应用层协议原理

网络应用程序体系结构

客户 - 服务器体系结构

有一个总是打开的主机称为**服务器**，它服务于来自许多其他称为**客户**的主机的请求。

在一个客户 - 服务器应用中，常常会出现一台单独的服务器主机跟不上它所有客户请求的情况。 例如，一个流行的社交网络站点如果仅有一台服务器来处理所有请求，将很快变得不堪重负。 为此，配备大量主机的**数据中心**常被用于创建强大的虚拟服务器 。 最为流行的因特网服务——如搜索引擎（如谷歌和 Bing）、因特网商务（如亚马逊和e-Bay）、基于 Web 的电子邮件（如 Gmail 和雅虎邮件）、社交网络（如脸谱和推特），就应用了一个或多个数据中心 。

P2P 体系结构

对位于数据中心的专用服务器有最小的（或者没有）依赖 。 相反，应用程序在间断连接的主机对之间使用直接通信，这些主机对被称为**对等方** 。

P2P 体系结构的最引人人胜的特性之一是它们的 自扩展性 。 例如，在一个 P2P 文件共享应用中，尽管每个对等方都由于请求文件产生工作量，但每个对等方通过向其他对等方分发文件也为系统增加服务能力 。 P2P 体系结构也是成本有效的，因为它们通常不需要庞大的服务器基础设施和服务器带宽（这与具有数据中心的客户 - 服务器设计形成反差） 。

然而，未来 P2P 应用面临三个主要挑战：

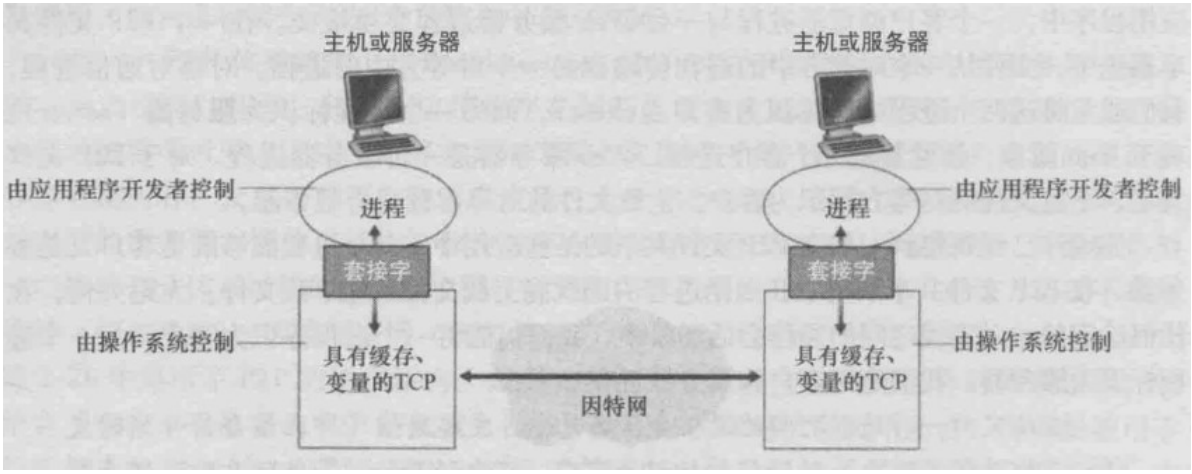
- ISP 友好 。 大多数住宅 ISP （包括 DSL 和电缆 ISP ）已经受制于“非对称的”带宽应用，也就是说，下载比上载要多得多 。 但是 P2P 视频流和文件分发应用改变了从服务器到住宅 ISP 的上载流量，因而给 ISP 带来了巨大压力 。 未来 P2P 应用需要设计对 ISP 友好的模式 。
- 安全性 。 因为它们的高度分布和开放特性， P2P 应用给安全带来挑战。
- 激励 。 未来 P2P 应用的成功也取决于说服用户自愿向应用提供带宽、存储和计算资源，这对激励设计带来挑战 。

主机间的进程通信

在两子不同端系统上的进程，通过跨越计算机网络交换**报文**而相互通信。发送进程生成并向网络中发送报文；接收进程接收这些报文并可能通过将报文发送回去进行响应 。

进程通过一个称为**套接字**（socket）的软件接口向网络发送报文和从网络接收报文。

如图显示了两个经过因特网通信的进程之间的套接宇通信（图 2-3 中假定由该进程使用的下面传输层协议是因特网的 TCP 协议） 。 如该图所示，套接字是同一台主机内应用层与传输层之间的接口 。 由于该套接字是建立网络应用程序的可编程接口，因此套接字也称为应用程序和网络之间的应用程序编程接口（ Application Programming Interface, API ） 。



应用程序开发者可以控制套接字在应用层端的一切，但是对该套接字的传输层端几乎没有控制权 。 应用程序开发者对于传输层的控制仅限于：

- 选择传输层协议；
- 设定几个传输层参数，如最大缓存和最大报文段长度等

进程寻址

在一台主机上运行的进程为了向在另一台主机上运行的进程发送分组，接收进程需要有一个地址。 为了标识该接收进程，需要定义两种信息：

- 主机的地址；

  > 在因特网中，主机由其 **IP 地址**标识。 P 地址是一个 32 比特的量且它能够唯一地标识该主机就够了。

- 定义在目的主机中的接收进程的标识符。

  > 除了知道报文送往目的地的主机地址外，发送进程还必须指定运行在接收主机上的接收进程（更具体地说，接收套接字） 。 因为一般而言一台主机能够运行许多网络应用，这些信息是需要的 。 **目的地端口号**用于这个目的 。 已经给流行的应用分配了特定的端口号 。
  >
  > - Web（HTTP）服务器端口号 80。
  > - 邮件服务器（SMTP）端口号 25。
  > - HTTPS端口号为使用443。
  > - DNS端口号为53。

可供应用程序使用的传输服务

可靠数据传输

当一个传输协议提供这种服务时，发送进程只要将其数据传递进套接字，就可以完全相信该数据将能无差错地到达接收进程。
当一个传输层协议不提供可靠数据传输时，由发送进程发送的某些数据可能不能够到达接收进程。 这可能能被**容忍丢失的应用（ loss- tolerant application ）**所接受，值得注意的是多媒体应用，如交谈式音频 / 视频，它们能够承受一定量的数据丢失。在这些多媒体应用中，丢失的数据引起播放的音频 / 视频出现小干扰，而不是致命的损伤。

吞吐量

具有吞吐量要求的应用程序被称为**带宽敏感的应用**(bandwidth- sensitive application)。 许多当前的多媒体应用是带宽敏感的，尽管某些多媒体应用程序可能采用自适应编码技术对数字语音或视频以与当前可用带宽相匹配的速率进行编码 。带宽敏感的应用具有特定的吞吐量要求，

而**弹性应用**（ elastic application ） 能够根据情况或多或少地利用可供使用的吞吐量 。 电子邮件、文件传输以及 Web 传送都属于弹性应用 。 当然，吞吐量是越多越好。

实时性

在因特网电话中，较长的时延会导致会话中出现不自然的停顿；在多方游戏和虚拟互动环境中，在做出动作并看到来自环境（如来自位于端到端连接中另一端点的玩家）的响应之间，较长的时延使得它失去真实感 。 对于非实时的应用，较低的时延总比较高的时延好，但对端到端的时延没有严格的约束 。

安全性

这种服务将在发送和接收进程之间提供机密性，以防该数据以某种方式在这两个进程之间被观察到 。 传输协议还能提供除了机密性以外的其他安全性服务，包括数据完整性和端点鉴别。

> TCP与UDP均不保证安全。
>
> 因特网界已经研制了 TCP 的加强版本，称为安全套接字层（ Secure Sockets Layer, SSL ）。用 SSL 加强后的 TCP 不仅能够做传统的 TCP 所能做的一切，而且提供了关键的进程到进程的安全性服务，包括加密 、 数据完整性和端点鉴别。 我们强调 SSL 不是与 TCP 和 UDP 在相同层次上的第三种因特网传输协议，而是一种对 TCP 的加强，这种强化是在**应用层**上实现的。

因特网提供的传输层服务

应用	数据丢失	带宽	时间敏感
文件传输	不能丢失	弹性	不
电子邮件	不能丢失	弹性	不
Web 文档	不能丢失	弹性（几 kbps）	不
因特网电话/视频会议	容忍丢失	音频（几 kbps ~ 1Mbps）	是，100ms
		视频（10kbps ~ 5Mbps）	
存储音频/视频	容忍丢失	同上	是，几秒
交互式游戏	容忍丢失	几 kbps ~ 10kbps	是，100ms
即时讯息	不能丢失	弹性	是和不是

各种网络应用的要求

TCP / UDP，详见传输层的讲解

电子邮件、远程终端访问、 Web 、文件传输都使用了 TCP。 这些应用选择 TCP 的最主要原因是 TCP 握供了可靠数据传输服务，确保所有数据最终到达目的地。 因为因特网电话应用（如 Skype ）通常能够容忍某些丢失但要求达到一定的最小座率才能有效工作，所以网特网电话应用的开发者涵常愿意将该应用运行在 UDP 上，从而设法避开 TCP 的拥塞控制机制和分组开销。但网为许多防火墙被配置成阻抖（大多数类型的） UDP 流量，所以因特网电话应用通常设计成如果 UDP 通信失败就使用 TCP 作为备份 。

应用	应用层协议	支撑的运输协议
电子邮件	SMTP［RFC 5321］	TCP
远程终端访问	Telnet［RFC 854］	TCP
Web	HTTP［RFC 2616］	TCP
文件传输	FTP［RFC 959］	TCP
流式多媒体	HTTP（如 YouTube）	TCP
因特网电话	SIP［RFC 3261］、RTP［RFC 3550］或专用的（如 Skype）	UDP 或 TCP

流行的因特网应用及其应用层协议和支撑的传输协议