物联网中间件设计

第一章 概述

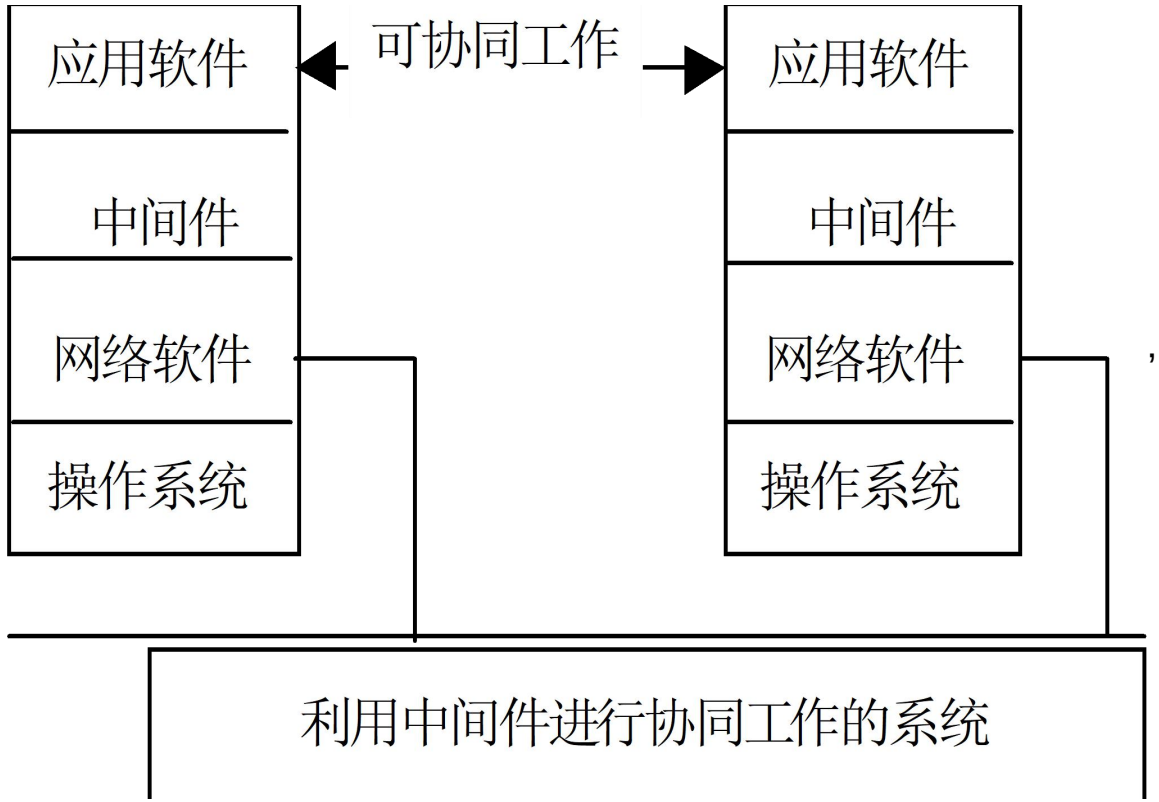
什么是中间件（Middleware）？

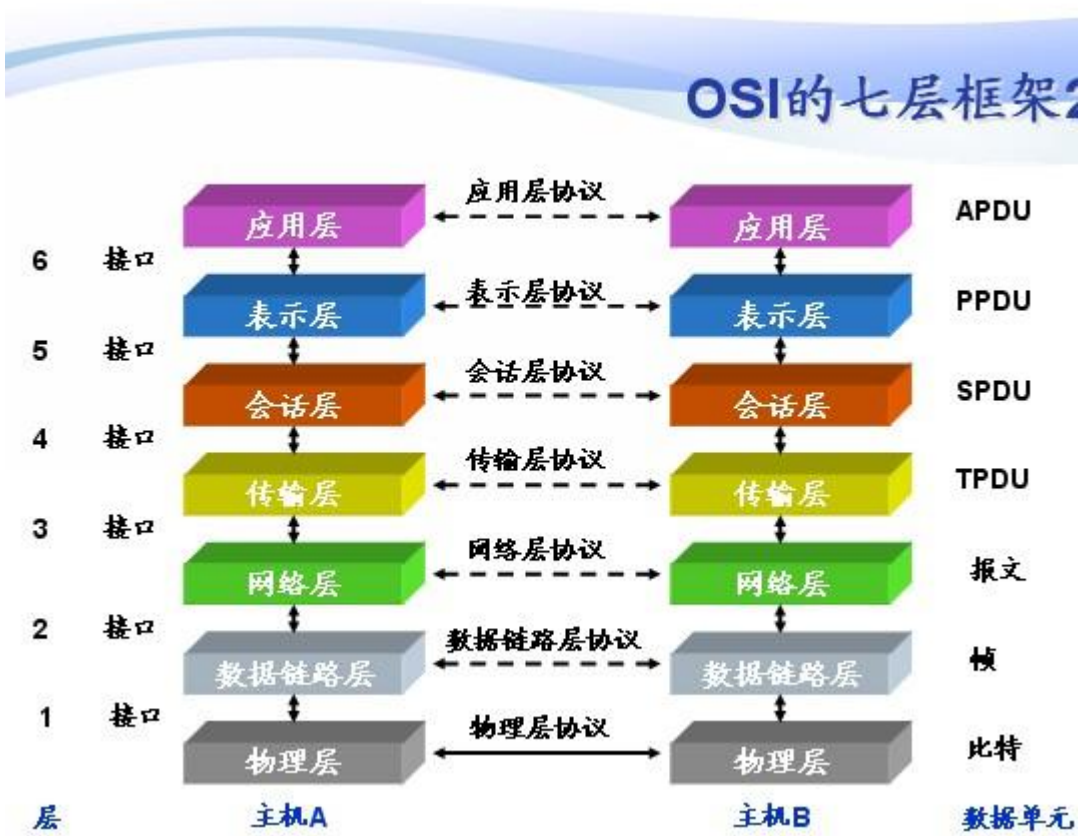
- 问题：如何解决位于两台不同的电脑上的两个计算机程序的通信问题。

什么是中间件（Middleware）？

- 中间件(Middleware)是一种软件
- 处于系统软件（操作系统和网络软件）与应用软件之间
- 能使应用软件之间进行跨网络的协同工作（也就是互操作)
- 允许各应用软件之下所涉及的“系统结构、操作系统、通信协议、数据库和其它应用服务”各不相同。

什么是中间件（Middleware）？





分层结构的优点

- 关于网络体系结构中的分层结构，下列哪一项是正确的
  - A、分层结构使得网络具有灵活性，易于实现和维护
  - B、所有的网络体系结构都用相同的层次名称和功能
  - C、分层结构把多种网络功能组合在同一层中，使用更方便
  - D、当某一层的具体实现方法发生变化时，必须对相邻层进行修改

答案

- 各层之间是独立的：某一层并不需要知道它的下一层是如何实现的，而仅仅需要知道该层通过层间的接口（即界面）所提供的服务。
- 灵活性好。当任何一层发生变化时（例如由于技术的变化），只要层间接口关系保持不变，则在这层以上或以下各层均不受影响。

广义的中间件

- 安全中间件
- 数据库中间件
- 所有不直接给客户提供业务价值的软件，与业务无关的软件都是中间件
- 任何软件有上层软件以及下层软件，那么中间层都可以称之为中间件

中间件简介

- 套接字（socket）
- 远程过程调用
- 分布式对象模型
- Web Services
- 云计算
- 消息中间件

套接字（socket）

- 套接字（socket）用于描述IP地址和端口，是一个通信链的句柄（句柄，类似C语言里的文件描述符）。应用程序通过套接字向网络发出请求或应答网络请求。

- 套接字是由操作系统直接提供的接口。最早是1983年BSD UNIX上定义的BSD sockets。目前，Linux 平台提供 POSIX sockets 接口，Window 平台提供 Winsock 接口。

套接字（socket）

- 套接字（socket）可以看作最原始的中间件。但是严格意义上来说套接字并不能算作中间件，因为应用程序使用套接字就是直接使用使用操作系统提供的网络通信接口。

- 高性能传输（点对点，传输数据量大）套接字依然是最合适的选择

远程过程调用（Remote Procedure Calls）

- 套接字提供的接口只有 write/read 的功能，只能传输字符流或者数据流，因此使用套接字不能直接调用其他电脑上的过程（函数）。

- 问题：本机如何使用套接字调用服务器上的过程function A(int x)?

远程过程调用（Remote Procedure Calls）

- 套接字提供的接口只有 write/read 的功能，只能传输字符流或者数据流，因此使用套接字不能直接调用其他电脑上的过程（函数）。

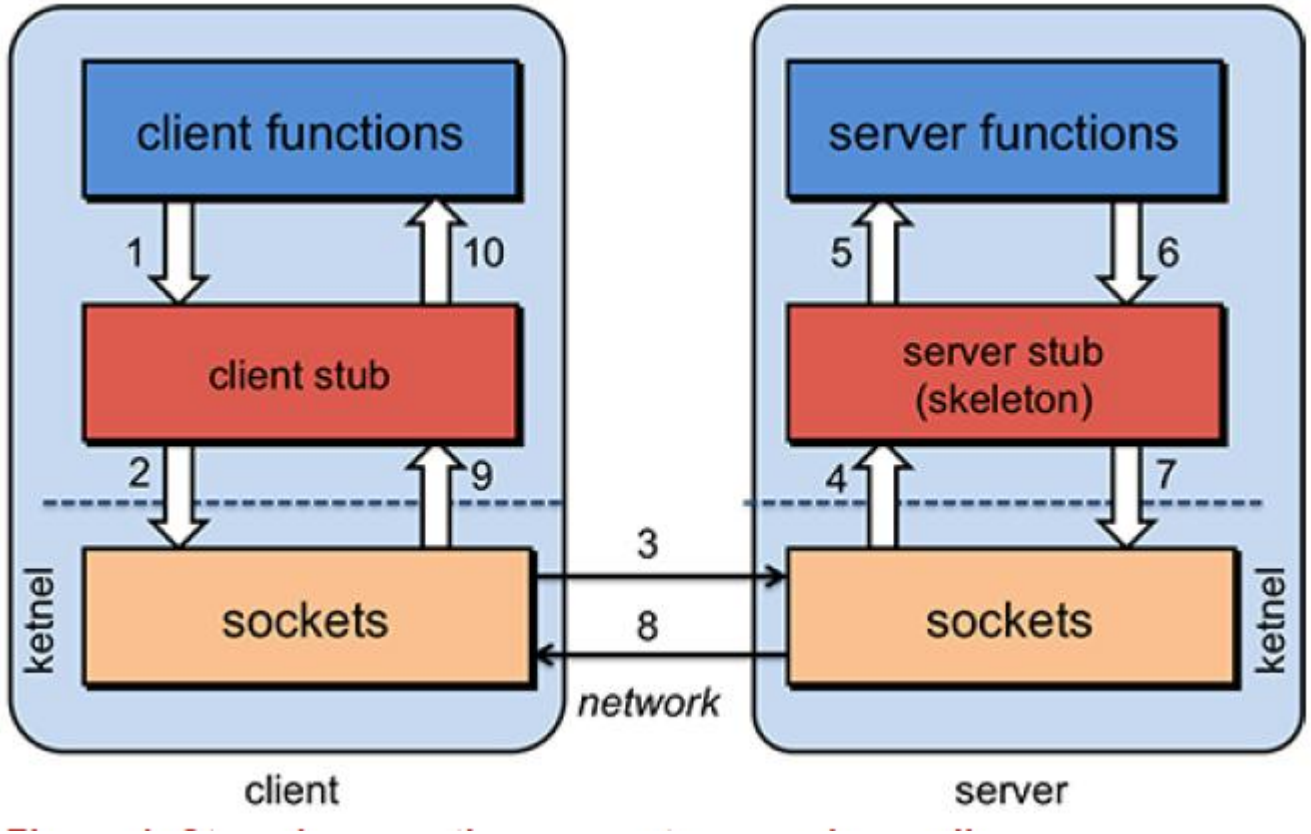
- 问题：本机如何使用套接字调用服务器上的过程A(int x)?
  - 将过程名以及参数打包（marshals），使用套接字传递
  - 服务器通过套接字接受，再解包（unmarshals）分析，完成过程调用后将结果打包，通过套接字返回
  - 本机通过套接字接受，再将结果解包

远程过程调用（Remote Procedure Calls）

- 直接调用系统函数（套接字）的缺陷：
  - 不易修改，不易测试
  - 只有过程A(int x)是业务相关的，打包、解包、传输等代码都是与业务无关的，业务的变动，以及不同的业务都需要重写这部分代码。
  - 整个系统耦合度很高，维护代码时修改一个地方会牵连到很多地方，如果修改时没有理清这些耦合关系，那么带来的后果可能会是灾难性的，特别是对于业务变化较多以及多人协作开发维护的项目，修改一个地方会引起本来已经运行稳定的模块错误，严重时会导致恶性循环。

远程过程调用（Remote Procedure Calls）

- RPCs 通过桩函数（stub functions) 对前述数据打包、传输、解包过程进行了封装。



远程过程调用（Remote Procedure Calls）

- 使用RPCs 通过网络调用其他电脑上的过程和调用本地的过程一样。
- 第一代RPCs：Sun公司 Open Network Couputing（ONC）RPCs, mid1980s
- 第二代RPCs：增加面向对象支持，分布式面向对象组件的中间件
- 第三代RPCs: Web Services

使用分布式面向对象组件的中间件(Middleware using Distributed Object-Oriented Components)

- Sun ONC RPCs 与C语言以及UNIX系统高度捆绑，不支持C++、Java等面向对象语言

- 上世纪八十年代末，面向对象编程兴起，因此也产生了调用远程对象（Remote Object）的需求。

使用分布式面向对象组件的中间件

- 在本机通过调用代理对象（proxy object) 来代替远程对象。本机调用代理对象过程与调用本地对象相同，而由代理对象完成网络读写远程调用等操作。

- 实现了对象的位置透明（location transparency）

CORBA

- Common Object Requst Broker Architecture(CORBA) 公共对象代理体系结构
  - Object Management Group（OMG）协会与1989年提出
    - 1991发布CORBA 1.0
    - 1997年发布CORBA 2.0
  - OMG是一个有300家公司参与的行业协会，几乎包含了当时所以的相关企业，UML也是协会的成果之一
  - 特点：Do everything for everybody
  - 上世纪90年代末最主流的中间件

CORBA 的兴衰

- 前沿风险技术->最受欢迎的中间件->默默无闻的小众技术（1991-2006）

- 目前只在大公司内网，以及嵌入式领域还有部分应用

CORBA 的衰落

- CORBA失败的原因：
  - 防火墙的兴起，CORBA无法穿越防火墙
  - 太复杂，以至于企业无法招聘到足够的训练有素的CORBA程序员
  - 参见《The Rise and Fall of CORBA - MICHI HENNING》， June 30, 2006, ACM Queue.

gRPC


gRPC
A high performance, open source universal RPC framework

gRPC

- gRPC 是Google开源的一款高性能的 RPC 框架
- 基于 ProtoBuf 序列化协议进行开发
- 支持多种开发语言（Golang、Python、Java、C/C++等）
- gRPC 提供了一种简单的方法来定义服务，同时客户端可以充分利用 HTTP/2 的特性，从而有助于节省带宽、降低 TCP 的连接次数、节省CPU的使用等。

其他分布式面向对象组件的中间件

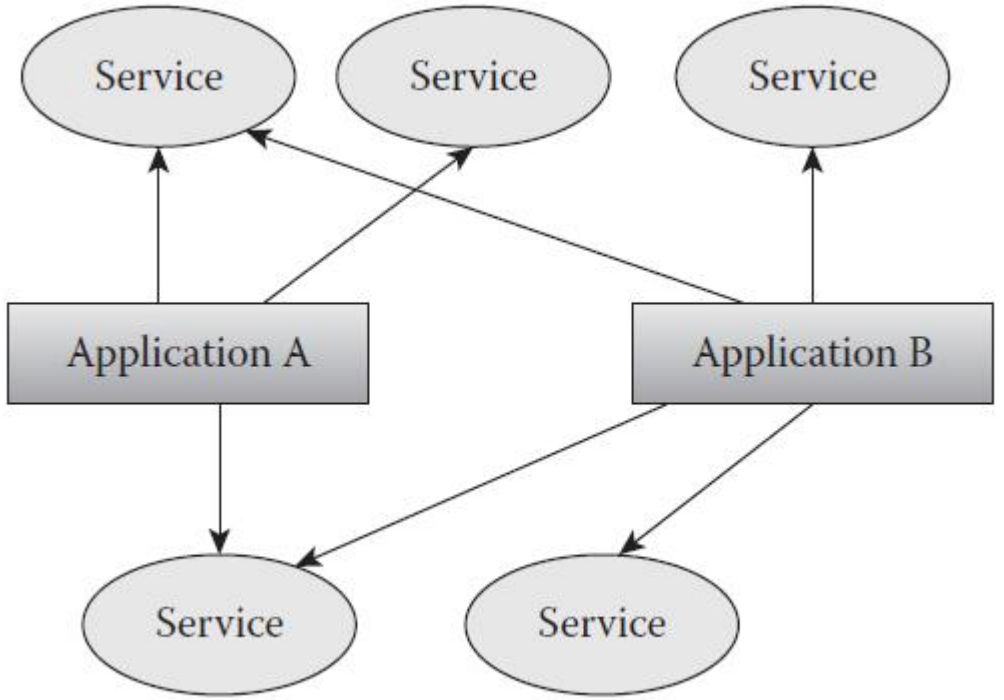
- Microsoft’s Distributed Component Model(DCOM)
- Microsoft’s .Net Remoting
- Enterprise Java Bean (EJB)

SOA & Web Service

- 面向服务的体系结构（Service-Oriented Architecture，SOA）是一种架构设计模式，设计思想
- SOA将应用程序的不同功能单元（称为服务）通过这些服务之间定义良好的接口和契约联系起来
- 服务之间的接口是采用中立的方式进行定义的，它应该独立于实现服务的硬件平台、操作系统和编程语言，使得构建在各种各样的系统中的服务可以使用一种统一和通用的方式进行交互

SOA & Web Service

- 面向服务的程序，程序由一系列服务组成；
- 面向对象的程序，程序由一系列对象组成
- 面向过程的程序，程序有一系列函数组成



什么是服务?

- 服务是一种可以逻辑表示的可重复商业行为:
  - 查看顾客信用卡余额
  - 提供天气数据

- 服务的消费者把服务看作黑盒

SOA & Web Service

- SOA并不是某一种具体的技术实现，它是一个系统架构的设计思想

- Web Service是实现SOA的一种具体的技术方案

Web的相关概念

- 网络（network）
- 互联网(internet; internetwork; internection network)
- 因特网(Internet)
- 万维网(WWW; world wide web; Web)
- HTTP(HyperText Transfer Protocol)
- HTML(HyperText Markup Language)

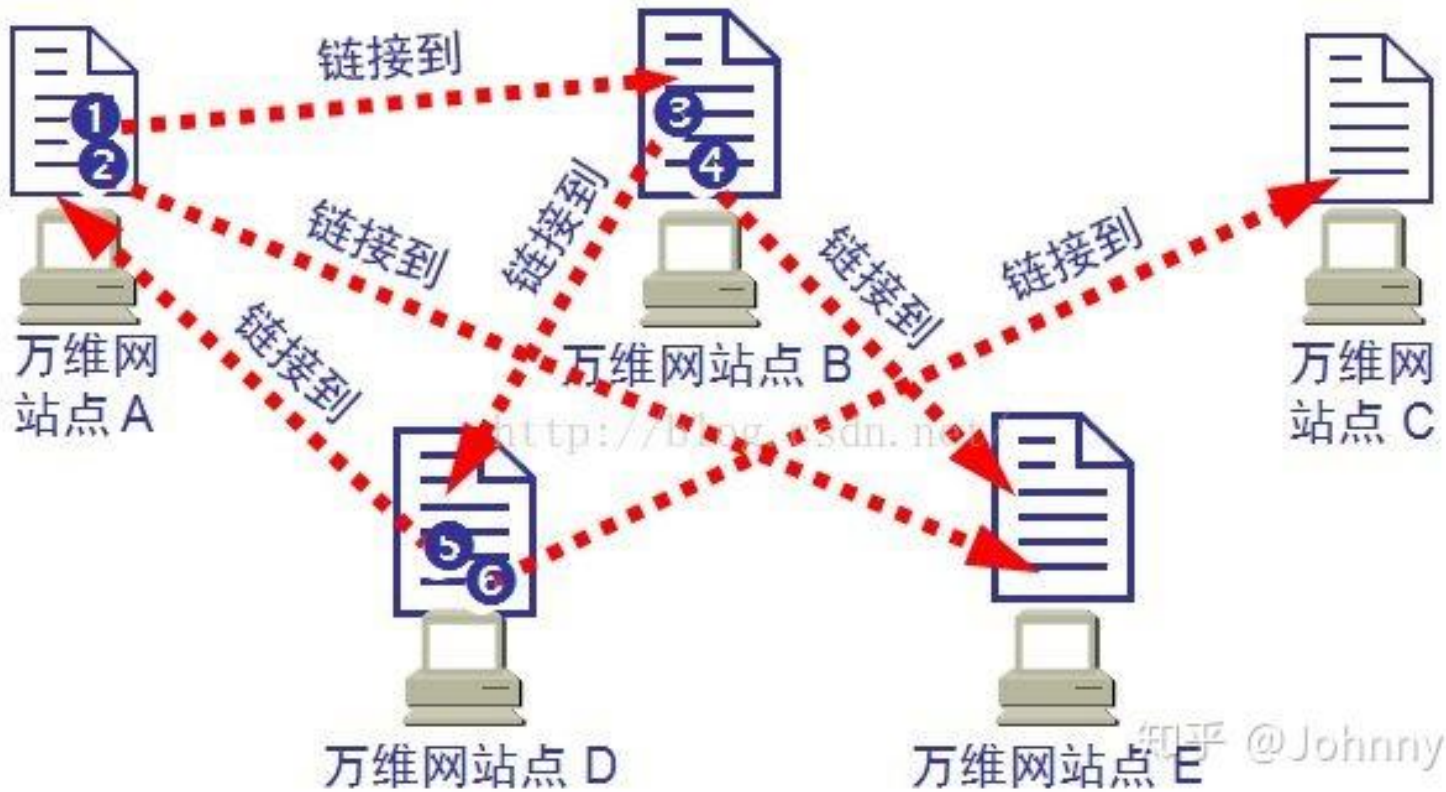
Web的相关概念

- **网络**：由若干结点（node）和连接这些结点的链路（link）组成。
- **互联网（internet）**：是**网络的网络（network of networks)**，是所有类型网络的母集。
- **因特网(Internet)**：**世界上最大的互联网网络。即 因特网概念从属于互联网概念。**习惯上，大家把连接在因特网上的计算机都成为主机。
- 网络把许多计算机连接在一起，而因特网则把许多网络联系在一起。

Web的相关概念

- **万维网（World Wide Web）：并非某种特殊的计算机网络**，万维网是一个**大规模的、联机式的信息贮藏所**，英文简称Web。
- 万维网用**链接的方法**能非常方便地**从因特网上的一个站点访问另一个站点**（超链技术），具有提供**分布式服务**的特点。
- 万维网是一个**分布式**的**超媒体系统，**是**超文本系统（hypertext）**的扩充。
- **万维网基于Client/Server 架构工作。客户程序向服务器程序发出请求，服务器程序向客户程序送回客户所要的万维网文档。**

Web的相关概念


链接到
链接到
链接到
链接到
链接到
链接到
万维网
站点 A
万维网站点 B
万维网
站点 C
万维网站点 D
万维网站点 E
1
2
3
4
5
6

Web的相关概念

- **URL**：万维网使用统一资源定位符（Uniform Resource Locator）来标志万维网上的各种文档，并使每个文档在整个因特网的范围内具有唯一的标识符URL。

小问题

- HTTP是( )
  - A、统一资源定位器
  - B、远程登录协议
  - C、文件传输协议
  - D、超文本传输协议

Web的相关概念

- **HTTP**：为解决“**用什么样的协议来实现整个因特网上的万维网文档**”这一难题，就要使万维网客户程序（以浏览器为主，但不限于浏览器）与万维网服务器程序之间的交互遵守严格的协议，这就是超文本传送协议（HyperText Transfer Protocol）。
- HTTP是处于应用层的协议，使用TCP传输层协议进行可靠的传送。
- **万维网是基于因特网的一种广泛因特网应用系统，且万维网采用的是HTTP（端口：80）/HTTPS（端口：43）的传输协议，但因特网还有其他的网络应用系统（如：FTP、SMTP等等）。**

Web的相关概念

- **HTML**：为了解决“**怎样使不同作者创作的不同风格的万维网文档，都能在因特网上的各种主机上显示出来，同时使用户清楚地知道在什么地方存在着链接**”这一问题，万维网使用**超文本标记语言（HyperText Markup Language）**
- **HTML与txt一样，仅仅是是一种文档，不同之处在于，这种文档专供于浏览器上为浏览器用户提供统一的界面呈现的统一规约。**且具备结构化的特征，这是txt所不具备的强制规定。

Web Service

- Web服务就是一个Web页面，和Web上的其他页面一样
- Web Service = Http + （XML or JSON）
  - 服务交互时通过http协议传输，传输数据遵循XML、JSON等通用数据标识语言
- 服务之间的接口是采用中立的方式进行定义的，它应该独立于实现服务的硬件平台、操作系统和编程语言，使得构建在各种各样的系统中的服务可以使用一种统一和通用的方式进行交互

Web Service的优势

- 低耦合度，方便增删功能
- 使用Http协议，数据都是通过80端口，可以穿过防火墙

云计算（Cloud Computing）

- 公有云（Public Cloud）： 用户通过Web来访问云计算公司来的数据存储以及计算资源。

- 私有云（Private Cloud）： 公司自己购买服务器来给雇员提供计算资源。

- 混合云（Hybrid Cloud）： 混合云是两个或多个云（私有云，社区云或公共云）的组合，它们保持不同的实体但绑定在一起，提供多个部署模型的好处。

云计算架构

- IaaS（Infrastructure as a service – 基础设施即服务）：用户可以在云服务提供商提供的基础设施上部署和运行任何软件，包括操作系统和应用软件。
- 用户没有权限管理和访问底层的基础设施，如服务器、交换机、硬盘等，但是有权管理操作系统、存储内容，可以安装管理应用程序，甚至是有权管理网络组件。
- 简单的说用户使用IaaS，有权管理操作系统之上的一切功能。我们常见的IaaS服务有虚拟机、虚拟网络、以及存储。

云计算架构

- PaaS（Platform as a service – 平台即服务）：PaaS给用户提供的能力是使用由云服务提供商支持的编程语言、库、服务以及开发工具来创建、开发应用程序并部署在相关的基础设施上。
- 用户无需管理底层的基础设施，包括网络、服务器，操作系统或者存储。他们只能控制部署在基础设施中操作系统上的应用程序，配置应用程序所托管的环境的可配置参数。
- 常见的PaaS服务有数据库服务、web应用以及容器服务。成熟的PaaS服务会简化开发人员，提供完备的PC端和移动端软件开发套件（SDK），拥有丰富的开发环境（Inteli、Eclipse、VS等），完全可托管的数据库服务，可配置式的应用程序构建，支持多语言的开发，面向应用市场。

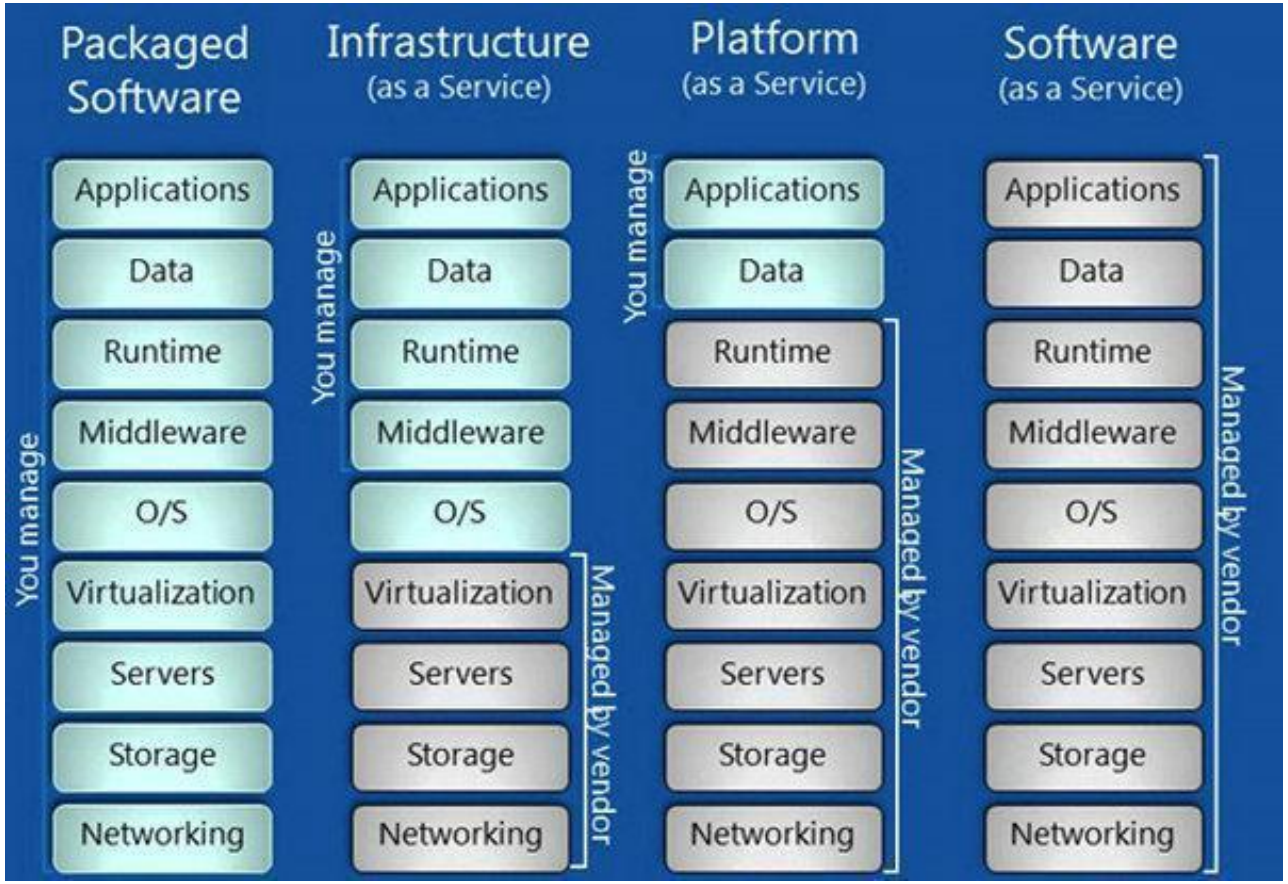
云计算架构

- SaaS（Software as a Service – 软件即服务）：SaaS给用户提供的能力是使用在云基础架构上运行的云服务提供商的应用程序。可以通过轻量的客户端接口（诸如web浏览器（例如，基于web的电子邮件））或程序接口从各种客户端设备访问应用程序。
- 用户无需管理或控制底层云基础架构，包括网络，服务器，操作系统，存储甚至单独的应用程序功能，可能的例外是有限的用户特定应用程序配置设置。
- 类似的服务有：各类的网盘(Dropbox、百度网盘等)，JIRA，GitLab等服务。而这些应用的提供者不仅仅是云服务提供商，还有众多的第三方提供商（ISV: independent software provider）。

云计算架构


IaaS
PaaS
SaaS
Amazon EC2
HEROKU
Gmail
阿里云
salesforce
PayPal
Azure
CLOUD FOUNDRY

云计算（Cloud Computing）



云计算的优势

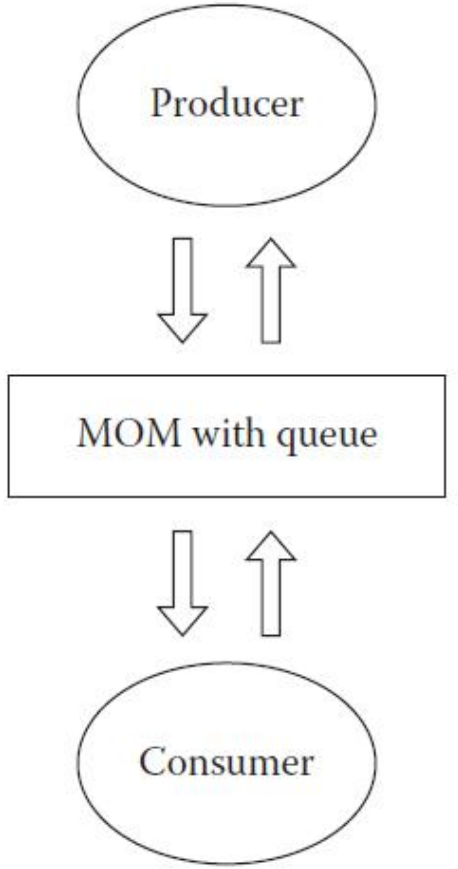
- 敏捷性：快速部署技术服务，并且从构思到实施的速度比以前快了几个数量级
- 弹性：可以根据业务需求的变化立即扩展或缩减计算资源
- 节省成本

消息中间件（Message-Oriented MiddleWare）

- 消息中间件（MOM）是一类在客户和服务器之间添加中间人的中间件。

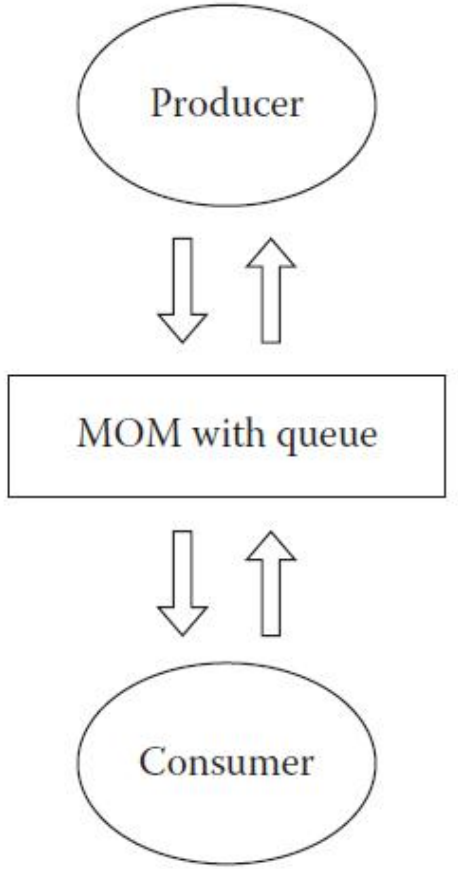
消息中间件（Message-Oriented MiddleWare）

- 使得客户和服务器只需要和中间人传递消息，而不需要互相传递消息。
- 中间人通过一个消息队列来存储消息生成者和消息接收者之间的消息。
- MOM使得客户和服务器之间可以异步操作

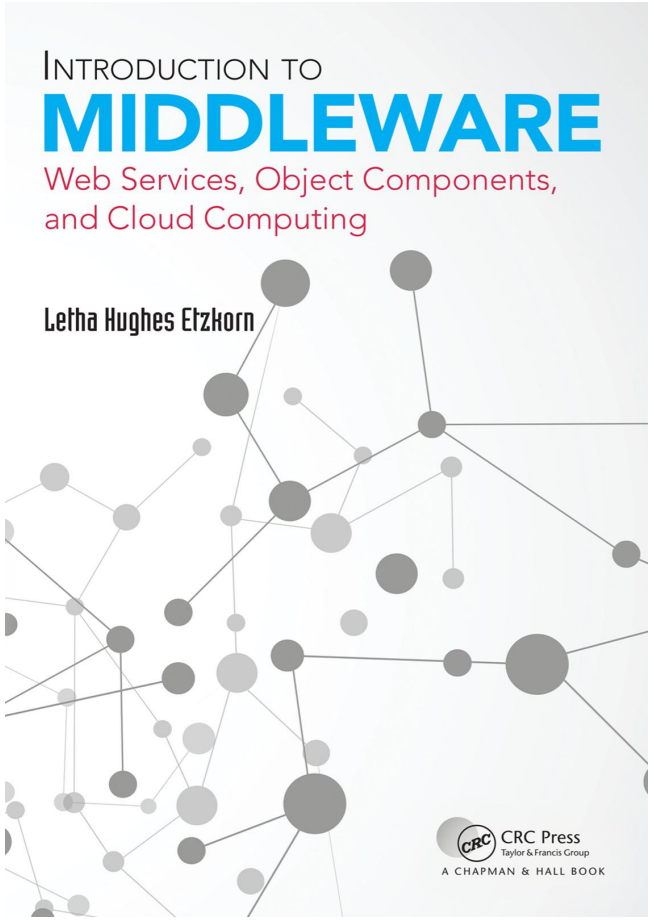


消息中间件（Message-Oriented MiddleWare）

- 客户发送消息到MOM
- MOM将消息加入队列
- 服务器空闲，可以处理消息，MOM将消息发往服务器
- 服务器完成计算，再把结果传回消息中间件
- 消息中间件再传回客户



教材



- 《INTRODUCTION TO MIDDLEWARE: Web Services, Object Components, and Cloud Computing》, Letha Hughes Etzkorn, CRC Press, 2017.

《Introduction to Middleware》

- Chapter 1 Introduction
- Chapter 8 Middleware Using Distributed Object-Oriented Components
- Chapter 9 Web Services Architectures
- Chapter 10 Non-RESTful Web Services
- Chapter 11 RESTful Web Services
- Chapter 13 Introduction to the Cloud and Introduction to the OpenStack Cloud
- Chapter 14 Introduction to Amazon Web Services and Introduction to the CloudStack Cloud

《Introduction to Middleware》

- Chapter 15 Message-Oriented MiddleWare
- Chapter 16 Introduction to Comparing MiddleWare

学习材料

- 《中间件技术原理与应用》，张云勇等，清华大学出版社，2004年第一版。

《中间件技术原理与应用》

- 第1章 中间件产生背景及分布式计算环境
- 第2章 面向对象中间件ODP
- 第3章 COM相关技术
- 第4章 J2EE技术
- 第5章 CORBA初步
- 第6章 CORBA服务
- 第7章 中间件中的事务处理
- 第8章 CORBA高级技术
- 第9章 无线、移动中间件
- 第10章 反射中间件
- 第11章 网络即插即用中间件
- 第12童Web服务
- 第13章 其他中间件技术
- 第14章 中间件的典型应用

开发平台

- MicroSoft .Net C#语言（2014年开源）
- Java

其他学习资料

- 《大型网站系统与Java中间件实践》，曾宪杰，电子工业出版社，2014年第一版。
- The Annual ACM/IFIP Middleware Conference
- IEEE Internet of Things Journal
- IEEE/ACM Transactions on Networking (TON)
- ACM/IEEE Conference on Internet of Things Design and Implementation
- Google, Baidu, Zhihu, 知网，……

课程安排

周次	学时数	教 学 主 要 内 容	教学方法
1	2	第一章 物联网中间件概述	主讲 2 学时
2/3/5	6	第二章 计算机网络基础以及套接字	主讲 6 学时
5/6/7/8	8	第三章 基于分布式组件的中间件	主讲 8 学时
9/10	2	第四章 基于 Web 服务的中间件概述	主讲 2 学时
10/11/12/13	8	第五章 Non-RESTful Web 服务	主讲 8 学时
14/15/16	6	第六章 RESTful Web 服务	主讲 6 学时
17	2	第七章 中间件比较	主讲 2 学时
18	2	复习课程内容	主讲 2 学时

考核方式

- 平时成绩：30%
  - 四次小作业
- 期末考试：70%

复习/答疑

- 什么是中间件?
- 为什么要用中间件？什么情况下可以不用中间件，直接使用网络通信?
- 套接字是中间件吗?
- Web服务的特点?