43 预习 | Socket通信之网络协议基本原理

2019-07-05 刘超

趋谈Linux操作系统 进入课程 >

讲述：刘超

时长 13:43 大小 11.00M

上一节我们讲的进程间通信，其实是通过内核的数据结构完成的，主要用于在一台 Linux 上两个进程之间的通信。但是，一旦超出一台机器的范畴，我们就需要一种跨机器的通信机制。

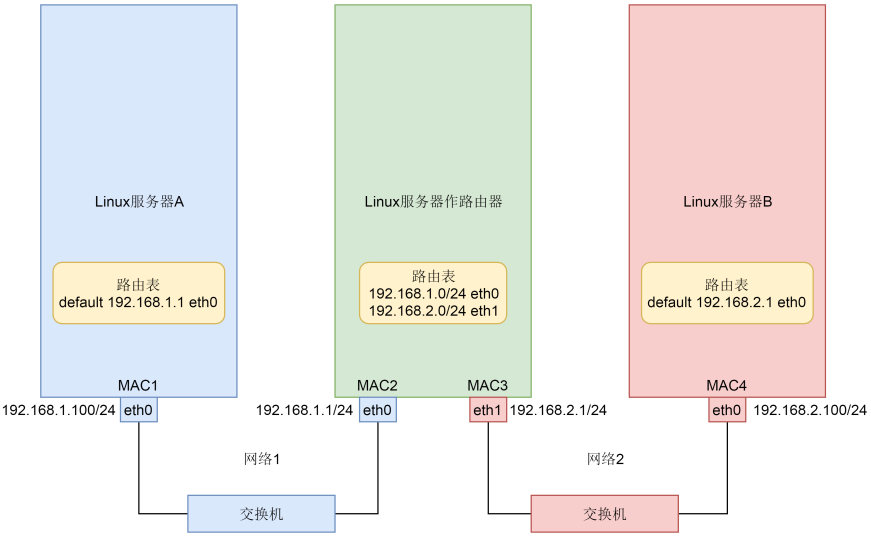
一台机器将自己想要表达的内容，按照某种约定好的格式发送出去，当另外一台机器收到这些信息后，也能够按照约定好的格式解析出来，从而准确、可靠地获得发送方想要表达的内容。这种约定好的格式就是**网络协议**（Networking Protocol）。

我们将要讲的 Socket 通信以及相关的系统调用、内核机制，都是基于网络协议的，如果不了解网络协议的机制，解析 Socket 的过程中，你就会迷失方向，因此这一节，我们有必要做一个预习，先来大致讲一下网络协议的基本原理。

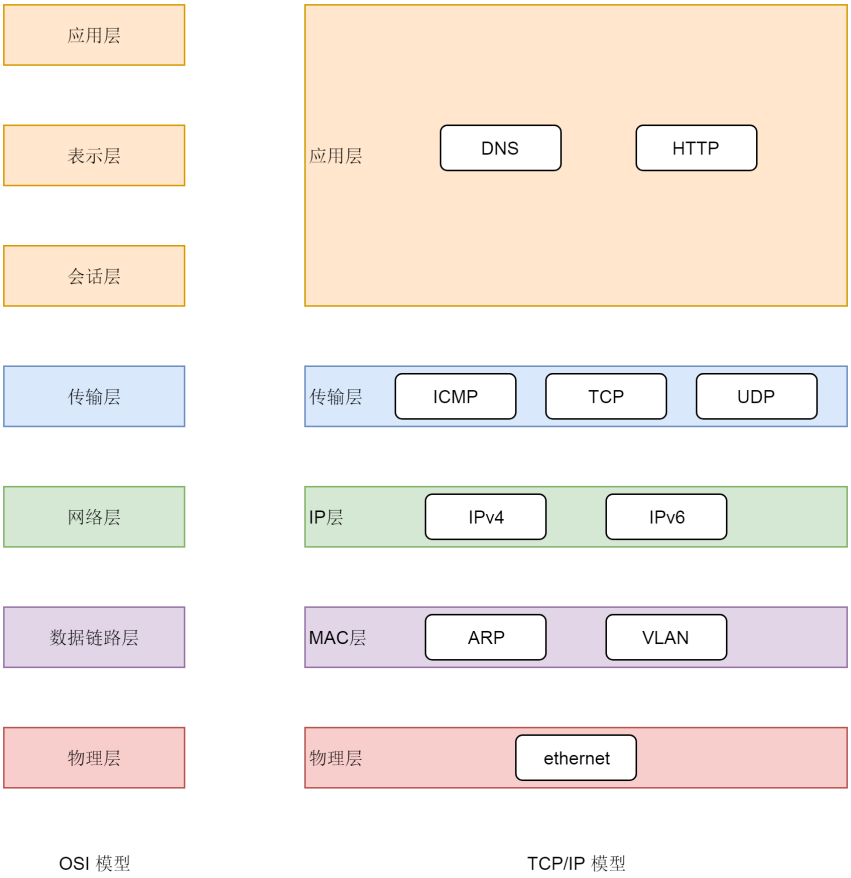
网络为什么要分层?

我们这里先构建一个相对简单的场景，之后几节内容，我们都要基于这个场景进行讲解。

我们假设这里就涉及三台机器。Linux 服务器 A 和 Linux 服务器 B 处于不同的网段，通过中间的 Linux 服务器作为路由器进行转发。



说到网络协议，我们还需要简要介绍一下两种网络协议模型，一种是**OSI 的标准七层模型**，一种是**业界标准的 TCP/IP 模型**。它们的对应关系如下图所示：



为什么网络要分层呢？因为网络环境过于复杂，不是一个能够集中控制的体系。全球数以亿记的服务器和设备各有各的体系，但是都可以通过同一套网络协议栈通过切分成多个层次和组合，来满足不同服务器和设备的通信需求。

我们这里简单介绍一下网络协议的几个层次。

我们从哪一个层次开始呢？从第三层，网络层开始，因为这一层有我们熟悉的 IP 地址。也因此，这一层我们也叫 IP 层。

我们通常看到的 IP 地址都是这个样子的：192.168.1.100/24。斜杠前面是 IP 地址，这个地址被点分隔为四个部分，每个部分 8 位，总共是 32 位。斜线后面 24 的意思是，32 位中，前 24 位是网络号，后 8 位是主机号。

为什么要这样分呢？我们可以想象，虽然全世界组成一张大的互联网，美国的网站你也能够访问的，但是这个网络不是一整个的。你们小区有一个网络，你们公司也有一个网络，联通、移动、电信运营商也各有各的网络，所以一个大网络是被分成个小的网络。

那如何区分这些网络呢？这就是网络号的概念。一个网络里面会有多个设备，这些设备的网络号一样，主机号不一样。不信你可以观察一下你家里的手机、电视、电脑。

连接到网络上的每一个设备都至少有一个 IP 地址，用于定位这个设备。无论是近在咫尺的你旁边同学的电脑，还是远在天边的电商网站，都可以通过 IP 地址进行定位。因此，**IP 地址类似互联网上的邮寄地址，是有全局定位功能的。**

就算你要访问美国的一个地址，也可以从你身边的网络出发，通过不断的打听道儿，经过多个网络，最终到达目的地址，和快递员送包裹的过程差不多。打听道儿的协议也在第三层，称为路由协议（Routing protocol），将网络包从一个网络转发给另一个网络的设备称为路由器。

路由器和路由协议十分复杂，我们这里就不详细讲解了，感兴趣可以去看我写的另一个专栏“趣谈网络协议”里的相关文章。

总而言之，第三层干的事情，就是网络包从一个起始的 IP 地址，沿着路由协议指的道儿，经过多个网络，通过多次路由器转发，到达目标 IP 地址。

从第三层，我们往下看，第二层是数据链路层。有时候我们简称为二层或者 MAC 层。所谓 MAC，就是每个网卡都有的唯一的硬件地址（不绝对唯一，相对大概率唯一即可，类比UUID）。这虽然也是一个地址，但是这个地址是没有全局定位功能的。

就像给你送外卖的小哥，不可能根据手机尾号找到你家，但是手机尾号有本地定位功能的，只不过这个定位主要靠“吼”。外卖小哥到了你的楼层就开始大喊：“尾号 xxxx 的，你外卖到了！”

MAC 地址的定位功能局限在一个网络里面，也即同一个网络号下的 IP 地址之间，可以通过 MAC 进行定位和通信。从 IP 地址获取 MAC 地址要通过 ARP 协议，是通过在本地发送广播包，也就是“吼”，获得的 MAC 地址。

由于同一个网络内的机器数量有限，通过 MAC 地址的好处就是简单。匹配上 MAC 地址就接收，匹配不上就不接收，没有什么所谓路由协议这样复杂的协议。当然坏处就是，MAC 地址的作用范围不能出本地网络，所以一旦跨网络通信，虽然 IP 地址保持不变，但是 MAC 地址每经过一个路由器就要换一次。

我们看前面的图。服务器 A 发送网络包给服务器 B，原 IP 地址始终是 192.168.1.100，目标 IP 地址始终是 192.168.2.100，但是在网络 1 里面，原 MAC 地址是 MAC1，目标 MAC 地址是路由器的 MAC2，路由器转发之后，原 MAC 地址是路由器的 MAC3，目标 MAC 地址是 MAC4。

所以第二层干的事情，就是网络包在本地网络中的服务器之间定位及通信的机制。

我们再往下看，第一层，物理层，这一层就是物理设备。例如连着电脑的网线，我们能连上的 WiFi，这一层我们不打

算进行分析。

从第三层往上看，第四层是传输层，这里面有两个著名的协议 TCP 和 UDP。尤其是 TCP，更是广泛使用，在 IP 层的代码逻辑中，仅仅负责数据从一个 IP 地址发送给另一个 IP 地址，丢包、乱序、重传、拥塞，这些 IP 层都不管。处理这些问题的代码逻辑写在了传输层的 TCP 协议里面。

我们常称，TCP 是可靠传输协议，也是难为它了。因为从第一层到第三层都不可靠，网络包说丢就丢，是 TCP 这一层通过各种编号、重传等机制，让本来不可靠的网络对于更上层来讲，变得“看起来”可靠。哪有什么应用层岁月静好，只不过 TCP 层帮你负重前行。

传输层再往上就是应用层，例如咱们在浏览器里面输入的 HTTP，Java 服务端写的 Servlet，都是这一层的。

二层到四层都是在 Linux 内核里面处理的，应用层例如浏览器、Nginx、Tomcat 都是用户态的。内核里面对于网络包的处理是不区分应用的。

从四层再往上，就需要区分网络包发给哪个应用。在传输层的 TCP 和 UDP 协议里面，都有端口的概念，不同的应用监听不同的端口。例如，服务端 Nginx 监听 80、Tomcat 监

听 8080；再如客户端浏览器监听一个随机端口，FTP 客户端监听另外一个随机端口。

应用层和内核互通的机制，就是通过 Socket 系统调用。所以经常有人会问，Socket 属于哪一层，其实它哪一层都不属于，它属于操作系统的概念，而非网络协议分层的概念。只不过操作系统选择对于网络协议的实现模式是，二到四层的处理代码在内核里面，七层的处理代码让应用自己去做，两者需要跨内核态和用户态通信，就需要一个系统调用完成这个衔接，这就是 Socket。

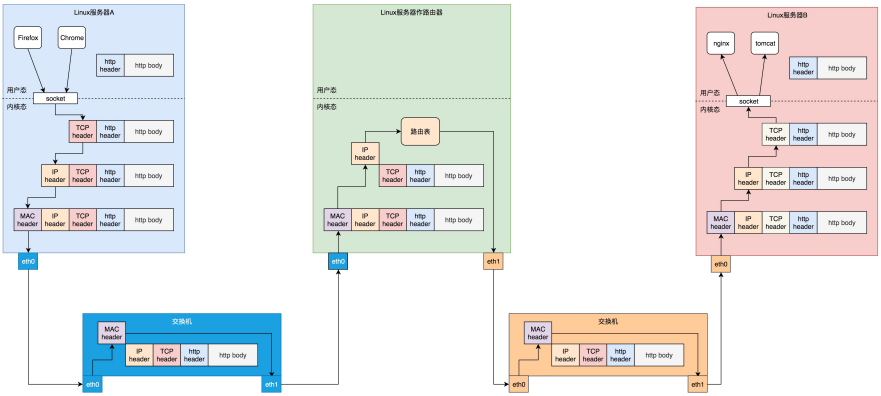
发送数据包

网络分完层之后，对于数据包的发送，就是层层封装的过程。

就像下面的图中展示的一样，在 Linux 服务器 B 上部署的服务端 Nginx 和 Tomcat，都是通过 Socket 监听 80 和 8080 端口。这个时候，内核的数据结构就知道了。如果遇到发送到这两个端口的，就发送给这两个进程。

在 Linux 服务器 A 上的客户端，打开一个 Firefox 连接 Ngnix。也是通过 Socket，客户端会被分配一个随机端口

12345。同理，打开一个 Chrome 连接 Tomcat，同样通过 Socket 分配随机端口 12346。



在客户端浏览器，我们将请求封装为 HTTP 协议，通过 Socket 发送到内核。内核的网络协议栈里面，在 TCP 层创建用于维护连接、序列号、重传、拥塞控制的数据结构，将 HTTP 包加上 TCP 头，发送给 IP 层，IP 层加上 IP 头，发送给 MAC 层，MAC 层加上 MAC 头，从硬件网卡发出去。

网络包会先到达网络 1 的交换机。我们常称交换机为二层设备，这是因为，交换机只会处理到第二层，然后它会将网络包的 MAC 头拿下来，发现目标 MAC 是在自己右面的网口，于是就从这个网口发出去。

网络包会到达中间的 Linux 路由器，它左面的网卡会收到网络包，发现 MAC 地址匹配，就交给 IP 层，在 IP 层根据 IP 头中的信息，在路由表中查找。下一跳在哪里，应该从哪个网口发出去？在这个例子中，最终会从右面的网口发出去。我们常把路由器称为三层设备，因为它只会处理到第三层。

从路由器右面的网口发出去的包会到网络 2 的交换机，还是会经历一次二层的处理，转发到交换机右面的网口。

最终网络包会被转发到 Linux 服务器 B，它发现 MAC 地址匹配，就将 MAC 头取下来，交给上一层。IP 层发现 IP 地址匹配，将 IP 头取下来，交给上一层。TCP 层会根据 TCP 头中的序列号等信息，发现它是一个正确的网络包，就会将网络包缓存起来，等待应用层的读取。

应用层通过 Socket 监听某个端口，因而读取的时候，内核会根据 TCP 头中的端口号，将网络包发给相应的应用。

HTTP 层的头和正文，是应用层来解析的。通过解析，应用层知道了客户端的请求，例如购买一个商品，还是请求一个网页。当应用层处理完 HTTP 的请求，会将结果仍然封装为 HTTP 的网络包，通过 Socket 接口，发送给内核。

内核会经过层层封装，从物理网口发送出去，经过网络 2 的交换机，Linux 路由器到达网络 1，经过网络 1 的交换机，到达 Linux 服务器 A。在 Linux 服务器 A 上，经过层层解封装，通过 socket 接口，根据客户端的随机端口号，发送给客户端的应用程序，浏览器。于是浏览器就能够显示出一个绚丽多彩的页面了。

即便在如此简单的一个环境中，网络包的发送过程，竟然如此的复杂。不过这一章后面，我们还是会层层剖析每一层做的事情。

总结时刻

网络协议是一个大话题，如果你想了解网络协议的方方面面，欢迎你订阅我写的另一个专栏“趣谈网络协议”。这个专栏重点解析在这个网络通信过程中，发送端和接收端的操作系统都做了哪些事情，对于中间通路上的的复杂的网络通信逻辑没有做深入解析。

如果只是为了掌握这一章的内容，这一节我们讲的网络协议的七个层次，你不必每一层的每一个协议都很清楚，只要记住 TCP/UDP->IPv4->ARP 这一条链就可以了，因为后面我们的分析都是重点分析这条链。

另外，前面那个简单的拓扑图中，网络包的封装、转发、解封装的过程，建议你多看几遍，了熟于心，因为接下来，我们就能从代码层面，看到这个过程。到时候，对应起来，你就比较容易理解。

了解了 Socket 的基本原理，下一篇文章，我们就来看一看在 Linux 操作系统里面 Socket 系统调用的接口是什么样的。

上一篇　42 | IPC（下）：不同项目组之间抢资源，如何协调？

下一篇

精选留言 (4)

写留言

LDxy
2019-07-06

为什么称为协议栈呢？这和栈这种数据结构有何关系？

展开

许童童
2019-07-05

老师：问个问题。我们家庭办理的宽带，都是运营商在哪一层把带宽给限制的呢？

W.jyao
2019-07-05

看到留言的问题了，这不就是arp干的事

展开

WL

2019-07-05

老师我想问一下在发送数据包的时候, Linux服务A是怎么拿到linux服务器B的mac地址的, Linux服务器B的mac地址是一开始就加上去的吗?