队列功能及性能测试

**1．测试目标**

为验证基于SDN的发布订阅系统在实际SDN网络环境中的性能相比虚拟环境下的提升情况，在控制相对条件不变的情况下对原有测试方案进行适度更改，现拟定如下测试方案。

**2．测试环境**

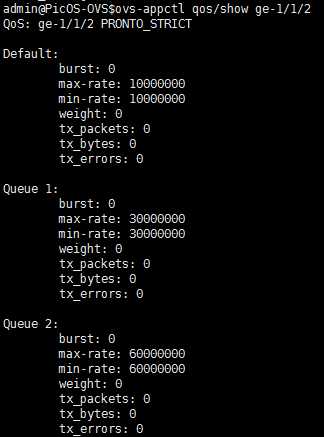
手中现有SDN交换机一台，PC机两台，交换机参数如下：

|  |  |
| --- | --- |
| 交换机型号 | Pica8-p3290 |
| 支持OpenFlow版本 | OpenFlow1.0、1.3、1.4 |
| 支持OVS版本 | 2.3 |
| 网口数 | 48 |
| 端口队列 | 最多支持8个优先级队列 |
| IPv6 | 支持Ipv6地址匹配 |

**3．测试内容**

**队列添加测试**

通过交换机自带的串口可以登录交换机内嵌的picos系统，通过队列下发命令给交换机的ge-1/1/2端口添加3个队列q0、q1、q2，初始带宽分别为10M、30M、60M，下图表示队列添加成功。



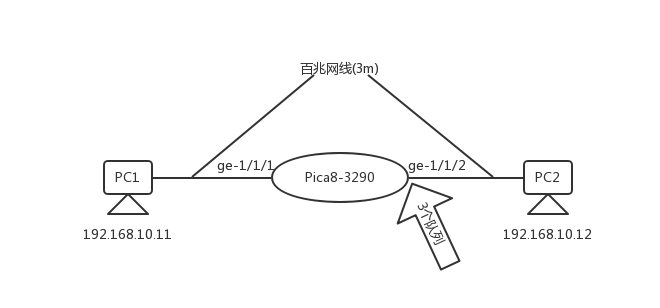
**入队流表下发测试**

添加队列之后还要测试队列的有效性，验证数据能否真正从设定的端口进入。首先要下发对应端口上的流表，流表下发方式与命令跟虚拟环境有所不同。下图表示在交换机上添加了两条流表，一条是2端口进1端口处，另一条是1端口进并进入2端口的2号队列，流表下发成功。

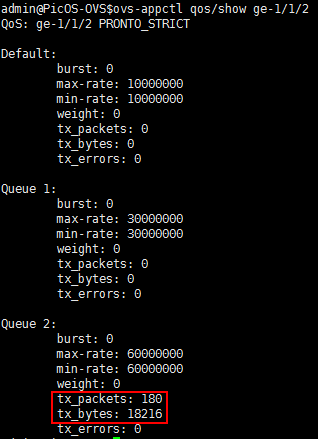


**流表与队列的有效性测试**

交换机的1、2端口分别有一台PC机，地址分别是192.168.10.11和192.168.10.12，拓扑如下所示。



根据以上流表情况，现在PC1上pingPC2，再次查看流表及队列情况如下图，流表和队列有效性得到验证。



**转发时延测试**

测试目的：为了验证队列带宽调整算法的有效性，验证算法能否在队列拥塞的情况下减小队列长度，降低数据包的排队时间。与虚拟环境不同的是，数据在链路中的时延将包括很多无效时间，我们需要把这些无效时间减掉才能得到更准确的数据。

a)单跳无压力情况

首先测试单跳无压力下数据包进入交换机，经过流表匹配并执行相应的actions所需的时延。测试步骤如下：

1． 将一台PC连接到交换机的其中一个网口上，PC启动控制器OpenDaylight，在拓扑界面能够正常看到交换机，表示连接成功。

2．令控制器发送packet\_out消息(主动发)，该消息到达交换机中由于没有流表项可以匹配而又以packet\_in消息形式发回控制器。

3．记录接收到发送的时延，记为T0。

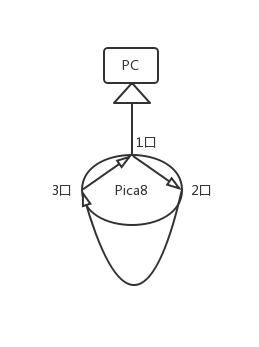
4．接入另一台PC到交换机端口1，同时下发两条流表用于数据包匹配，流表如下：

ovs-ofctl add-flow br0 priority=50,in\_port=1,actions=output:2

ovs-ofctl add-flow br0 priority=50,in\_port=3,actions=output:1

5．用网线将交换机2、3两端口连接，形成机内环路，在PC上发送数据包，数据会经过两次匹配并执行actions的时延，最终返回PC。

6．记录发送到接收的时延，记为T1。示意图如下：



由此可得出在单跳无压力下数据包匹配并执行actions的时延为：

T = ( T1 - T0 ) / 2 （多次测试求平均值）

测试过程中的数据记录下表（单位us）

|  |  |
| --- | --- |
| 发包数 | T0 |
| 100 |  |
| 500 |  |
| 1000 |  |
| 2000 |  |
| 5000 |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 发包数 | 发包时间 | 第一次  收包时间 | 第二次  收包时间 | T1 |
| 100 |  |  |  |  |
| 500 |  |  |  |  |
| 1000 |  |  |  |  |
| 2000 |  |  |  |  |
| 5000 |  |  |  |  |

预期结果：SDN交换机流表匹配速率很快，在没有拥塞的情况下可以达到微秒级。

b)单跳有压力情况

测试目的：测试在队列拥塞的情况下，数据包在不同的队列排队等候的时延。

由于队列初始带宽不同（设定高优先级为60M，中优先级为30M，低优先级为10M），因而对数据包的处理能力也不同。我们增大发包频率是队列产生拥塞，然后按照上述T1的计算方法重新计算端到端时延T2，并由此计算出排队时延，计算公式为： = （T2 – T1）/ 2 （T1不同表项的影响）

测试步骤如下：

1．在上述无压力情况下逐步加大发包频率，同时计算队列长度（队列长度=进队列数据-出队列数据），找出临界点（即最大可接受频率）。

2．逐步增大发包频率，使得队列长度有序增长，排队时延增大，记录不同压力下三条队列的长度和排队时延。

原则上，数据包的流表匹配时间是不变的，链路传输时间也可以认为是不变的，我们将拥塞情况下的端到端时延减去无拥塞下端到端时延即可得到数据排队时延，记录表格如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 发包数 | 队列长度 | | | 排队时延 | | |
| 0 | 1 | 2 | 0 | 1 | 2 |
| 10000 |  |  |  |  |  |  |
| 20000 |  |  |  |  |  |  |
| 30000 |  |  |  |  |  |  |
| 50000 |  |  |  |  |  |  |
| 80000 |  |  |  |  |  |  |
| 100000 |  |  |  |  |  |  |

预期结果：发包频率越大，数据包排队时间越长，并且相同频率下队列带宽越小，排队时间越长，即排队时延跟发包频率成正比，跟队列带宽成反比。

\*\***不同条件下的队列带宽调整算法对排队时延的影响 \*\***

**（调整算法核心）高优先级队列也要有丢包策略**

原始算法：首先为队列下发初始带宽，默认60M、30M、10M，之后根据队列时延判断队列拥塞情况，若发生拥塞，则根据拥塞程度（一般、严重、非常严重）则查看其它队列有无空闲带宽进行带宽抢占，高优先级队列抢占能力高于低优先级队列的抢占能力。一般情况下，高优先级队列总能保证一定带宽，而低优先级队列往往总被抢占。因此，如果低优先级队列发生拥塞，并且又无法抢占其它队列的带宽，此时会触发尾丢弃算法禁止数据继续进入队列而造成更严重的拥塞。

算法分析：根据以上原始算法的描述，该算法主要由以下几个弊端：

1. 初始带宽的设定没有根据网络实际情况，而是主观臆定；

用户订阅时提出时延需求，管理节点收到LSA，订阅信息

1. 低优先级队列因为带宽抢占能力弱可能造成“饿死”现象；
2. 没有考虑数据流量超过端口总带宽时如何处理丢包问题；

尾丢弃，反馈

1. 没有考虑用户的实际需要。
2. 队列拥塞状况判断标准没有统一管理和配置。

针对以上弊端，新的算法提出以下几点创新点：

针对弊端1）：由于优先队列只在集群对外端口上添加，而且集群是按序上线的，因此我们在添加初始队列的时候已经存在其他邻居集群（第一个集群单独处理），可以统计集群内各种优先级消息的比例【发布端在发布消息时要指定消息的优先级，节点在接收到各自的发布消息后做出优先级统计并实时共享所有发布端消息的优先级比例告知管理员】，管理员按照这个比例进行初始带宽的分配，保证带宽都可以物尽其用。（备用方案：管理员流量管理模块，可以看到全局流量状况，根据全局比例分配带宽）。

针对弊端2）：为了防止低优先级队列“饿死”，我们为低优先级队列设置一个最低阈值，如果带宽在阈值之上，我们允许高优先级队列抢占它的带宽；如果带宽低于这个阈值，则抢占停止，任何高优先级队列都不能再抢占，以防止其“饿死”。

针对弊端3）：我们在优先队列的基础上引入Meter概念，Meter又叫计量表，类似于流表。每个Meter Table都由若干个Meter Enties组成，每个Enties定义每个流表的其中一个匹配域meters.每个流表项都可以在Instruction Set里指定一个Meter，之后若有数据包成功匹配这条流表项，数据就会按照Meter对应的速率限制和动作集得到处理。通俗的说，优先队列是在出端口上对数据进行管理，而计量表是在匹配时对数据进行管理，二者结合可以实现更加复杂的QoS保证。

在这里，我们根据网络实际状况或交换机端口本身的性能可以得到每个端口的速率上限。同时，我们也可以根据统计信息获取每种优先级消息的比例，针对每种优先级的消息设定不同的Meter进行速率上限的设定，防止队列发生严重拥塞。

针对弊端4）：对于用户的实际需求（服务等级），若用户服务等级较高，一方面可以设定其订阅的消息为高优先级，另一方面也可以针对其转发路径进行带宽调整，保证可靠传输的同时也满足其对时延的需要。

丢包率过大时，时延不能保证时，管理员重新分配带宽，保持稳定，平均情况，长期现象，降低调整频率。

针对弊端5）：对于某一个主题，订阅者在订阅成功后可以根据实际情况（付费标准等）在订阅消息的同时提出一个时延要求，记为（j表示第j个订阅者，即第j条路径，下同，）作为消息从发布者该到订阅者的时延约束，亦即端到端时延。在转发树中可能存在多个订阅者，每个订阅者都会提出自己的时延约束，这就会针对一个特定主题形成一个约束集。

而对路径中的每一跳即每个SDN交换机而言，会有一个最低的时延下限，其值与交换机性能、端口带宽、线路标准有关，是一个固定值,记为(其中p是路径数组，表示路径j所经过的交换机编号为i，下同),交换机只能保证在此下限之上的时延约束。但由于不同的节点在网络中的位置不同，导致节点的时延要求也不同。我们将单个节点的时延约束记为，对整个转发树而言，从发布者到所有订阅者的时延应尽量满足用户提出的需求，即有：

在分配带宽时我们有一个原则：即选择最佳适应算法尽量接近用户的需要，根据每个节点的最低时延比例来分配每个节点的时延约束，可得出每个节点每条线路的时延要求，在此基础上计算出最低时延要求作为节点时延配比。

如果上述方程组有解，管理员按照计算出的作为交换机的时延参照并下发到交换机所在的控制器作为本地调整参照；如果无解，管理员通知订阅者该标准无法满足，双方协商一个新的时延约束继续上述过程。

初始化完成之后，管理员还要对时延配比进行实时调整，而调整的基础是节点的实时反馈。当前队列长度可以用交换机提供的接口通过一定的计算方式获得，而平均队列长度则采用随机早期检测公式（RED公式）得出，该公式为：

AvgQ = (1 - Wq) \* AvgQ + Wq \* Q

其中AvgQ为平均队列长度，Wq是一个权值，Q是实际得出的当前队列长度，而队列的实际处理时延则根据平均队列长度和队列带宽的比值计算得到。由此，我们得到了队列的实际处理时延，记为,本地控制器将该实际处理时延反馈给全局管理员，管理员按照如下公式重新计算每个交换机的时延约束。

如果计算得出的，说明此时交换机无法满足当前的传输速率，管理员则通知消息的发布者降低发包频率；否则管理员将此新的时延约束下发至交换机控制器

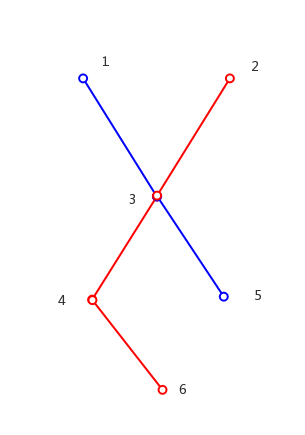
而在本地交换机控制器，我们根据交换机中当前数据包转发情况实时监控端口上的优先队列的平均队列长度AvgQ并调整队列的带宽w使之满足管理员分配的时延约束。每个对外端口上都有3条优先队列，编号为q=0,1,2，其中高优先级消息进入0队列，低优先级消息进入2队列，其他消息进入1队列。端口的总带宽是一定的,记为,每条队列的带宽总和等于端口总带宽，即,每条队列的平均队列长度通过上述RED公式获取，经过计算之后，我们的队列带宽调整要满足以下几个约束条件：

/, q=0,1,2

, q=0,1,2

而带宽调整的目的是将3条队列的带权时延降低到最小，即minimize（）,按照该目标计算带宽配比，由控制程序下发到交换机中使之生效。

举例说明，下图中有红色和蓝色两条转发路径，每条路径都要尽量满足订阅用户提出的时延要求。对于节点3，既要满足红色线路又要满足蓝色线路，我们取能同时满足两者的最小时延作为该节点的时延要求，该过程由管理员负责计算并将结果下发，节点按照该时延要求配置相应的带宽。



测试目的：为验证队列带宽调整算法的有效性，亦即验证在相同发包频率下算法能否改善队列拥塞状况，减少数据排队时延。队列调整算法，是根据三个队列的不同拥塞状况（由实时获取的队列长度判断）采取相互之间借取带宽来缓解自身拥塞的算法，带宽借取原则是高优先级队列优先借取，低优先级队列借取能力最次，借取后不能使被借取得队列产生严重拥塞并能保证最低带宽（比如1MB/s），算法具体细节详见上节。

测试步骤：

1. 按上述过程的环境连接拓扑，添加初始化队列，下发初始流表。

2．在PC上启动队列带宽调整程序，分别调整算法的检测频率（相邻两次检测队列长度的时间间隔）为500ms,1000ms,3000ms,5000ms，并实时监测队列拥塞状况。

3．保持发包频率相同，再次发送相同的数据包，验证相同条件下队列长度以及排队时延情况，并检测当前队列的实际带宽。

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 算法监测频率 | 发包数 | 队列长度 | | | 排队时延 | | | 队列带宽 | | |
| 0 | 1 | 2 | 0 | 1 | 2 | 0 | 1 | 2 |
| 500ms  1000ms  3000ms  5000ms | 10000 |  |  |  |  |  |  |  |  |  |
| 20000 |  |  |  |  |  |  |  |  |  |
| 30000 |  |  |  |  |  |  |  |  |  |
| 50000 |  |  |  |  |  |  |  |  |  |
| 80000 |  |  |  |  |  |  |  |  |  |
| 100000 |  |  |  |  |  |  |  |  |  |

预期结果：队列带宽调整算法会对数据的排队时延产生明显的影响，高优先级的队列效果尤其显著。并且，不同的检测频率会有不同的效果，原则上频率越高效果越好，但随之而来的是性能的下降。因此会有一个拐点，在拐点上效果达到最佳。

**队列抗干扰测试**

测试目的：SDN交换机端口最多支持8个优先级队列，该测试主要验证队列的独立性和抗干扰性，验证一条队列在任何情况下都不会对其他队列产生明显的干扰。

测试步骤：

1.两台PC分别连接交换机的1、2两端口，在2端口上添加3个优先级队列分别为q0,q1,q2，并在交换机中下发相关流表。

2.在PC1上分别无压力持续发送数据包至PC2。

3.不断增大队列0的发包频率，观察其他两个队列的时延波动和丢包率变化，结果记录在下表。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 队列0发包数 | 1000 | 3000 | 5000 | 8000 | 10000 |
| 队列1时延 |  |  |  |  |  |
| 队列2时延 |  |  |  |  |  |
| 队列1丢包率 |  |  |  |  |  |
| 队列2丢包率 |  |  |  |  |  |

预期结果：队列0的变化不会对队列1和队列2产生明显干扰，队列具有较强的独立性和抗干扰性。

**队列抗压测试**

队列的优先级是由队列带宽以及队列调整策略决定的。原则上，队列的优先级不同，其抗压能力也不同，抗压能力是指在允许的时延和丢包率下可以接收的最大发送频率。数据发送频率不同将对数据接收比率产生直接影响。在虚拟环境下，随着发送频率的不断增大，数据丢包率逐渐增大以致达到50%以上，这在真实情况下是不能容忍的。

测试目的：测试不同带宽的队列随着发包频率的变大而变化的情况，观察队列的抗压能力。

测试步骤：

1.PC1，PC2连接交换机1、2端口，2端口添加3个队列，下发相关流表。

2.PC1分别发送不同优先级的消息至PC2，发包数见下表，记录数据的收包数、丢包率、平均时延和吞吐量。

由于三个队列初始带宽不同，因而在实际数据传输过程中丢包率也不同，我们选择不同的发包频率，记录接收方收到的数据包数求得丢包率，进而求得系统吞吐量，测试三条队列对数据的处理能力，每条队列拟定测试情况如下：

队列0/1/2抗压测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 预计每秒  发包数 | 收包数 | 丢包率 | 接收时间(ms) | 平均时延(ms) | 吞吐量  (包/秒) |
| 500 |  |  |  |  |  |
| 1000 |  |  |  |  |  |
| 3000 |  |  |  |  |  |
| 6000 |  |  |  |  |  |
| 10000 |  |  |  |  |  |
| 30000 |  |  |  |  |  |
| 60000 |  |  |  |  |  |
| 90000 |  |  |  |  |  |

预期结果：高优先级队列的抗压性能明显优于低优先级队列，在相同发送频率下，高优先级能保证更好的可靠性，QoS得到较好的保证。

**尾丢弃测试**

测试目的：尾丢弃是指队列在发生拥塞的时候如果无法通过借流量缓解拥塞状况，就要禁止后续数据继续进入队列，防止更严重的拥塞发生。就上述描述来看，一般情况下尾丢弃只发生在低优先级队列中，因为任何高优先级队列都有较大的可能从其他队列获取流量来缓解拥塞状况。

具体测试步骤为：

1. 在上述拓扑环境中，PC1发送大量不同优先级的数据包至PC2使得每条队列的带宽都能得到有效利用（发包数根据上述压力测试得出临界点）。

2. PC1上启动队列带宽调整程序，记录队列2的队列长度和时延分布情况，观察时延拐点。

3. 查看交换机中的流表情况，验证尾丢弃流表是否下发成功。

尾丢弃原理为：程序一旦检测到低优先级队列q2拥塞，与此同时高优先级的队列没有流量可以借取，则遍仿交换机中所有actions为set\_queue:2的流表并保存，然后将对应的actions更改为丢弃动作drop，该动作可以禁止任何数据进入该队列，等队列拥塞状况好转，再将事先保存起来的原始流表下发回交换机。

为了更好地观察到尾丢弃效果，PC1上持续发包时间应该适当延长，我们按顺序发送5次，每次发包100000，观察队列时延的变化情况，测试数据记录如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 队列2发包数 | 收包数 | 丢弃个数 | 队列长度 | 队列时延 |
| 100000 |  |  |  |  |
| 100000 |  |  |  |  |
| 100000 |  |  |  |  |
| 100000 |  |  |  |  |
| 100000 |  |  |  |  |

预期结果：队列2一开始的时延随着数据包的涌入而逐渐上升，直到尾丢弃算法执行生效，后续数据drop，队列长度不再上升，并开始下降直到一个合适的位置，尾丢弃动作被清除，以此呈现一种动态稳定状态。

**流表项个数对数据匹配速度的影响 (T1，T2)**

测试目的：交换机中的流表由一个个的流表项组成，每个流表可以包含多个流表项，但是随着流表项个数的不断增大，流表匹配速度会不会受影响是一个值得研究的问题，基于此，我们设计如下方案来测试流表项个数对交换机转发速度的影响。

测试步骤：

1. 按上述拓扑连接PC1、PC2和交换机，添加相关队列，下发流表。

2. PC1以相同压力发送数据包至PC2，并逐步改变交换机中流表项个数分别为10、100、500、1000、3000，记录不同个数下数据时延和丢包率，记录如下：

发包数固定为5000包

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 流表项个数 | 交换机处理  总时延 | 单个时延 | 丢包率 | 吞吐量 |
| 10 |  |  |  |  |
| 100 |  |  |  |  |
| 500 |  |  |  |  |
| 1000 |  |  |  |  |
| 3000 |  |  |  |  |

预期结果：流表项个数会对流表匹配速度产生影响，并随着个数增大呈现一定程度的增长。

**抖动测试**

测试目的：抖动描述的是网络延迟的变化量，它是由同一节点的任意两个相邻数据包在传输路由中经过网络延迟而产生，可由相邻数据包延迟时间差除以数据包序号差得到。基于此概念，可设计如下实验测试系统抖动情况。

测试步骤：

1. 在上述拓扑下，固定发包压力不变，多次发送相同数据包。

2．记录数据传输总时延变化情况。

3. 求得时延标准差（抖动率）。

PC1发送一定数量的数据包至PC2，在PC2上记录总时间差（最后一包接收时间减去第一包发送时间），用此时间差除以发送的数据包数（序号差）可得系统抖动率，可多次测量求平均值。

|  |  |  |  |
| --- | --- | --- | --- |
| 发包数 | 收包数 | 时延 | 抖动率 |
| 5000 |  |  |  |
| 10000 |  |  |  |
| 30000 |  |  |  |
| 50000 |  |  |  |

预期结果：系统抖动会在一定范围内波动，整体呈现动态稳定性。

拓扑、路由、订阅测试

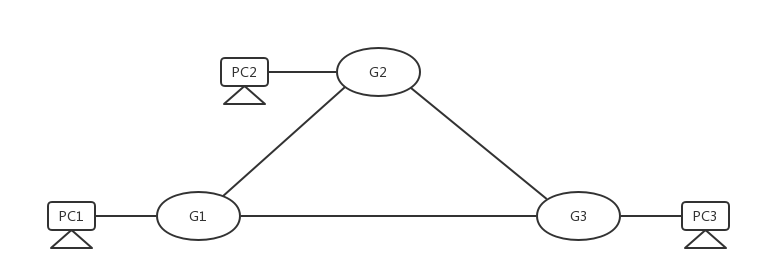
**测试目标**

本部分测试主要分为功能测试和性能测试两部分，前者主要测试程序中相关模块在实体SDN环境中的基本功能能否正确运行，后者在前者的基础上对比虚拟环境中的相关数据测试相关功能的运行效率。

**测试环境**

|  |  |
| --- | --- |
| SDN交换机 | 4台 |
| PC | 8台 |
| 网线 | 若干 |

由于实际条件的限制，假设我们有3台交换机，最多可以组成3个集群，每个集群1台交换机，拓扑图如下所示：



实验中，每个集群包含一台交换机和一台PC，PC上运行控制器和节点程序，控制器负责管理本集群的相关事务，包括拓扑获取，流表下发等。

功能测试 （拥塞时改变路由路径缓解时延，空闲路径）

路由重计算算法描述：

为缓解单个节点的转发压力，我们对原系统的路由算法进行优化。原路由计算的核心算法是基于OSPF（开放最短路径优先）的Dijkstra算法，该算法用于计算单源最短路径。全网拓扑结构既保存了集群间的连接关系，又包含了连接上的权值，该权值综合了物理距离、链路带宽和流量信息。初始值设定为集群间的物理距离。若有拥塞发生，我们根据拥塞程度综合一个新的链路权值通过LSA广播到全网，每个集群收到之后更新自己保存的链路状态数据库LSDB.

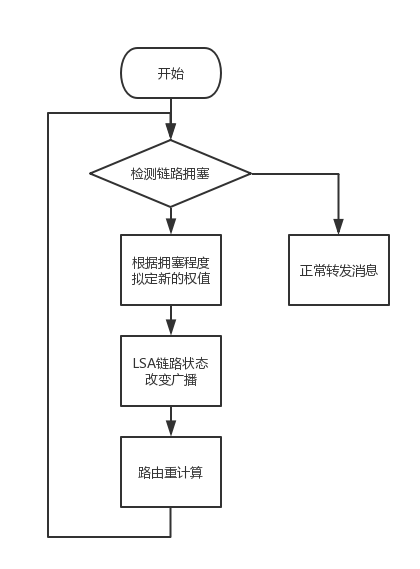
寻找空闲路径，动态规划算法，

同时触发路由计算模块进行重计算，以寻求其他空闲路径减轻当前链路的压力，达到缓解拥塞的目的。权值更新公式为：

新权值 = 旧权值 + E \*

其中E为链路拥塞程度，E = 时延 / 带宽。

流程图如下：



**拓扑维护测试**

**集群丢失测试**

**邻居构建测试**

**订阅测试**

**主题分裂与聚合测试 （聚合分裂缓解时延）全局路由重计算**

时延无法满足时触发

**路由计算测试**

性能测试

**拓扑收敛测试**

**路由时间测试**

**网络性能测试**

**断线重连测试**