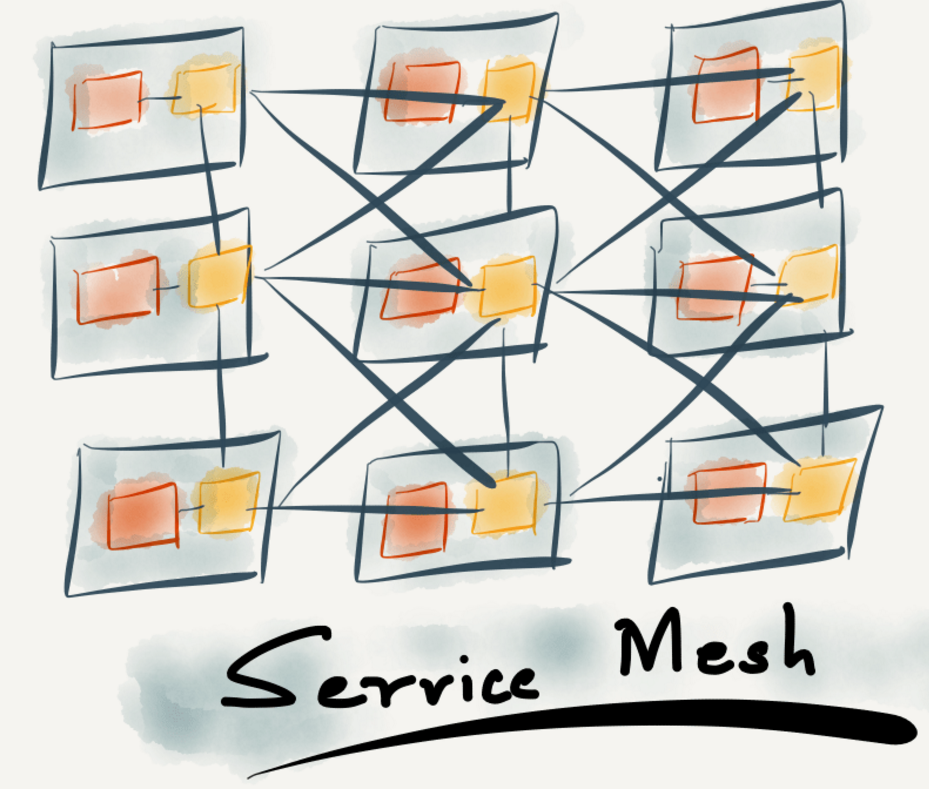
**ServiceMesh**

**1 Service Mesh（服务网格）会是今年微服务生态的主角吗？**

从趋势来看，众多企业正在将这项理微服务复杂性的技术/工具，搬进他们的IT“火药库”之中。

**2 什么是Service Mesh？**

根据Linkerd CEO William Morgan定义，Service Mesh是用于处理服务间通信的基础设施层，用于在云原生应用复杂的服务拓扑中实现可靠的请求传递。在实践中，Service Mesh通常是一组与应用一起部署，但对应用透明的轻量级网络代理。Service Mesh与传统基础设施层不同之处在于，它形成了一个分布式的互连代理网络，以sidecar形式部署在服务两侧，服务对于代理无感知，且服务间所有通信都由代理进行路由。



**3 为什么需要Service Mesh？**

**（1）“Smart endpoint and dumb pipes”**是微服务架构在集成服务时采用的一个核心理念，这一理念改变了过去臃肿集中的ESB（企业服务总线），无疑是正确方向上的一大进步，但同时也给我们出了一些难题——多智能才不会过于智能，而服务轻重大小的程度如何拿捏？我们应该如何处理微服务系统中服务间交互的复杂性？放在服务内部还是外部？如果是内部，如何处理业务逻辑关系，或者应该与基础设施更为相关？如果是外部，如何避免重蹈ESB的覆辙？

**（2）处理服务间通信时需要关注的点：**

服务发现

负载均衡

路由

流量控制

通信可靠性

弹性

安全

监控/日志

似乎都是老生常谈，存在于任何需要处理网络的分布式系统之中，区别在于，当所涉及微服务数量呈指数级增加，这些问题也会被相应放大。

1. **一个已经被广泛应用的解决方案是利用api网关来处理**服务外部和服务之间的请求，提供例如服务发现、路由、监控、流量控制等。然而，api网关有一个比较致命的缺陷，它容易出现单点故障并且实践不当很有可能会变得异常臃肿。另一方面，api网关核心是面向用户，也就是说它可以解决从用户到微服务的流量问题，但不能解决所有问题，而我们需要的是一个完整的方案，或者至少是一些能够与api网关互补的方案和工具。

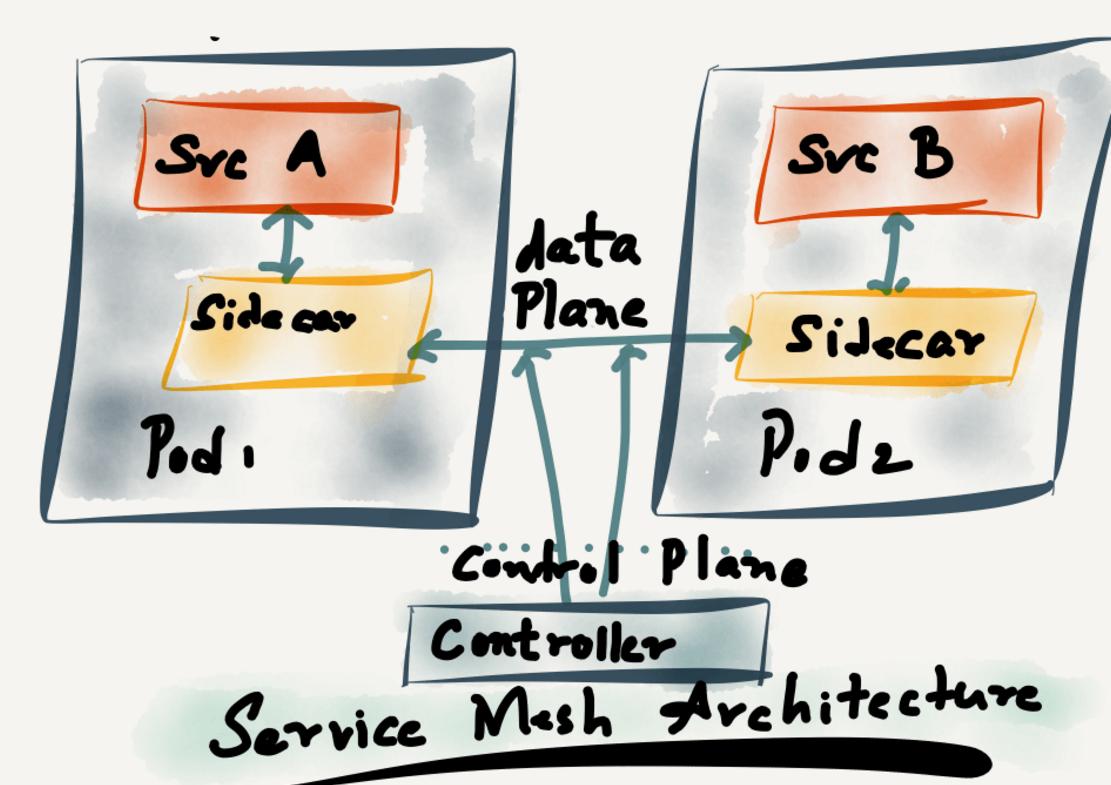
**（4）另一种选择是在网络堆栈的较低层级（4/3）进行可靠性、**监控、流量控制等方面处理。这种选择的问题是，在较低较低的操作难易满足应用层的问题。联想end-to-end（端到端）的理论，我们前面提到的那几个关注点实际上还是集中在应用层，也只能在应用层成功实现。

像Netflix、Twitter等SOA/微服务的早期采用者，他们通过建立内部库的方式处理这些问题，然后提供给所有服务使用。这种方法的问题在于，把库扩展到成百上千个微服务中难度极高，而且这些库相对来说是比较”脆弱“的，我们很难保证他们可以适应所有的技术堆栈选择。

程度上来说，Service Mesh与这些库很类似，但Service Mesh是与服务相邻的独立进程。服务连接到代理，代理反过来又与其他代理（HTTP1.1/2、GRPC）进行通信。它们是相对独立的进程，在应用层或应用层之下分布和运行，进而解决了上述两个方案存在的缺陷。

**4 Service Mesh架构**

Service Mesh由data plane构成，其中所有服务通过sidecar代理进行服务通信。（所有代理相互连接形成一个Mesh，Service Mesh由此得名）网格同时包含一个control plane——可以将所有独立的sidecar代理连接到一个分布式网络中，并设置网格还包括一个控制平面——它将所有独立的sidecar代理连接到一个分布式网络中，并设置由data plane指定的策略。



Control plane定义服务发现、路由、流量控制等策略。这些策略可以是全局的，也可以是限定的。Data plane负责在通信时应用和执行这些策略。

5总结来说

Service Mesh是“时间的产物”，Docker、Kubernetes等容器技术直接推进了对于Service Mesh的需求，让复杂的系统可以被轻松部署和管理。

未来Service Mesh将如何发展还未可知，或许会像TCP/IP一样形成标准，或许不同工具和平台会独具一格……但有一点是确认的，Service Mesh对于微服务生态的价值令人难以忽视，能够将繁重的服务治理工作变得简单高效，何乐而不为？