**SDN实验环境的拓扑搭建说明**

使用四台linux服务器，搭建SDN网络环境。其中一台服务器上部署OpenDaylight作为控制平面，另外三台服务器上要搭建一个数据平面。下面详细说一下数据平面拓扑的搭建。主要应用了基于vlan的路径隔离技术和私有协议。

**一 数据平面搭建说明**

数据平面拓扑需求：能在三台服务器上搭建了一个形如Internet2的拓扑。

**1.1 Internet2拓扑如图：**

****

**1.2 拓扑相关元素：**

数据平面使用ovs作为虚拟交换机节点，使用linux的network namespace作为虚拟主机节点，使用linux的Tap设备以及veth设备作为虚拟链路。

**1.2.1拓扑组成**

* 三台物理服务器；
* 17台ovs；
* 11台虚拟主机；
* 若干虚拟网络设备（tap设备，veth设备等）；
* VLAN隔离模块：三台使用VLAN的OvS
* 私有协议模块：实现增删广播头
* 一台普通二层交换机

Internet2拓扑总一共有17个主要的交换节点，我们将17个节点根据地理位置分割成三部分，分别部署在三台服务器上。在第一台服务器上有s6、s12、s13、s14、s15、s16六个交换节点；在第二台服务器上有s1、s2、s3、s4、s5、s7六个交换节点；在第三服务器上s8、s9、s10、s11、s17有五个交换机节点。

**1.2.2 虚拟主机搭建**

虚拟主机采用Linux network namespace（以下简称namespace）技术实现。一个namespace可以理解为主机内核协议栈的一份拷贝，它与主机之间是两个独立的网络协议栈，但是共享主机上的应用程序（ping，iperf，Wireshark等工具共享）。

namespace创建脚本（以创建192.168.11.21为例）：

ip netns add h21

ip link add h21-eth0 type veth peer name s48-eth3

ip link set h21-eth0 netns h21

ip netns exec h21 ip addr add 192.168.11.13/24 dev h21-eth0

ip netns exec h21 ifconfig h21-eth0 hw ether 08:00:27:11:af:76

ip netns exec h21 ip link set lo up

ip netns exec h21 ip link set dev h21-eth0 up

ip link set dev s48-eth3 up

由于namespace拥有独立的协议栈，在实验时可以把它当作一个主机来使用。如查看namespace h1 的ip地址，只需输入命令：

sudo ip netns exec h1 ifconfig

namespace的创建脚本见env文件夹下的server2.sh脚本。

**1.2.3 OvS的搭建**

为了实现更好的隔离性，每个OvS都搭建在虚拟机内。开启OvS只需要运行官方给出的启动命令即可。

OvS版本统一采用2.11.1

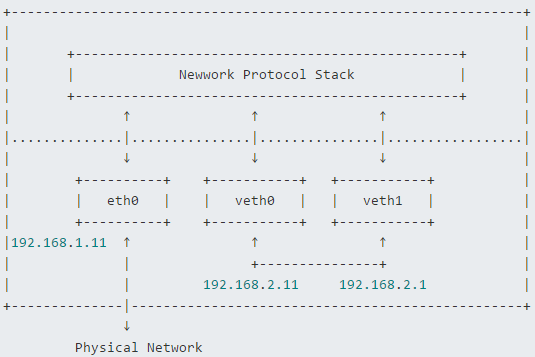
**1.2.3.1 OvS 和namespace（主机）之间的连接**

OvS 和namespace之间通过veth设备连接，veth设备是一种虚拟网络设备，它有两端，用于连接两个协议栈。

创建namespace时，同时创建一个veth peer（假设一端叫veth1，另一端叫veth2）。使用ip netns 命令将veth设备的一端分配给namespace。另一端则使用虚拟机网卡的桥接功能，桥接之后虚拟机内就会有一张物理网卡eth1，通过ovs-vsctl命令将这块网卡绑定到OvS上。这样namespace和OvS就通过veth设备连接起来。

**veth设备原理解析：**veth是一种网络虚拟化设备，如下图所示，veth和其他的网络设备一样，一段连接的是内核协议栈。veth是成对出现的，另一端两个设备彼此相连。一个设备收到协议栈的数据发送请求后，会将数据发送到另一个设备上去。

利用veth端到端的特性，和veth能连接两个协议栈的特性。可以将veth一段配置到network namespace上，当做虚拟Host的网卡，另一端连到ovs上。这样就好比一条虚拟链路连接了ovs和虚拟Host。



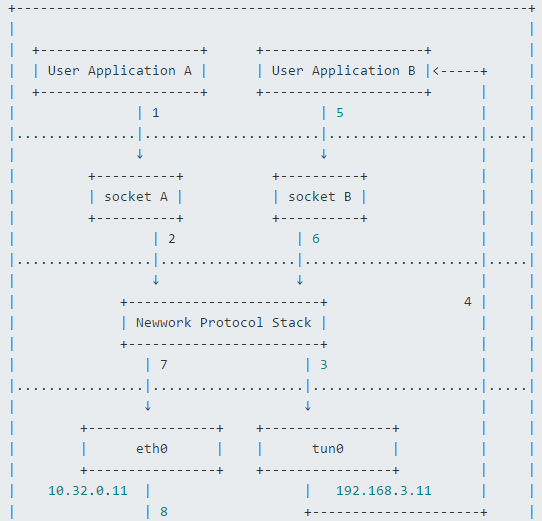
**1.2.3.2 OvS之间的连接**

OvS之间的连接通过tap设备来连接。假设有OvS1和OvS2，连接的方法为：OvS1用过桥接的方式连接tap设备，OvS2也通过桥接方式连接tap设备。这样两个ovs虚拟机内部都会出现一张网卡。在两个虚拟机中都是用ovs-vsctl命令将这个网卡绑定到OvS上。这样OvS1和OvS2都桥接了同一个设备，这样他们就连接在了一起。

**TAP设备原理解析：**TAP设备是一种虚拟网络设备，内核层把它当做一个正常的网卡看待，和物理网卡无异。

TUN /TAP设备是一种虚拟网络设备，通过此设备，程序可以方便地模拟网络行为。其中TUN 模拟的是一个三层设备,也就是说,通过它可以处理来自网络层的数据。而TAP设备是一个二层设备，可以收发MAC层数据包，

通过在Linux内核中添加了一个TUN/TAP虚拟网络设备的驱动程序和一个与之相关连的字符设备/dev/net/tun，字符设备tun作为用户空间和内核空间交换数据的接口。用户空间的应用程序可以通过这个设备文件来和内核中的驱动程序进行交互，其操作方式和普通的文件操作无异。当内核将数据包发送到虚拟网络设备时，数据包被保存在设备相关的一个队列中，直到用户空间程序通过打开的字符设备tun的描述符读取时，它才会被拷贝到用户空间的缓冲区中，其效果就相当于，数据包直接发送到了用户空间。通过系统调用write发送数据包时其原理与此类似。从结构上来说，Tun/tap驱动并不单纯是实现网卡驱动，同时它还实现了字符设备驱动部分。以字符设备的方式连接用户态和核心态。下面是示意图：



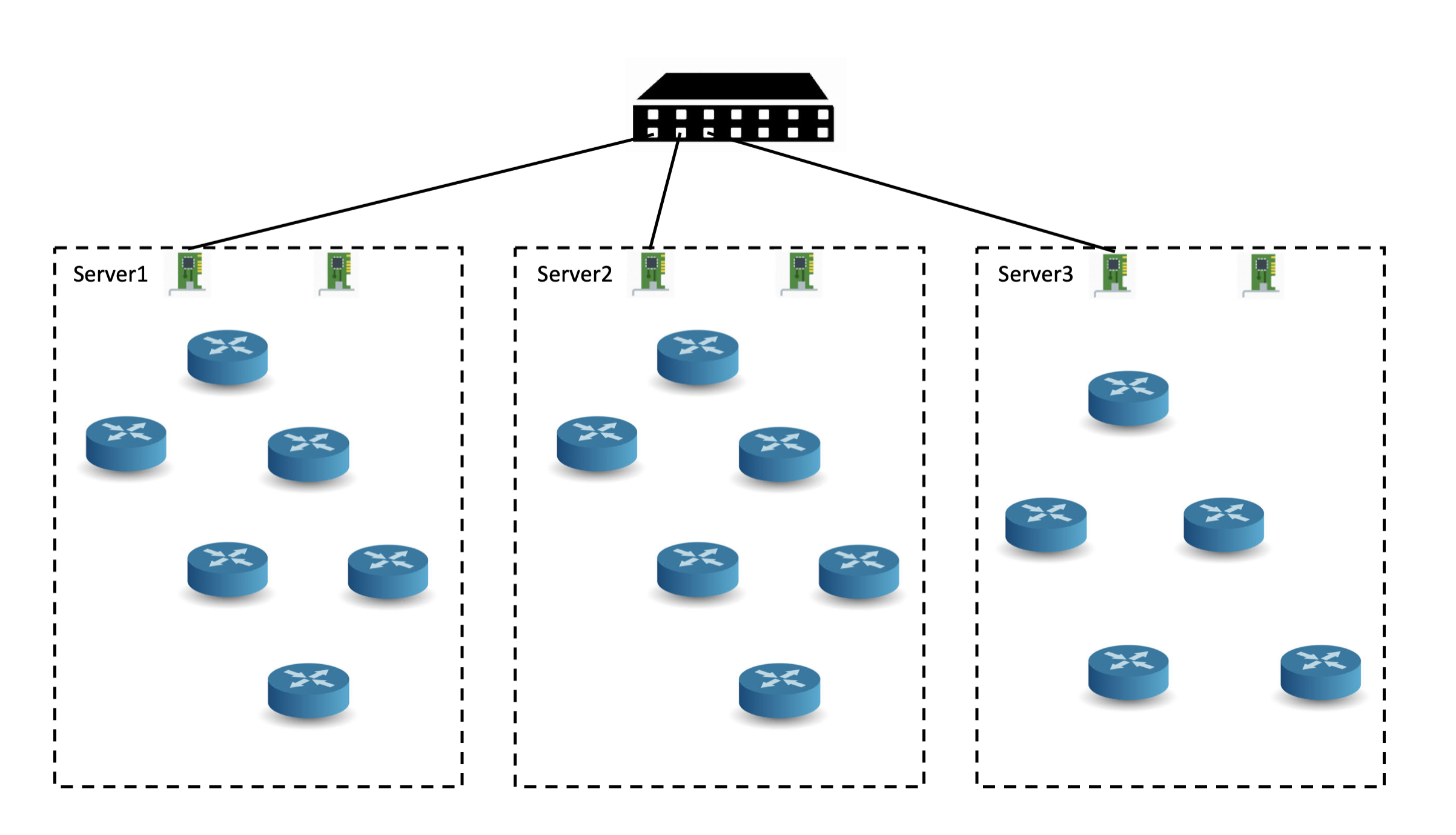
上图中有两个应用程序A和B，都在用户层，而其他的socket、协议栈和网络设备部分都在内核层。TUN/TAP是一个虚拟设备，从上图可以看出它和物理设备的差别。它们的一段虽然都连着协议栈，但另一端不一样，eth0的另一端是物理网络，而TUN/TAP设备的另一端是一个用户层的程序，协议栈发给它的包会被这个应用程序读取，应用程序能直接向它写数据。

也就是说TUN/TAP设备提供了应用层网络数据到内核层的直接连接（不需要通过协议栈）。Tun/tap 驱动程序中包含两个部分，一部分是字符设备驱动，还有一部分是网卡驱动部分。利用网卡驱动部分接收来自TCP/IP协议栈的网络分包并发送或者反过来将接收到的网络分包传给协议栈处理，而字符驱动部分则将网络分包在内核与用户态之间传送，模拟物理链路的数据接收和发送。Tun/tap驱动很好的实现了两种驱动的结合。

**二 使用VLAN技术和私有协议搭建Internet2拓扑**

**2.0 问题描述（可以省略不看 只是提供参考）**

起初服务器之间只简单的利用交换机来连接，保持他们的通信正常，每台服务器只需要提供一个额外的物理网卡连接到交换机。每台ovs都桥接到服务器的物理网卡，这样就能使三台服务器上的交换机相连。



**问题：拓扑上原来没有直接相连的交换机出现了直接连接**

原因： 跨服务器连接的交换机直接桥接到服务器物理网卡上，通过物理交换机互连。导致LLDP报文的广播会发送给所有与物理交换机相连的OvS，使原本不相连的交换机收到了彼此的LLDP报文，让他们认为有多个不对的邻居OvS。

解决方案：查阅资料得知OvS支持Vlan功能，这样就可以在不影响拓扑的前提下解决这个问题。

1）在每个服务器内再创建一个使用VLAN的ovs（以下简称vlan-ovs）。

2）这个vlan-ovs绑定到连接各个需要跨服务器的OVS和服务器的物理网卡。

3）vlan-ovs与服务器物理网卡连接的端口设置为trunk模式

4）vlan-ovs为需要跨服务器互连的每对交换机提供一个VLAN ID

这样有直接连接的OvS可以正常的跨过Vlan，没有直接连接的OvS会因为Vlan标签不符无法进行LLDP报文的交换。

由于Vlan-OvS放置在服务器中OvS的“外面”，所有的Vlan标签不会对正常通信产生影响，实现了“透明性”： 因为OvS发送LLDP给邻居交换机，通过收发邻居的LLDP报文判断邻居交换机。从交换机发出的LLDP报文会通过VLAN隔离和私有协议模块正确的转发到另一个交换机上，所以在交换机眼里拓扑还是Internet2拓扑。并没有改变。

具体Vlan-OvS的创建脚本如下：

ip netns add h21

ip link add h21-eth0 type veth peer name s48-eth3

ip link set h21-eth0 netns h21

ip netns exec h21 ip addr add 192.168.11.13/24 dev h21-eth0

ip netns exec h21 ifconfig h21-eth0 hw ether 08:00:27:11:af:76

ip netns exec h21 ip link set lo up

ip netns exec h21 ip link set dev h21-eth0 up

ip link set dev s48-eth3 up

**2.1使用vlan的路径隔离**

**2.1.1基于端口的Vlan技术原理介绍**

* VLAN简介：

VLAN（虚拟局域网）是对连接到的第二层交换机端口的网络用户的逻辑分段，不受网络用户的物理位置限制而根据用户需求进行网络分段。一个VLAN可以在一个交换机或者跨交换机实现。通过划分VLAN能够将一个局域网划分成多个广播域。VLAN通过在数据包头增加VLAN ID的方式来区分不同广播域的数据包。

* 基于交换机端口的VLAN技术：

在支持VLAN的交换机上配置交换机端口和VLAN ID的对应关系，该端口在接受和发送数据包时会有如下行为：之后会对从这个端口出去的数据包加特定的VLAN ID，而且只能收到带有特定VLAN ID的数据包。交换机中，有两种类型的接口：接入端口（Access）和中继端口（Trunk），Access接口只能用来连接用户主机，只能属于一个VLAN，因此该接口只传输本VLAN的数据；而Trunk端口允许多个VLAN通过，可以接受和发送多个VLAN数据包，常用于两台中继交换机之间的端口。

* Acess端口收报文：收到一个报文,判断是否有VLAN信息：如果没有则打上端口的PVID，并进行交换转发,如果有则直接丢弃（缺省）
* Acess端口发报文：将报文的VLAN信息剥离，直接发送出去
* trunk端口收报文：收到一个报文，判断是否有VLAN信息：如果没有则打上端口的PVID，并进行交换转发，如果有判断该trunk端口是否允许该 VLAN的数据进入：如果可以则转发，否则丢弃
* trunk端口发报文：比较端口的PVID和将要发送报文的VLAN信息，如果两者相等则剥离VLAN信息，再发送，如果不相等则直接发送
* 使用VLAN实现链路隔离：

通过划分VLAN可以分离广播域，除了Trank链路会传输多个VLAN ID的数据包，其他链路上只能传输带有特定VLAN ID的数据包。因此通过对不同的链路划分VLAN ID，让他们只能传输特定VLAN ID的数据包，做到隔离链路的功能。因此，在跨服务器的Internet2拓扑中，对于每条跨服务器链路可以设置一个VLAN ID。

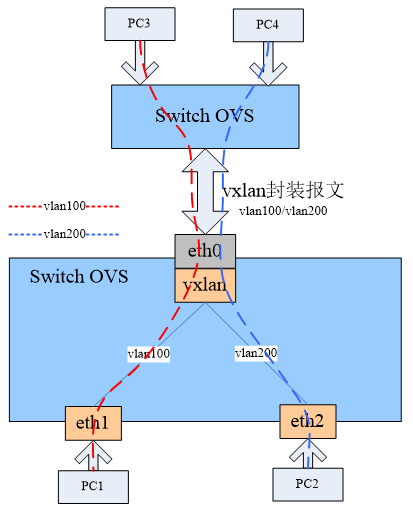
**2.1.2 ovs-vlan技术介绍：**

* 使用OVS实现VLAN组网：

在OVS中，VLAN的概念和普通交换机一样，不同的是，在OVS中可以通过流表进行VLAN值的修改，这就使得VLAN在OVS中的应用更加灵活。本实验主要使用传统方式设置vlan tag，和普通交换机工作原理相同，通过access进行设置vlan tag，通过trunk进行转发vlan报文来完成vlan组网。

* ovs配置vlan命令

以下图场景为例，4台PC设备分别接入不同的SDN交换机，通过vlan组成大二层网络。其中交换机中的eth1和eth2两个端口用于接入PC，OVS通过VLAN组网把PC1和PC3划分为VLAN 100，把PC2和PC4换分为VLAN 200，从而实现二层的网络隔离。其中PC1-PC3上的数据包 不会被发送到PC2-PC4链路上，即PC1到PC3的链路与PC2到PC4的链路实现了隔离。



设置ovs端口为trunk类型并配置可以转发的vlan：

ovs-vsctl set Port eth0 trunks=100,200

设置eth1的接口为access类型并配置tag：

ovs-vsctl set port eth1 tag=100  
ovs-vsctl set Port eth2 tag=200

**2.1.3 方法说明：**

1）在每个服务器内再创建一个使用VLAN的ovs（以下简称vlan-ovs）。

2）这个vlan-ovs绑定到连接各个需要跨服务器的OVS和服务器的物理网卡。

3）vlan-ovs与服务器物理网卡连接的端口设置为trunk模式

4）vlan-ovs为需要跨服务器互连的每对交换机提供一个VLAN ID

**最终形成的拓扑样貌为：每个服务器内的虚拟链路连接到一个vlan-ovs交换机，各个服务器的vlan-ovs交换机借助外部网络连通，外部网络就是使用二层以太网交换机的物理网络。**

这样有直接连接的OvS可以正常的跨过Vlan，没有直接连接的OvS会因为Vlan标签不符无法进行LLDP报文的交换。

由于Vlan-OvS放置在服务器中OvS的“外面”，所有的Vlan标签不会对正常通信产生影响，实现了“透明性”： 因为OvS发送LLDP给邻居交换机，通过收发邻居的LLDP报文判断邻居交换机。从交换机发出的LLDP报文会通过VLAN隔离和私有协议模块正确的转发到另一个交换机上，所以在交换机眼里拓扑还是Internet2拓扑。并没有改变。

**2.1.4 转发流程：**

每条虚拟链路配一个vlan标签，用不同vlan标签区分不同的虚拟链路。vlan-ovs收到虚拟链路一端的数据包后，打上正确的标签，通过物理网络转发到其他服务器上的vlan-ovs，它们收到后，根据数据包头的标签将流量从特定端口转发，即转发到到同一虚拟链路的另一端。

**2.1.5 小结**

通过使用基于vlan的路径隔离技术，我们在三台服务器之间实现了不同虚拟链路的隔离。虽然跨服务器的多条虚拟链路共享同一个物理网络，但是这些链路上的流量在物理网络两端会通过vlan-ovs隔离，达到逻辑上的多条链路，从而能够构建各种想要的拓扑。

**2.2 私有协议**

我们实际物理网络中使用的交换机是一台普通的以太网二层交换机，没有vlan功能。即这个交换机是基于mac地址学习的方法转发流量的。而使用这样的交换机是无法将多个服务器之间连通的，下面说明存在的问题。

**使用普通二层交换机带来的mac地址学习两次的问题：**

假设服务器1和服务器2之间有两条不同的虚拟链路分别是vlan100和vlan200。这两条虚拟链路建立在物理网络（二层交换机）之上。那么当一个数据包经过虚拟链路vlan100从服务器1发给服务器2时，交换机记录一下mac地址，当数据包经过虚拟链路vlan200从服务器2又发给服务器1时，交换机又记录了一次mac地址。

这种mac地址与端口映射的不断变化会导致交换机失效。导致转发出问题。

**使用私有协议解决：**

解决方案，将中间的交换机通过软件的方式改成集线器，这样数据包就可以正常的转发，而且复制产生的多余数据包会因为vlan标签不符而被丢弃，不会增加三台服务器的负载。

具体的方法是（以服务器1为例），新建一对veth peer（veth1，veth2），当veth2收到包时添加一层mac头部，并将目的地址改成广播地址，从物理网卡发出。当物理网卡收到数据包时，去掉其mac头部，从veth2发出交给vlan交换机检查vlan标签。

具体实现采用Linux内核模块开发，修改网卡的数据包处理函数，并插入内核模块。

以下是Linux网络协议栈处理数据包的流程图，我通过修改函数指针到我自定义的处理函数上来实现功能。

rx\_handler\_result\_t add\_fun(struct sk\_buff \*\*pskb)

{

printk(KERN\_INFO "[liuhy]veth2 receive a packet\n");

struct net\_device \*indev;

struct sk\_buff \*skb = \*pskb;

skb = skb\_share\_check(skb, GFP\_ATOMIC);

struct ethhdr \*mh = eth\_hdr(skb);

printk(KERN\_EMERG "Source MAC=%x:%x:%x:%x:%x:%x\n",\

mh->h\_source[0],mh->h\_source[1],mh->h\_source[2],mh->h\_source[3],mh->h\_source[4],mh->h\_source[5]);

char mac\_header\_buf[14];

memset(mac\_header\_buf, 0, sizeof mac\_header\_buf);

// set src and dst mac address

int i;

for (i = 6; i < 12; i ++)

{

mac\_header\_buf[i] = mh->h\_source[i - 6];

mac\_header\_buf[i - 6] = 0xff;

}

// set protocol type

mac\_header\_buf[12] = 0x08;

mac\_header\_buf[13] = 0x06;

struct net\_device \*go = dev\_get\_by\_name(&init\_net, "enp0s3");

if (go == NULL)

{

printk(KERN\_INFO "[liuhy]enp0s3 not find \n");

return -1;

}

/\* add ethernet header code here\*/

indev = skb->dev;

skb->dev = go;

skb\_push(skb, ETH\_HLEN);

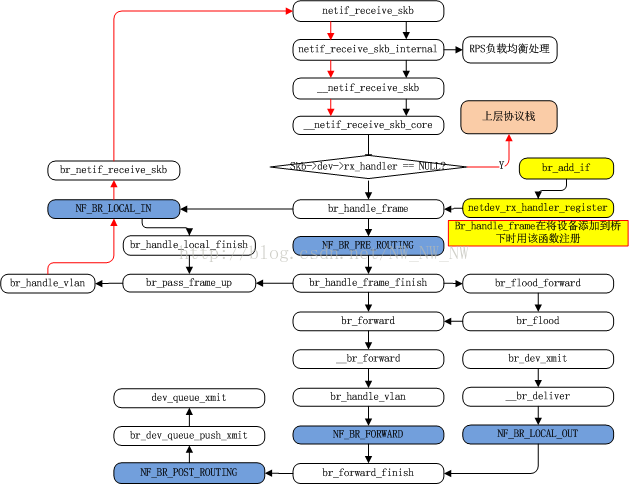
skb\_push(skb, ETH\_HLEN);

memcpy(skb->data, mac\_header\_buf, 14);

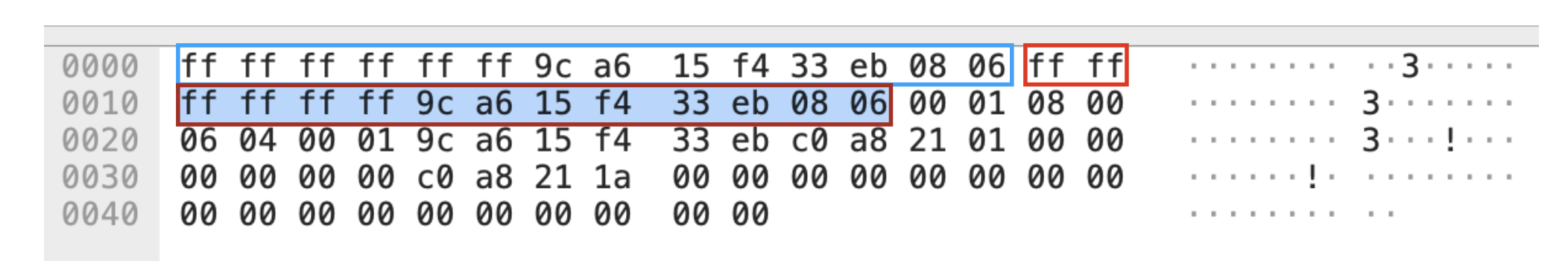
dev\_queue\_xmit(skb);

return RX\_HANDLER\_CONSUMED;

}



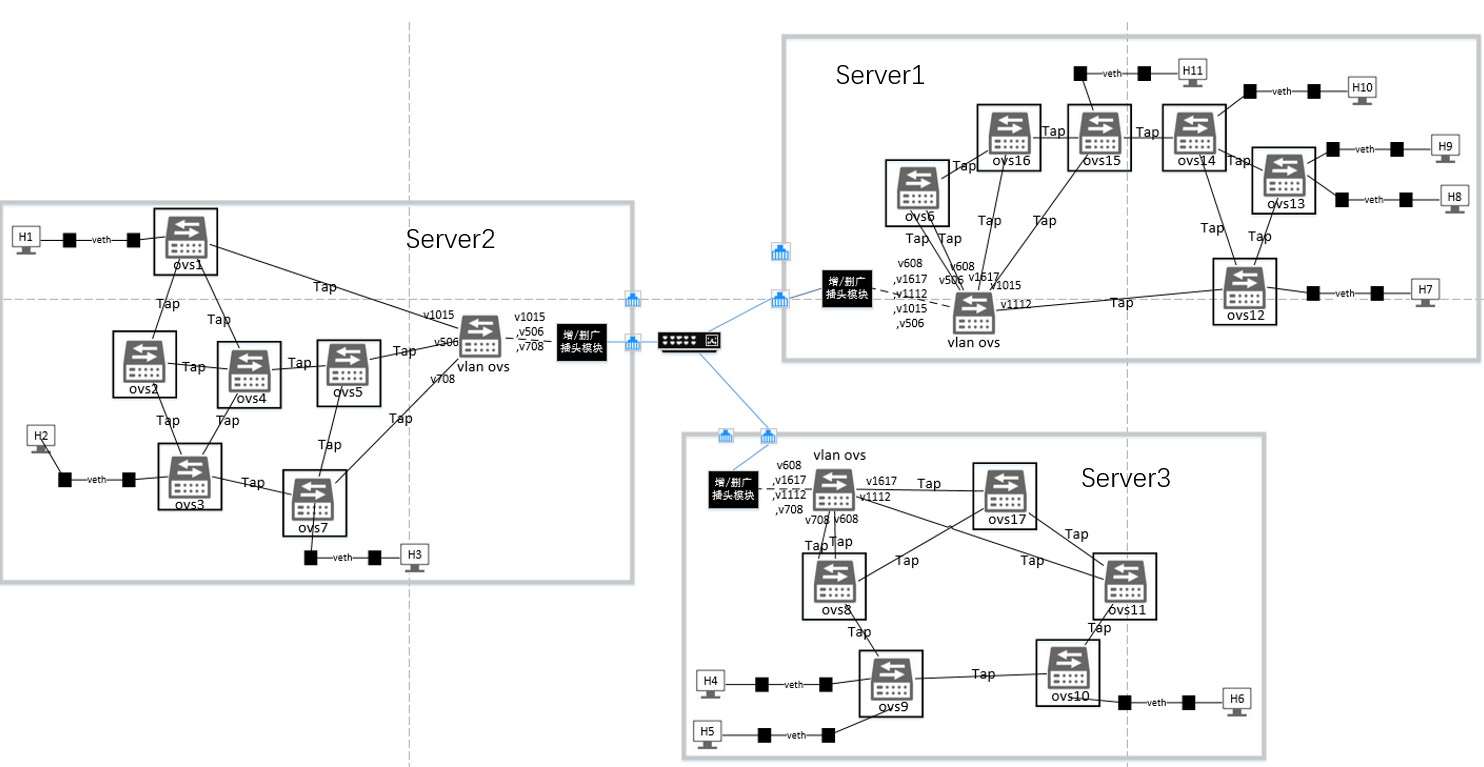
以下是修改结果：



至此，Internet 2 拓扑搭建完成，可以在上面进行相关实验。

**三 实际网络拓扑详细图：**

**如图所示是实际网络拓扑的详细图：**



在图中详细的描述了如何在三台服务器上搭建一个Internet2拓扑。其中：

1. 使用OpenvSwitch 2.11.1作为交换节点。Internet2拓扑总一共有17个主要的交换节点，我们将17个节点根据地理位置分割成三部分，分别部署在三台服务器上。在第一台服务器上有s6、s12、s13、s14、s15、s16六个交换节点；在第二台服务器上有s1、s2、s3、s4、s5、s7六个交换节点；在第三服务器上s8、s9、s10、s11、s17有五个交换机节点。
2. 通过network namespace创建虚拟主机。在Server1中有H7-H11五台虚拟Host；在Server2中有H1-H3三台虚拟Host；在Server3中有H4-H6三台虚拟Host
3. 同一服务器内的OvS间使用tap设备构建链路。
4. 边缘OvS和虚拟主机通过veth构建链路。
5. 跨服务器的OvS使用VLAN技术和私有协议方法构建链路。
6. 如图所示，跨服务器的虚拟链路构建在VLAN和私有协议模块之上，该模块是对OvS “透明的”：因为OvS发送LLDP给邻居交换机，通过收发邻居的LLDP报文判断邻居交换机。从交换机发出的LLDP报文会通过VLAN隔离和私有协议模块正确的转发到另一个交换机上，所以在交换机眼里拓扑还是Internet2拓扑。并没有改变。
7. 三台使用了VLAN技术的OvS，其VLAN配置如下：

a）首先，每条跨服务器的虚拟链路需要设置一个不同的VLAN。具体的VLAN-OvS的端口的VLAN ID如图所示。

b）其次，在Server1中：与跨服器的交换机S6、S12、S15、S16之间的端口配置为Access端口。与服务器相连的端口配置为Trunk端口。

c）在Server2中：与跨服器的交换机S1、S5、S7之间的端口配置为Access端口。与服务器相连的端口配置为Trunk端口。

d）在Server3中：与跨服器的交换机S8、S11、S17之间的端口配置为Access端口。与服务器相连的端口配置为Trunk端口。

1. VLAN-OvS通过私有协议模块与服务器的物理网卡连接
2. 服务器之间通过一个普通的二层交换机连接

**接下来以H1到H11的通信为例，分析数据流H1-H11是如何在数据平面上传输的：**

1. 数据流H1-H11从主机H1发出，到达S1

2. S1将数据包发送给S15，数据包到达VLAN-OvS

3. VLAN-OvS将数据包打上v1015标签，并从Trunk端口发出去，到达私有协议模块

4. 私有协议模块给数据包打上广播头，并从物理网卡发出去，到达二层交换机

5. 由于是广播包，二层交换机将数据包发送到服务器Server1和Server3。

6. Server1和Server3上的私有协议模块将广播头去掉，然后把包发给VLAN-OVS，Server3上的VLAN-OvS比较VLAN ID，由于不存在v1015标签，所以丢弃。而Server1上的VLAN-OvS会接受并从v1015的端口发出去，到达S15

7. 最终由S15发给H11。因此，数据流得到正确的传输。