第30讲 | 容器网络之Flannel：每人一亩三分地

2018-07-25 刘超

第30讲 | 容器网络之Flannel：每人一亩三分地
朗读人：刘超　11′40″ | 5.35M

上一节我们讲了容器网络的模型，以及如何通过 NAT 的方式与物理网络进行互通。

每一台物理机上面安装好了 Docker 以后，都会默认分配一个 172.17.0.0/16 的网段。一台机器上新创建的第一个容器，一般都会给 172.17.0.2 这个地址，当然一台机器这样玩玩倒也没啥问题。但是容器里面是要部署应用的，就像上一节讲过的一样，它既然是集装箱，里面就需要装载货物。

如果这个应用是比较传统的单体应用，自己就一个进程，所有的代码逻辑都在这个进程里面，上面的模式没有任何问题，只要通过 NAT 就能访问进来。

但是因为无法解决快速迭代和高并发的问题，单体应用越来越跟不上时代发展的需要了。

你可以回想一下，无论是各种网络直播平台，还是共享单车，是不是都是很短时间内就要积累大量用户，否则就会错过风口。所以应用需要在很短的时间内快速迭代，不断调整，满足用户体验；还要在很短的时间内，具有支撑高并发请求的能力。

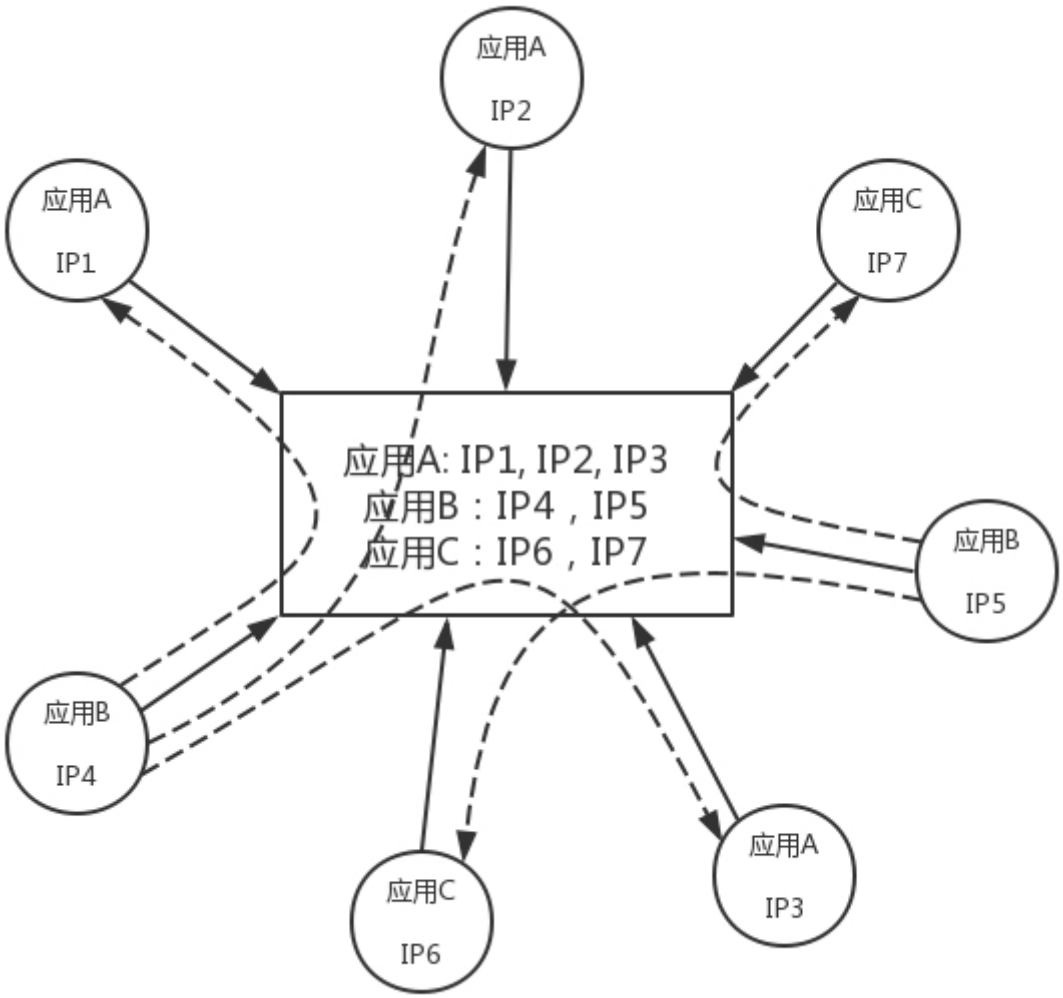
单体应用作为个人英雄主义的时代已经过去了。如果所有的代码都在一个工程里面，开发的时候必然存在大量冲突，上线的时候，需要开大会进行协调，一个月上线一次就很不错了。而且所有的流量都让一个进程扛，怎么也扛不住啊！

没办法，一个字：拆！拆开了，每个子模块独自变化，减少相互影响。拆开了，原来一个进程扛流量，现在多个进程一起扛。所以，微服务就是从个人英雄主义，变成集团军作战。

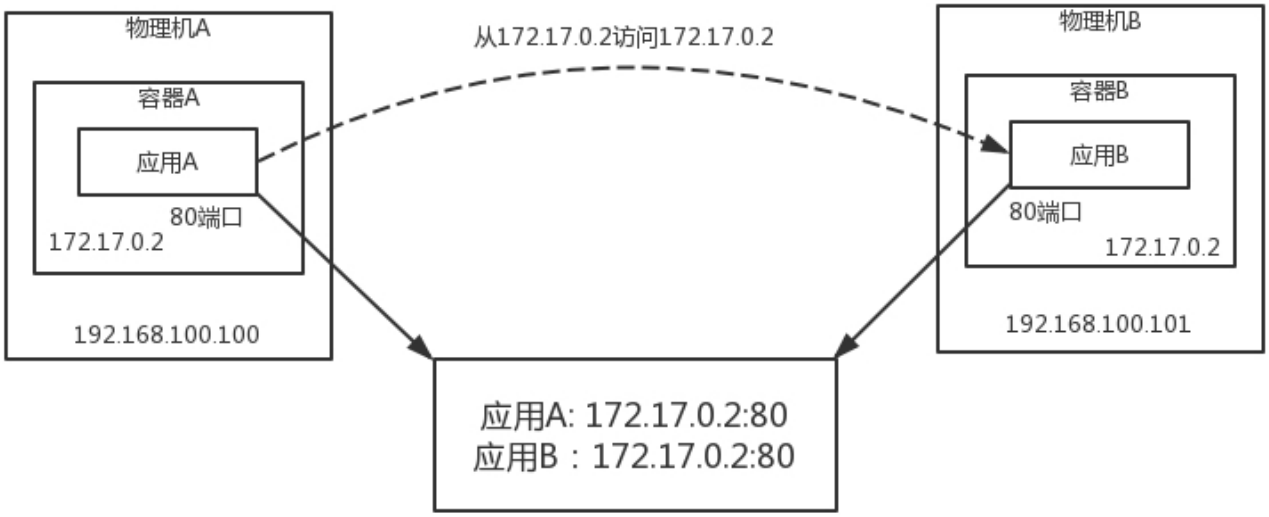
容器作为集装箱，可以保证应用在不同的环境中快速迁移，提高迭代的效率。但是如果要形成容器集团军，还需要一个集团军作战的调度平台，这就是 Kubernetes。它可以灵活地将一个容器调度到任何一台机器上，并且当某个应用扛不住的时候，只要在 Kubernetes 上修改容器的副本数，一个应用马上就能变八个，而且都能提供服务。

然而集团军作战有个重要的问题，就是通信。这里面包含两个问题，第一个是集团军的 A 部队如何实时地知道 B 部队的位置变化，第二个是两个部队之间如何相互通信。

第一个问题位置变化，往往是通过一个称为注册中心的地方统一管理的，这个是应用自己做的。当一个应用启动的时候，将自己所在环境的 IP 地址和端口，注册到注册中心指挥部，这样其他的应用请求它的时候，到指挥部问一下它在哪里就好了。当某个应用发生了变化，例如一台机器挂了，容器要迁移到另一台机器，这个时候 IP 改变了，应用会重新注册，则其他的应用请求它的时候，还是能够从指挥部得到最新的位置。

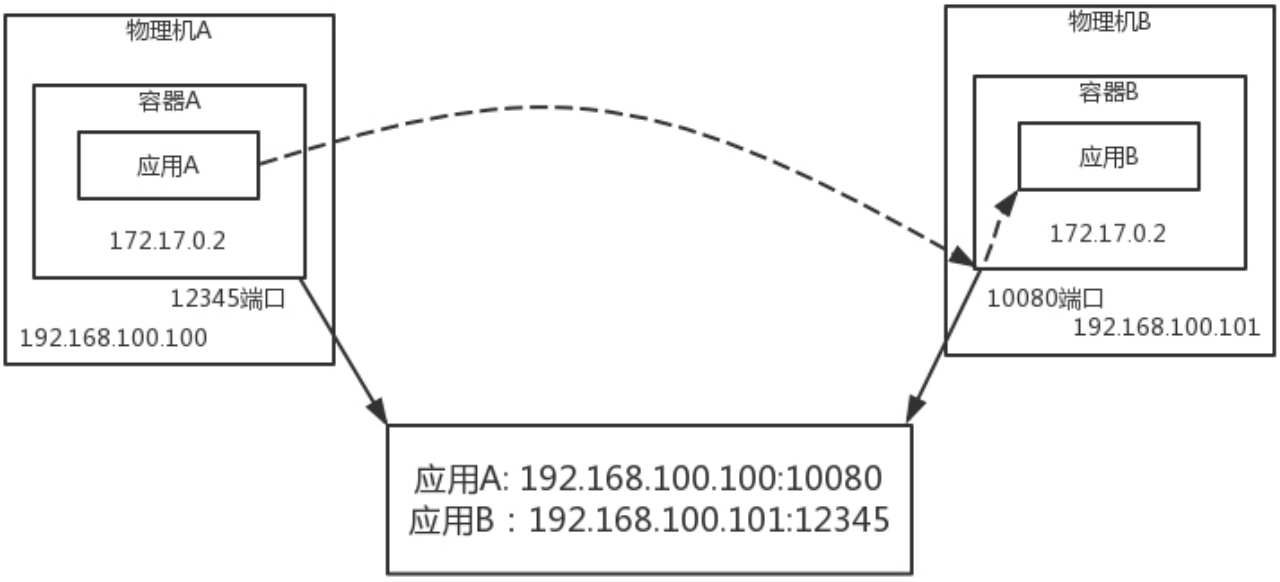


接下来是如何相互通信的问题。NAT 这种模式，在多个主机的场景下，是存在很大问题的。在物理机 A 上的应用 A 看到的 IP 地址是容器 A 的，是 172.17.0.2，在物理机 B 上的应用 B 看到的 IP 地址是容器 B 的，不巧也是 172.17.0.2，当它们都注册到注册中心的时候，注册中心就是这个图里这样子。



这个时候，应用 A 要访问应用 B，当应用 A 从注册中心将应用 B 的 IP 地址读出来的时候，就彻底困惑了，这不是自己访问自己吗？

怎么解决这个问题呢？一种办法是不去注册容器内的 IP 地址，而是注册所在物理机的 IP 地址，端口也要是物理机上映射的端口。



这样存在的问题是，应用是在容器里面的，它怎么知道物理机上的 IP 地址和端口呢？这明明是运维人员配置的，除非应用配合，读取容器平台的接口获得这个 IP 和端口。一方面，大部分分布式框架都是容器诞生之前就有了，它们不会适配这种场景；另一方面，让容器内的应用意识到容器外的环境，本来就是非常不好的设计。

说好的集装箱，说好的随意迁移呢？难道要让集装箱内的货物意识到自己传的信息？而且本来 Tomcat 都是监听 8080 端口的，结果到了物理机上，就不能大家都用这个端口了，否则端口就冲突了，因而就需要随机分配端口，于是在注册中心就出现了各种各样奇怪的端口。无论是注册中心，还是调用方都会觉得很奇怪，而且不是默认的端口，很多情况下也容易出错。

Kubernetes 作为集团军作战管理平台，提出指导意见，说网络模型要变平，但是没说怎么实现。于是业界就涌现了大量的方案，Flannel 就是其中之一。

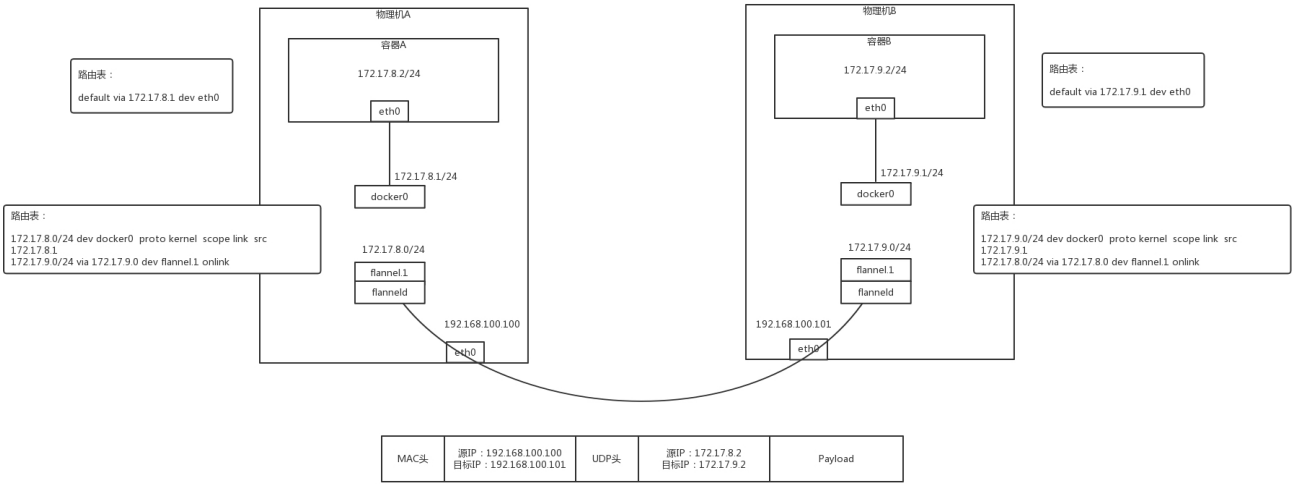
对于 IP 冲突的问题，如果每一个物理机都是网段 172.17.0.0/16，肯定会冲突啊，但是这个网段实在太大了，一台物理机上根本启动不了这么多的容器，所以能不能每台物理机在这个大网段里面，抠出一个小的网段，每个物理机网段都不同，自己看好自己的一亩三分地，谁也不和谁冲突。

例如物理机 A 是网段 172.17.8.0/24，物理机 B 是网段 172.17.9.0/24，这样两台机器上启动的容器 IP 肯定不一样，而且就看 IP 地址，我们就一下子识别出，这个容器是本机的，还是远程的，如果是远程的，也能从网段一下子就识别出它归哪台物理机管，太方便了。

接下来的问题，就是物理机 A 上的容器如何访问到物理机 B 上的容器呢？

你是不是想到了熟悉的场景？虚拟机也需要跨物理机互通，往往通过 Overlay 的方式，容器是不是也可以这样做呢？

这里我要说 Flannel 使用 UDP 实现 Overlay 网络的方案。



在物理机 A 上的容器 A 里面，能看到的容器的 IP 地址是 172.17.8.2/24，里面设置了默认的路由规则 default via 172.17.8.1 dev eth0。

如果容器 A 要访问 172.17.9.2，就会发往这个默认的网关 172.17.8.1。172.17.8.1 就是物理机上面 docker0 网桥的 IP 地址，这台物理机上的所有容器都是连接到这个网桥的。

在物理机上面，查看路由策略，会有这样一条 172.17.0.0/24 via 172.17.0.0 dev flannel.1，也就是说发往 172.17.9.2 的网络包会被转发到 flannel.1 这个网卡。

这个网卡是怎么出来的呢？在每台物理机上，都会跑一个 flanneld 进程，这个进程打开一个 /dev/net/tun 字符设备的时候，就出现了这个网卡。

你有没有想起 qemu-kvm，打开这个字符设备的时候，物理机上也会出现一个网卡，所有发到这个网卡上的网络包会被 qemu-kvm 接收进来，变成二进制串。只不过接下来 qemu-kvm 会模拟一个虚拟机里面的网卡，将二进制的串变成网络包，发给虚拟机里面的网卡。但是 flanneld 不用这样做，所有发到 flannel.1 这个网卡的包都会被 flanneld 进程读进去，接下来 flanneld 要对网络包进行处理。

物理机 A 上的 flanneld 会将网络包封装在 UDP 包里面，然后外层加上物理机 A 和物理机 B 的 IP 地址，发送给物理机 B 上的 flanneld。

为什么是 UDP 呢？因为不想在 flanneld 之间建立两两连接，而 UDP 没有连接的概念，任何一台机器都能发给另一台。

物理机 B 上的 flanneld 收到包之后，解开 UDP 的包，将里面的网络包拿出来，从物理机 B 的 flannel.1 网卡发出去。

在物理机 B 上，有路由规则 172.17.9.0/24 dev docker0 proto kernel scope link src 172.17.9.1。

将包发给 docker0，docker0 将包转给容器 B。通信成功。

上面的过程连通性没有问题，但是由于全部在用户态，所以性能差了一些。

跨物理机的连通性问题，在虚拟机那里有成熟的方案，就是 VXLAN，那能不能 Flannel 也用 VXLAN 呢？

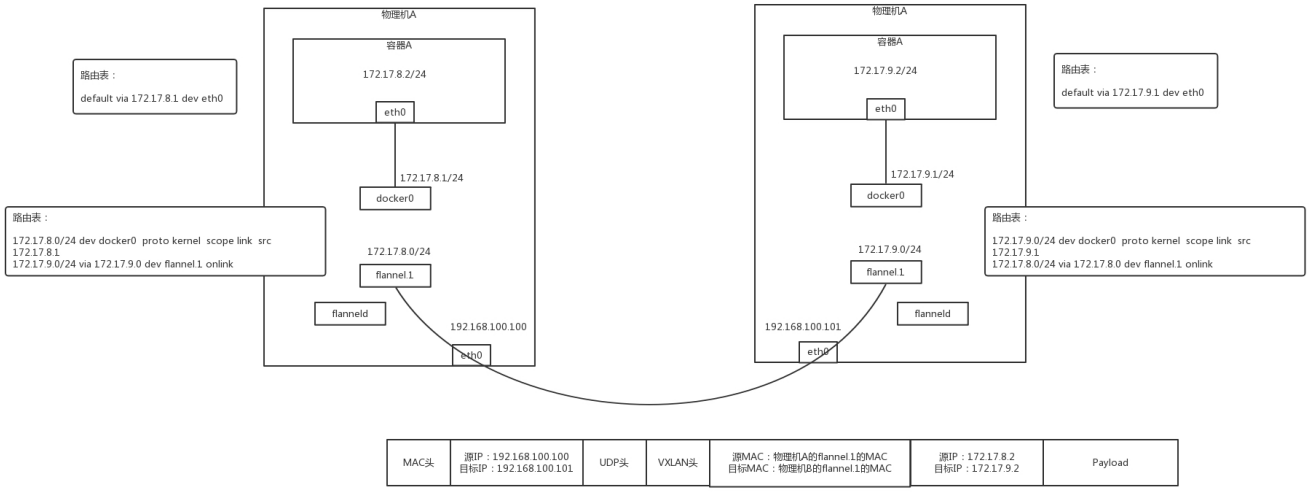
当然可以了。如果使用 VXLAN，就不需要打开一个 TUN 设备了，而是要建立一个 VXLAN 的 VTEP。如何建立呢？可以通过 netlink 通知内核建立一个 VTEP 的网卡 flannel.1。在我们讲 OpenvSwitch 的时候提过，netlink 是一种用户态和内核态通信的机制。

当网络包从物理机 A 上的容器 A 发送给物理机 B 上的容器 B，在容器 A 里面通过默认路由到达物理机 A 上的 docker0 网卡，然后根据路由规则，在物理机 A 上，将包转发给 flannel.1。这个时候 flannel.1 就是一个 VXLAN 的 VTEP 了，它将网络包进行封装。

内部的 MAC 地址这样写：源为物理机 A 的 flannel.1 的 MAC 地址，目标为物理机 B 的 flannel.1 的 MAC 地址，在外面加上 VXLAN 的头。

外层的 IP 地址这样写：源为物理机 A 的 IP 地址，目标为物理机 B 的 IP 地址，外面加上物理机的 MAC 地址。

这样就能通过 VXLAN 将包转发到另一台机器，从物理机 B 的 flannel.1 上解包，变成内部的网络包，通过物理机 B 上的路由转发到 docker0，然后转发到容器 B 里面。通信成功。



小结

好了，今天的内容就到这里，我来总结一下。

- 基于 NAT 的容器网络模型在微服务架构下有两个问题，一个是 IP 重叠，一个是端口冲突，需要通过 Overlay 网络的机制保持跨节点的连通性。
- Flannel 是跨节点容器网络方案之一，它提供的 Overlay 方案主要有两种方式，一种是 UDP 在用户态封装，一种是 VXLAN 在内核态封装，而 VXLAN 的性能更好一些。

最后，给你留两个问题：

1. 通过 Flannel 的网络模型可以实现容器与容器直接跨主机的互相访问，那你知道如果容器内部访问外部的服务应该怎么融合到这个网络模型中吗？
2. 基于 Overlay 的网络毕竟做了一次网络虚拟化，有没有更加高性能的方案呢？

我们的专栏更新到第 30 讲，不知你掌握得如何？每节课后我留的思考题，你都有没有认真思考，并在留言区写下答案呢？我会从**已发布的文章中选出一批认真留言的同学**，赠送学习奖励礼券和我整理的独家网络协议知识图谱。

欢迎你留言和我讨论。趣谈网络协议，我们下期见！

精选留言

Jay 1

“例如物理机 A 是网段 172.17.8.0/24，物理机 B 是网段 172.17.9.0/24”，这里应该是指物理机A给docker分配的网段吧？老师，这个容易造成误解哦……

2018-08-03

刘工的一号马由 1

1默认路由发到容器网络的网关

2underlay网络

2018-07-26

donson 0

flannel网络的地址段是10.242.0.0/16，默认每个Node上的子网划分是10.242.x.0/24，请问这个子网划分在哪里可以配置，比如我想划分成/21

2018-09-03

小宇宙 0

flannel的backend除了UDP和vxlan还有一种模式就是host-gw，通过主机路由的方式，将请求发送到容器外部的应用，但是有个约束就是宿主机要和其他物理机在同一个vlan或者局域网中，这种模式不需要封包和解包，因此更加高效。

2018-08-27

作者回复

对的

2018-08-27

selboo 0

物理机A和物理机B是以access模式连接到交换机。此时flannel.1 在加上 VXLAN 头可以通信吗？？？求老师解答。

2018-08-07

_CountingStars 0

1.容器访问外面还是应该用nat

2.使用bgp协议的其他实现 calico kube-router

2018-07-25