参考：<http://blog.csdn.net/yinwenjie/article/details/46845997>

[**架构设计：负载均衡层设计方案（4）——LVS原理**](http://blog.csdn.net/yinwenjie/article/details/46845997)

之前我们花了两篇文章的篇幅，详细讲解了Nginx的原理、安装和特性组件。请参看《负载均衡层设计方案（2）——Nginx安装》（[http://blog.csdn.net/yinwenjie/article/details/46620711](http://blog.csdn.net/yinwenjie/article/details/46620711" \t "_blank)）和《**[架构](http://lib.csdn.net/base/architecture" \o "大型网站架构知识库" \t "_blank)**设计：负载均衡层设计方案（3）——Nginx进阶》（[http://blog.csdn.net/yinwenjie/article/details/46742661](http://blog.csdn.net/yinwenjie/article/details/46742661" \t "_blank)）两篇文章。虽然不包括Nginx的所有知识（也不可能全部包括），但是足够读者将Nginx应用到实际的生产中，并进行重要特性的优化，后面有时间我们还会重新回到Nginx的讲解上。从本篇文章开始，我们将开始介绍LVS技术，包括基本概念、简单使用和进阶使用。

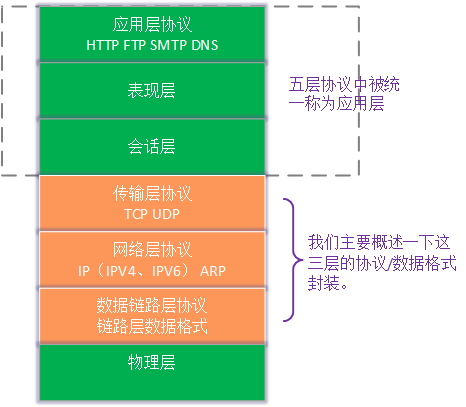
1、LVS介绍

请自行Google或者百度。

2、网络协议基础知识

根据官方文档LVS支持三种负载工作方式：NAT方式、TUN方式和DR方式。为了说明这三种方式的工作原理，我们首先需要了解一下基础的IP/TCP报文（注意，IP报文和TCP报文是两种不同的报文格式），以及链路层对IP数据的封装方式。然后我们采用看图说话的方式，以图文结合的方式为您介绍这三种工作方式中对报文或重写或封装的过程。

为了说清楚我们将要讲解的基础知识，就要提到OSI7层网络模型。



章文嵩博士及其团队是我的偶像，LVS体系之所以高效是因为其直接对链路层报文、IP报文、TCP报文进行了修改或封装。所以要真正理解LVS的三种工作方式，就不能像网络上的抄袭贴一样转几句不知所以的文字就完了，必须对链路层报文、网络层报文、传输层报文有所了解。下面我们就进行概述。

为了保证这个系列的博文不发生偏差，我们只讲解其中需要用到的属性和含义，如果读者对网络的核心原理有兴趣，可以读读《TCP/IP详解，卷1：协议》这本书。

2.1、链路层报文

链路层的数据格式有一个共同特点，都包括目标MAC地址和源MAC地址。下面这个图主要说明了我们最常用的Ethernet帧（以太帧）的报文格式：



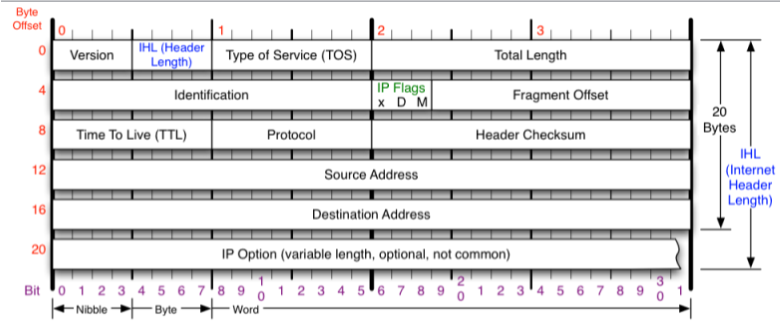
* 目标MAC地址/源MAC地址：00:00:00:00:00:00——FF:FF:FF:FF:FF:FF这个范围是全球MAC地址的可用范围。一张物理网卡肯定有一个唯一的MAC地址。实际上网络层常用的IP协议，就是基于MAC地址的。一个子网范围内某个IP对应的MAC地址是通过ARP查询协议从NAT设备（可能是路由器、交换机或者网络代理设备）上查询得到的。
* 上层协议类型：链路层的报文是为了承载网络层的协议而存在的，所以链路层的数据格式中需要有一个属性说明这个链路层所承载的上层协议是什么类型。
  + IPv4: 0x0800
  + ARP:0x0806
  + PPPoE:0x8864
  + IPV6: 0x86DD
* 封装的上层数据：最多可以有1500个字节。

请记住这个链路层的数据格式，因为讲到LVS-DR方式的时候，主要就是对链路层的数据格式进行修改，而不会对IP数据报文和TCP数据报文进行修改。

2.2、网络层IP报文

* TCP协议和IP协议是两种不同的协议。对应的，也就是两种不同的描述格式。
* IP协议是网络层协议，顾名思义，就是用来描述整个网络构成情况的；TCP协议是通讯层协议，是用来表示两个或多个网络上的点如何进行通信和当前通信状态的。
* 这两种协议有很多共同特点，例如这两种协议都分为“头部”和“数据部”；针对IP协议来说，TCP协议的描述就存放在其“数据部”。

我们首先来看看IP协议是怎么描述的：



好吧，图我是直接从百度百科直接粘过来的，因为要我来画，结构也是这样^\_^。其中有几个重要的我们后面要使用的属性，要给大家说一下：

* header.version：IP协议的版本号，你猜对了就是IPV4还是IPV6。4表示IPV4版本，6表示IPV6。你要问我什么是IPV4，还是IPV6：192.168.220.141，这就是IPV4的格式；FE80:0000:0000:0000:AAAA:0000:00C2:0002，这就是IPV6的格式。
* header.Total Length：总长度，这个总长度是IP头和IP数据两个区域的总长度。主要还是用于生成头验证码和为了**[操作系统](http://lib.csdn.net/base/operatingsystem" \o "操作系统知识库" \t "_blank)**处理方便。
* header.IP Flags：这个位置有3位000，但实际只有后两位才有值010，这个就是“D”位=1，这个时候表示由于要传输的整个数据不大，所以这个IP数据报的数据部分已经描述了整个数据描述，不需要进行IP数据报的分片；”D”为=0，表示要传输的整个数据比较大，所以IP数据报进行了拆分，这个时候就要用到最后一位了：最后一位“M”中“1”表示还有后续分片；0表示这个数据报就是IP分片的最后一个数据片了。
* header.Protocol：IP协议是网络层协议，在网络层以上是传输层协议。TCP、UDP、ICMP和IGMP是传输层协议。这个位置的8位说明IP协议数据部分携带的是哪种上层协议。
* header.Source Address：这个当然就是IP数据报的来源地址咯。
* header. Destination Address：这个当然就是IP数据报的目标地址咯。
* header.checksum：首部校验值。这个值校验IP数据报首部的传输完整性（注意校验不包括IP数据报的数据部分）。这就意味着NAT设备重写这个数据报的来源或者目标IP后，校验值要重新进行计算。Source Address、Destination Address、Checksum是各种NAT设备主要的改写属性。而且很多时候NAT设备只改写这三个值就可以实现IP数据报的转发（当然TCP报文中的端口也会被改写，以便在端口映射的情况下进行端口转换）。

2.3、传输层TCP报文

上文已经说过，TCP的报文信息是装在到IP报文的数据部分的，当成网络上进行传输的数据从Srouce Address传到Destination Address中。下面是TCP报文的信息：



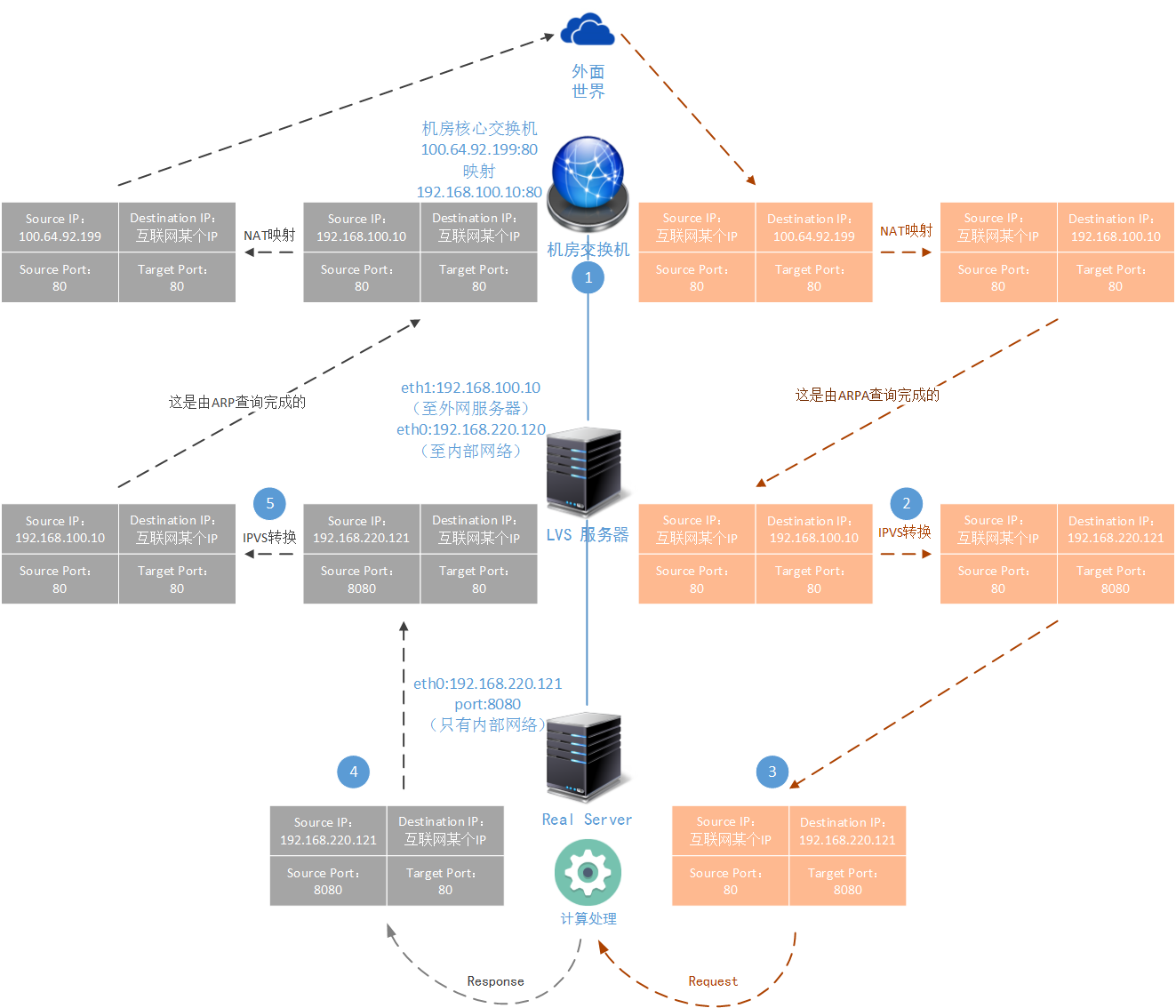
* 头.源端口号：TCP信息来源的端口号。
* 头.目的端口号：TCP信息数据的目标端口。
* 头.状态位（URG/ACK/PSH/RST/SYN/FIN）：如果您已经看到我之前写的《标准Web系统的架构分层》（[http://blog.csdn.net/yinwenjie/article/details/46480485](http://blog.csdn.net/yinwenjie/article/details/46480485" \t "_blank)）这边文章的第3.2小节，那么您对ACK、SYN、FIN这三个标记肯定不陌生，因为TCP的三次握手和连接中断就要用到这三个标记，需要注意SYN SEQ和ACK SEQ就是TCP数据报的确认号；另外解释一下PSH和RST两个状态标记。应用层的TCP数据报有一个缓存区，也就是说多个正确的TCP数据报会首先放到这个缓存区，达到一定条件后，再推送给上层的应用层协议，例如http。PSH为1的时候，表示不需要再等到后续的TCP数据报文了，直接将目前接收方缓存中的tcp数据报进行数据段组合后推送给上层协议，并且清空缓存区；RST表示复位，您可以理解成放弃当前缓存区的所有未发送给上层协议的TCP数据报文，一般这种情况都是TCP报文传输出现了问题。
* 头.TCP校验和：TCP报文的校验和比起IP头校验要稍微复杂点。TCP校验的输入包括三部分：TCP伪首部、TCP首部长度和TCP数据部长度。TCP伪首部是一个虚拟概念，它包括承载TCP数据报文的IP报文的一部分，和TCP首部的一部分数据（源IP、目标IP、IP报文中的protocoly、以及TCP报文的报文头长度和TCP报文的数据长度）。

从上面的描述可以看出，一旦IP报文中的源IP和目标IP发生改变了，TCP报文校验信息就会改变。

3、LVS的三种工作方式

3.1、LVS-NAT工作方式

NAT方式是一种由LVS Master服务节点收到数据报，然后转给下层的Real Server节点，当Real Server处理完成后回发给LVS Master节点然后又由LVS Master节点转发出去的工作方式。LVS的管理程序IPVSADMIN负责绑定转发规则，并完成IP数据报文和TCP数据报文中属性的重写。请用几分钟时间仔细看看下图（为了简单，图里面只画了一个Real Server。如果看不清楚，可点击右键“查看原图”）：



* 1、在正式的机房环境中，一般有两种方式为一个机器分配外网地址：在核心交换机上直接绑定外网地址到主机网卡的，这样使用ifconfig命令看到的IP地址为外网地址；在核心交换机上使用映射规则，将一个外网地址映射到内网地址，这样使用ifconfig命令看到的IP地址为内网地址。上图中我们采用的是后一种映射规则。如果使用前一种外网IP的分配规则，也不会影响LVS NAT的工作方式，因为这个IP被限制在LVS NAT工作以外。只不过eth1的IP从192.168.100.10换成100.64.92.199而已。
* 2、我们用中文描述一下转换规则：**凡是发送到“192.168.100.10:80”的数据报，目标地址全部改写为“192.168.220.121:8080”**，所以来自于100.64.92.199:80的报文被改写了。被改写的属性包括：IP.header.destinationIP、IP.header.checksum、TCP.header.sourcePort、TCP.header.targetPort、TCP.header.checksum。注意IP报文的Source IP不会发生变化，还是“互联网某个IP”。
* 3、这个包最终被送到了192.168.220.121的8080端口进行处理，并由下层的Real Server生成了返回的数据报（至于这个Real Server是不是“真正的Real Server”，LVS不会关心）。你要问它是怎么被发送过去的，请参考ARP查询协议。
* 4、注意：因为LVS服务器和Real Server（可能有多个），组成了一个封闭的局域网。除了LVS节点以外，这个子网的任何节点都是无法访问外网的。所以要求192.168.220.121这个Real Server直接把数据报给“互联网某个IP”这个外网地址，显然是不行的，因为在局域网中根本就找不到这个IP。Real Server只能将数据报返给网关，再由网关去寻找这个外网地址。整个服务器中只有LVS节点能够找到这个外网地址，**这就是为什么在LVS-NAT工作模式下，所有的Real Server节点必须设置自己的Gateway为LVS节点的原因**。
* 5、收到来源于“192.168.220.121:8080”的数据报文后，IPVS又要进行数据报文的重写了。重写规则是：**凡是来源于“192.168.220.121:8080”的数据报，源地址全部改写为“192.168.100.10:80”**。于是数据报文的Source IP、Source Port被改写成“192.168.100.10:80”。**在外层的核心交换机（或者是机房以外的请求方）看来，LVS接受了数据报，并进行了处理，返回了结果。它并不知道LVS节点的下层还有什么。**

LVS-NAT的优点在于：

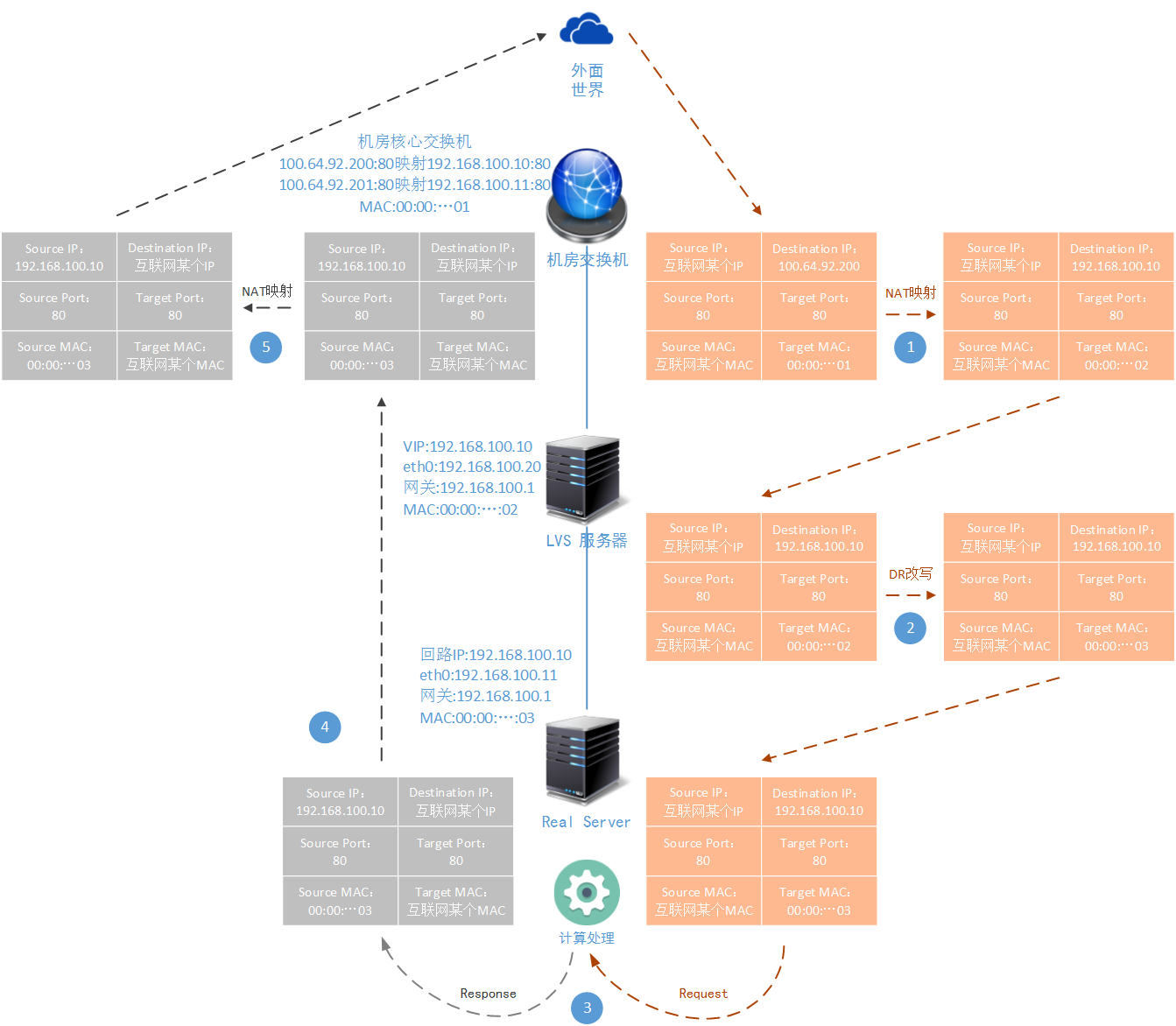
* 配置管理简单。LVS-NAT的工作方式是LVS三种工作模式中最容易理解、最容易配置、最容易管理的工作模式。
* 节省外网IP资源，一般机房分配给使用者的IP数量是有限的，特别是您购买的机架的数量不多时。LVS-NAT工作方式将您的系统架构封装在局域网中，只需要LVS有一个外网地址或外网地址映射就可以实现访问了。
* 系统架构相对封闭。在内网环境下我们对防火墙的设置要求不会很高，也相对容易进行物理服务器的运维。您可以设置来源于外网的请求需要进行防火墙过滤，而对内网请求开放访问。
* 另外改写后转给Real Server的数据报文，Real Server并不会关心它的真实性，只要TCP校验和IP校验都能通过，Real Server就可以进行处理。所以LVS-NAT工作模式下Real Server可以是任何操作系统，只要它支持TCP/IP协议即可。
* 当然作为[**Linux**](http://lib.csdn.net/base/linux)系统忠实拥护者，我并不建议使用Window服务器。但如果您的Real Server是.Net系统，又有业务场景需要用到LVS，那么LVS-NAT可能是一个不错的选择。

LVS-NAT的缺点是由于这种转发模式本身所造成的：

* 转发点就是瓶颈点。您可以想象100台Real Server将处理结果全部转到一个LVS进行发送是一个怎么样的场景。事实上，LVS-NAT的极限负载是达不到100台Real Server的。

3.2、LVS-DR工作方式

LVS的DR工作模式，是目前生产环境中最常用的一种工作模式，网上的资料也是最多的，有的文章对DR工作模式的讲解还是比较透彻的。这里我们通过图文的方式再向您介绍一下DR的工作模式（同样，如果看不清楚，请右键“查看原图”）：



上图反映了DR模式的整个工作过程，同样为了简单起见，这里的Real Server也只画了一个。如果是多个Real Server的话，LVS会通过调度**[算法](http://lib.csdn.net/base/datastructure" \o "算法与数据结构知识库" \t "_blank)**来决定发往哪台Real Server。LVS-DR工作模式的几个关键点在于：

* 被Real Server处理后形成的响应报文，不再回发到LVS节点，而是直接路由给中心交换机然后发送出去。省去了LVS-NAT方式中的LVS回发过程。
* LVS节点只会改写链路层的报文封装，对网络层和传输层报文是不进行改写的。
* 有网帖说DR工作模式，不能跨子网，也就是说LVS节点和各个Real Server节点必须处于同一个网段中。这是为什么呢？事实又真的是这样吗？很多网络帖子没有回答这个问题，这篇文章马上回答一下（实际上章文嵩先生已经回答过这个问题）。
* 使用DR模式时，需要Real Server设置LVS上的VIP为自己的一个回环IP，不然包会被丢弃。这又是为什么呢？很多网贴同样没有回答这个问题，好吧，我们马上回答一下。

先来说一说上图的工作原理：

* 1、同样的，我们为了演示整个生产环境中，从机房中心交换机收到一个数据报文后开始讲解。中心交换机同样采取的IP映射方式。但是与LVS-NAT方式不一样，Real Server在机房的中心交换机上也需要绑定一个外网映射。这样保证Real Server回发的响应报文能够被发送到外网。
* 2、LVS节点接收到请求报文后，会改写报文的数据链路层格式。将Target Mac改写成Real Server的Mac，**但是网络层和传输层报文不会改写**，然后重新回发给交换机。这里就涉及一个问题，现在**target Mac和Destination IP的对应关系的错误的，这个数据报文到了交换机后，由于这种错位的关系，是不能进行三层交换的，只能进行二层交换（一旦进行IP交换，数据报文的验证就会出错，被丢弃）。所以LVS-DR方式要求Real Server和LVS节点必须在同一个局域网内，或者这样说更确切：LVS节点需要找到一个二层链路，将改写了Mac地址的报文发送给Real Server，而不能进行三层交换的校验**。这样来看，实际上LVS节点和Real Server界面不一定要在同一个子网，您用一个独立网卡独立组网，传送报文也是可行的。
* 3、通过二层交换，数据被发送到Real Server节点。那么Real Server节点怎么来判断这个包的正确性呢？首先当然是传输层TCP/IP报文校验没有问题，LVS-NAT没有改写TCP/IP，当然校验就没有问题（除非报文本身就存在问题）；然后是链路层的MAC地址能够被识别，这时就是回环IP的功劳了。对于Real Server节点来说，192.168.100.10这个VIP就是自己的回环IP，绑定的MAC也就是被LVS替换后的target mac。**那么Real Server会认为这个包是在本机运行的某一个应用程序通过回环IP发给自己的，所以这个包不能被丢弃，必须处理**。
* 4、被处理后的生成的响应报文，被直接发送给网管。这个就没有太多的解释的了，只要保证Real server的默认路由设置成到核心交换机的192.168.100.1就OK了。另外，需要说明的是，**由于LVS-DR模式并没有更改原有的IP报文和TCP报文，所以LVS-DR模式本身是不支持端口映射的**，实际上在日常使用实践中，我们一般使用Nginx做端口映射，因为: 灵.活.。

LVS-DR工作模式的优点在于：

* 解决了LVS-NAT工作模式中的转发瓶颈问题，能够支撑规模更大的负载均衡场景。
* 比较耗费网外IP资源，机房的外网IP资源都是有限的，如果在正式生产环境中确实存在这个问题，可以采用LVS-NAT和LVS-DR混合使用的方式来缓解。

LVS-DR当然也有缺点：

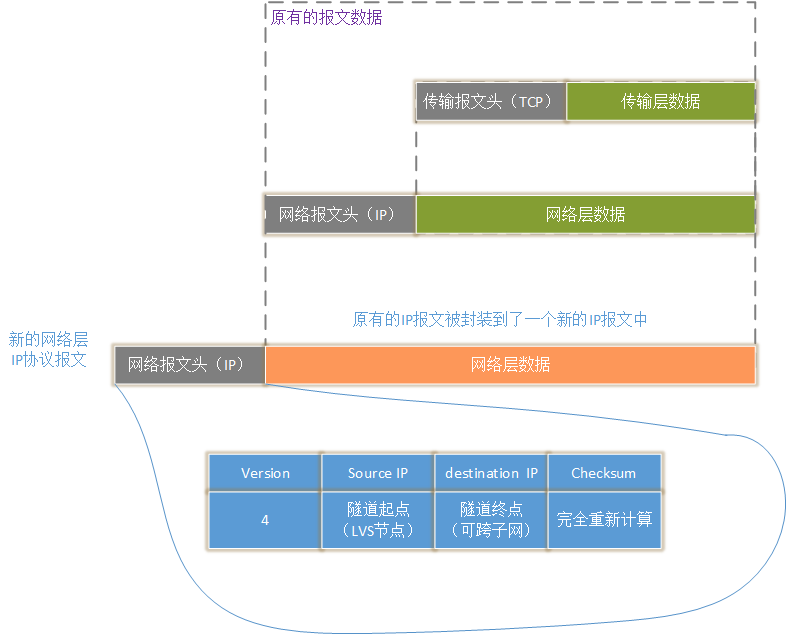
* 配置工作较LVS-NAT方式稍微麻烦一点，您至少需要了解LVS-DR模式的基本工作方式才能更好的指导自己进行LVS-DR模式的配置和运行过程中问题的解决。
* 由于LVS-DR模式的报文改写规则，导致LVS节点和Real Server节点必须在一个网段，因为二层交换是没法跨子网的。但是这个问题针对大多数系统架构方案来说，实际上并没有本质限制。

3.3、LVS-TUN工作方式

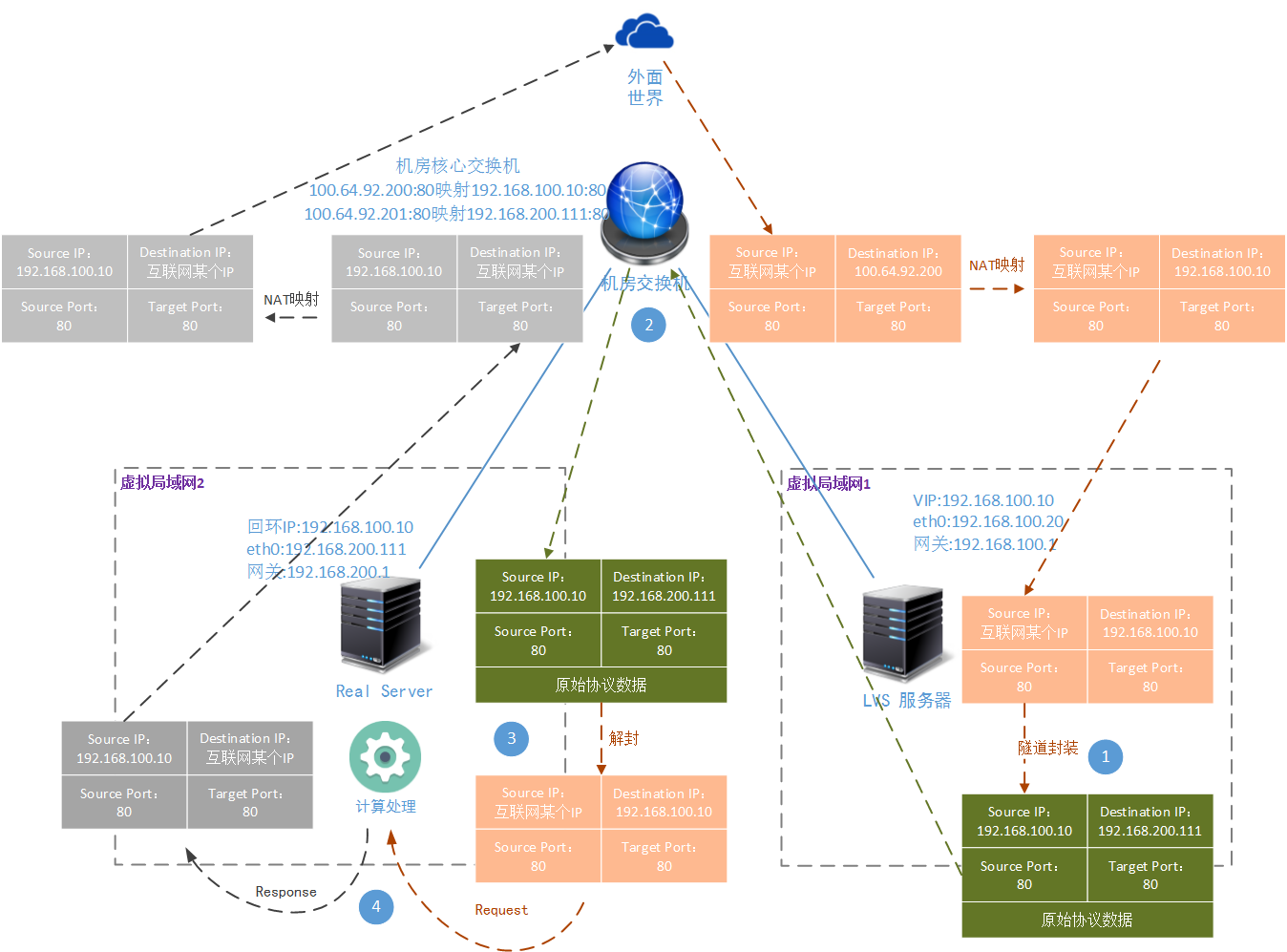
很多网络上的文章都为读者介绍DR和TUN的工作方式类似，要么就是直接讲解DR模式和TUN模式的安装配置方式，然后总结两种模式类似。那为什么有了DR模式后还需要TUN模式呢？为什么ipvsadmin针对两种模式的配置参数不一样呢？

实际上LVS-DR模式和LVS-TUN模式的工作原理完全不一样，工作场景完全不一样。DR基于数据报文重写，TUN模式基于IP隧道，后者是对数据报文的重新封装。下面我们就来讲解一下LVS-TUN模式的工作原理。

首先要介绍一个概念IPIP隧道。将一个完整的IP报文封装成另一个新的IP报文的数据部分，并通过路由器传送到指定的地点。在这个过程中路由器并不在意被封装的原始协议的内容。到达目的地点后，由目的地方依靠自己的计算能力和对IPIP隧道协议的支持，打开封装协议，取得原始协议。如下图：



可以说隧道协议就是为了解决跨子网传输准备的，在生产环境中由于业务需要、技术需要或者安全需要，可能使用交换机进行VLAN隔离（即形成若干个虚拟的独立的局域网），我们可能需要LVS节点在局域网A，而需要进行负载的多台**[MySQL](http://lib.csdn.net/base/mysql" \o "MySQL知识库" \t "_blank)**读服务器可能在局域网B中。这个时候，我们就要配置LVS的隧道方式。LVS-TUN模式如下图所示（注意，目标节点要能够解开隧道协议，好消息是Linux支持IPIP隧道协议）：



上图中的线有点多，您只需要关注 “有箭头”的虚线就可以了。

* 1、一旦LVS节点发现来目标（source IP）为192.168.100.10VIP的请求，就会使用IPIP隧道协议对这个请求报文进行封装。而不是像LVS-DR模式重写数据报文的MAC信息。如果配置了多个Real Server，那么LVS会使用设置的调度算法确定一个Real Server（这里为了简单，就只画了一个Real Server节点）。
* 2、重新封装后的IPIP隧道协议报文会重新被回发到路由器，路由器（或三层交换机）会根据设置的LVAN映射情况，找到目标服务器，并将这个IPIP隧道报文发送过去。
* 3、Real Server收到这个IPIP隧道报文后，会将这个报文进行解包。这里注意一下，一般情况下IPIP隧道报文会进行分片，就如同IP报文分片一样，只是为了讲解方便，我们假定这个报文不需要分片。解压后得到的数据报文就是原来发送给VIP的请求报文。
* 4、Real Server设置的回环IP，让Real Server认为原始的请求报文是从自己本地的某个应用程序发出的，完成原始报文的校验后，它会对这个报文进行处理。剩下的过程就和LVS-DR相同了，这里就不再进行复述了。

可以说LVS-TUN方式基本上具有LVS-DR的优点。在此基础上又支持跨子网间穿透。这样的敷在方案能够给我们架构师足够的系统设计场景。

4、LVS调度方式

在本文第3节中，为了集中介绍LVS的三种工作模式，我们在三幅图中都为LVS画了一个Real Server。但实际应用中，一般都是多个Real Server。**LVS使用多种调度算法来决定“当前的数据报文”由哪个Real Server进行处理**。在我的上篇文章《架构设计：负载均衡层设计方案（2）——Nginx安装》（[http://blog.csdn.net/yinwenjie/article/details/46620711](http://blog.csdn.net/yinwenjie/article/details/46620711" \t "_blank)）中，已经花了较多的篇幅介绍了Nginx中的调度方式，并明确说明了这些调度方式是**万变不离其宗的**。

文章中包括了Hash算法，并且说明了任何属性都可以做Hash，包括IP、用户名等；还介绍了轮询和加权轮询，加权轮询也可以依据各种属性作为权值，例如节点的CPU使用情况、内存使用情况、或者管理员自己设置的一个固定权值。LVS的调度也是这样的。

* 利用一致性Hash算法完成调度
  + 目标地址Hash（DH）：调度算法根据请求的目标IP地址，作为散列键（Hash Key）从静态分配的散列表找出对应的服务器，若该服务器是可用的且未超载，将请求发送到该服务器，否则返回空。
  + 原地址Hash（SH）：根据请求的源IP地址，作为散列键（Hash Key）从静态分配的散列表找出对应的服务器，若该服务器是可用的且未超载，将请求发送到该服务器，否则返回空。
* 轮询调度
  + 最简轮询（RR）：调度算法将外部请求按顺序轮流分配到集群中的真实服务器上，它均等地对待每一台服务器，而不管服务器上实际的连接数和系统负载。
  + 最少连接轮询（LC）：请注意“最少连接轮询”和“最少连接加权轮询”两种调度算法的区别。调度器通过“最少连接”调度算法动态地将网络请求调度到已建立的链接数最少的服务器上。注意请求肯定会被分配到这台目前连接数最少的Real Server上面，不会考虑几率问题
* 加权轮询调度：
  + 性能加权轮询（WRR）：调度算法根据真实服务器的不同处理能力来调度访问请求。这样可以保证处理能力强的服务器能处理更多的访问流量。调度器可以自动问询真实服务器的负载情况，并动态地调整其权值。
  + 最少连接数的加权轮询（WLC）：具有较高权值的服务器将承受较大比例的活动连接负载。调度器可以自动问询真实服务器的负载情况，并动态地调整其权值。注意，是按照一个比例，有较高的分配几率，而不是LC一样“肯定分配”。

在LVS官方中文资料中，提到了更为完整的调度算法。可以进行参考（[http://zh.linuxvirtualserver.org/node/2903](http://zh.linuxvirtualserver.org/node/2903" \t "_blank)）。**但一定记住各种调度算法肯定是逃不开哈希一致性、轮询、加权轮询这大思路的**。

5、后文介绍

这篇文章我们的重点是说明LVS的工作原理，并详细介绍了LVS三种工作模式的工作过程、优缺点、应用场景。只要您理解了原理，那么下一篇介绍的LVS的安装和配置就是小菜一碟。我们在下一篇文章中，将介绍LVS三种工作模式的安装和配置方式，之后我们介绍LVS + Keepalived的安装和配置方式。有了这个知识基础，我们将回到Nginx，介绍LVS + Keepalived + Nginx的安装和配置方式。