虚拟设备之TUN和TAP

LLLibra1...
会修电脑的程序猿

TAP与TUN是什么

不同于硬件物理网卡，TAP/TUN 是在 Linux 内核 2.4.x 版本之后完全由软件实现的虚拟网络设备，在功能上 TAP/TUN 和物理网卡没有区别，它们同样都是网络设备，都可以设置 IP 地址，而且都属于 Linux 网络设备管理模块，由 Linux 网络设备管理模块统一来管理。

虚拟设备和物理设备的区别

在 Linux 内核里，有一个叫做网络设备管理层的东西，它处在网络设备驱动和协议栈之间，负责处理它们之间的数据交互细节。网络驱动不需要知道协议栈的处理细节，协议栈也不需要知道网络驱动的处理细节。这样可以很方便的将两层分隔开来，和 TCP/IP 类似，每一层只需要干好自己的工作就好了，简化了开发。

对于一个物理网卡来说，它的两端分别是内核协议栈和外面的物理网络，从内核协议栈接收到的数据会通过物理网络发送出去，通过物理网络接收到的数据会转发到内核协议栈进行处理。

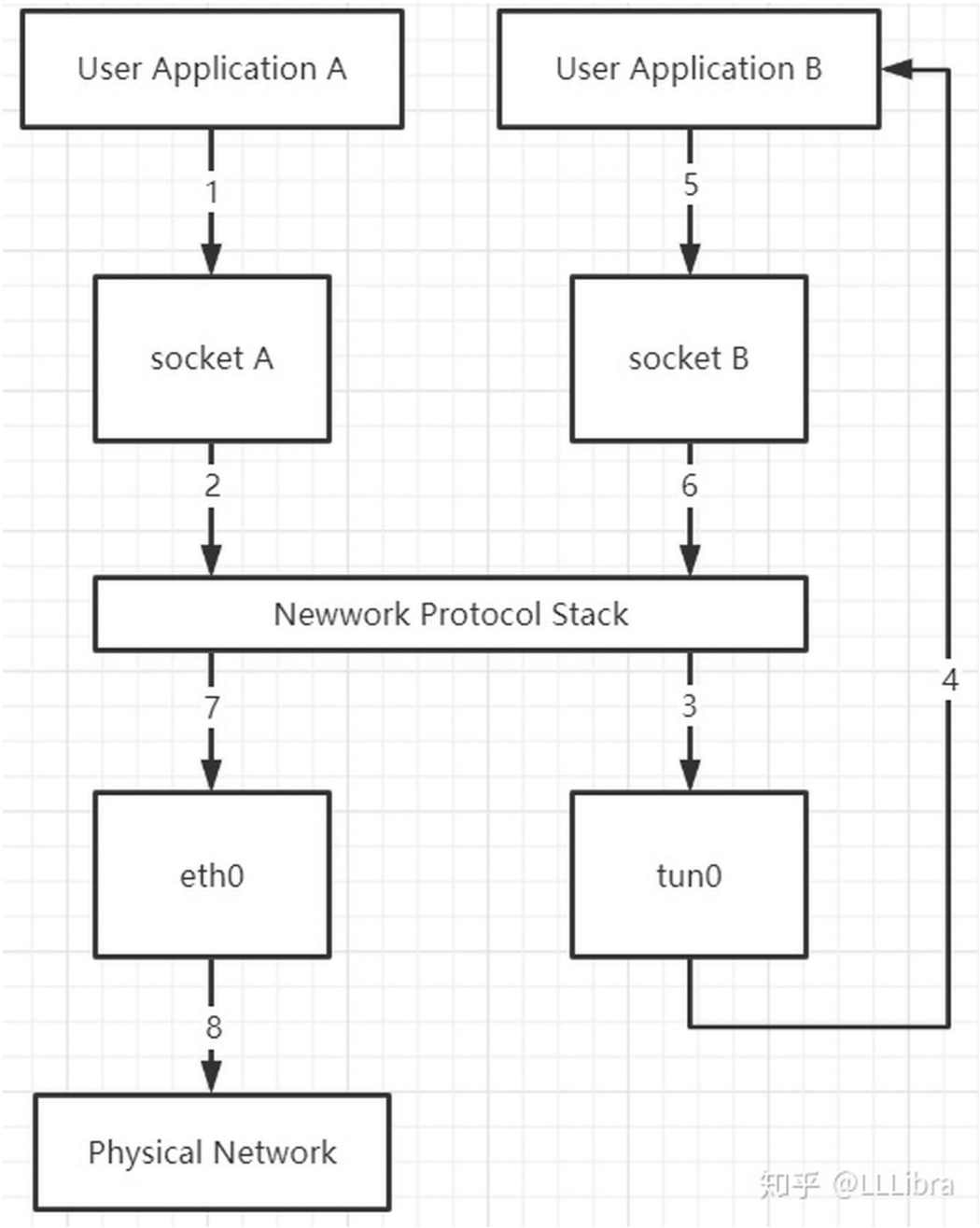
对于一个虚拟网络设备来说，它的两端分别是内核协议栈和网络设备驱动，从网络驱动接收到的数据会发送到内核协议栈，应用程序从内核协议栈发送过来的数据会发送到网络驱动，至于网络驱动怎么处理它，那是驱动层面的事情，取决于网络驱动怎么实现。

TUN和TAP的区别

TUN 是一个虚拟网络设备，它模拟的是一个三层设备，通过它可以处理来自网络层的数据包，也就是 IP 数据包。由于它只模拟到了 IP 层，所以它无法与物理网卡做 bridge，也没有 MAC 地址，但是可以通过三层交换的方式来与物理网卡相互通信。

TAP 模拟的是一个二层设备，它比 TUN 更加深入，它可以处理数据链路层的数据包，拥有 MAC 地址，可以与物理网卡做 bridge，支持 MAC 层广播，也可以给它设置 IP 地址。

数据包发送/接收流程



如图所示，两个应用程序 A 和 B，都处于用户层，其他的设备都处于内核层，数据的流向可以顺着数字方向来查看。

TUN0 是一个虚拟网卡设备，可以看到它与物理网卡相同的地方在于它们的一端都连接内核协议栈，不同的地方在于物理网卡另一端连接的是外面的交换机或者路由器等硬件设备，TUN0 连接的是处于用户层的应用程序，协议栈发送给虚拟网卡的数据都被发送到了应用程序中，通过加密，转换，校验等方式后通过物理网卡发送出去。

数据包发送的整个流程为：

用户层的应用程序 A 发送了一个普通的数据包，socket 将数据包发送给内核协议栈，内核协议栈根据目的地址进行路由，查询路由表，发现数据包的下一跳地址应该为 TUN0 网卡，所以内核协议栈将数据包发送给虚拟网卡设备 TUN0，TUN0 接收到数据之后通过某种方式从内核空间将数据发送给运行在用户空间的应用程序 B，B 收到数据包后进行一些处理，然后构造一个新的数据包，通过 socket 发送给内核协议栈，这个新的数据包的目的地址变成了一个外部地址，源地址变成了

eth0 的地址，内核协议栈通过查找路由表之后发现目的地址找不到，就会将数据包通过 eth0 网卡发送给网关，eth0 接收到数据之后将数据包发送到和 eth0 网卡物理相连的外部设备。

至于 TUN/TAP 设备是怎么让内核层和用户层进行数据交换的，这里暂不讨论，对于 Linux 来说有好多种方法可以让内核层和用户层进行数据交换。

通过上面的发送流程可以看出，数据包转发的路径由路由表来控制，如果需要使某个应用程序的流量走虚拟网卡，只需要在路由表中添加路由即可使对应的流量发送到虚拟网卡中即可。

总结

以上就是 Linux 中虚拟网卡的实现原理了，在 Windows 中原理类似，基本路径也是一致的， 只不过可能有一些实现细节会有所区别。

发布于 2020-09-29