## 总体设计

因为PBFT算法，相当于随机选择leader，任何节点都是对等的，所以可以选择任一节点进行操作，效果都是对等的。

1. 通过iptables，我们可以控制p2p端口的丢包率
2. 通过netem和tc，我们可以控制网络的时延和抖动，模拟网络长距离传输
3. 通过tc和iptables可以限制网络带宽，将网络带宽控制到用户设定的范围内
4. 其中网络时延和带宽的值，会在基准值上乘上一个随机浮点数，会模拟出比较多变的网络的场景
5. 写多线程程序，每隔5秒发送一次交易，使快高一直在增长
6. 一直去监控出快状态，当不出快时，程序停止

针对以上设计，该方案应该提供如下功能：

第一步：在任一节点上起一个线程，进行TC控制

第二步：再起一个线程往区块链一直发送交易，使区块链的状态一直在改变。

第三步：两个进程要能够进行通信，当检测到特定状态后，程序退出（exit）

## 具体实现

## 参考资料

1. iptables和TC介绍

Iptables主要功能

1. 封包过滤
2. 封包重定向
3. 网络地址转换（NAT）

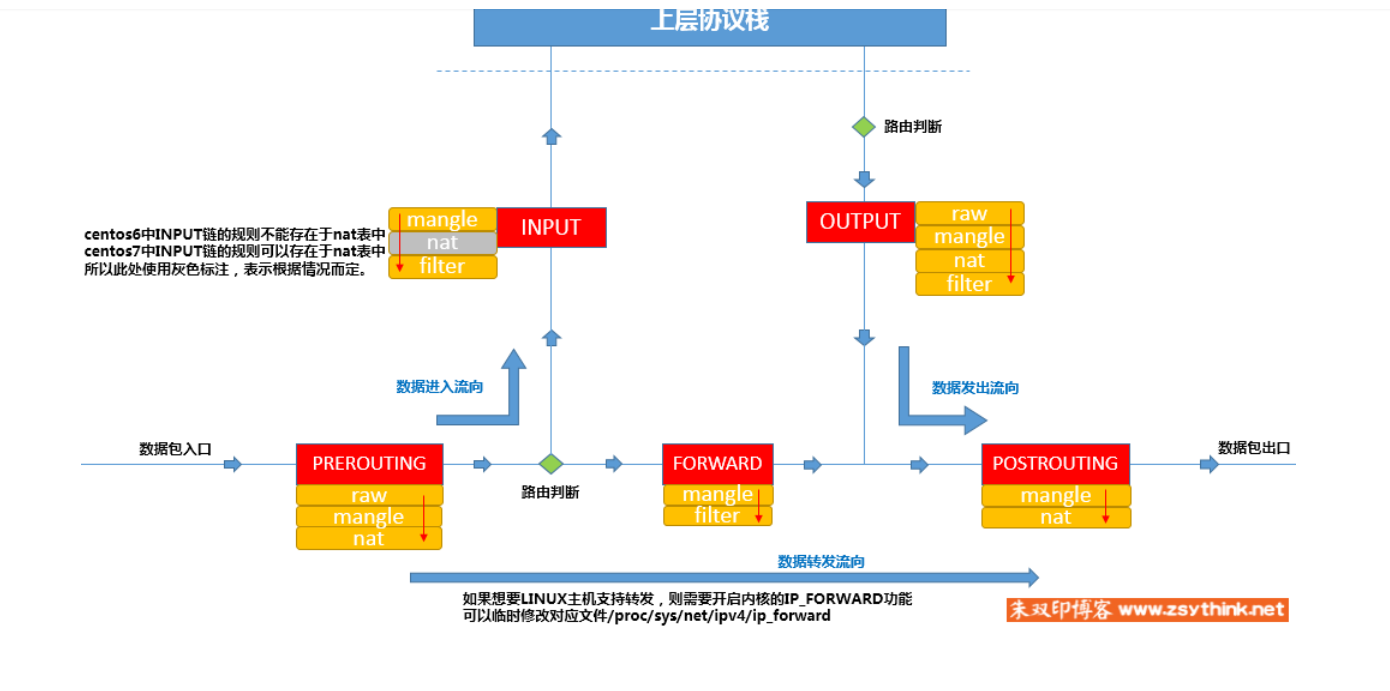
Filtre表：负责过滤功能，防火墙

Nat表：网络地址转换

Mangle：报文解包封包处理表

Raw表：关闭nat表上启用的链接追踪机制

网络数据包通过iptables的处理流程如下：



## 网络带宽时延丢包测量

### 带宽测量方法

使用iperf工具测试服务器带宽，它分为服务端与客户端，两边都要安装iperf3工具。在Linux通过yum或者apt-get即可直接安装。  
  
服务端(假设IP为106.14.26.11)运行iperf服务:  
  
#iperf -s  
------------------------------------------------------------                                
Server listening on TCP port 5001              
TCP window size: 85.3 KByte (default)          
这是一台上下行带宽都接近100Mbits/sec的服务器，实测数值不应该与该数值偏离太远。  
  
客户端  
  
测试上行和下行带宽  
  
$iperf  -c 106.14.26.11 -t 60  -d  
------------------------------------------------------------  
Server listening on TCP port 5001  
TCP window size: 85.3 KByte (default)  
------------------------------------------------------------  
------------------------------------------------------------  
Client connecting to 45.32.64.187, TCP port 5001  
TCP window size:  197 KByte (default)  
------------------------------------------------------------  
[  4] local 104.250.144.8 port 50040 connected with 106.14.26.11 port 5001  
[  5] local 104.250.144.8 port 5001 connected with 106.14.26.11 port 34386  
[ ID] Interval       Transfer     Bandwidth  
[  4]  0.0-60.0 sec   658 MBytes  92.0 Mbits/sec #上行带宽  
[  5]  0.0-60.4 sec   628 MBytes  87.4 Mbits/sec #下行带宽  
从测试结果看，上行带宽为92.0Mbits/sec, 下行带宽为87.2Mbits/sec，低于服务器的100Mbits/sec，但是没有低得很离谱。

如果你的客户端无法连接到iperf服务端，可能是主机的防火墙挡住或者VPS提供商做了限制。对于前者，通过iptables -F即可解决；对于后者，有几种解法：进VPS提供商的控制台修改、联系VPS提供商、修改端口为外部可以访问得到的端口。

测量方法  
测量服务器带宽  
使用iperf工具测试服务器带宽，它分为服务端与客户端，两边都要安装iperf工具。在Linux通过yum或者apt-get即可直接安装。  
  
服务端(假设IP为106.14.26.11)运行iperf服务:  
  
#iperf -s  
------------------------------------------------------------                                
Server listening on TCP port 5001              
TCP window size: 85.3 KByte (default)          
这是一台上下行带宽都接近100Mbits/sec的服务器，实测数值不应该与该数值偏离太远。  
  
客户端  
  
测试上行和下行带宽  
  
$iperf  -c 106.14.26.11 -t 60  -d  
------------------------------------------------------------  
Server listening on TCP port 5001  
TCP window size: 85.3 KByte (default)  
------------------------------------------------------------  
------------------------------------------------------------  
Client connecting to 45.32.64.187, TCP port 5001  
TCP window size:  197 KByte (default)  
------------------------------------------------------------  
[  4] local 104.250.144.8 port 50040 connected with 106.14.26.11 port 5001  
[  5] local 104.250.144.8 port 5001 connected with 106.14.26.11 port 34386  
[ ID] Interval       Transfer     Bandwidth  
[  4]  0.0-60.0 sec   658 MBytes  92.0 Mbits/sec #上行带宽  
[  5]  0.0-60.4 sec   628 MBytes  87.4 Mbits/sec #下行带宽  
从测试结果看，上行带宽为92.0Mbits/sec, 下行带宽为87.2Mbits/sec，低于服务器的100Mbits/sec，但是没有低得很离谱。  
  
如果你的客户端无法连接到iperf服务端，可能是主机的防火墙挡住或者VPS提供商做了限制。对于前者，通过iptables -F即可解决；对于后者，有几种解法：进VPS提供商的控制台修改、联系VPS提供商、修改端口为外部可以访问得到的端口。

### 测量延迟、抖动和丢包率

延迟，抖动和丢包率，直接使用ping做测试即可。如果想测量这个值，建议抓100个包测试：

$ ping -c 100 www.baidu.com  
PING www.a.shifen.com (163.177.151.109): 56 data bytes  
64 bytes from 163.177.151.109: icmp\_seq=0 ttl=55 time=26.639 ms  
64 bytes from 163.177.151.109: icmp\_seq=1 ttl=55 time=24.329 ms  
64 bytes from 163.177.151.109: icmp\_seq=2 ttl=55 time=24.794 ms  
64 bytes from 163.177.151.109: icmp\_seq=3 ttl=55 time=22.709 ms  
64 bytes from 163.177.151.109: icmp\_seq=4 ttl=55 time=39.994 ms  
64 bytes from 163.177.151.109: icmp\_seq=5 ttl=55 time=50.786 ms  
。。。省略  
^C  
--- www.a.shifen.com ping statistics ---  
100 packets transmitted, 100 packets received, 0.0% packet loss #最后这个百分比就是丢 包率  
round-trip min/avg/max/stddev = 21.505/26.743/50.786/6.613 ms #分别表示最小延迟、平均延迟、最大延迟、抖动（抖动就是延迟标准差）  
根据上面的结果，我们得到结论:  
  
延迟：26.743ms  
抖动: 6.613ms  
丢包率: 0.0%  
符合指标要求，所以访问百度一定很快。（这里假设带宽是没问题的，测量到服务器的带宽需要服务器支持，所以是测量不了的）

## 网络场景模拟

### 带宽控制

Linux操作系统，提供TC工具进行流量控制，流量控制包括如下功能

流量控制包括以下几种方式：

SHAPING(整形限速)

当流量被限制，它的传输速率就被控制在某个值以下。限制值可以大大小于有效带宽，这样可以平滑突发数据流量，使网络更为稳定。shaping（限制）只适用于向外的流量。

SCHEDULING(调度)

通过调度数据包的传输，可以在带宽范围内，按照优先级分配带宽，高优先级的以太网报文会优先通过。SCHEDULING(调度)也只适于向外的流量。

POLICING(策略)

SHAPING用于处理向外的流量，而POLICIING(策略)用于处理接收到的数据。

DROPPING(丢弃)

如果流量超过某个设定的带宽，就丢弃数据包，不管是向内还是向外。

Example：

Linux流量控制主要分为建立队列、建立分类和建立过滤器三个方面。  
          基本实现步骤。

1. 针对网络物理设备（如以太网卡eth0）绑定一个队列qdisc
2. 在该队列上建立分类class

                   3)  为每一分类建立一个基于路由的过滤器filter

1. 最后与过滤器相配合，建立特定的路由表

环境模拟实例。

 sudo tc qdisc del dev eth0 root 2>/dev/null #清空配置

sudo iptables -F #清空iptables

sudo tc qdisc add dev eth0 root handle 1:0 netem delay 198ms 9.9ms

 sudo tc qdisc add dev eth0 parent 1:1 handle 10: htb default 256 sudo tc class add dev eth0 parent 10: classid 10:1 htb rate 5.0Mbit ceil 5.0Mbit

sudo tc class add dev eth0 parent 10:1 classid 10:143 htb rate 5.0Mbit ceil 5.0Mbit prio 1 sudo tc qdisc add dev eth0 parent 10:143 handle 100143: pfifo sudo tc filter add dev eth0 parent 10: protocol ip prio 100 handle 143 fw classid 10:143 sudo iptables -t mangle -A PREROUTING -s 10.107.105.143 -j MARK --set-mark 143

sudo iptables -t mangle -A PREROUTING -s 10.107.105.143 -j RETURN

sudo iptables -t mangle -A POSTROUTING -d 10.107.105.143 -j MARK --set-mark 143 sudo iptables -t mangle -A POSTROUTING -d 10.107.105.143 -j RETURN

### 丢包控制

可以通过脚本设置丢包率 random.random()会产生一个0-1之间的随机数

指定source IP进行百分比丢包

iptables -I INPUT -s 121.14.48.1 -m statistic --mode random --probability 0.5 -j DROP   
指定dport 进行百分比丢包

iptables -I INPUT -p tcp --dport 8309 -m statistic --mode random --probability 0.5 -j DROP

总体想法：

因为PBFT算法，相当于随机选择leader，任何节点都是对等的，所以可以选择任一节点进行操作，效果都是对等的。

第一步：在任一节点上起一个进程或者线程，进行p2p端口限速、丢包控制、时延控制（时延应该可通过带宽和tps进行模拟）

第二步：再起一个进程，往区块链一直发送交易，使区块链的状态一直在改变。

* 带宽、时延应该提供接口，可以让用户进行操作
* 可以选择对端口或者网卡级别进行网络控制
* 时延时间最好也可以提供api进行控制
* 做一个管理层，用户控制并行进行网络控制，还是串形
* 需要完整的log网络控制log，方便开发定位

需要能够检查链的共识状态，出来问题，让程序停住（鱼哥的monitor脚本）