通信

进程间的内部通信是分布式系统的核心。如果不自信的研究不同机器之间的进程是如何通信的，那么学习分布式系统就没有意义了。这种通信都是基于底层的网络的。通过消息传递来表达通信比使用基于共享内存的原语要困难。现代分布式系统通常由分散在网络(如Internet)上的数千个甚至数百万个进程组成，然而这些进程之间的通信并不可靠。除非计算机网络的原始通信设施被其他东西取代，否则大规模分布式应用程序的开发是极其困难的。

在本章中，我们首先讨论通信过程必须遵循的规则(称为协议)，并着重于以层的形式构造这些协议。之后我们将着眼于两个广泛使用的模型：Remote Procedure Call（RPC）和 Message-Oriented Middleware（MOM）。同样的，我们将讨论多播问题。

我们在分布式系统中的第一个通信模型是远程过程调用(RPC)。RPC旨在隐藏大多数复杂的消息传递，非常适合客户机-服务器应用程序。然而，以透明的方式实现rpc说起来容易做起来难。我们将查看一些不可忽略的重要细节，同时深入研究实际代码，以说明在性能可以接受的情况下，能将透明性实现到什么程度。

在许多分布式应用程序中，通信并不遵循相当严格的客户机-服务器交互模式。在这些情况下，事实证明用信息来思考更合适。计算机网络的底层通信设施在很多方面都是不合适的，同样是由于他们缺乏分布式的透明性。另一个替代方法是使用一种高级别的消息队列模型，就像使用邮件系统通信一样。面向消息通信是一个重要的主题。我们研究了许多方面，包括应用级别的路由。

最后，由于我们对组播设施的理解有所提高，出现了新颖而优雅的数据传播解决方案。

我们将在本章的最后一节单独关注这个主题，讨论传统的确定性组播方法，以及泛洪和gossiping中使用的概率方法。由于其优雅和简单，后者在过去几年中受到越来越多的关注。

4.1 基础

在开始讨论分布式系统中的通信之前，我们首先概述一些与通信相关的基本问题。在下一节中，我们将简要讨论网络通信协议，因为这些协议构成了任何分布式系统的基础。然后，我们采用不同的方法，对分布式系统中通常发生的不同类型的通信进行分类。

**层级协议**

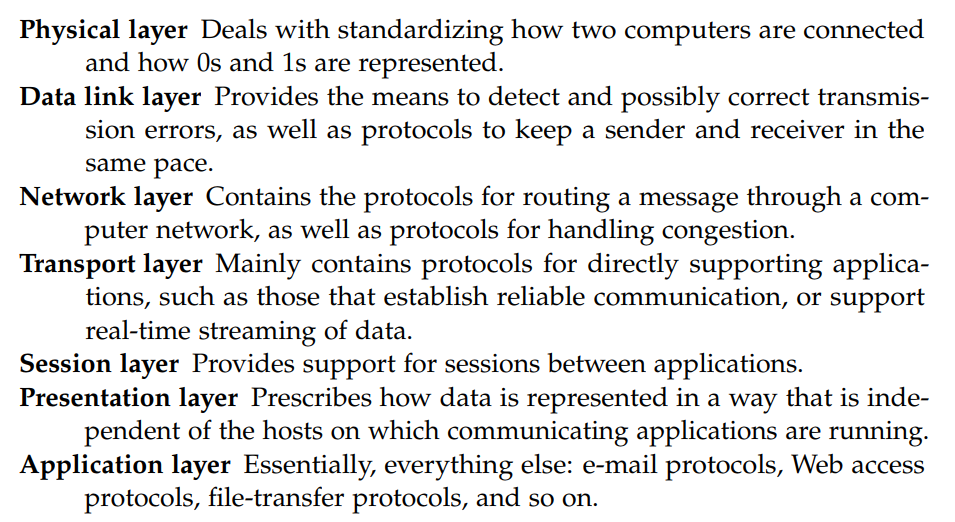
由于缺乏共享内存，分布式系统中的所有通信都基于发送和接收(低层)消息。当进程P希望与进程Q通信时，它首先在自己的地址空间中构建一条消息。然后它执行一个系统调用，使操作系统通过网络向Q发送消息。虽然这个基本的想法听起来很简单，但是为了防止混乱，P和Q必须就发送的比特的意义达成一致。

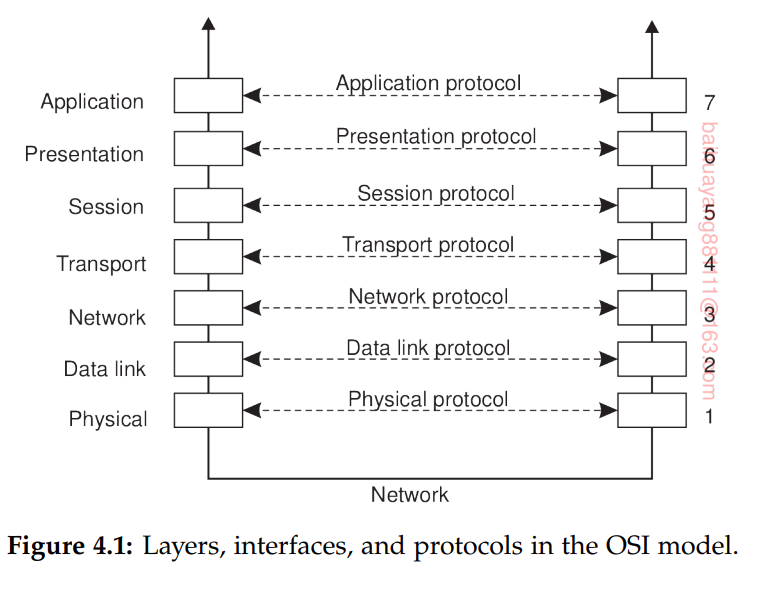
**OSI相关模型**

为了更容易处理通信中涉及的众多级别和问题，国际标准组织(ISO)开发了一个参考模型，该模型清楚地标识了涉及的各个级别，给出了它们的标准名称，并指出哪个级别应该做哪个工作。这个模型被称为开放系统互连参考模型[Day and Zimmerman, 1983]，通常缩写为ISO OSI或者有时只是OSI模型。应该强调的是，作为OSI模型的一部分而开发的协议从来没有被广泛使用，而且基本上已经死了。然而，事实证明，底层模型本身对于理解计算机网络非常有用。虽然我们不打算在这里对这个模型及其所有含义进行全面描述，但是做一个简短的介绍还是会有用的。

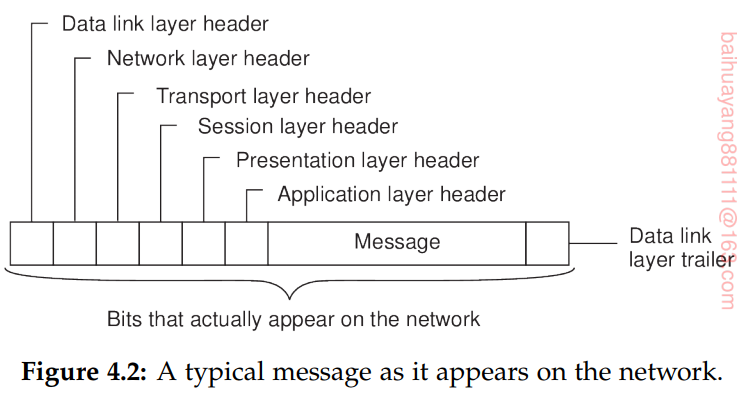
OSI模型是为了让开放系统通信而设计的。开放系统是一种通过使用标准的格式，内容和消息内容的规则，来和其他的系统进行通信的系统。这些规则称之为**通信协议**。为了能够使得计算机可以通过网络进行通信，通信协议是必须要被使用的。一种协议其实就是一种通信服务。这里有两种类型的服务。对于面向连接的服务，在交换数据之前，发送方和接收方首先显式地建立连接，并可能协商它们将使用的协议的特定参数。当它们完成时，它们释放(终止)连接。电话是一种典型的面向连接的通信服务。对于无连接的服务，不需要预先设置。当第一个消息准备好时，发送方就发送它。将信件放入邮箱是使用无连接通信服务的一个例子。在计算机世界中，面向连接和无连接通信都很常见。

在OSI模型中，通信被分为了7层，如图4.1。每一层都为上一层提供了一个或者更多个的特殊的通信服务。通过这种方式，将消息从A发送到B的问题就可以划分成许多可管理的部分，每一部分都可以独立的和其他层来处理。每一层都为上一层提供一个接口。接口由一组操作组成，这些操作一起定义了层准备提供的服务。这七层分别是：





当进程P希望与某个远程进程Q通信时，它构建一条消息，并通过接口将该消息传递给应用程序层。这个接口通常以库过程的形式出现。然后，应用层软件在消息的前面添加一个头，并通过层6/7接口将结果消息传递给表示层。表示层依次添加自己的头并将结果传递到会话层，以此类推。有些图层不仅在前面添加了一个header，而且在末尾添加了一个tail。当它到达底部时，物理层实际上通过将消息放到物理传输介质上来传输消息(到目前为止，看起来可能如图4.2所示)。



当消息到达承载Q的远程机器时，它向上传递，每一层都被剥离并检查它自己的报头。最后，消息到达接收方进程Q，进程Q可能使用相反的路径对消息进行应答（如图4.2，进程p按照从内向外的顺序增加请求头，进程q按照从外向内的顺序解析请求头）。第n层标头中的信息用于第n层协议。

在OSI模型中，一共有7层，如图4.1。在特定系统中使用的协议集合称为协议套件或协议栈。将引用模型与其实际协议区分开来是很重要的。如上所述，OSI协议从来没有流行过，这与为Internet开发的协议(如TCP和IP)形成了鲜明的对比。

**中间件协议**

中间件是一个逻辑上存在于OSI应用层中的应用程序，但它包含许多通用协议，这些协议保证它们自己的层独立于其他更具体的应用程序。我们主要看几个例子。

域名系统(DNS) [Liu and Albitz, 2006]是一种分布式服务，用于查找与名称相关的网络地址，例如www.distributedsystems.net之类的所谓域名的地址。就OSI参考模型而言，DNS是一个应用程序，因此在逻辑上被放置在应用程序层。然而，DNS提供的是一种通用的、独立于应用程序的服务，这一点应该是相当明显的。可以说，它是中间件的一部分。

另一个例子是，有多种方法可以建立身份验证，即提供已声明身份的证明。身份验证协议并不与任何特定的应用程序紧密相连，而是可以作为通用服务集成到中间件系统中。同样，授权协议通常具有与应用程序无关的特性，通过授权协议，经过身份验证的用户和进程只能访问他们具有授权的资源。作为OSI参考模型中的应用程序，这些都是属于中间件的清晰示例。

分布式提交协议规定，在一组进程中(可能分布在许多机器上)，要么所有进程执行特定的操作，要么根本不执行操作。这种现象也称为**原子性**，在事务中得到了广泛的应用。事实证明，提交协议可以独立于特定应用程序呈现接口，从而提供通用事务服务。在这种形式下，它们通常属于中间件，而不属于OSI应用层。

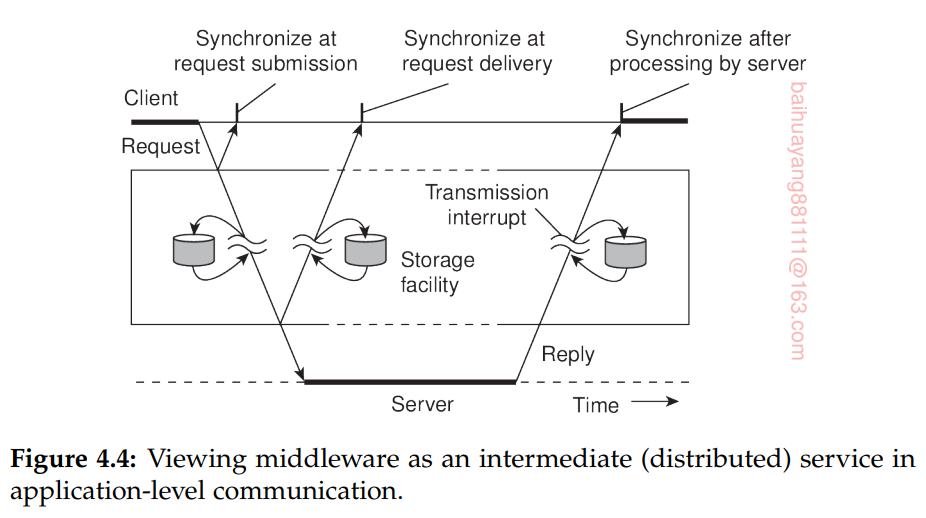
作为最后一个示例，考虑一个分布式锁定协议，通过该协议，可以保护资源不受分布在多台机器上的进程集合的同时访问。不难想象，这样的协议可以以独立于应用程序的方式设计，并且可以通过相对简单的、同样独立于应用程序的接口进行访问。因此，它们通常属于中间件。

这些协议示例并不直接绑定到通信，但是也有许多中间件通信协议。例如，所谓的**远程过程调用**，就是提供了一个工具来在本地调用实现于远程机器的方法过程。该通信服务属于最古老的中间件服务类型之一，用于实现访问透明性。同样的，有用于设置和同步流以传输实时数据的高级通信服务，例如多媒体应用程序所需的流。最后一个例子是，一些中间件系统提供可靠的多播服务，可以扩展到广域网上的数千个接收器。

采用这种方法进行分层，可以得到经过调整和简化的通信参考模型，如图4.3所示。与OSI模型相比，会话和表示层已经被一个包含独立于应用程序协议的中间件层所取代。这些协议不属于我们刚才讨论的底层。网络和传输服务由管理着底层硬件建立通信的操作系统分成了几组通信服务。

**通信类型**

在本章的其余部分中，我们将集中讨论高级中间件通信服务。在此之前，还有其他一些用于区分(中间件)通信的通用标准。为了理解中间件可以为应用程序提供的各种通信替代方案，我们将中间件视为客户机-服务器计算中的附加服务，如图4.4所示。例如电子邮件系统。原则上，邮件传递系统的核心可以看作是中间件通信服务。每个主机运行一个用户代理，允许用户编写、发送和接收电子邮件。发送用户代理将此类邮件传递给邮件传递系统，并期望它最终将邮件传递给目标收件人。同样，接收方的用户代理连接到邮件传递系统，查看是否有邮件进入。如果是，则将消息传输到用户代理，以便用户可以显示和读取它们。



电子邮件系统是一个典型的例子，其中通信是持久的。使用**持久通信**，通信中间件存储已提交用于传输的消息，直到将其交付给接收方为止。在这种情况下，中间件将在图4.4所示的一个或多个存储设施中存储消息。因此，发送应用程序无需在提交消息后继续执行（意思就是提交之后，会有别的中间件帮你进行发送消息）。同样，在提交消息时，接收应用程序不需要执行。

相反，使用**瞬态通信**，通信系统只在发送和接收应用程序执行时才存储消息。更准确地说，从图4.4中可以看出，如果中间件由于传输中断而不能传递消息，或者因为接收方当前没有活动，那么这个消息将被简单地丢弃。通常，所有传输级通信服务只提供瞬时通信。在这种情况下，通信系统由传统的存储转发路由器组成。如果路由器不能向下一个或目标主机发送消息，它将直接丢弃该消息。

通信除了分为持久的和短暂的，还可以分为异步的或同步的。**异步通信**的特征是，发送方在提交消息后可以立即继续执行其他操作（不用等待发送消息响应）。这意味着消息(临时)在提交时立即由中间件存储。使用**同步通信**，发送方将被阻塞，直到知道其请求被接受为止。同步通信有三个特点，第一，发送方会一直阻塞，直到中间件接管了请求。第二，发送方可以同步，直到其请求被发送到预期的接收方为止。第三，同步可以通过让发送方等待，直到其请求被完全处理，也就是说，直到接收方返回响应。

实践中，经常使用的是持久性和同步的各种结合。流行的方法是结合请求提交时的持久性和同步，这是许多消息队列系统的常见方案，我们将在本章后面讨论。同样，在请求被完全处理之后，使用同步的瞬态通信也得到了广泛的应用。这个方案与远程过程调用相对应，我们接下来将讨论这个问题。