测试方案

本系统的测试分为功能测试和性能测试，测试内容主要包括发布订阅系统和多媒体数据传输等模块的基本功能及运行效率。

//网络架构

## 测试目标

为保证基于SDN发布订阅系统运行稳定，我们对各个功能模块进行了单元测试和集成测试。本文主要针对发布订阅系统的设计与实现、多媒体数据传输QoS保障这两方面，因此需要对拓扑探测、路由维护、用户协商、数据传输进行详细的测试。即通过功能测试验证拓扑探测、路由维护、用户交互的可靠性，通过性能测试验证数据传输方案的优越性。

## 测试环境

//控制器/wsn/管理员环境/用户

测试环境使用3台实体交换机相互连接，每台交换机开放多个端口，通过指定local通道将发布订阅系统所在主机连接在交换机上，同时针对指定的对外端口下发队列信息。具体测试环境如表所示：

|  |  |
| --- | --- |
| 硬件环境 | |
| SDN交换机型号 | Pica-p3290 |
| 交换机数量 | 3台 |
| 支持OpenFlow版本 | OpenFlow1.0、1.3、1.4版本 |
| 支持OVS版本 | OVS 2.3版本 |
| 是否支持IPV6地址 | 是 |

## 测试内容

### 功能测试

1. 拓扑探测

测试目的：通过功能测试，验证拓扑模块的可靠性。拓扑模块采用的是OSPF协议，通过三次握手进行拓扑感知，收到的邻居集群关系保存在本地LSDB中，定期交换LSA信息达到全局拓扑统一的目的，并使用心跳信息维护拓扑。因此，该部分测试具体分为：新增集群测试、拓扑信息交互测试、集群丢失测试。

1. 新增集群测试

|  |  |
| --- | --- |
| 测试目的 | 验证当有新集群加入时，通过OSPF三次握手进行拓扑探测 |
| 测试步骤 | 1. G1集群启动管理员，G2、G3集群启动控制器 2. 经过初始化、预下发流表后观察拓扑信息 |
| 预期结果 | 经过三次握手，集群间彼此感知，在管理员界面上还可以看到详细的集群信息 |
| 实际结果 | 与预期相同，三次握手流程见下图1、图2，管理员收到邻居集群信息可以再ui界面上展示，结果见图3 |
| 结果分析 | G2集群向所有对外端口发送Hello消息，到达G1集群后经过解析，反向下发拓扑流表并发送ReHello消息，此时集群间达到单向连接状态，G2集群收到第二次握手信息后返回FinalHello消息，G1接收后完成拓扑探测过程 |

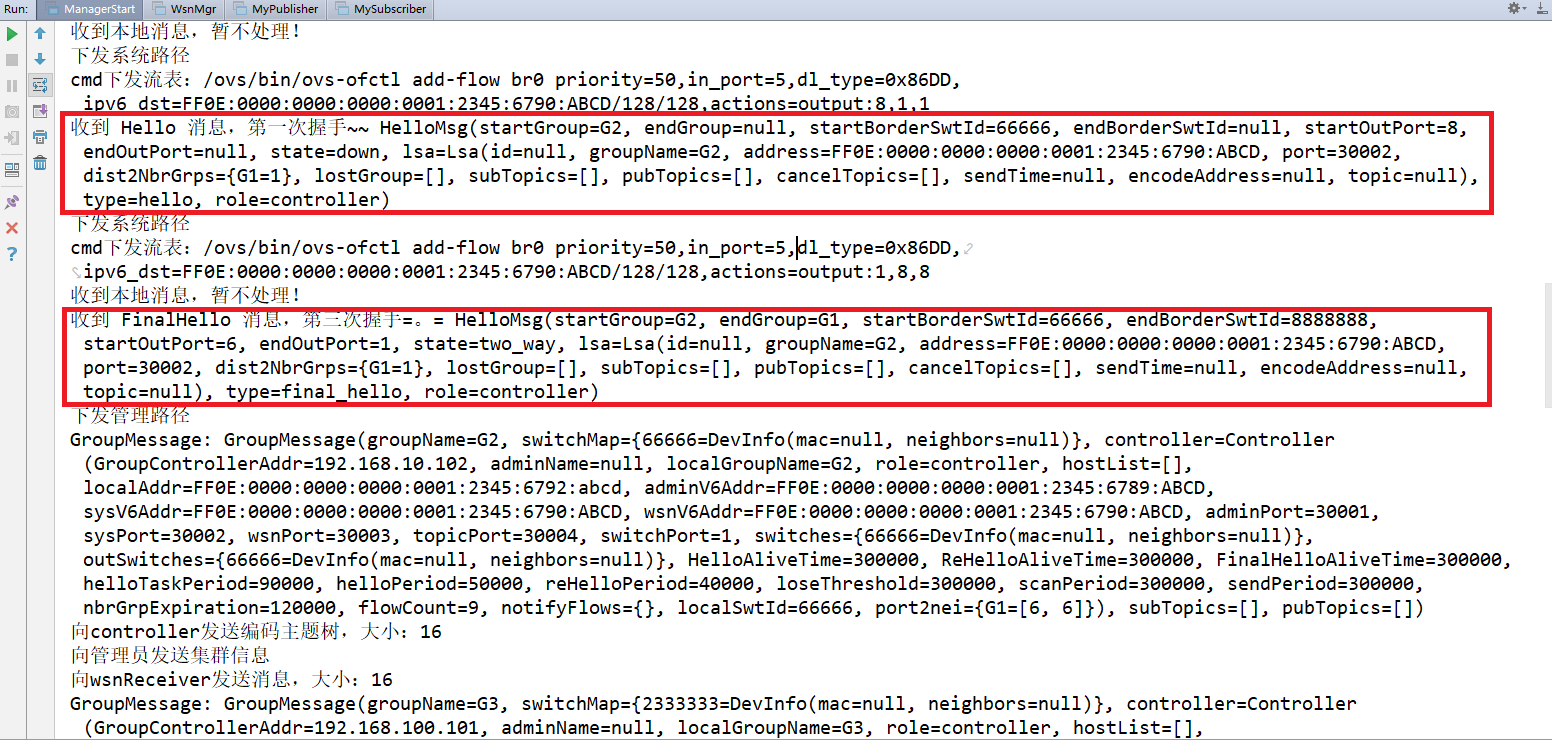


图1 G1三次握手过程

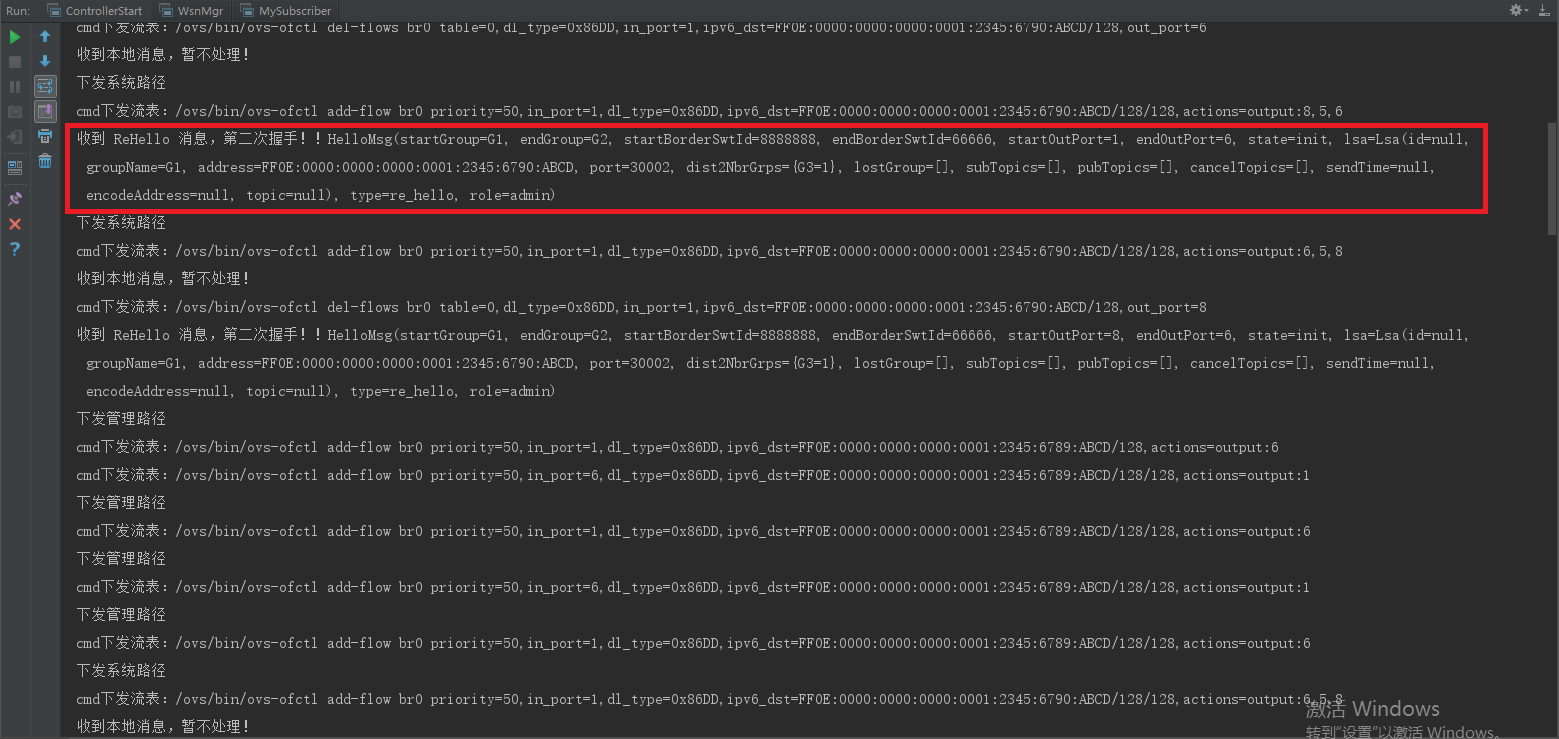


图2 G2三次握手过程



图3 新增集群展示

1. 拓扑信息交互测试

|  |  |
| --- | --- |
| 测试目的 | 验证当拓扑维护阶段，集群间彼此交换LSA信息 |
| 测试步骤 | 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 |
| 预期结果 | G1、G2集群在构建了邻居关系后，相互之间交换了LSA信息 |
| 实际结果 | 与预期相同，具体见图4：G1广播lsa内容 |
| 结果分析 | 拓扑模块采用的是OSPF协议，集群保存了邻居集群的所有信息，包括集群名、跳数、连接端口等，保存在本地LSDB数据库中，并通过定期交换LSA消息保证拓扑的全局一致。集群间交换的LSA内容见图4 |

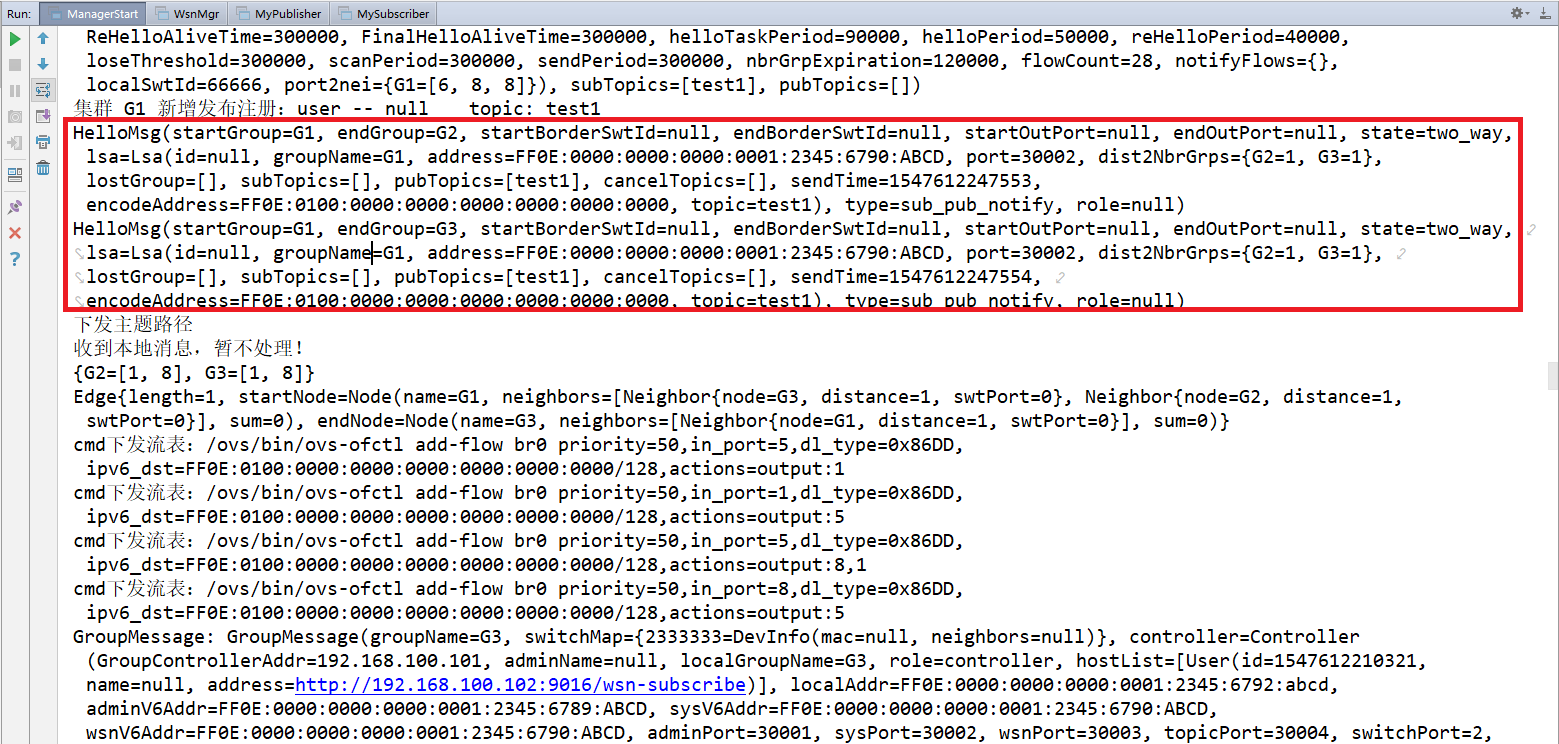


图4 G1广播lsa内容

1. 集群丢失测试
2. 路由维护

测试目的：通过功能测试，验证路由维护模块的可靠性。路由模块负责转发路径的计算，需要结合全网拓扑，转换为连通图，使用dijkstra最短路径算法、steiner多源生成树算法，计算出合理的路径，由于全网拓扑的统一，各集群独立计算结果相同。该模块测试具体分为：管理路径测试、主题路径测试。

1. 管理路径测试

|  |  |
| --- | --- |
| 测试目的 | 验证当有新集群加入时，通过判断是否为管理员身份，进而计算管理路径 |
| 测试步骤 | 1. 启动G2 集群的控制器模块  2. 启动G3 集群的控制器模块  3. 启动G1 集群的管理员模块 |
| 预期结果 | 1. G2、G3通过拓扑发现后并没有计算管理路径  2. 当管理员G1 计入后，三个集群分别计算管理路径 |
| 实际结果 | 与预期相同，具体见图5：管理路径计算结果 |
| 结果分析 | 管理路径是以管理员节点为根节点的树形结构，在计算时必须确保管理员角色在全网拓扑中，若不存在则无需计算，采用的是Dijkstra最短路径算法求解两集群间的最短路径，Kruskal算法求解最小连通图算法；因此测试的整体过程为：  1. 当G2、G3通过拓扑发现感知到邻居节点的存在后，由于两者都为控制器角色，因此无需计算管理路径  2. 当G1加入拓扑后，由于存在了管理员身份，因此需要计算管理路径 |

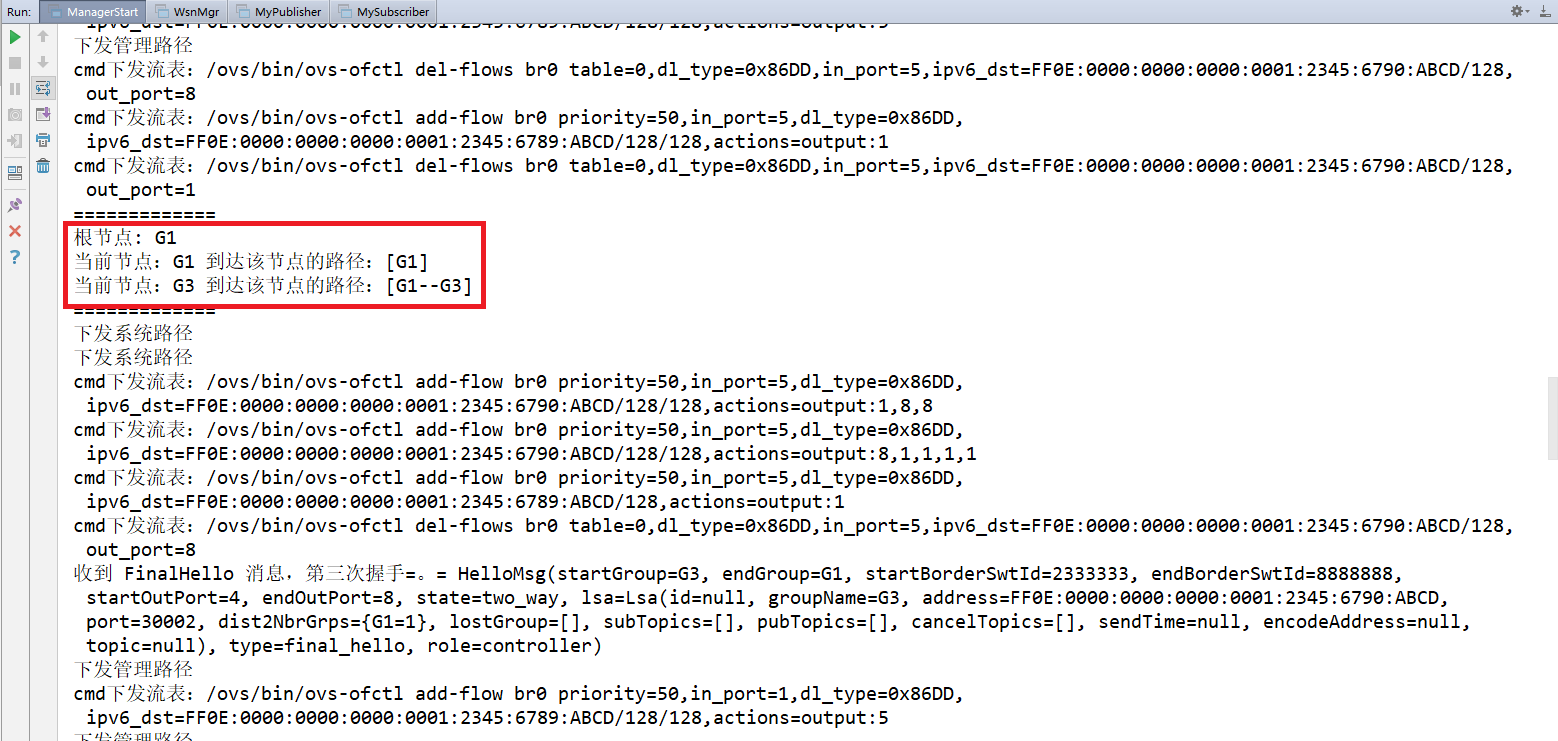


图5 管理路径计算结果

1. 主题路径测试

|  |  |
| --- | --- |
| 测试目的 | 验证当全网中存在多个发布订阅节点时，会计算相应的主题路径 |
| 测试步骤 | 1. 分别启动G1、G2、G3集群  2. 在集群G1 上开启针对主题“test1”的发布  3. 在集群G2 上开启针对主题“test1”的订阅  4. 在集群G3 上开启针对主题“test1”的订阅 |
| 预期结果 | 1. 在获知集群G1、G2针对相同主题存在发布订阅行为后，计算主题“test1”的主题路径  2. 集群G3 由于订阅了相同主题，计算新的主题路径 |
| 实际结果 | 与预期相同，具体见图6：G1-G2 主题路径计算结果，图7：G1-G2-G3 主题路径计算结果 |
| 结果分析 | 主题路径是根据相同主题计算的树形结构，由于在网络中会存在针对同一主题的多个发布源、订阅源，因此采用的是steiner算法，求解多源最小生成树，每当有新节点加入或更改时都需要重新计算，测试的整体过程为：  1. 当集群G1、G2中存在相同主题的发布订阅行为后，计算主题路径  2. 新节点G3对相同主题有订阅行为，计算新的主题路径 |

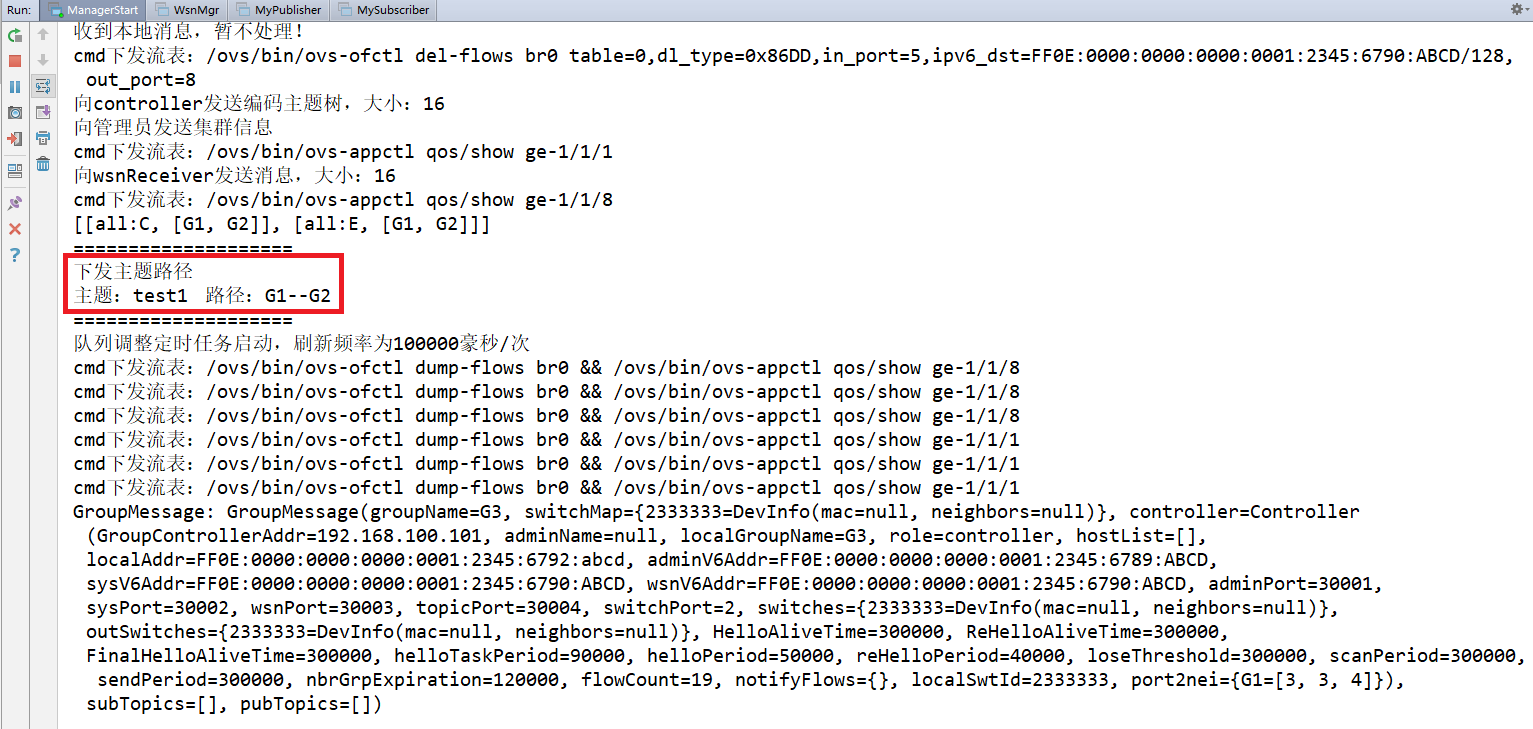


图6 G1-G2 主题路径计算结果

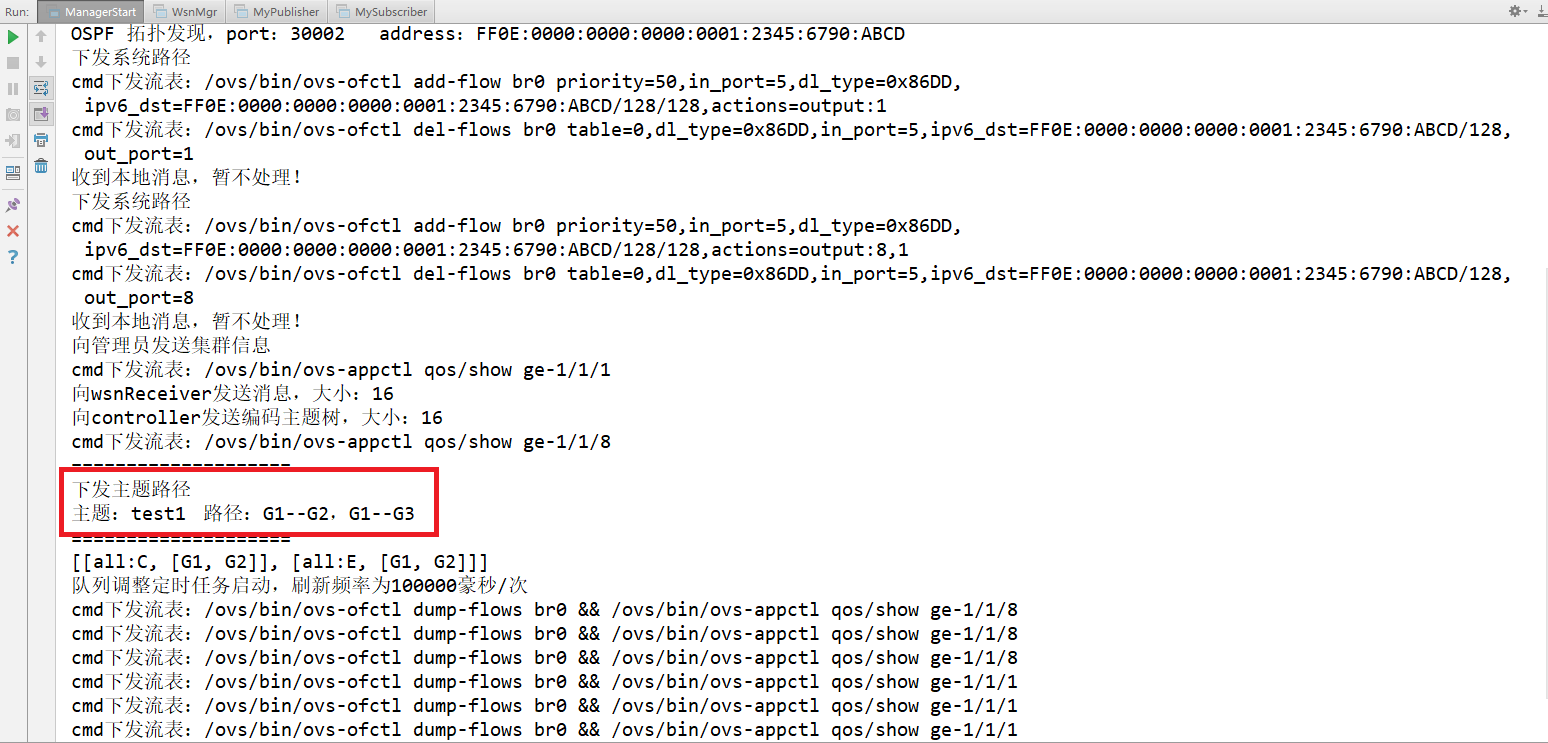


图7 G1-G2-G3 主题路径计算结果

1. 用户交互

测试目的：通过功能测试，验证用户交互模块的可靠性。用户交互模块负责将用户的请求传给控制器，并根据消息内容的不同，处理发布、订阅请求和用户协商。该模块测试具体分为：发布测试、订阅测试、用户协商测试。

1. 发布测试

|  |  |
| --- | --- |
| 测试目的 | 验证当wsn模块启动后，可以接收到下属用户的发布请求 |
| 测试步骤 | 1. 启动wsn模块  2. 启动订阅用户，触发订阅请求  3. 启动发布用户，触发发布请求 |
| 预期结果 | 1. wsn收到用户的订阅请求  2. wsn收到用户的发布请求 |
| 实际结果 | 与预期相同，具体见图8：wsn收到用户订阅请求 |
| 结果分析 | Wsn模块与用户之间采取的是ws传输方式，wsn收集用户发来的发布请求，由于发布数据量相对于用户请求较为庞大，需要单独开启监听，因此在发布的处理逻辑中wsn顺序使用可用端口，并将发布地址返回给用户，这样用户只需向针对主题的发布地址传输即可 |



图8 wsn收到用户发布请求

1. 订阅测试

|  |  |
| --- | --- |
| 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  3. 启动发布用户，触发发布请求 |
| 预期结果 | 1. wsn收到用户的订阅请求  2. wsn收到用户的发布请求 |
| 实际结果 | 与预期相同，具体见图9：wsn收到用户订阅请求 |
| 结果分析 | Wsn模块与用户之间采取的是ws传输方式，wsn收集用户的订阅请求，系统采用的是主题树编码方式，相同主题对应的ipv6编码一致，监听主题对应的地址就能收集到该主题的所有消息，再根据订阅用户的详细信息分别推送即可 |

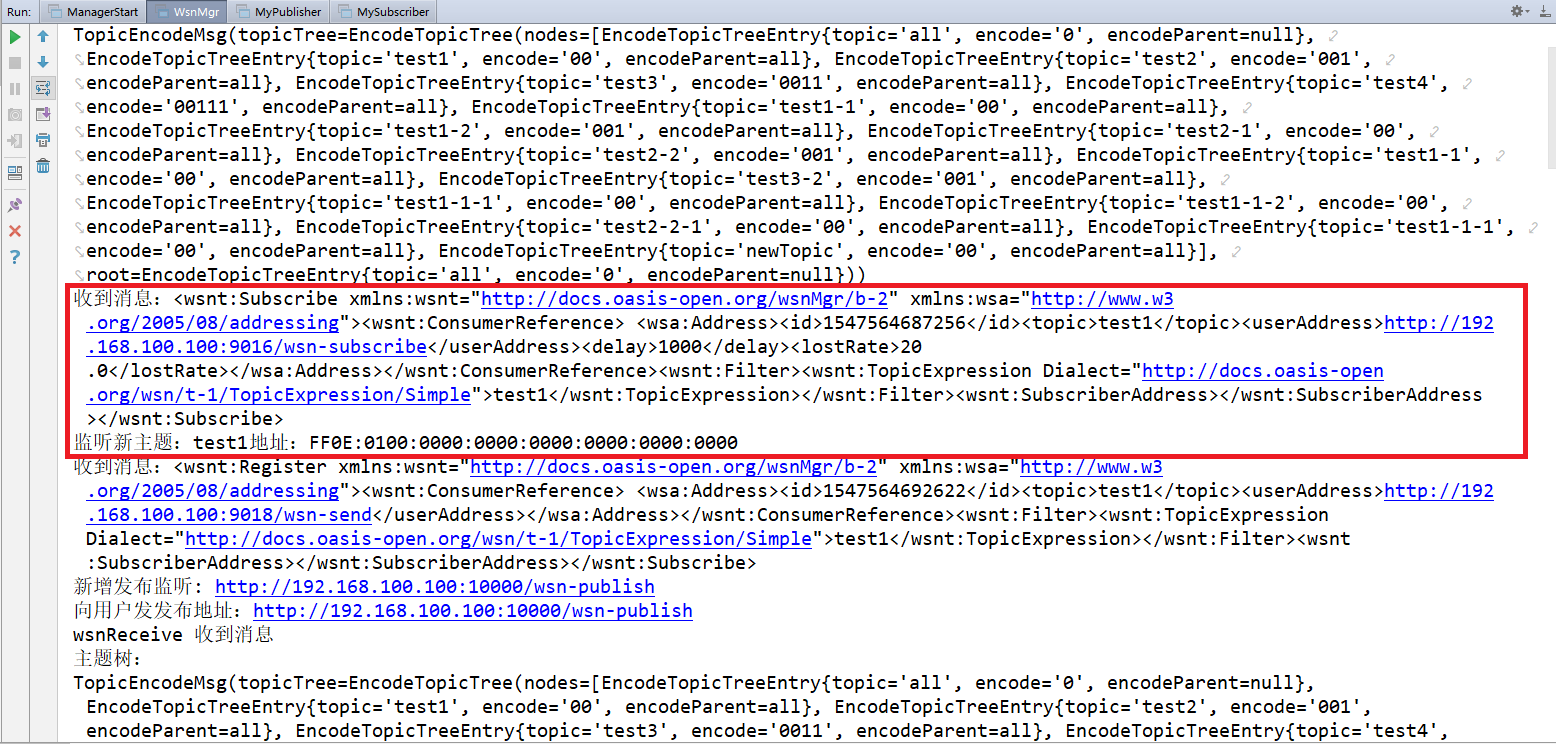


图9 wsn收到用户订阅请求

1. 用户协商测试

|  |  |
| --- | --- |
| 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  3. 启动发布用户，触发发布请求 |
| 预期结果 | 1. wsn收到用户的订阅请求  2. wsn收到用户的发布请求 |
| 实际结果 | 与预期相同，具体见图10：wsn收到用户协商请求 |
| 结果分析 | Wsn模块与用户之间采取的是ws传输方式，用户可以针对自己的订阅发起时延、抖动上的要求，由于只有管理员存储了全网链路信息，wsn在收到用户请求后发给控制器，控制器再沿管理路径上报给管理员，经过计算分析，若结果可以满足，则将新的时延带宽分配结果下发给具体控制器，再由控制器处理；若用户需求不能满足，则进一步通知到具体用户，由用户决定是接受当前结果还是进一步发起协商 |

### 性能测试

多媒体传输

测试目的：验证多媒体传输QoS模块的有效性。多媒体传输模块采用的是Reactor+负载均衡策略，在高并发环境下多线程无法承载系统压力，而传统的串行传输方式又导致高丢包率，因此，需要在内存中缓存，并采用负载均衡策略对阻塞队列进行控制。

1. 连续发包测试

|  |  |
| --- | --- |
| 测试目的 | 验证不同发包数量下用户丢包率的变化 |
| 测试步骤 | 1. 启动控制器，开启用户发布订阅注册服务 2. 调用订阅注册服务 3. 调用发布服务，发布主题需要与订阅主题相同 4. 在订阅方记录接受的数据包，计算用户丢包率 5. 控制其他条件不变，增大发包数量，判断不同发包压力下的丢包率结果 |
| 预期结果 | 随着发包数量的增大，用户丢包率增加，但总体稳定在一个范围内 |
| 实际结果 | 与预期相同，总体结果见表 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 发包方式：连续发送 | | | | |
| QoS策略 | 发包数量 | 数据包大小 | 时延 | 丢包率 |
| 单线程传输 | 500 | 1KB | 13.65 | 76.20% |
| 1000 | 1KB | 12.41 | 81.40% |
| 1500 | 1KB | 13.61 | 81.67% |
| 2000 | 1KB | 15.04 | 83.60% |
| 2500 | 1KB | 17.03 | 97.68% |
| 3000 | 1KB | 17.54 | 90.37% |
| 3500 | 1KB | 17.25 | 88.57% |
| 4000 | 1KB | 18.17 | 88.93% |
| 4500 | 1KB | 19.21 | 89.07% |
| 5000 | 1KB | 20.57 | 88.44% |
| Reactor+负载均衡策略 | 500 | 1KB | 2395.79 | 0.80% |
| 1000 | 1KB | 5956.87 | 0.20% |
| 1500 | 1KB | 9543.59 | 0.20% |
| 2000 | 1KB | 11651.86 | 0.70% |
| 2500 | 1KB | 15077.17 | 0.12% |
| 3000 | 1KB | 21290.90 | 0.20% |
| 3500 | 1KB | 23633.17 | 0.74% |
| 4000 | 1KB | 34454.96 | 3.60% |
| 4500 | 1KB | 29550.34 | 0.09% |
| 5000 | 1KB | 32010.39 | 0.28% |

将上表中数据绘制成折线图如下所示：

从图中可以看出，随着发包总量的增大，单线程和Reactor方式的丢包率都有一定程度的提升，总体稳定在一个范围内；但是在数值上比较，单线程下的平均时延明显低于Reactor方式，这是因为Reactor模式采用了阻塞队列，将接收到的数据包暂存在内存中，这将导致数据线性传输，必须等待前面的数据包先传输完成，才能进行处理，这必然导致平均时延的上升。

从图中可以直观的看到，随着发包数量的增加，单线程和Reactor方式的丢包率都有一定的提升，但总体稳定在一个范围内；；但是在数值上比较，单线程下的平均时延明显高于Reactor方式，部分条件下的丢包率甚至达到90%以上，这是因为Reactor模式采用了阻塞队列，将接收到的数据包暂存在内存中，当用户连续传输时，wsn层处理速率远远小于数据包到达速率，若单线程处理，则必然导致很多数据包的丢失。

从上述数据可以看出，在连续发包、数据包大小为1KB条件下，平均时延、丢包率将随着发包总量的增加而提升，与单线程传输相比，Reactor

1. 发包速率调整测试
2. 调整数据包大小测试