LVS四种实现模式详解

一、集群cluster

当后端服务器承受不住访问的压力，提高服务器性能的解决方案会极大增加成本时，人们提出了横向扩展的解决方案。增加一台或几台服务器，提供相同的服务，通过前段分发器将访问量均匀的分配到后台服务器上。这种多台服务器组成的数组集合就叫做集群。

集群按功能划分有三种模型：

- 负载均衡集群（loadBalance）
- 高可用性集群（High Availability）
- 高性能集群（High Performance）

二、负载均衡集群

根据某种算法将负载压力合理的分配到集群中的每一台计算机上，以减轻主服务器的压力。

负载均衡实现的方式一般有三种：

- 通过DNS轮询进行负载均衡
- 通过反向代理进行负载均衡
- 通过网络地址转换（NAT）进行负载均衡

下面我要说的linux virtual server就是通过NAT技术进行负载均衡的

三、Linux Virtual Server （linux虚拟服务）

负载均衡的实现可以使用硬件如著名的F5设备，也可以使用软件如：LVS、NGINX反向代理，毕竟硬件是要花钱的，软件是开源免费的，你懂的。

lvs是国内章文嵩教授牵头开发的开源项目，现在已经被收到linux2.6以上的内核版本中，不需要对系统打补丁就可以轻松实现。lvs工作于IOS七层模型的传输层，通过对TCP、UDP、SCTP、IPsec ESP、AH这些工作在四层的协议的支持，根据目标地址和端口做出转发与否的决策，根据调度算法做出转发至哪一个端口的决策。

LVS将其控制程序ipvs嵌套至传输层数据流的Input钩子函数上，ipvs将发送至本控制器主机（director）上的数据流在input链上进行截流，通过对数据报文的分析根据自身的算法将数据流转发至后台真正提供服务的主机（Real Server）上，达到根据后端服务器负载能力均衡分配处理任务的效果。

lvs中的术语：

ClientIP：CIP----------客户端ip

Dirvector Virtual IP：VIP----------控制器上对外开放的ip

Dirvector IP：DIP----------控制器上连接后台服务器的ip

Realserver IP：RIP----------后台服务器的ip

Director----------控制器或调度器

Real Server----------后台提供服务的主机

四、LVS的四种类型

1、lvs-nat(net adress translation)

类似于DNAT，但支持多目标转发。通过修改请求报文的目标地址为根据调度算法所挑选出的某RS的RIP来进行转发；

架构特性：

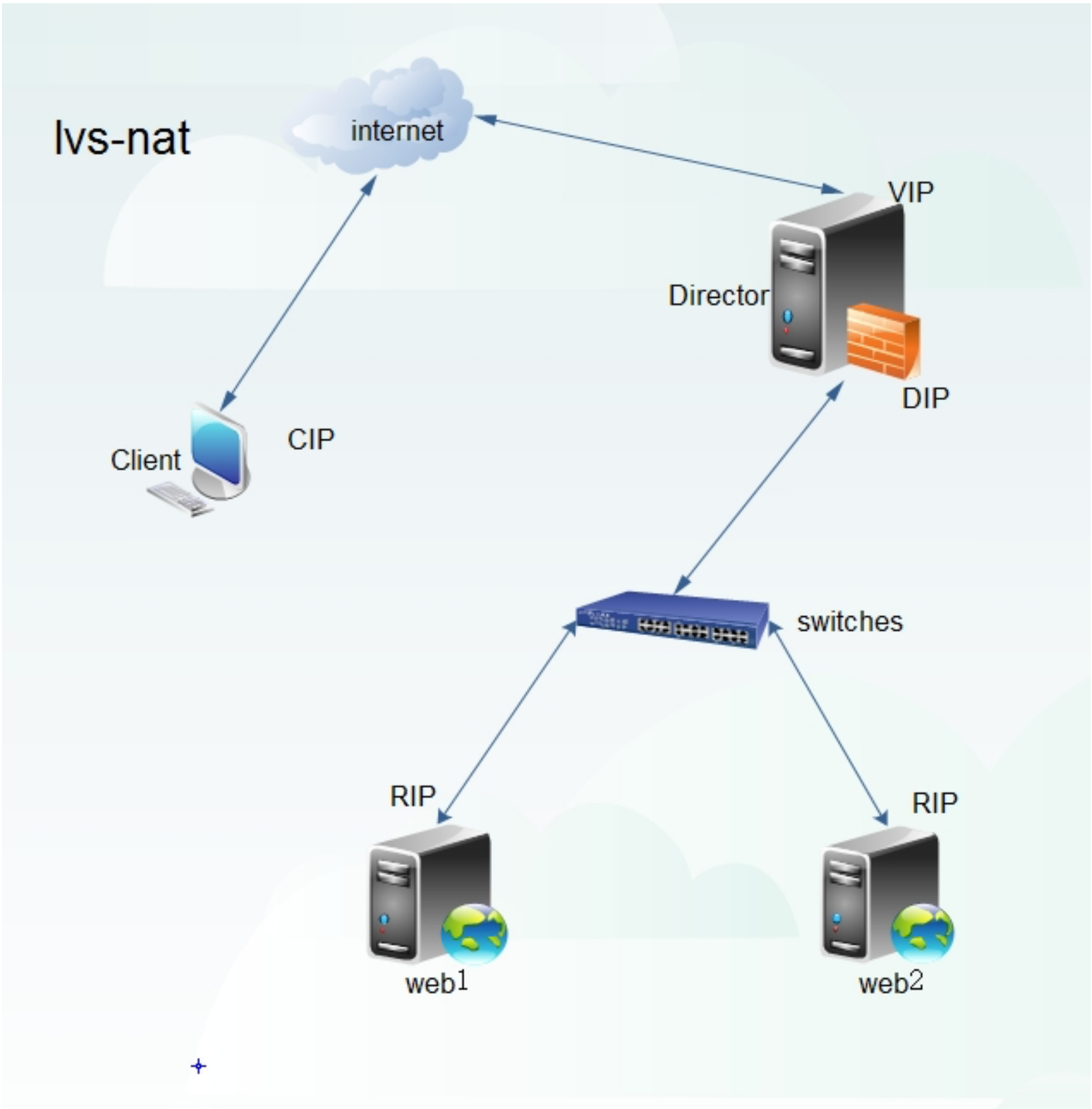
（1）RS应该使用私有地址，即RIP应该为私有地址：各RS的网关必须指向DIP；

（2）请求和响应报文都经由director转发：高负载场景中，dircetor可能成为瓶颈；

（3）支持端口映射；

（4）RS可以使用任意OS；

（5）RS的RIP必须与director的DIP在同一网络；



2、lvs-dr(direct route)

director在实现转发时不修改请求的ip首部，而是通过直接封装MAC首部完成转发：目标MAC是Dircetor根据调度算法挑选出某RS的MAC地址，此类型中，RS也有同Director一样的VIP。

架构特点：

（1）通过静态绑定或内核参数修改或arptables规则实现只有Director上的VIP响应服务请求，RS上的VIP拒绝响应服务请求；

（2）RS上的RIP可以是私有地址，也可以是公网地址；

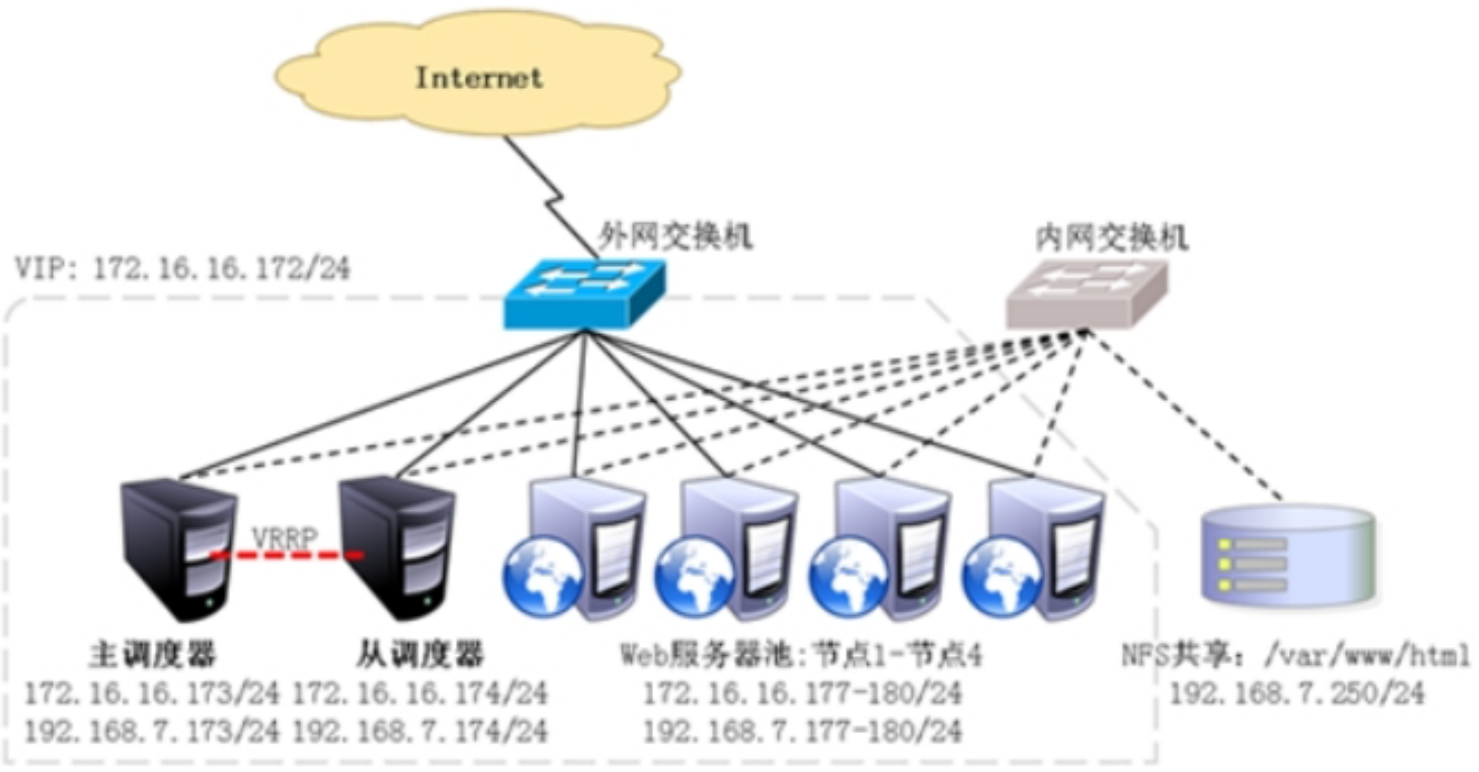
（3）请求报文必须经过Director调度，响应报文直接由RS通过VIP返回给用户；

（4）各RIP必须与DIP在同一网络中；

（5）不支持端口映射；

（6）RS可以使用大多数的OS；

（7）RS的网关一定不能指向Director；



3、lvs-tun(Tunnel transmission)

隧道传输ipip：不修改请求报文ip首部，而是通过ip隧道机制在原有的ip报文之外在封装ip首部，经由互联网把请求报文交给选定的rs；

架构特性:

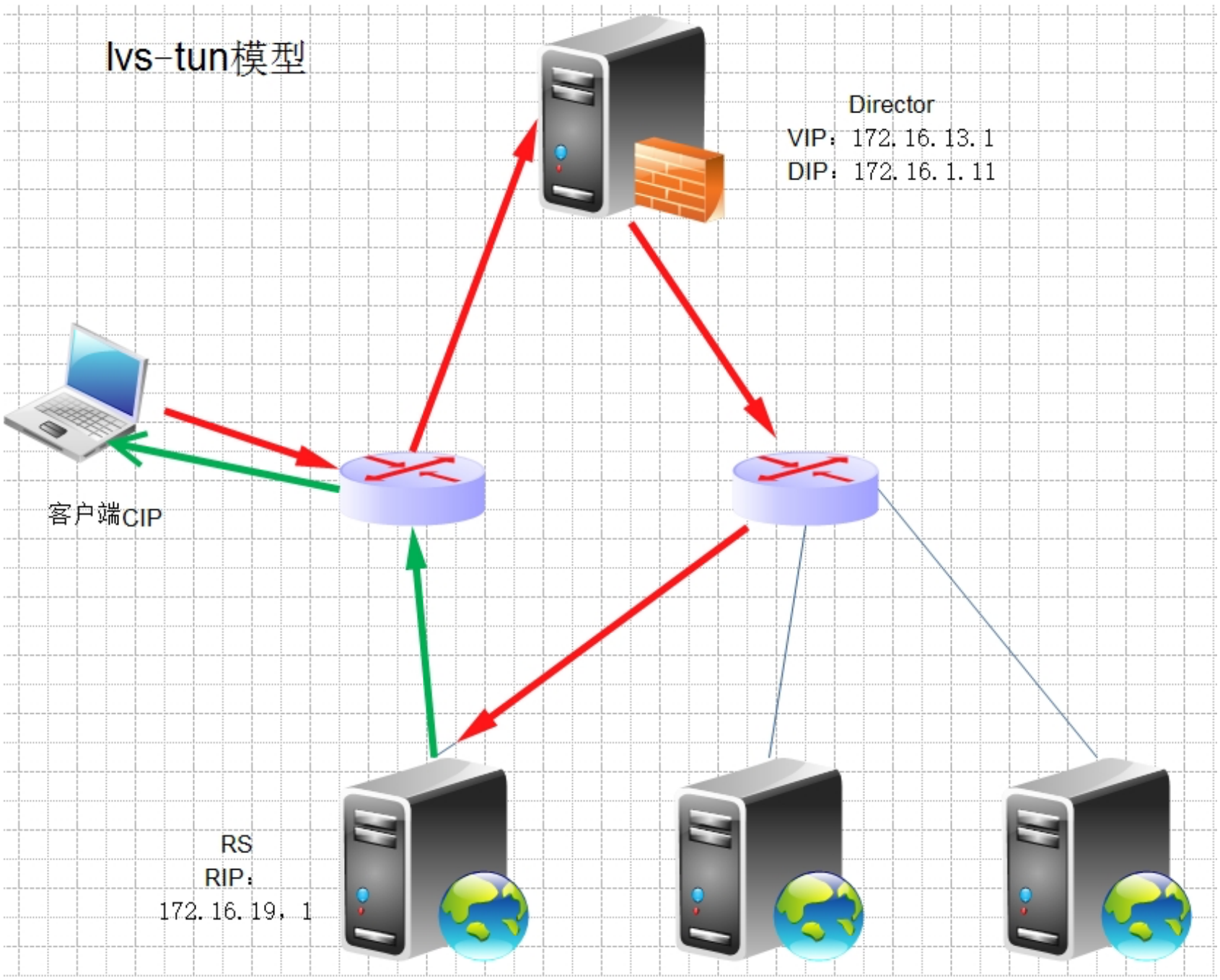
（1）RIP,DIP,VIP都是公网地址；

（2）RS的网关不能，也不可能指向DIP；

（3）请求报文由Director分发，但响应报文直接由RS响应给Client；

（4）不支持端口映射；

（5）RS的OS必须得支持IP隧道，现在只有linux系统支持，windows，bsfdb等不支持；



4、lvs-fullnat（双向转换）

通过请求报文的源地址为DIP，目标为RIP来实现转发：对于响应报文而言，修改源地址为VIP，目标地址为CIP来实现转发：

架构特点：这是一种对nat模型的改进，是一个扩展，使得RS与Director可以处于不同网络。

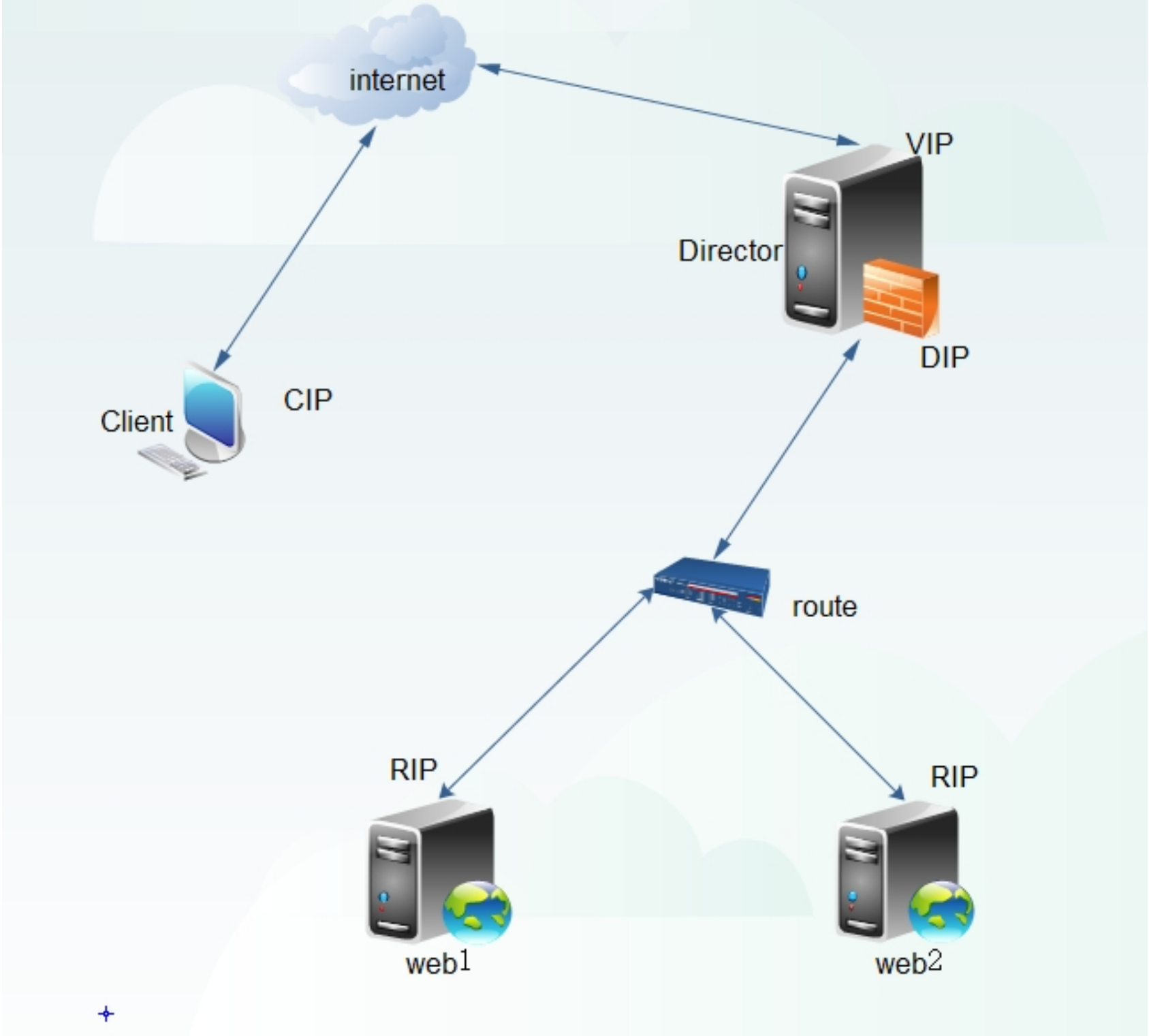
（1）RIP，DIP可以使用私有地址；

（2）RIP和DIP可以不再同一个网络中，且RIP的网关未必需要指向DIP；

（3）支持端口映射；

（4）RS的OS可以使用任意类型；

（5）请求报文经由Director，响应报文也经由Director；



五、LVS的十种调度算法

四种静态算法，不考虑后端服务器实际负载情况：

1、RR

根据规则依次论调,不考虑RS的性能。轮到谁就转发给谁。

2、WRR

加权轮询，加入了weight（权重），可以根据RS的性能为其设置权重值，权重越大功能越强，但是不能发硬当前的服务器的运行的情况。

3、DH

目标地址hash，适用于前段是一个drector后端是几个缓存服务器，当客户端第一次访问到的是RS1的时候，DH这种算法保证，在客户端刷新后还是访问的是RS1。

4、SH

源地址hash，用于保证响应的报文和请求的报文是同一个路径。

六种动态算法，考虑后端服务器当前负载后再进行分配：

1、LC

least connection，当一个用户请求过来的时候，就计算下哪台RS的链接谁最小，那么这台RS就获得了下次响应客户端请求的机会，计算的方法Overhead=active*256+inactive,如果两者的结果是相同的则从LVS中的规则依次往下选择RS。这种算法也是不考虑服务器的性能的。

2、WLC

这个就是加了权重的LC，考虑了RS的性能，即是性能好的就给的权重值大一些，不好的给的权重值小一些。缺点就是如果Overhead相同，则会按规则表中的顺序，由上而下选择RS，Overhead=(active*256+inactive)/weight

3、SED

就是对WLC的情况的补充，Overhead=(active+1)*256/weight，加一，就是为了让其能够比较出大小。

4、NQ

never queue 基本和SED相同，避免了SED当中的性能差的服务器长时间被空闲的弊端，它是第一个请求给性能好的服务器，第二个请求一定是给的空闲服务器不论它的性能的好与坏。以后还是会把请求给性能好的服务器

5、LBLC

它就是动态DH和LC的组合，适用于cache群，对于从来没有来过的那些新的请求会分给当前连接数较少的那台服务器。

6、LBLCR

带有复制功能的LBLC，它的适用场景这里举例说明一下，比如说现在又RS1和RS2,第一次访问RS1的5个请求第二次又来了，理所应到Director将会将其交给RS1，而此时在RS2是非常闲的，所以此时最好的处理方法就是可以将后来的这5个请求分别交给RS1和RS2，所以此时就需要把客户端第一次请求的资源复制下来。（特殊情况）

六、ipvsadm使用说明

1、编译安装或yum源安装

下载：http://www.linuxvirtualserver.org/software/

注意对应自己的内核版本

```
1    ipvsadm-1.24.tar.gz
2    tar zxvf ipvsadm-1.24.tar.gz
3    cd ipvsadm-1.24
4    make
5    make install
```

2、添加一个集群服务director

ipvsadm -A|E -t|u|f service-address [-s scheduler]

-A:添加
-E：修改
-t：tcp

-u：udp

-f：打防火墙标记的tcp或udp

-D：删除

-s：指定调度算法

3、给一个集群服务添加一条RS规则

ipvsadm -a|e -t|u|f service-address -r server-adddress [-g|-i|-m] [-w wegiht]

-a:添加
-e：修改
-d：删除

-g：dr模式直接路由gatway

-i：ipip，tun隧道

-m：NAT模式，地址转换

4、 查看规则：
ipvsadm -L -n -c --stats

-c:列出当前所有的connection

--stats：列出统计数据

--rate：速率数据

5、保存规则：default save to /etc/sysconfig/ipvsadm

6、重载规则：

7、清空所有已匹配的计数器

ipvsadm -Z [-t|u|f service-address]

七、LVS-NAT示例

虚拟网络类型为仅主机模式host-only

```
1
    Client ： 172.16.13.21
2
```

3	Director VIP：172.16.13.10 DIP:192.168.13.10
4	WebServer1:192.168.13.13 WebServer2:192.168.13.14

Director配置：

Webserver配置：

安装nginx当作web服务器，web1主页写上192.168.13.13；web2主页写上192.168.13.14；

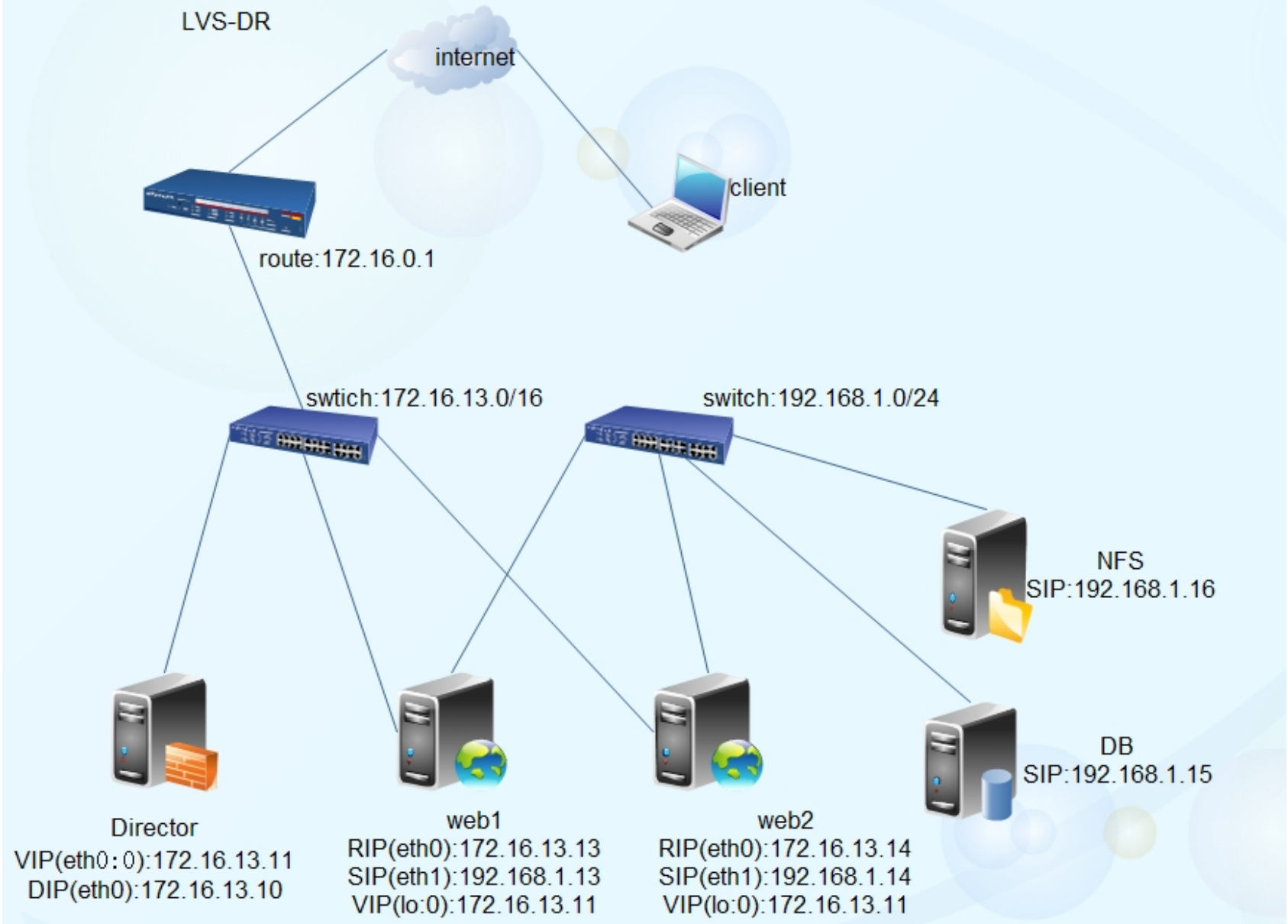
将WebServer网关指定为192.168.13.10

Client测试：

浏览器输入http://172.16.13.10/index.html

八、LVS-DR示例

在同一网段网络拓扑如下



Director配置:

```
1
echo 1 > /proc/sys/net/ipv4/ip_forward
2
ifconfig eth0 172.16.13.10 up
3
ifconfig eth0:0 172.16.13.11 netmask 255.255.255.255 broadcast 172.16.1
4
route add -host 172.16.13.11 dev eth0:0       //以防万一，不是必须
5
ipvsadm -A -t 172.16.13.11:80 -s rr
6
ipvsadm -a -t 172.16.13.11:80 -r 172.16.13.13:80 -g
7
ipvsadm -a -t 172.16.13.11:80 -r 172.16.13.14:80 -g
```

WebServer1配置:

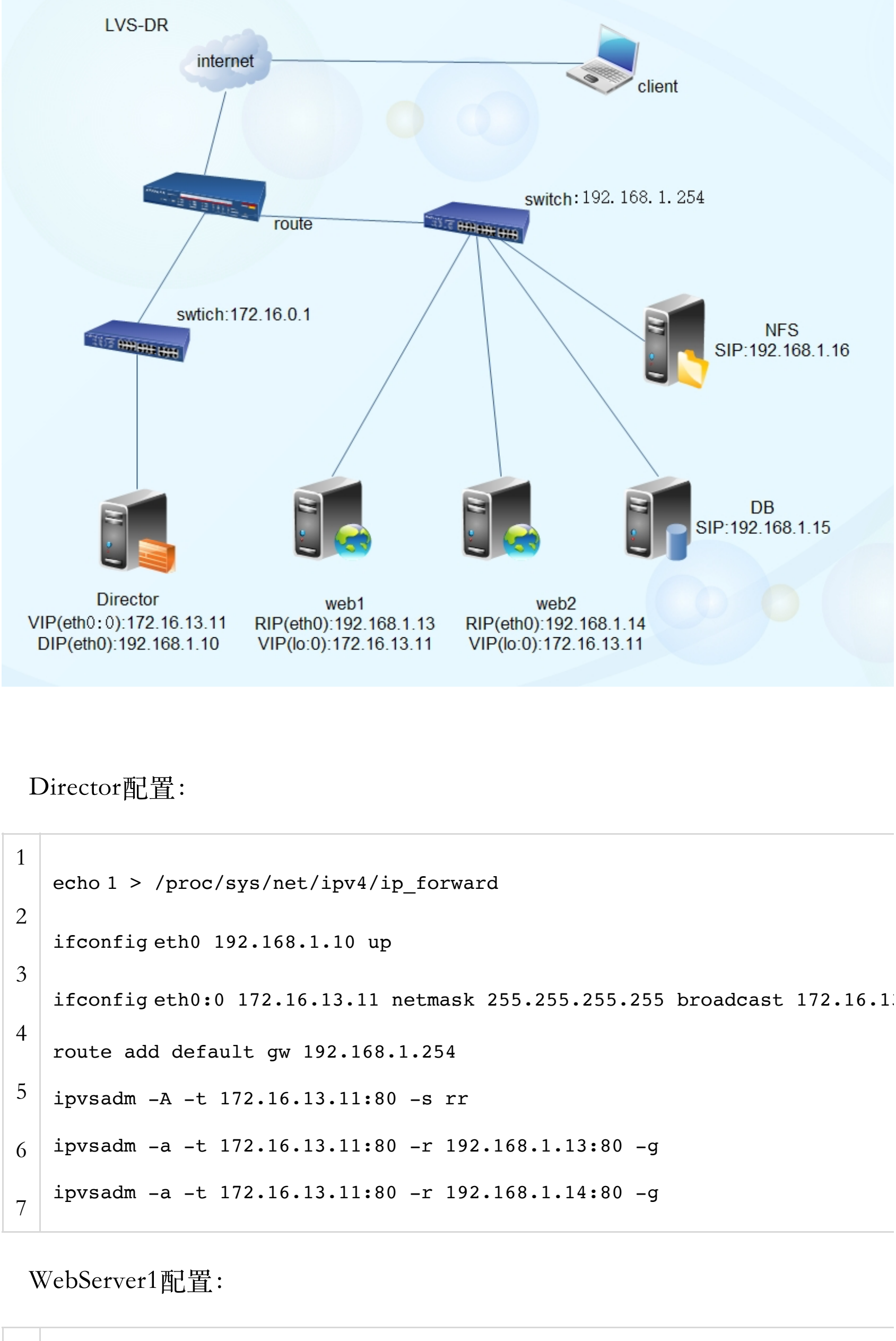
```
1
ifconfig eth0 172.16.13.13 up
```

```
ifconfig eth1 192.168.1.13 up

echo 1 > /proc/sys/net/ipv4/conf/all/arp_ignore

echo 1 > /proc/sys/net/ipv4/conf/lo/arp_ignore

echo 2 > /proc/sys/net/ipv4/conf/all/arp_announce

echo 2 > /proc/sys/net/ipv4/conf/lo/arp_announce

ifconfig lo:0 172.16.13.11 netmask 255.255.255.255 broadcast 172.16.13.

route add -host 172.16.13.11 dev lo:0
```

WebServer2配置：

```
ifconfig eth0 172.16.13.14 up

ifconfig eth1 192.168.1.14 up

echo 1 > /proc/sys/net/ipv4/conf/all/arp_ignore

echo 1 > /proc/sys/net/ipv4/conf/lo/arp_ignore

echo 2 > /proc/sys/net/ipv4/conf/all/arp_announce

echo 2 > /proc/sys/net/ipv4/conf/lo/arp_announce

ifconfig lo:0 172.16.13.11 netmask 255.255.255.255 broadcast 172.16.13.

route add -host 172.16.13.11 dev lo:0
```

在不同网段网络拓扑图



Director配置:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
ifconfig eth0 192.168.1.10 up
ifconfig eth0:0 172.16.13.11 netmask 255.255.255.255 broadcast 172.16.1
route add default gw 192.168.1.254
ipvsadm -A -t 172.16.13.11:80 -s rr
ipvsadm -a -t 172.16.13.11:80 -r 192.168.1.13:80 -g
ipvsadm -a -t 172.16.13.11:80 -r 192.168.1.14:80 -g
```

WebServer1配置:

```
ifconfig eth0 192.168.1.13 up
```

```
route add default gw 192.168.1.254

echo 1 > /proc/sys/net/ipv4/conf/all/arp_ignore

echo 1 > /proc/sys/net/ipv4/conf/lo/arp_ignore

echo 2 > /proc/sys/net/ipv4/conf/all/arp_announce

echo 2 > /proc/sys/net/ipv4/conf/lo/arp_announce

ifconfig lo:0 172.16.13.11 netmask 255.255.255.255 broadcast 172.16.13.

route add -host 172.16.13.11 dev lo:0
```

WebServer2配置:

```
ifconfig eth1 192.168.1.14 up

route add default gw 192.168.1.254

echo 1 > /proc/sys/net/ipv4/conf/all/arp_ignore

echo 1 > /proc/sys/net/ipv4/conf/lo/arp_ignore

echo 2 > /proc/sys/net/ipv4/conf/all/arp_announce

echo 2 > /proc/sys/net/ipv4/conf/lo/arp_announce

ifconfig lo:0 172.16.13.11 netmask 255.255.255.255 broadcast 172.16.13.

route add -host 172.16.13.11 dev lo:0
```

九、关于内核参数

arp_igonre

0: 如果路由向我的一个网络接口发送广播，请求的是本机上的其他接口，就告知它（默认值）；

1：如果路由向我的一个网络接口发送广播，请求的是本机上的其他接口，就拒绝它；

arp_announce

0: 一旦我接入一个网络，就向这个网络内的所有主机通告自己所有的网络接口信息，不隐藏（默认值）；

1：一旦我接入一个网络，就将接入网络这一个接口的信息通告其他主机，本机上的其他接口信息可以通告也可以不通告；

2：一旦我接入一个网络，就将接入网络这一个接口的信息通告其他主机，本机上的其他接口信息绝对不通告出去；

十、脚本

lvs-dr脚本

Director脚本示例

```
#!/bin/bash
vip=172.16.13.11
rip=('172.16.13.13' '172.16.13.14')
weight=('1' '2')
port=80
scheduler=rr
ipvstype='-g'
case $1 in
start)
    iptables -F -t filter
    ipvsadm -C
    ifconfig eth0:0 $vip broadcast $vip netmask 255.255.255.255 up
```

```
14      route add -host $vip dev eth0:0
15      echo 1 > /proc/sys/net/ipv4/ip_forward
16      ipvsadm -A -t $vip:$port -s $scheduler
17      [ $? -eq 0 ] && echo "ipvs service $vip:$port added." || exit 2
18      for i in `seq 0 $[${
19          ipvsadm -a -t $vip:$port -r ${rip[$i]} $ipvstype -
w ${weight[$i]}
20          [ $? -eq 0 ] && echo "RS ${rip[$i]} added."
21      done
22      touch /var/lock/subsys/ipvs
23  ;;
24  stop)
25      echo 0 > /proc/sys/net/ipv4/ip_forward
26      ipvsadm -C
27      ifconfig eth0:0 down
28      rm -f /var/lock/subsys/ipvs
29      echo "ipvs stopped."
30  ;;
31  status)
32      if [ -f /var/lock/subsys/ipvs ]; then
33  echo "ipvs is running."
34  ipvsadm -L -n
35  else
36  echo "ipvs is stopped."
37  fi
38  ;;
39  *)
```

```
    echo "Usage: `basename $0` {start|stop|status}"
exit 3
;;
esac
```

RS脚本示例

```
#!/bin/bash
vip=172.16.13.11
interface="lo:0"
case $1 in
start)
    echo 1 > /proc/sys/net/ipv4/conf/all/arp_ignore
    echo 1 > /proc/sys/net/ipv4/conf/lo/arp_ignore
    echo 2 > /proc/sys/net/ipv4/conf/all/arp_announce
    echo 2 > /proc/sys/net/ipv4/conf/lo/arp_announce
    ifconfig $interface $vip broadcast $vip netmask 255.255.255.255 up
    route add -host $vip dev $interface
;;
stop)
    echo 0 > /proc/sys/net/ipv4/conf/all/arp_ignore
    echo 0 > /proc/sys/net/ipv4/conf/lo/arp_ignore
    echo 0 > /proc/sys/net/ipv4/conf/all/arp_announce
```

```
    echo 0 > /proc/sys/net/ipv4/conf/lo/arp_announce
    ifconfig $interface down
;;
status)
    if ifconfig lo:0 |grep $vip &> /dev/null; then
echo "ipvs is running."
    else
echo "ipvs is stopped."
    fi
;;
*)
    echo "Usage: `basename $0` {start|stop|status}"
    exit 1
esac
```

lvs-nat脚本

Director脚本示例

```

#!/bin/bash
chmod 755 /etc/rc.d/init.d/functions
. /etc/rc.d/init.d/functions
VIP=172.16.13.11
DIP=192.168.1.11
RIP1=192.168.1.13
RIP2=192.168.1.14
```

```
case "$1" in
start)
  /sbin/ifconfig eth0:1 $VIP netmask 255.255.255.0 up
  echo 1 > /proc/sys/net/ipv4/ip_forward
  /sbin/iptables -F
  /sbin/iptables -Z
  /sbin/ipvsadm -C
  /sbin/ipvsadm -A -t $VIP:80 -s rr
  /sbin/ipvsadm -a -t $VIP:80 -r $RIP1 -m
  /sbin/ipvsadm -a -t $VIP:80 -r $RIP2 -m
  /bin/touch /var/lock/subsys/ipvsadm.lock
;;
stop)
  echo 0 > /proc/sys/net/ipv4/ip_forward
  /sbin/ipvsadm -C
  ifconfig eth0:1 down
  rm -rf /var/lock/subsys/ipvsadm.lock
;;
status)
  [ -
e /var/lock/subsys/ipvsadm.lock ] && echo "ipvs is running..." || echo
;;
*)
  echo "Usage: $0 {start|stop}"
;;
esac
```

```
32
33
```

RS脚本示例

```
1
2  #!/bin/bash
3  VIP=172.16.13.11
4  chmod 755 /etc/rc.d/init.d/functions
5  . /etc/rc.d/init.d/functions
6  case "$1" in
   start)
7  echo " start LVS of REALServer"
8  /sbin/ifconfig lo:0 $VIP broadcast $VIP netmask 255.255.255.255 up
9  /sbin/route add -host $VIP dev lo:0
10 echo "1" >/proc/sys/net/ipv4/conf/lo/arp_ignore
11 echo "2" >/proc/sys/net/ipv4/conf/lo/arp_announce
12 echo "1" >/proc/sys/net/ipv4/conf/all/arp_ignore
13 echo "2" >/proc/sys/net/ipv4/conf/all/arp_announce
14 sysctl -p >/dev/null 2>&1
15 ;;
16 stop)
17 /sbin/ifconfig lo:0 down
18 echo "close LVS Directorserver"
19 echo "0" >/proc/sys/net/ipv4/conf/lo/arp_ignore
20 echo "0" >/proc/sys/net/ipv4/conf/lo/arp_announce
```

```
echo "0" >/proc/sys/net/ipv4/conf/all/arp_ignore
echo "0" >/proc/sys/net/ipv4/conf/all/arp_announce
;;
*)
echo "Usage: $0 {start|stop}"
exit 1
esac
```

十一、健康状态检查

在Director上实现后端RS健康状态检查的脚本：

lvs本身不支持对RS的健康状态作检测，所以应该写脚本实现；对脚本有如下要求：

健康：周期性检查机制

状态发生转变时，要作出相应处理

up --> down: 建议要至少确认三次；

down --> up: 建议一次以上（含一次）；

下线处理机制：

⑴ 设置权重为0；

⑵ 将相应的RS从ipvs的可用RS列表中移除；

上线处理机制：

⑴ 设置为正常权重；

⑵ 将相应的RS添加至ipvs的可用RS列表；

如何做健康状态检查：

三种方案：

IP层：ping等主机在线状态探查工具；

传输层：端口扫描工具探查服务在线状态；

应用层：请求专用于健康状态检查的资源或者某正常资源；

备用服务器：

sorry server, backup server

可以在Director上直接实现：即配置director成为web服务，仅提供有限资源，在所有RS都故障时，方才启用此server；

```
#!/bin/bash
fwm=10
sorry_server='127.0.0.1'
lvstype='-m'
checkloop=3
logfile=/var/log/ipvs_health_check.log
rs=('192.168.1.13' '192.168.1.14')
rw=('1' '1')
rsstatus=(0 0)
addrs() {
ipvsadm -a -f $fwm -r $1 $lvstype -w $2
```

```
[ $? -eq 0 ] && return 0 || return 1
}
delrs() {
ipvsadm -d -f $fwm -r $1
[ $? -eq 0 ] && return 0 || return 1
}
chkrs() {

local i=1
while [ $i -le $checkloop ]; do
if curl --connect-timeout 1 -s http://$1/index.html | grep -i "real[[:space:]]* server" &> /dev/null; then
return 0
fi
let i++
sleep 2
done
return 1
}

initstatus() {
for host in `seq 0 $[${
        if chkrs ${rs[$host]}; then
 if [ ${rsstatus[$host]} -eq 0 ]; then
rsstatus[$host]=1
        fi
else
```

```
39 if [ ${rstatus[$host]} -eq 1 ]; then
40 rsstatus[$host]=0
41 fi
   fi
42 done
43 }
44 initstatus
45 while :; do
46 for host in `seq 0 $[${
47     if chkrs ${rs[$host]}; then
48         if [ ${rsstatus[$host]} -eq 0 ]; then
49         addrs ${rs[$host]} ${rw[$host]}
50         [ $? -eq 0 ] && rsstatus[$host]=1
51         fi
    else
52         if [ ${rsstatus[$host]} -eq 1 ]; then
53         delrs ${rs[$host]}
54         [ $? -eq 0 ] && rsstatus[$host]=0
55         fi
56      fi
done
57 sleep 10
58 done
59
60
61
62
```

63	