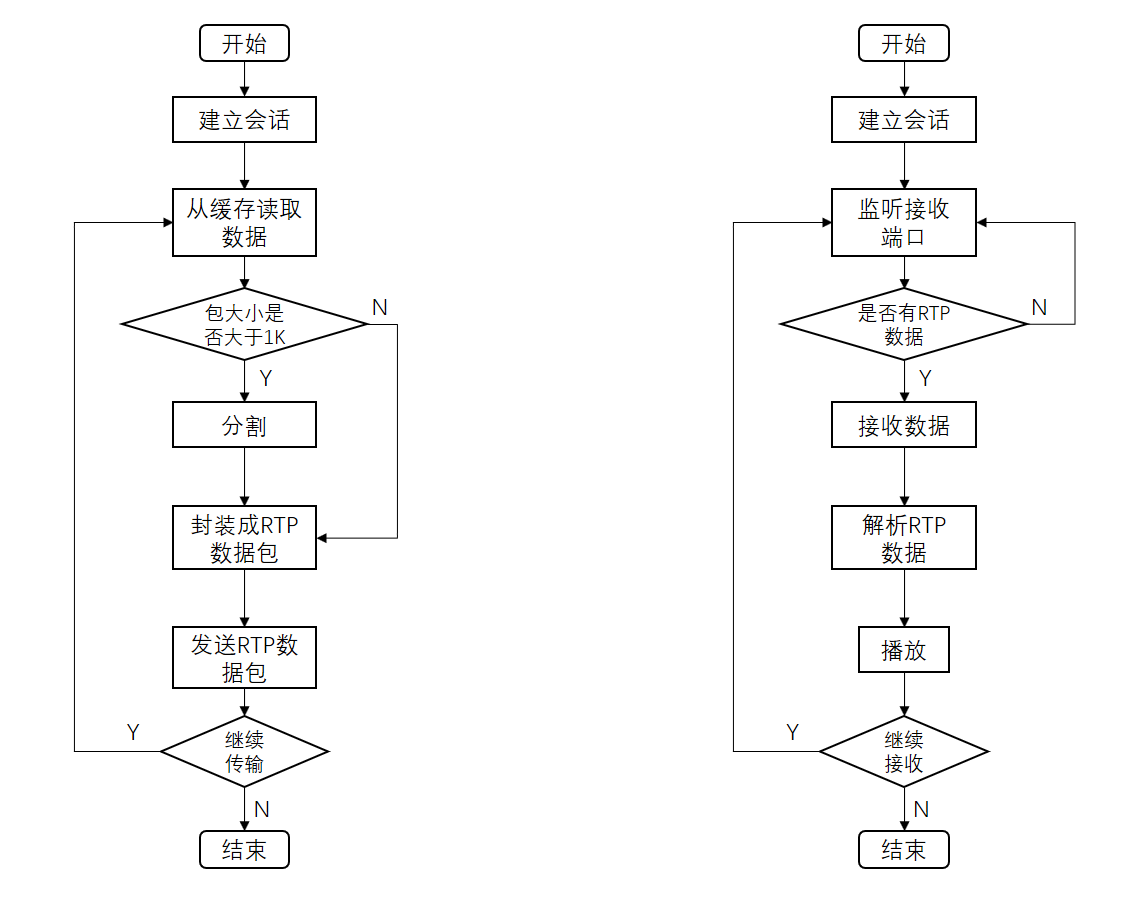


控制器系统与SDN交换机采用的是ssh连接方式，通过jsch软件包可以指定交换机内部系统的ip地址、用户名、密码，访问到交换机的内部系统，此时指定执行语句即可获取控制器信息。从流程图中可以看出，控制器需要定时获取交换机信息，包括端口流量、队列进出字节数等，封装在集群信息中上报给管理员，同时根据路由算法、全网拓扑和主题树编码计算出相应的转发流表，并下发给交换机实现物理转发功能。

1. 多媒体传输机制

发布订阅系统的最终目的是实现消息的解耦，而在本系统中，消息的种类繁多，既包括用于与管理员通信的管理消息，也包括发布订阅的主题消息，还包括与wsn层通信的系统消息。

这里是借鉴RTP实时传输协议，对视频数据进行了封装，同时可以根据数据包序号提供相应的QoS保障。根据RTP协议编写的发送方、接收方的处理流程如下图所示。



发送方在会话连接建立后，需要使用字节流的方式，依次从缓存中读取待传输数据，若数据包过大（超过1k），则人为切割，并填充RTP数据包首部，生成大小为1k的RTP数据包，然后遵循传输协议，向目标接收地址依次传输。

接收方首先需要构建与发送方的连接，通过监听接受端口，当有RTP数据包到来时，触发解析操作，这里可以使用相应的jar包将字节流转换为图片image，最终以数据帧的形式展现在用户界面上。

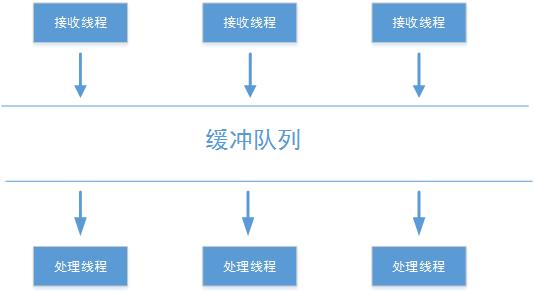
1. 消息传输的QoS保障方案

为了防止突发情况，已有的系统中提供了包括单线程、线程池两种方式，经过测试验证，这两种方案在某些程度上存在不足，因此本系统提供了新的基于Reactor模式的传输方式，以及用户协商方式，下面分别进行阐述。

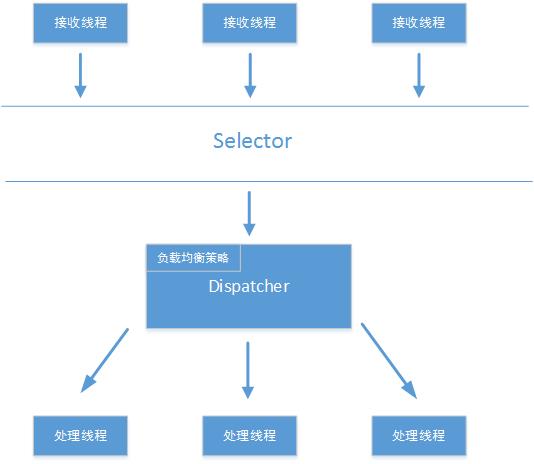
1. Reactor模式

已有系统中，

这里系统在设计之初，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，生产者-消费者的运行示意图如下所示。



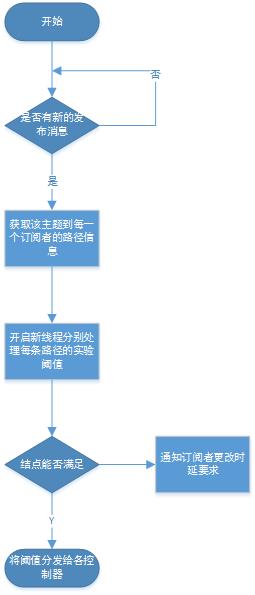
当接收到订阅信息时，生产者将消息放在缓冲队列中，同时触发处理线程的处理功能，将特定主题的消息从队列中取出，并转发给订阅该主题的用户。这有两个好处，一方面，这是基于队列实现，使用队列缓存到来的消息，能够有效降低丢包率，另一方面，解决了生产者、消费者的处理速度不同步问题，但是在实际使用过程中，我们发现虽然存在队列，但一方面队列的大小有限，不可能一直缓存，另一方面，我们无法控制队列，不能对其中的消息采取负载均衡策略，因此，在此基础上，本系统最终使用了Reactor模式来解决上述问题。Reactor模式的运行示意图如下所示。



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。

1. 用户协商模式

作为一款多媒体传输软件，除了基本的视频传输提供外，还需要提供传输质量保障方案。本系统在Reactor模式的传输保障上，还提供了用户协商机制，用户输入时延、丢包率请求，系统得到后根据用户需求动态调整，用户协商的流程图如下所示。



从图中可以看到，用户可以对当前订阅的特定主题，发起时延、丢包率等请求，经过wsn汇总，发送给集群控制器，本集群控制器再向管理员提起需求，经过处理后由管理员将结果下发，若可以满足，则控制器根据计算的结果动态调整带宽等参数，若无法满足，则处理结

果将返回给用户，由用户自行决定是继续请求还是接受当前处理结果。相比于传统的发布订阅系统，这种用户协商机制更加具有现实意义，即系统可以根据用户请求动态调整网络带宽分配，从而满足用户需求。

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、多媒体传输的实现，以及QoS保障方案，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图，关于它们的详细设计将在第五章进行具体的介绍。

1. 系统详细设计与实现

第四章完成了本系统的系统设计内容。根据第四章中各个模块的设计原理和

工作流程，完成系统实现。本章将详细介绍基于服务容器的服务语义重构系统是

如何实现的。

1. 工程概览

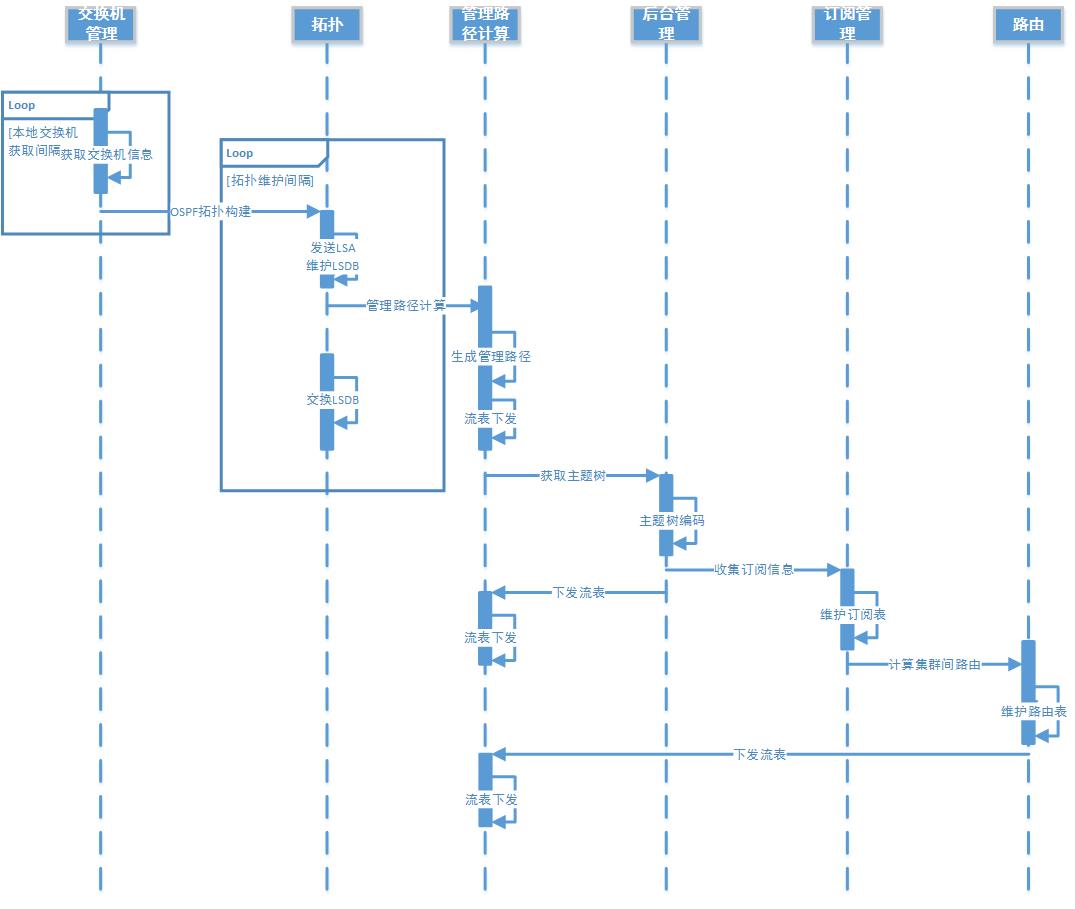
由于本系统功能较为复杂，因此在设计之初，就采用划分模块的方式，单独编写。通过参与角色划分为控制器、管理员、wsn层、用户等模块，又根据各模块的主要功能划分为ui展示、队列、路由、交换机、主题树、拓扑等模块，模块之间彼此独立，又通过预留接口的方式相互关联，这样不论是前期编码还是后期维护都带来了便利。系统结构图如下所示。

图

除此之外，本系统还具备了QoS方案和日志系统功能，为系统的可靠性提供保障。下面分别阐述各个模块的功能。

1. 参与角色
2. 控制器

集群控制器负责处理集群内的发布订阅情况、接收来自管理员的管理消息，并直接与wsn层进行交互，控制器时序图如下所示。

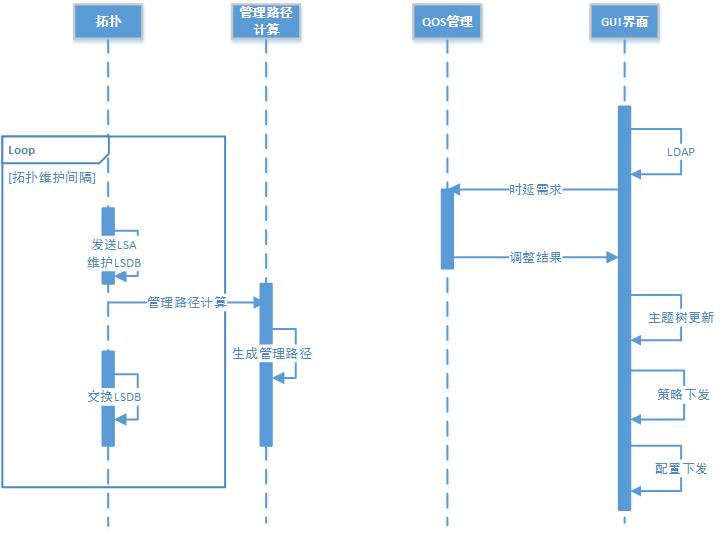


从图中可以看出，当集群控制器启动后，首先会下发初始化流表，并根据OSPF协议进行拓扑探测，对收集到的邻居信息保存至本地LSDB数据库中；若本集群存在发布订阅信息，则分析wsn层发来的用户请求，广播本地集群的新增发布订阅消息，由于全网拓扑一致，经过各个集群的单独计算，根据计算结果得到组播树，生成转发路径并下发相应的流表。

1. 管理员

管理员的基本职责与控制器相同，除拓扑探测外，管理员还需要获取主题树消息并进行广播，系统提供了LDAP数据库和xml文本两种方式对主题树数据进行持久化操作，管理员在获得主题树的全部信息后，根据下发的管理路径可以将主题树发送给全网。

同时管理员还需要收集集群控制器上报的用户请求，根据全网情况，计算出是否能够满足用户需求。管理员时序图如下所示。



从图中可以看出，管理员模块在控制器职责的基础上，还具备管理路径的计算、QoS管理、GUI界面展示等功能。

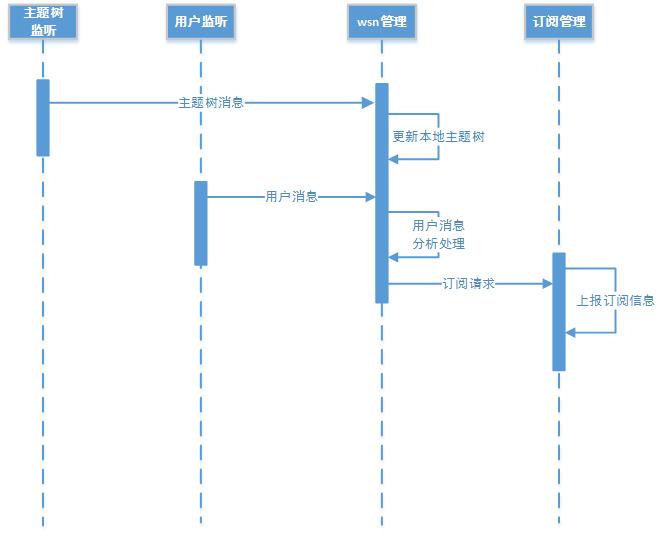
1. 管理路径的计算是在拓扑管理的基础上，当系统发现LSDB中已经存在管理员角色时，会以管理员为跟几点，主动计算出一个完全非连通图，所有已知节点都能通过管理路径访问到。
2. 管理员ui模块负责管理员界面的展示，运行效果图如下所示。



从图中可以看出，GUI管理界面 了集群基本信息，包括所有集群列表，各集群成员，集群订阅信息，各集群配置信息等，其中各集群的成员信息，订阅信息是通过向集群代表查询得到。在界面上还可以进行相应的操作，比如修改Hello消息失效阈值、扫描周期等，修改后的数据将由管理员通过管理路径，下发给各个控制器。

1. QoS管理模块，用户提出时延、丢包率请求，经过分析处理后，若可以满足，则管理员将计算后的网络带宽分配结果下发给控制器，由控制器重新下发流表。
2. Wsn层

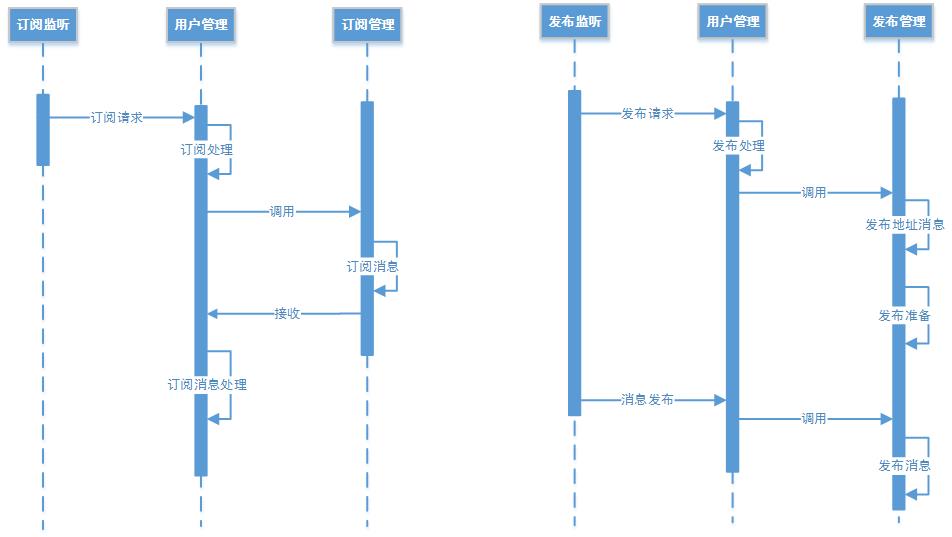
Wsn层是中介层，用于承接用户和控制器（管理员），用户发起的所有发布订阅请求、时延需求都是先由wsn收集，再统一上报给控制器，在wsn和控制器间采用的是web service的传输方式。Wsn层时序图如下所示。



从图中可以看到，主题树监听模块负责接收来自控制器的主题树编码信息，用户监听模块负责接收来自用户的消息，若经过处理，得到的是用户订阅请求，则wsn层将代替用户想控制器发起订阅请求。

1. 用户

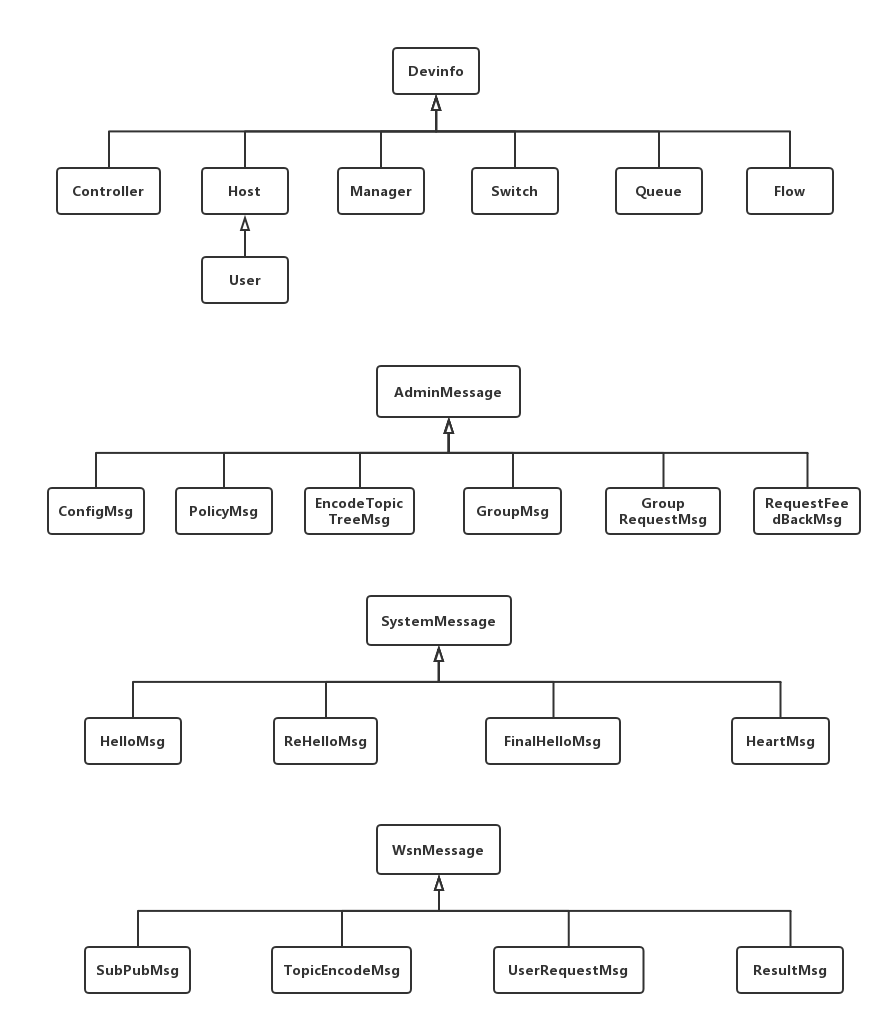
用户是本系统的实际使用者，由于发布订阅的存在，发布者和订阅者之间实现了解耦，他们不需要知道彼此的存在，只需要针对特定主题发起请求，作为纽带连接两者的是主题树编码机制、发布订阅系统、ipv6组播方式等。用户的运行时序图如下所示。



从图中可以看到，用户发起订阅请求，由用户管理模块接收并分析处理，然后调用订阅管理模块向wsn汇报，当接收到订阅主题相关的内容时就接收到本地；若用户端为发布请求，则调用发布管理模块进行注册，注册成功将得到wsn返回的发布地址，用户经过准备后，向该地址推送消息即可。

1. 消息的分类与定义

在本系统中，由于存在拓扑管理、主题管理、发布订阅管理等多个消息交互模块，因此消息的种类繁多，本系统在构建之初就详细设计了各种消息的格式，具体消息分类见下图。

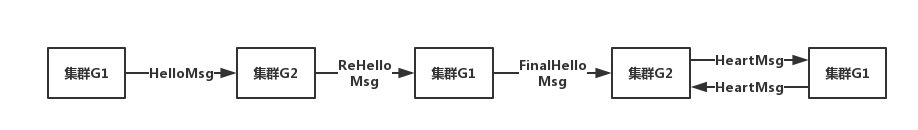


图中列出了消息的分类和继承关系，详细内容见下表。

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

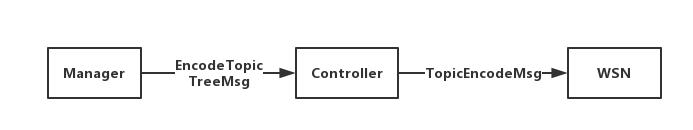
系统中存在多条通信链路，结合消息的分类，可以给出主要的消息交互流程。

1. 拓扑消息



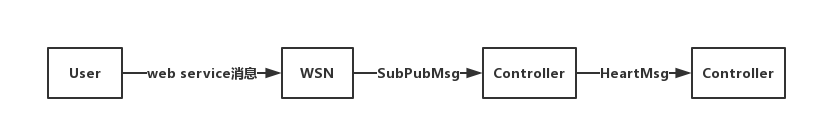
从图中可以看到，在集群G1、G2进行拓扑探测过程中，使用的是Hello、ReHello、FinalHello消息来达到三次握手的目的，拓扑维护阶段集群间交互的是HeartMsg心跳消息。

1. 主题树消息



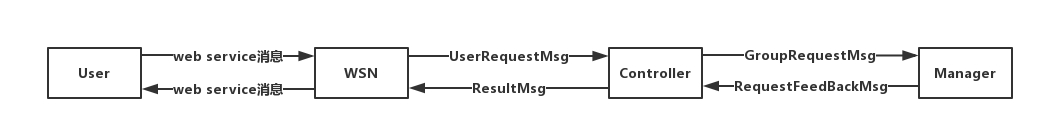
如上图所示，管理员处将编码后的主题树封装为EncodeTopicTreeMsg消息下发给各个集群控制器，控制器分析处理后，再将包含主题编码的TopicEncodeMsg发送给wsn层。

1. 发布订阅消息



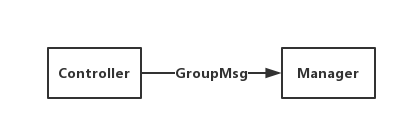
由上图分析可知，用户通过web service消息接口，向WSN层发起发布订阅请求，wsn层将用户请求封装为SubPubMsg消息发动给控制器，不同集群间通过控制器的HeartMsg心跳消息广播集群内的发布订阅情况。

1. 用户请求消息



分析上图可知，用户通过web service接口发起时延、带宽请求，WSN收到后封装为UserRequestMsg消息上传给控制器，控制器收集本集群内的用户请求，将GroupRequestMsg发送给管理员，经过分析处理后，相应的计算结果由管理员的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。

1. 集群状态消息

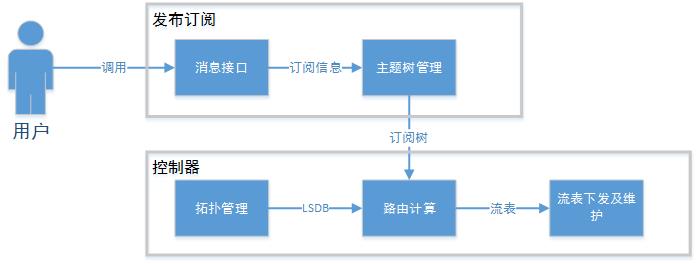


从图中可以知道，控制器定期将集群内的运行情况封装为GroupMsg消息发送给管理员。

在定义了系统消息的分类，以及主要消息交互流程后，就可以开始对系统架构进行详细设计，下面进行介绍。

1. SDN发布订阅系统架构设计

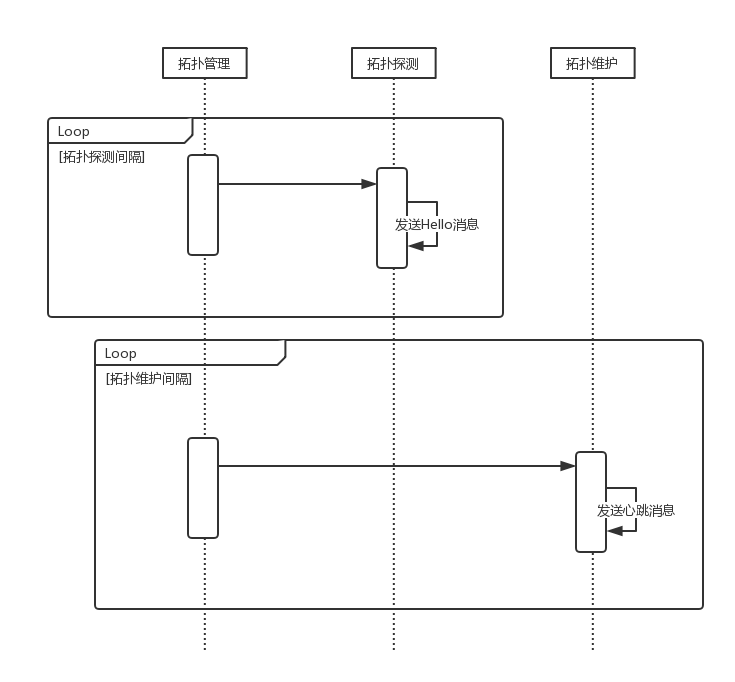
本系统的核心目的是自行开发一套SDN控制器，向上对用户开放接口，使得网络动态可编程，向下连接SDN交换机，能够生成流表并下发、获取SDN交换机的时实时运行情况，同时，系统还应该具备发布订阅功能，用户在此基础上能够针对特定主题发起多媒体传输操作。因此，要达到上述目的，基本的架构设计图为：



从图中可以看到，消息接口模块、主题树管理模块负责完成发布订阅功能，用户通过调研消息接口，可以发起针对特定主题的订阅请求，主题树管理负责将特定主题编码为ipv6组播地址，并维护用户的订阅树信息；控制器功能由拓扑管理、理由计算、流表管理这三个模块组成，拓扑管理模块负责控制器的拓扑探测，最终得到的邻居集群信息保存在本地LSDB数据库中，路由模块根据收集的用户订阅树信息，结合全网拓扑可以得到某个主题的转发路径，最终生成转发流表并下发给SDN交换机。各个模块相互协调共同完成SDN发布订阅系统功能。下面详细介绍各个模块的设计思路。

1. 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。



从图中可以看到，除事件驱动工作方式外，拓扑管理模块还包括定时处理机制，拓扑管理模块在拓扑探测间隔内调用拓扑探测功能，对外发送Hello消息；同时在拓扑维护周期中，对所有邻居集群发送心跳消息。下面分别介绍这两个子模块。

1. 拓扑探测

拓扑管理首先需要对涉及的消息进行定义，在OSPF的拓扑探测过程中，消息包括Hello、ReHello、FinalHello这三种，本质上包含的信息相似，这里以Hello消息为例，Hello消息类图如下所示。



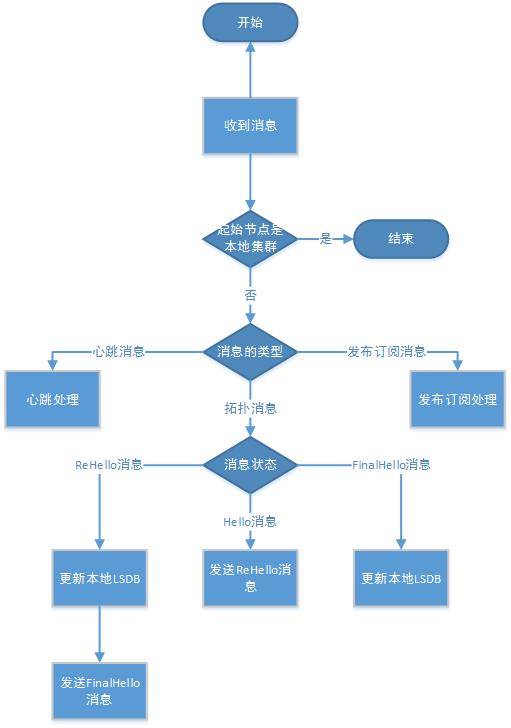
图中概括了Hello消息的消息格式，下面介绍各字段含义。

|  |  |  |
| --- | --- | --- |
| 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，下面以G1、G2集群为例，展示具体交互过程中的主要消息内容。

|  |  |  |  |
| --- | --- | --- | --- |
| G1：发送 Hello、Final\_Hello 消息，收到 ReHello 消息后在本地保存 G2 集群信息  G2：发送 ReHello 消息，收到 Final\_Hello 消息后在本地保存 G1 集群信息 | | | |
|  | Hello | ReHello | FinalHello |
| G1 | startGroup -- G1  endGroup – null  state -- down |  | startGroup -- G1  endGroup -- G2  state -- two\_way |
| G2 |  | startGroup -- G2  endGroup -- G1  state -- init |  |

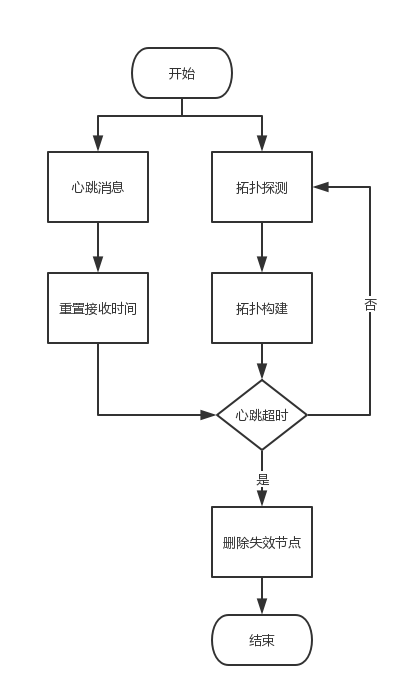
拓扑探测处理的流程图如下所示。



伪代码

1. 拓扑维护

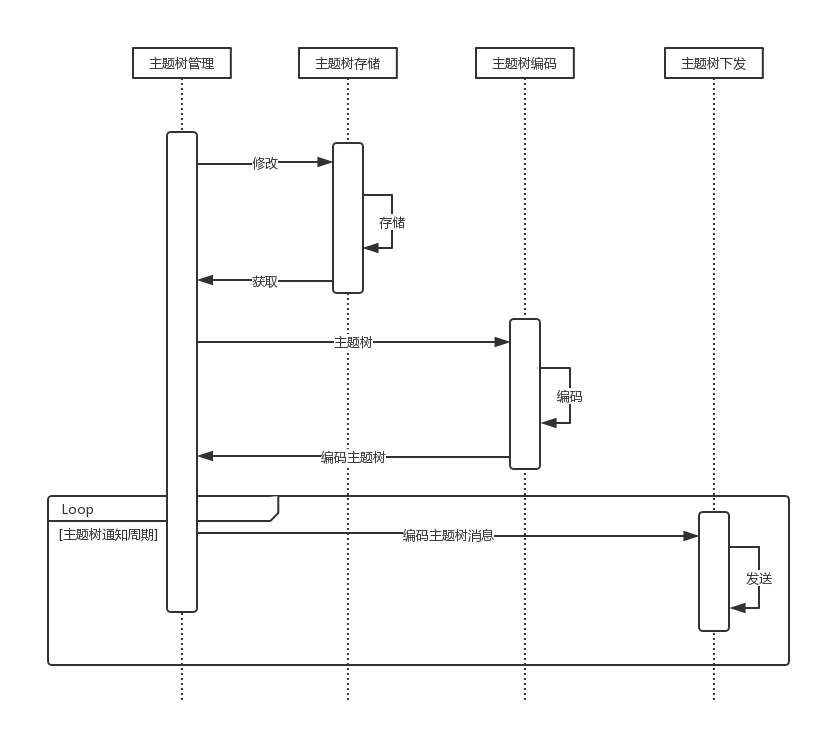
拓扑探测负责与邻居进群交互，并维护本地LSDB数据库，在邻接关系建立后，还需要后续的拓扑维护过程，本系统采用的是定时发送心跳消息的方式，拓扑维护模块具体流程见下图。



从图中可以得知，正常的拓扑构建阶段完成后，会有专门的维护线程定期判断邻居集群是否失效，若收到心跳消息，则刷新接收时间；若超时或未收到心跳消息，则认为该集群已经取消或出现重大故障，无法继续使用，因此需要删除失效节点，重新进行拓扑感知。

1. 主题树管理

主题是发布订阅系统的纽带，发布者、订阅者彼此不相识，但可以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，模块整体功能如下图所示。



从图中可以发现，主题树管理模块调用接口，可以实现岁主题树的增删改查功能，同时这些修改后的数据也将写回到持久化文件中，获取到的主题树信息通过编码子模块，得到的是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，定期将编码主题树消息发送给各个控制器。下面依次介绍这三个子模块。

1. 主题存储

主题的定义多种多样，但不同主题间存在着耦合关系，比如用户订阅的“球类运动”主题与“足球”主题间就存在父子关系，我们这样定义主题：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，多个主题间不辞独立，若订阅了父主题则默认获得所有子主题的订阅内容。从上述概念可以看出，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。

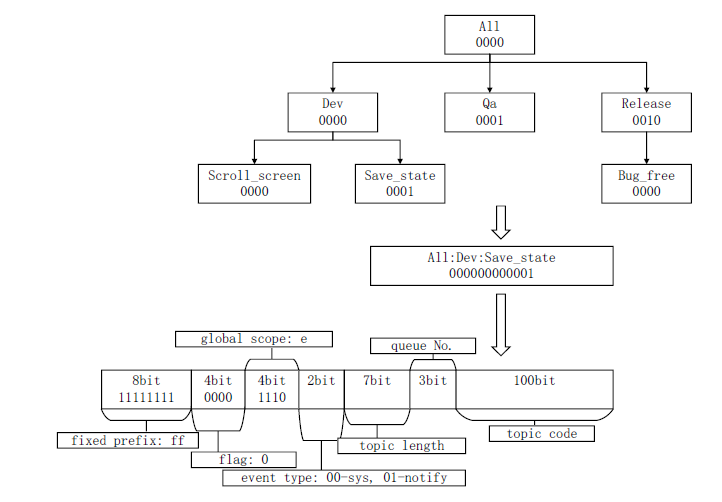


图中一共有个16主题，在虚拟“all”主题下包含“test1”、“test2”、“test3”等一级主题，各级主题递归定义，在xml中存储具有直观性。

1. 主题树编码

系统采用ipv6组播地址进行传输，IPv6 地址有128bit，因此除了固定的组播地址头部外，可以使用剩余的100bit 的空间预留给了主题树的编码，

由于主题存在依赖、包含关系，与树形结构相似，因此在内存中采用树形结构保存主题树信息。管理员通过xml方式读取文件中的主题树信息，将持久化文件中的主题树内容转换为树形结构，然后采用霍夫曼编码方式，层序遍历，依次将每层主题读取出来，根据节点的数量，使用最小比特数表示，即当该层存在n个主题节点，则使用log2n个比特位表示，依次为0000-> 0001 -> 0010 等，将每一层的编码拼接起来，并在头部封装ipv6组播地址，这样我们就可以得到主题树编码对应的ipv6组播地址。主题树编码流程见下图。



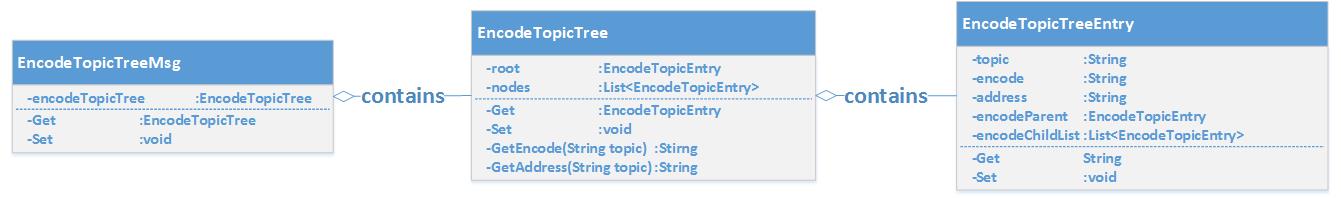
具体的伪代码如下图所示。

伪代码图

图中，在获取到所有主题信息后，需要从根节点开始，使用层序遍历的方式，依次计算出每一层的节点数目，然后采用最小比特数进行表示，这样将每层编码结果进行拼接，不足100位的用0进行填充，再加上ipv6组播地址头部，就能得到最终的编码结果。

1. 主题树下发

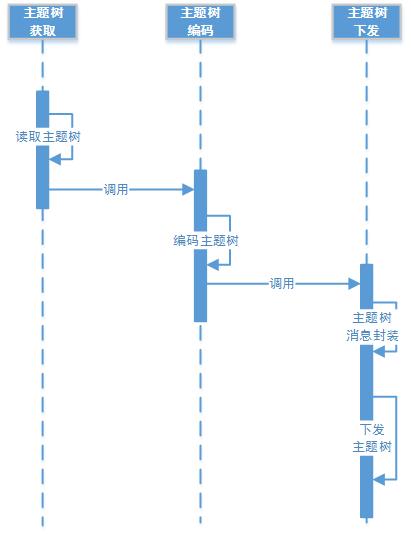
在本系统中，主题树管理功能由管理员负责，用户可以通过管理员ui界面或开放接口对主题树进行增删改查等操作，而控制器与wsn层的主题树信息是由管理员在读取主题树内容后，封装为主题树消息格式，通过管理员路径下发给各个控制器，主题树消息类定义如下所示。



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。

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

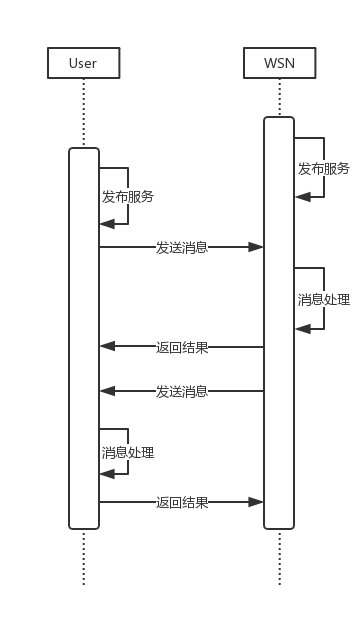
定义了消息格式后，就可以将xml文件中获取的主题消息编码，并下发给控制器，主题树下发的时序图如下所示。



通过读取xml持久化文件中的主题树信息，将所有主题转换为内存中的树形结构，经过编码、消息封装，最终调用转发模块将主题树下发到控制器。

1. 消息接口

系统在用户与WSN层之间是有的是web service通信机制，关于web service的相关知识已经进行了讲解，下面结合程序代码详细介绍，消息接口的时序图如下所示。



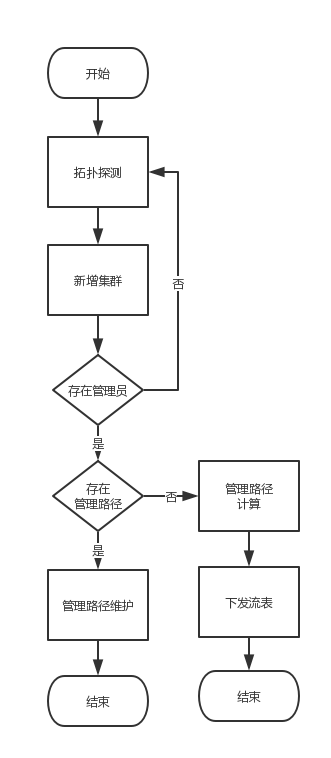
以一次通信流程为例，开始时，服务端暴露服务地址（http://192.168.10.101:9010/wsn-core），这样客户端可以通过SendWSNCommand 方法进行服务调用，本质是将自己的套接字、服务原语封装进soap协议的xml内容中，使用http post方法将数据传送给服务端，这样服务端通过监听并解析，就可以得到用户端的ip地址、监听端口等信息，这样相当于获知了用户在网络中的位置，接着向用户指定地址发送处理结果，从而形成WSN与订阅者之间的双向通信。

1. 路由计算

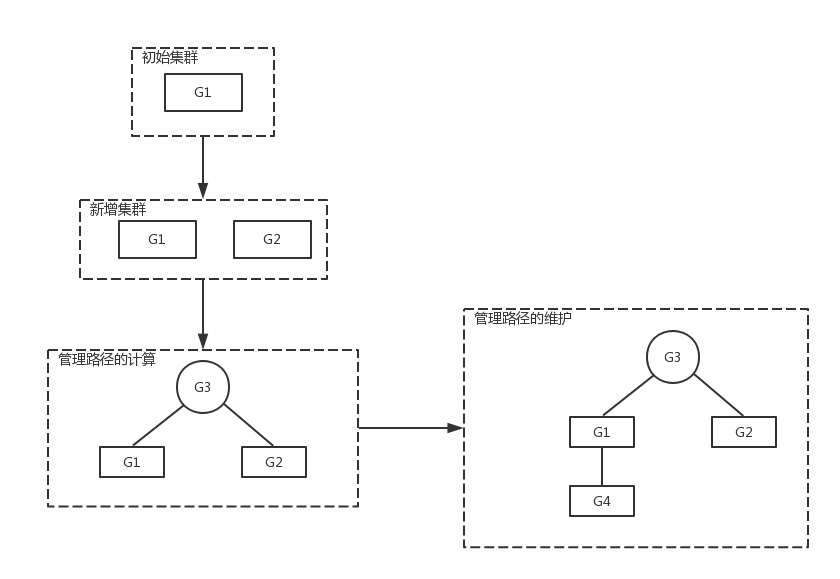
通过拓扑探测，可以获知邻居进群信息，结合LSA广播，各个集群控制器维护了全网链路状态，并保存在本地LSDB数据库中，但这些数据只是描述性内容，需要将集群信息抽象出来，表示成网络中的节点形式，这样当用户发起发布订阅请求时，就可以计算出相应的转发路径。在本系统中，路由计算功能应用于管理路径的计算、主题路径的计算等，使用了Dijkstra最短路径算法和Steiner多元最小生成树算法，下面分别进行介绍。

1. 管理路径计算

管理路径是一条以管理员为根节点的路径，分为初始计算过程和后续维护过程，计算出的管理路径结果会保存在内存中，当新增集群加入时，直接加入当前链路，具体计算流程如下图所示。



从图中可以看到，经过拓扑探测过程后，若发现新增了邻居集群，则首先会对当前网络中的节点身份进行判断，当存在管理员节点后，进行管理路径的计算并下发相应的流表；若当前网络中已经存在管理路径，则需要对新增集群调用管理路径维护方法，将其添加至当前的管理路径中，这里计算的方法就是Dijkstra最短路径算法。下面使用具体的样例来演示管理路径的计算过程。

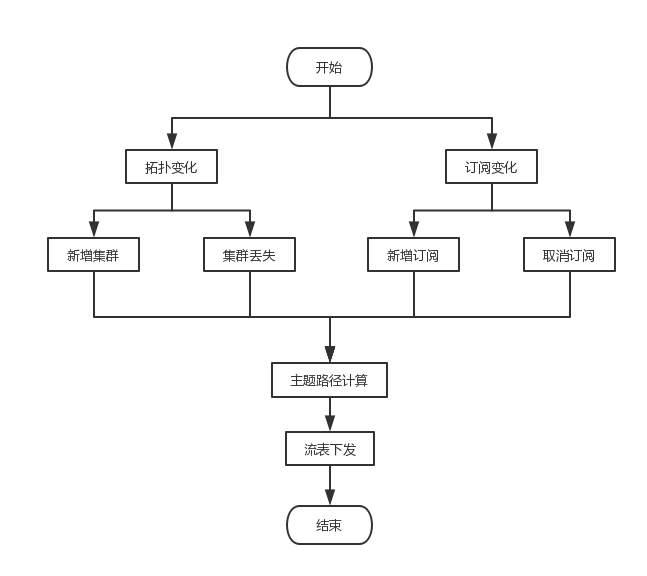


初始状态下启动G1集群，经过拓扑探测，发现了邻居集群G2，由于两个集群的身份都是控制器，因此无需计算管理路径，启动管理员节点G3，经过拓扑探测，三个集群互相交换了拓扑信息，此时全网链路中存在管理员节点，调用Dijkstra算法依次计算节点与管理员节点间的最短路径，并将最短路径上的节点添加进来，此后新增控制器集群G4，由于网络中已经存在管理路径，因此只需遍历当前节点，计算G4节点与其他集群间的最短路径，并将最短的结果添加进管理路径中。

由于管理员身份的特殊性和唯一性，因此管理路径本质上是一颗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网络中，与所有节点距离最短的路径，因此使用最短路径算法来实现管理路径的计算。

1. 主题路径计算

主题路径是LSDB数据库与用户订阅信息共同作用的结果，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。具体流程图如下所示。



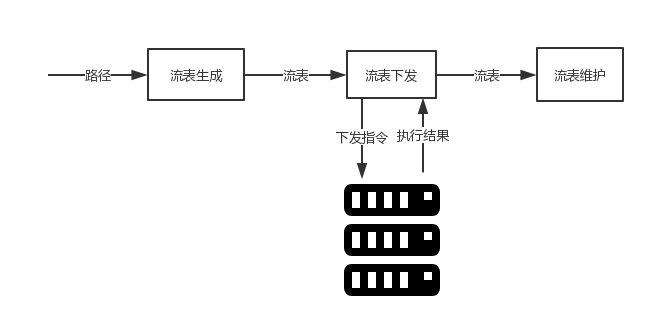
从图中可以看出，触发主题路径计算的原因包括新增集群、集群丢失、新增订阅、取消订阅等，进一步分析可以知道，这四种条件的深层原因正是主题路径计算的两个数据来源：全网拓扑信息和用户的订阅情况，因此无论是网络拓扑的变化，还是用户订阅的改变，都将重新计算对应主题的转发路径。下面使用具体样例演示主题路径计算的过程。

Steiner图

主题路径应用于发布订阅消息的传输，它的计算结果对于传输的性能有着重要作用，在内存中需要实时计算，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，在本系统中，使用了Steiner算法来实现主题路径的实时计算。

1. 流表管理

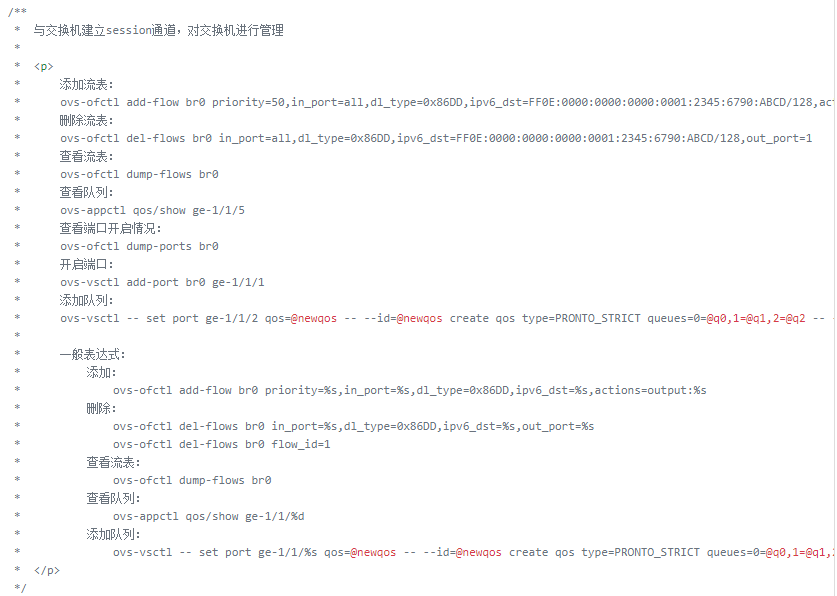
经过路由算法模块的计算后，生成了管理路径、主题路径的结果，但这是内存中网络拓扑的连接形式，还需要进一步解析成流表，这样才能结合SDN物理交换机，充分发挥SDN软件定义网络的强大功能。该部分具体划分为流表的生成、流表的下发、流表的维护，模块整体的功能如下图所示。



从图中可以看出，经过路由计算模块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

交换机识别的是流表的形式，路由计算的结果是拓扑链路的数据，因此就需要由该模块在两者之间进行转换。关于流表的概念在第二章中已经进行了介绍，流表项的一般格式见下图。



图中给出了交换机执行指令的具体样例，以及一般表达式，根据这些情况可以编写出对应的流表转换模块，流表类的定义就是根据流表项的一般表达式，填充不同的目标字段即可，结合从路由计算中提取出来的具体转发路径、端口内容，最终得到SDN交换机识别的流表指令形式。

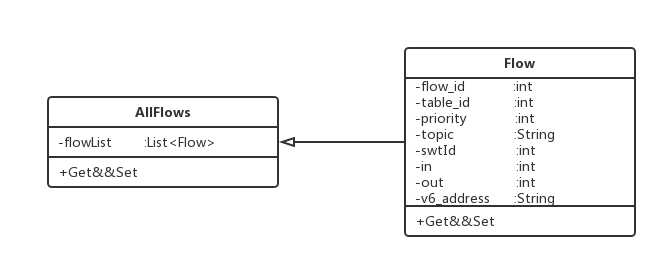
1. 流表的下发

本系统的基本功能是作为SDN控制器与SDN交换机交互，达到下发流表的目的，因此系统需要与SDN物理交换机核心处理器交互。在系统代码编写中，使用的是ssh连接方式，用网线连接控制器所在主机和SDN交换机，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，进而下发指令，完成相应的需求。流表下发具体执行结果见下图。



1. 流表的维护

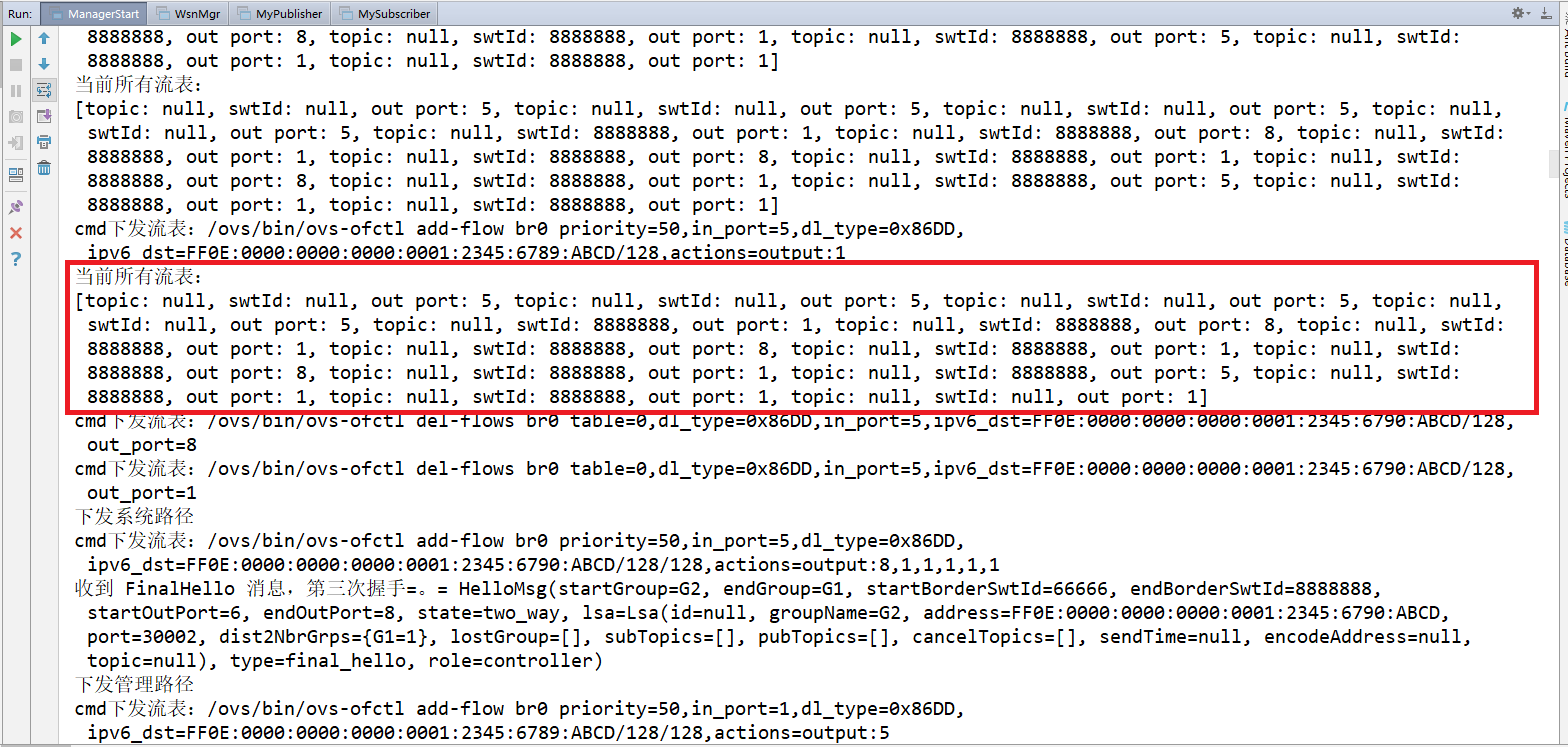
流表项在交换机中通过下发模块可以执行，但是实际操作过程中，流表可能存在重复、冲突等情况，重复的流表下发只是影响交换机处理性能，但是极端情况下会产生冲突流表，即某一流表项的目的转发地址、出端口号、优先级字段一致，区别在于进端口号，本系统采用的SDN交换机在面对冲突流表时会使用后者替代先下发的流表，这与我们真实目的相反，因此在内存中也需要有流表维护模块，对生成、下发的所有流表进行维护，流表维护的相关类图如下所示。



图中展示了流表类的消息格式，下面介绍各字段含义。

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

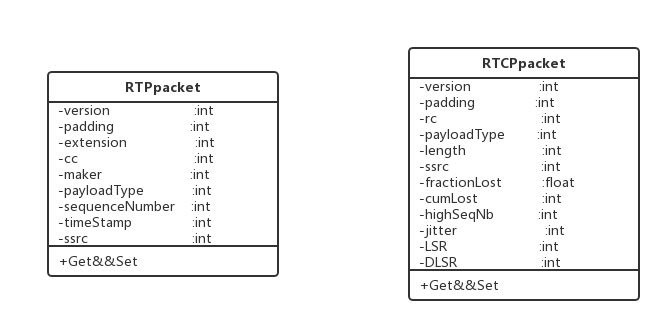
可以看到，在定义了基本的流表项（Flow）类后，内存中使用List链表的形式存储所有的流表项，这样当生成新的流表项时，可以在当前所有流表项中判断是否存在重复或者发生流表冲突，进一步通过合并操作解决冲突。经过流表维护后，内存中流表的打印信息见下图。



可以看到，经过流表维护模块的工作，SDN交换机最终能获得稳定、正确的流表。

1. 多媒体传输

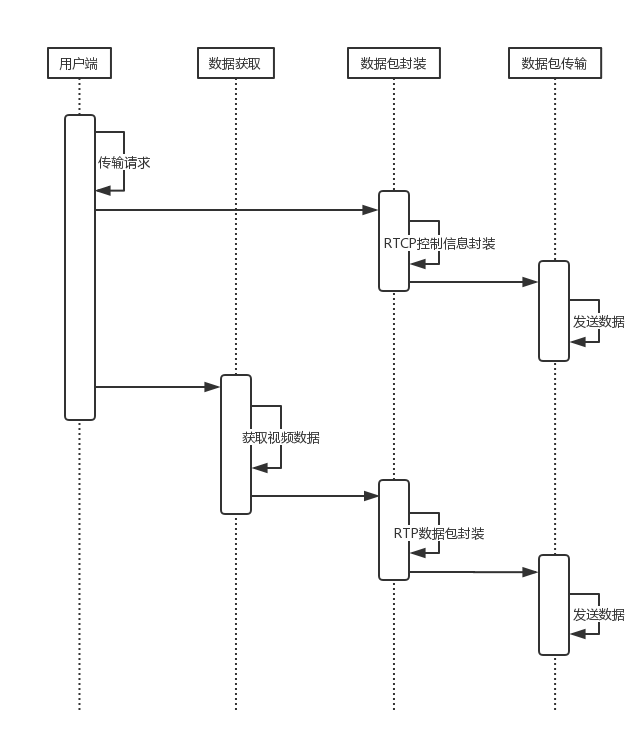
基于SDN的发布订阅系统建立后，用户可以在该系统上进行多媒体数据的传输，传输过程遵循的是RTP实时传输协议，关于该协议的基本概念已经进行了介绍，这里不再赘述，在编码过程中，首先需要针对系统使用过程中的消息格式进行定义，消息包括RTCP（实时传输控制消息）数据包，以及RTP（实时传输）数据包，具体消息类图如下所示。



图中概括了RTP、RTCP的消息格式，下面介绍各字段含义。

|  |  |  |  |
| --- | --- | --- | --- |
| 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。具体传输流程如下图所示。



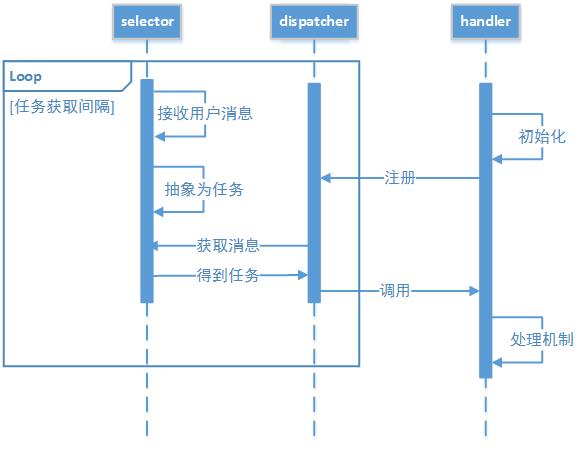
从图中可以看出，用户端发起视频传输请求，首先需要向接收方发送RTCP控制信息，包括数据流格式、RTP版本信息等，在收发双方确认了传输协议后才能进一步传输数据。数据获取模块通过字节流的方式从硬盘上获取视频内容，经过填充数据包头部、校验位后，封装为RTP数据包，这样就可以调用发送模块将视频数据传输到接收方处，接收方经过反向解析就能得到视频数据信息。

1. Qos保障

本系统的一个重要目标是提供QoS保障方案，已有代码只能够在程序设计层面提供相应的质量保证，本系统除采用Reactor模式提升了这方面的性能外，还提供了用户协商机制，在用户与管理员间构建了消息通路，使得系统更具体现实意义，下面分别进行介绍。

1. Reactor模式

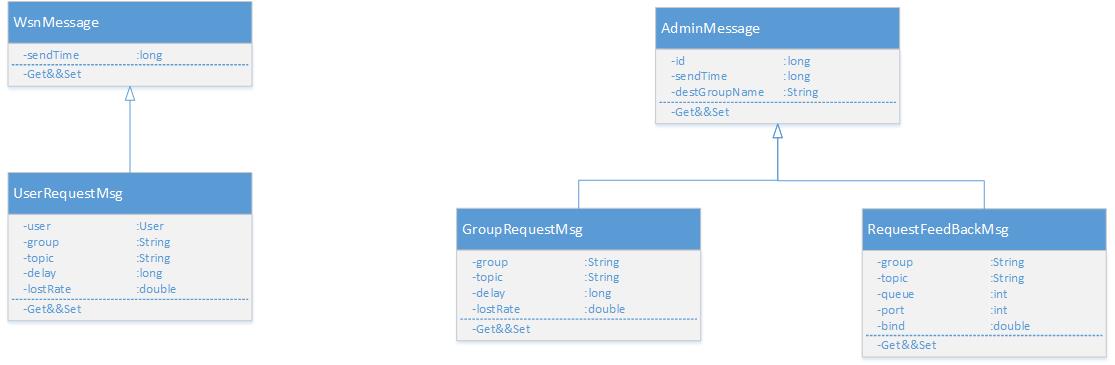
已有系统中，程序使用生产者-消费者模式提供可靠性保障，关于生产者-消费者的基本概念已经进行了介绍，这是一种协调处理速率的方法，但是还不够完善。Reactor模式是基于事件驱动，具体时序图如下所示。



从图中可以看到，Reactor模式遵循的是注册—调用机制，用户自定义handler，并将其注册到dispatcher分派器中，同时selector使用阻塞队列缓存接收到的用户消息，进一步抽象出任务task的形式保存起来，这样分派器在任务获取间隔中，不停地访问selector获取任务，并最终调用注册到dispatcher上的用户处理逻辑，完成后续的消息处理。

1. 用户协商机制

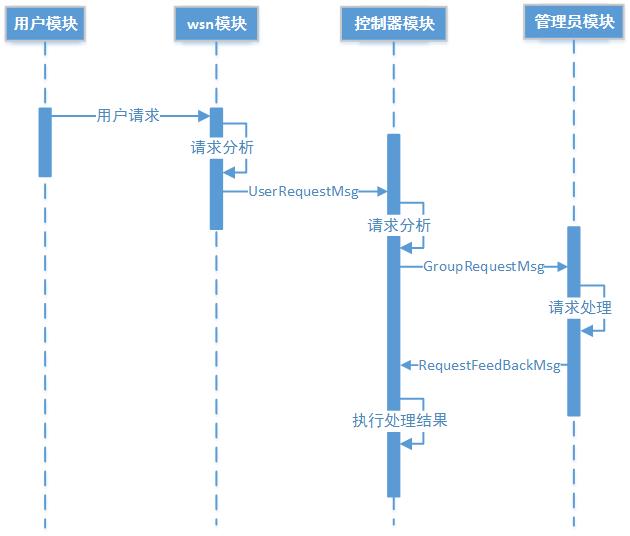
用户与系统进行协商是本系统的一个重要创新，这是因为在实际使用过程中，用户确实有这样的需求。首先需要对协商过程中出现的各种消息类型进行定义，据诶消息类图如下所示。



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。

|  |  |  |  |
| --- | --- | --- | --- |
| 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，具体实现时序图如下所示。

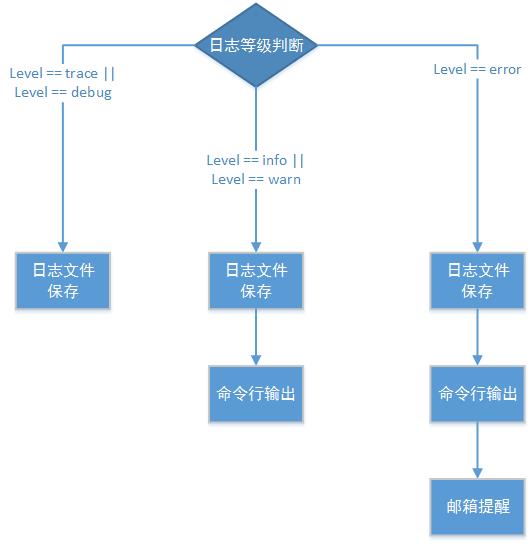


从图中可以看到用户协商的基本流程，首先由用户模块发起时延、丢包率请求，到达wsn模块后，提取相关参数，封装为UserRequestMsg消息格式发送给控制器，控制器收集并分析后，封装为GroupRequestMsg消息格式发送个管理员，在管理员处结合全网拓扑情况对用户请求进行处理，并将处理结果沿着管理路径下发给控制器，控制器收到RequestFeedBackMsg后将处理结果转换为流表并下发，用户协商过程至此完成。

1. 日志系统

已有的系统中，针对数据展示、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，还是紧迫性都需要一套新的提示机制。

本系统采取的是log4j框架，将消息的按照紧迫性划分为不同不同级别，常用的包括trace、debug、info、warn、error，紧迫性逐级递增。日志系统的通知机制如下图所示。



本系统提供了多级通知机制，普通消息标记为trace、debug会打印在日志文件中，警告消息warn输出在命令行窗口中，系统错误消息除基本展示外，还提供了邮箱提示功能。

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、SDN发布订阅系统架构、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。

1. 系统测试与验证
2. 测试目标
3. 测试环境
4. 系统功能测试
5. 性能测试
6. 本章总结
7. 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，同时尽可能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，以及性能上的QoS保障，设计的知识点颇多。从2017年12月开题，到目前过了一年半的时间，期间除了两个月的实习期外，都在完成本系统的功能，经过努力，目前功能要求全部满足，性能上也提供了相应的保障。在开题阶段，提出了关于SDN与发布订阅系统的结合，这与以往系统中的虚拟环境、开源控制器大为不同，相当于借鉴别人的思路，自己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，而是先捋清功能模块的划分及其之间的交互关系，同时，从消息定义出发，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，上述要求都得到了合理的验证。

当然，由于系统较为负责，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每一个队列运行情况，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
2. 消息持久化
3. 队列
4. RTP协议本身具有QoS保障机制，如何将两者结合使用？