# 巴比伦塔的故事

*……他们说，来吧！我们要建造一座城，和一座塔，塔顶通天。*
*为要传扬我们的名，免得我们分散各地。*
*——《圣经·创世纪》*

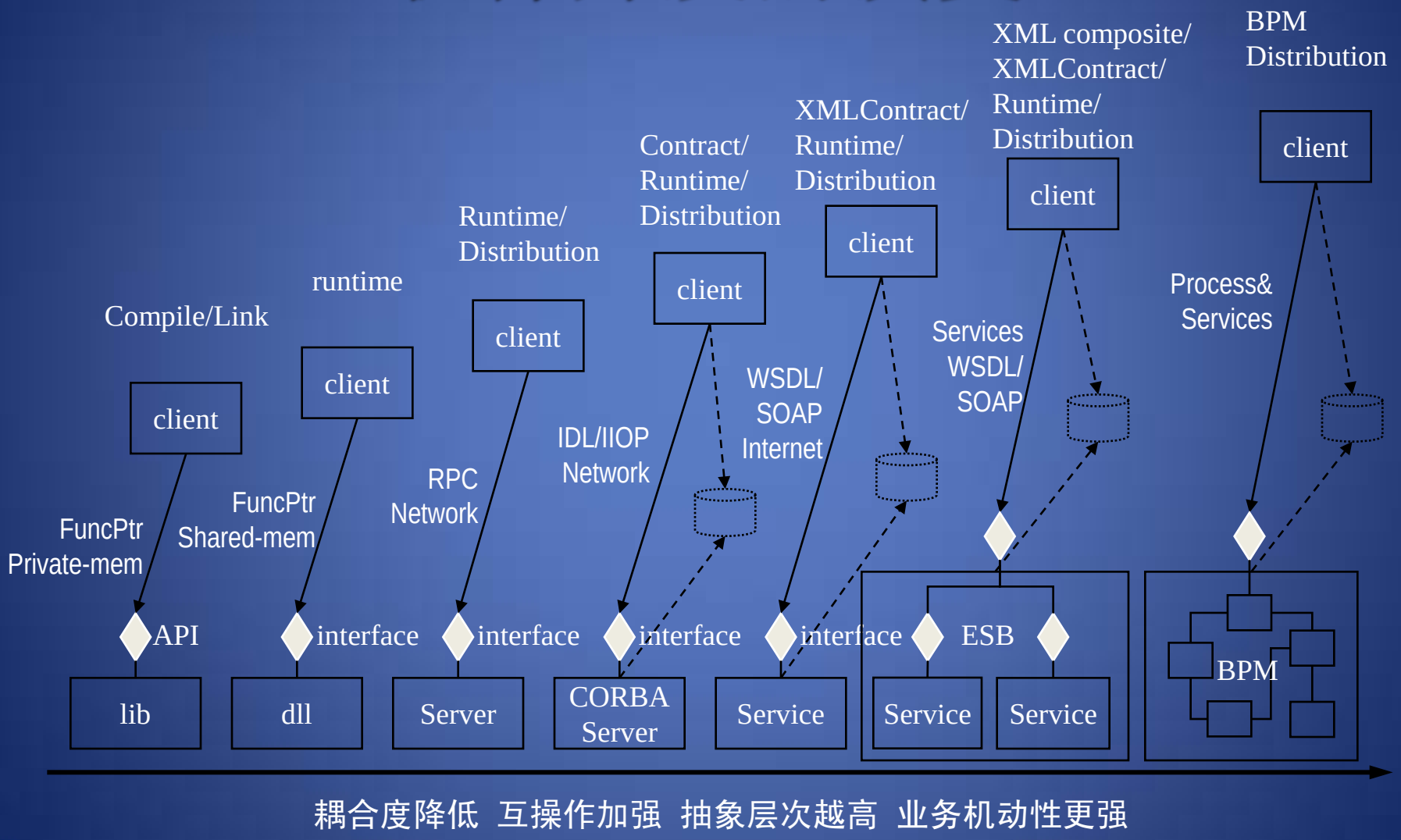
人到西内阿定居下来，说：让我们用砖石建一个城、一个塔，塔尖摩天，给我们自己一个名字，否则我们就要分散各方了。上帝看到人造的城和塔说：他们是一群人，讲的是一种语言，他们要做什么，就不会停止的。让我把他们的语言搞乱，互相不通。于是，上帝就把他们驱散到各方，他们因此必须停止建城和塔。

# 第 三 讲

# 软件互操作

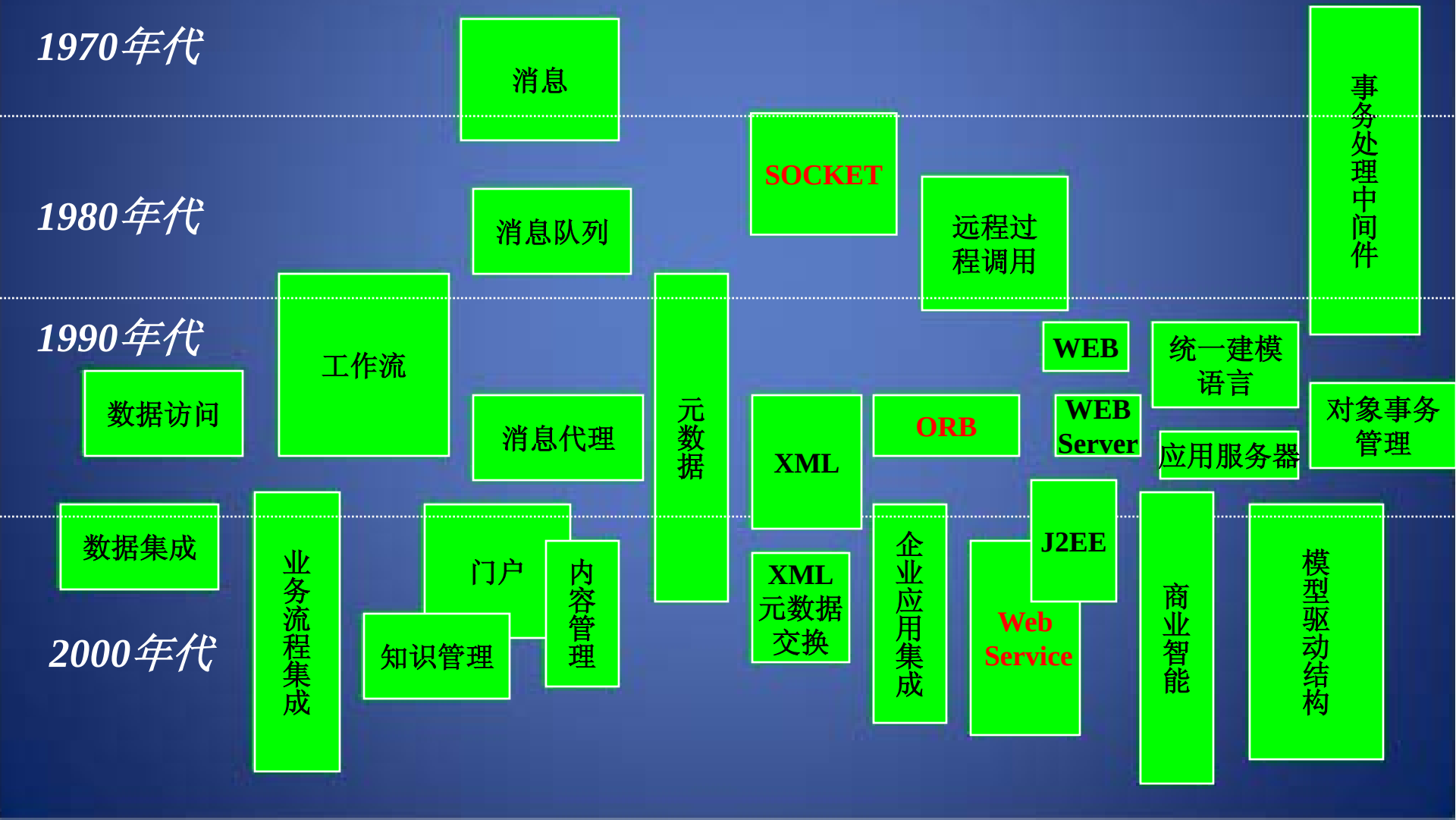
周明辉
zhmh@pku.edu.cn
北京大学软件研究所

# 软件开发的演变




耦合度降低 互操作加强 抽象层次越高 业务机动性更强

# 互操作(中间件)技术的演变



# 内　容

一、网络协议和Socket编程

二、互操作框架和中间件

三、面向服务体系结构SOA

1、TCP/IP

2、基于Socket的编程

# 一、网络协议和SOCKET编程

# 1、TCP/IP

（1）ISO/OSI 参考模型 与TCP/IP的对照

| ISO/OSI 参考模型 | TCP/IP |
| --- | --- |
| Application | 应用层：<br>telnet、ftp、smtp、snmp、<br>dns、http、nntp |
| Presentation | |
| Session | |
| Transport | TCP, UDP |
| Network | 因特网层： IP |
| Data link | 主机与网络的连接：<br>ethernet、token-ring |
| Physical | |

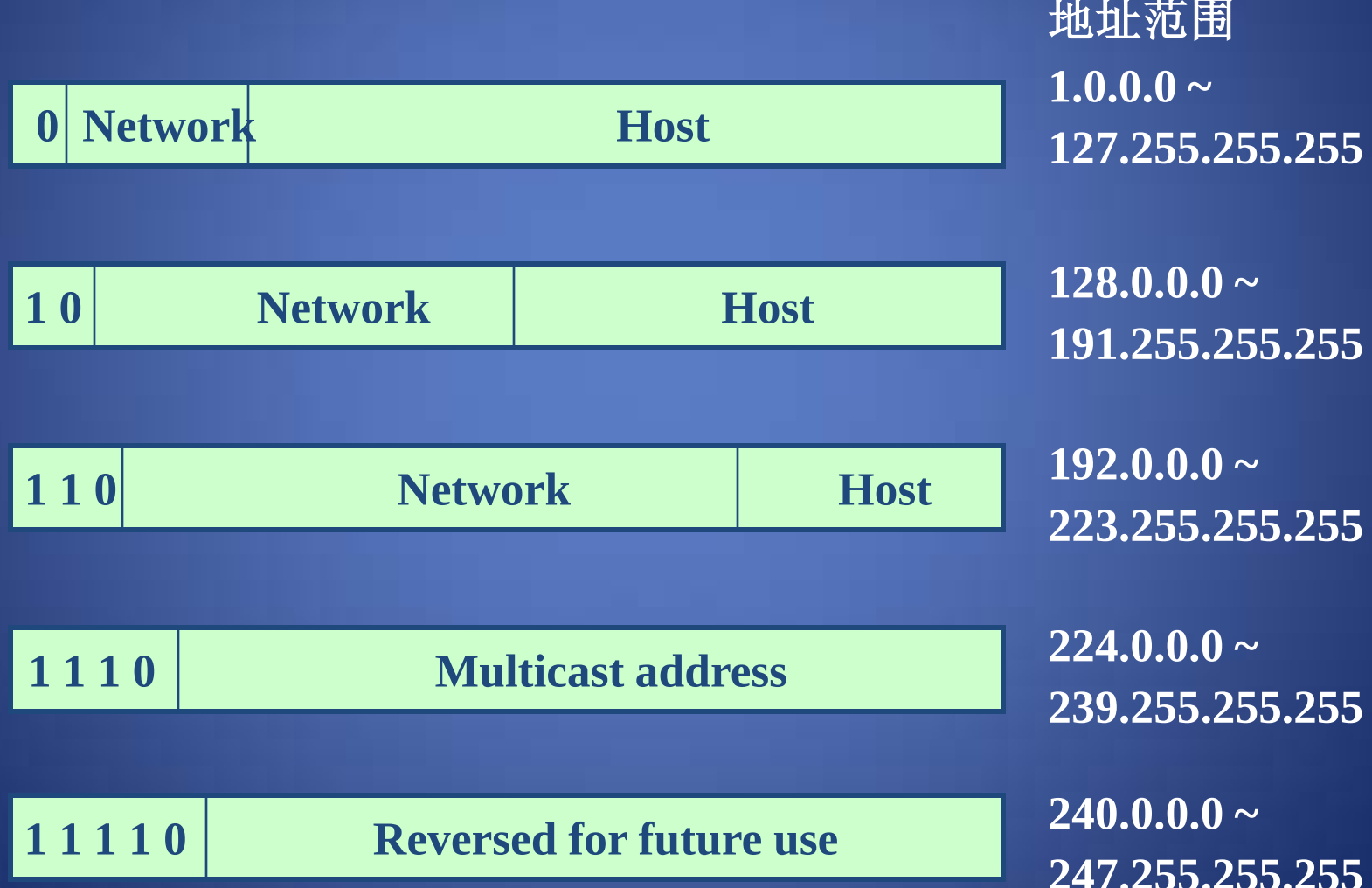
# 网络协议三要素

- **语法**，用来规定信息格式；数据及控制信息的格式、编码及信号电平等。
- **语义**，用来说明通信双方应当怎么做；用于协调与差错处理的控制信息。
- 时序，定义了何时进行通信，先讲什么，后讲什么，讲话的速度等。比如是采用同步传输还是异步传输。
-

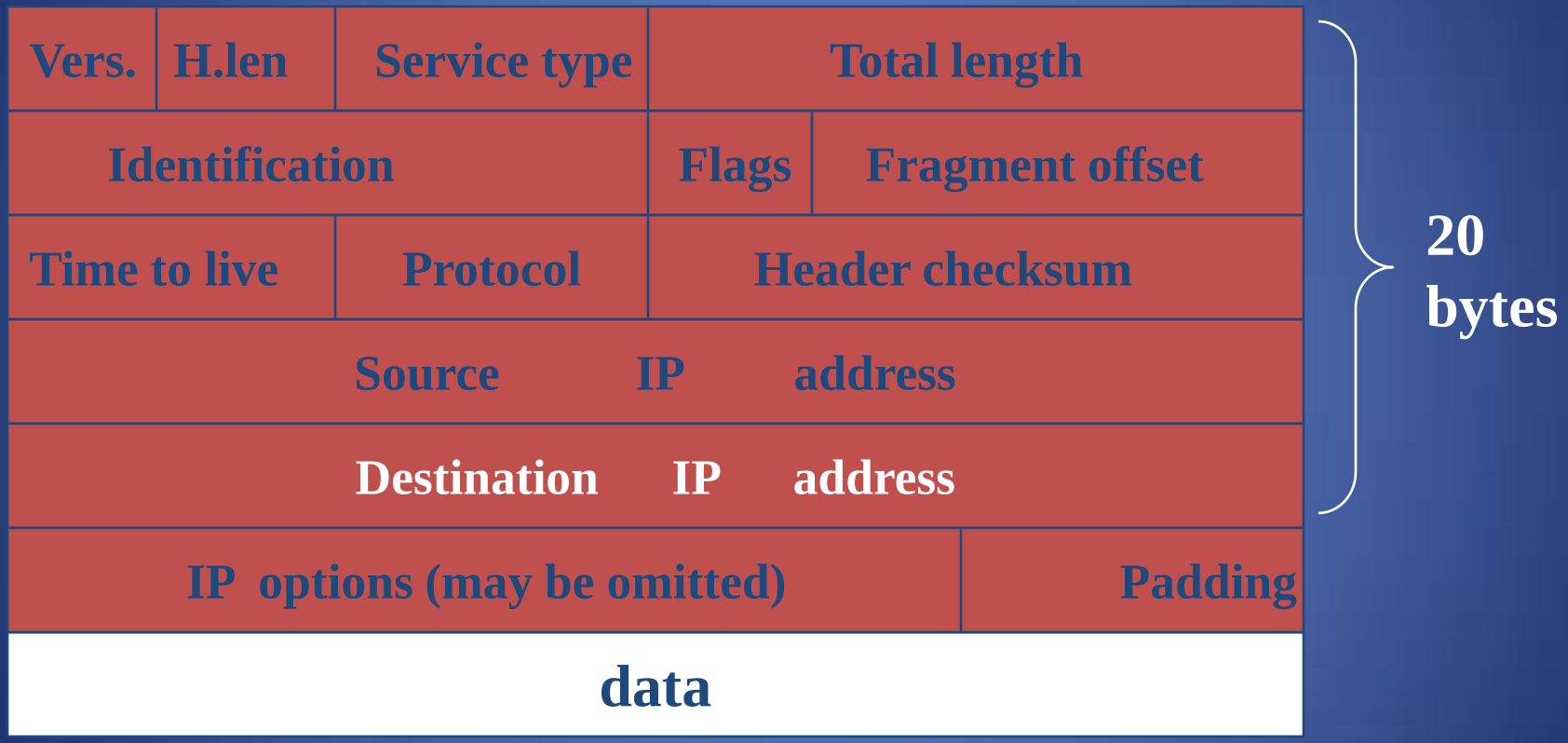
# （2）IP：Internet Protocol

- 实现两个不同IP地址的计算机的通信，它们可能位于同一网络或互连的两个不同网络中
- 具体工作包括形成IP数据报和寻址。如果目的主机不在本网，就要经路由器予以转发直到目的主机
- IP协议提供的是不可靠的数据传输功能，并且没有提供流量控制和差错控制功能

# IP地址


类
地址范围
A
0
Network
Host
1.0.0.0 ~
127.255.255.255
B
1 0
Network
Host
128.0.0.0 ~
191.255.255.255
C
1 1 0
Network
Host
192.0.0.0 ~
223.255.255.255
D
1 1 1 0
Multicast address
224.0.0.0 ~
239.255.255.255
E
1 1 1 1 0
Reversed for future use
240.0.0.0 ~
247.255.255.255

## IP 数据报



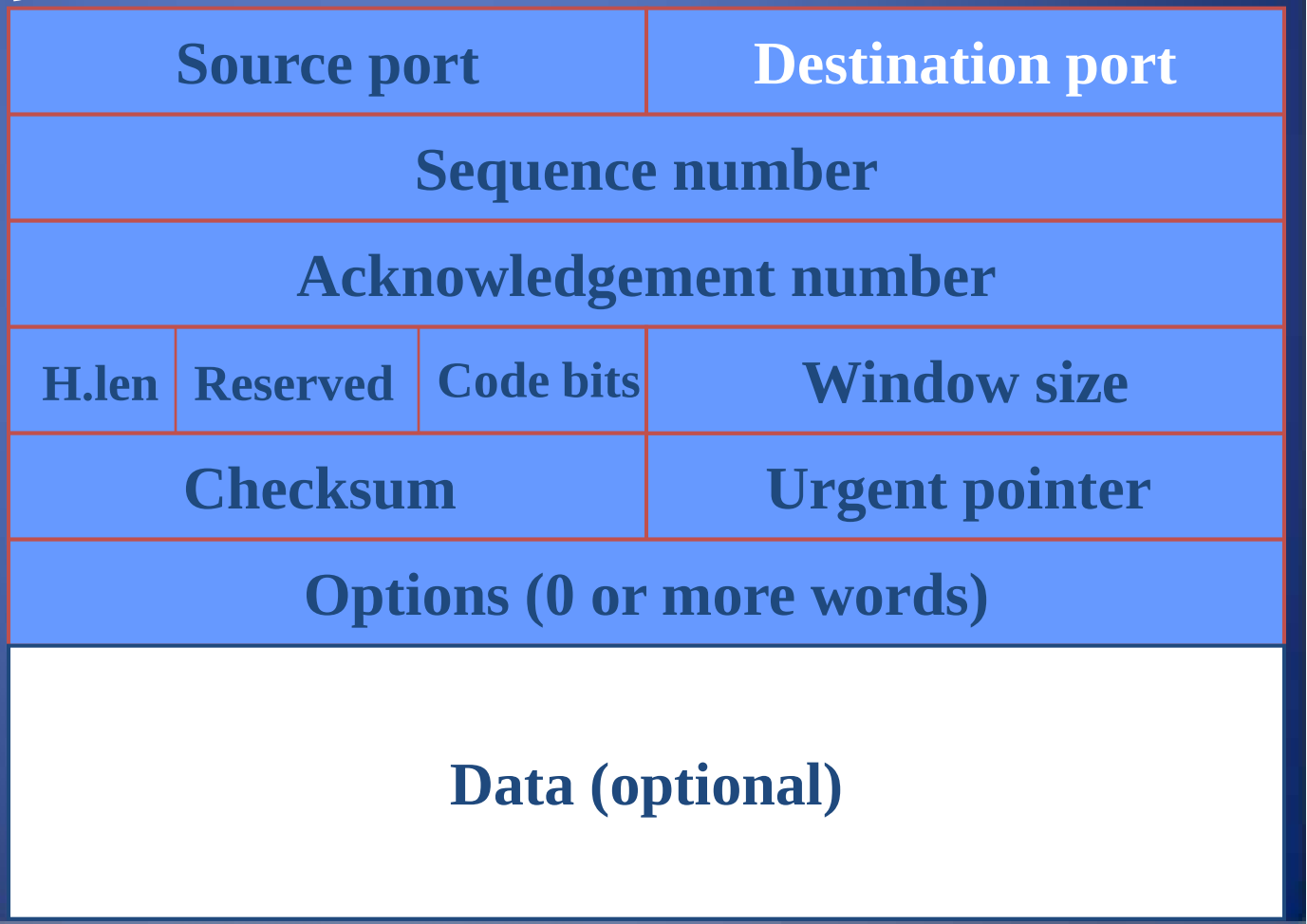

Service type

| Precedence | D | T | R | unused |
|---|---|---|---|---|

# （3）传输层协议

- 传输层协议提供应用程序间(端到端)的可靠通信
- 包括传输控制协议TCP和用户数据报协议UDP

- **TCP**， Transmission Control Protocol，用于一次传输大批数据的情形(如文件传输、远程登录等)，并适用于要求得到响应的应用服务
- 可靠但缓慢
- **TCP头格式**



# UDP： User Datagram Protocol

- 提供了无连接通信，且不对传送数据报进行可靠保证，适合于一次传输少量数据(如数据库查询)的场合，其可靠性由其上层应用程序提供
- 不可靠但快速的协议
- 消息长度固定，消息在接受方排队
- UNIX rwho 命令基于UDP

1、TCP/IP

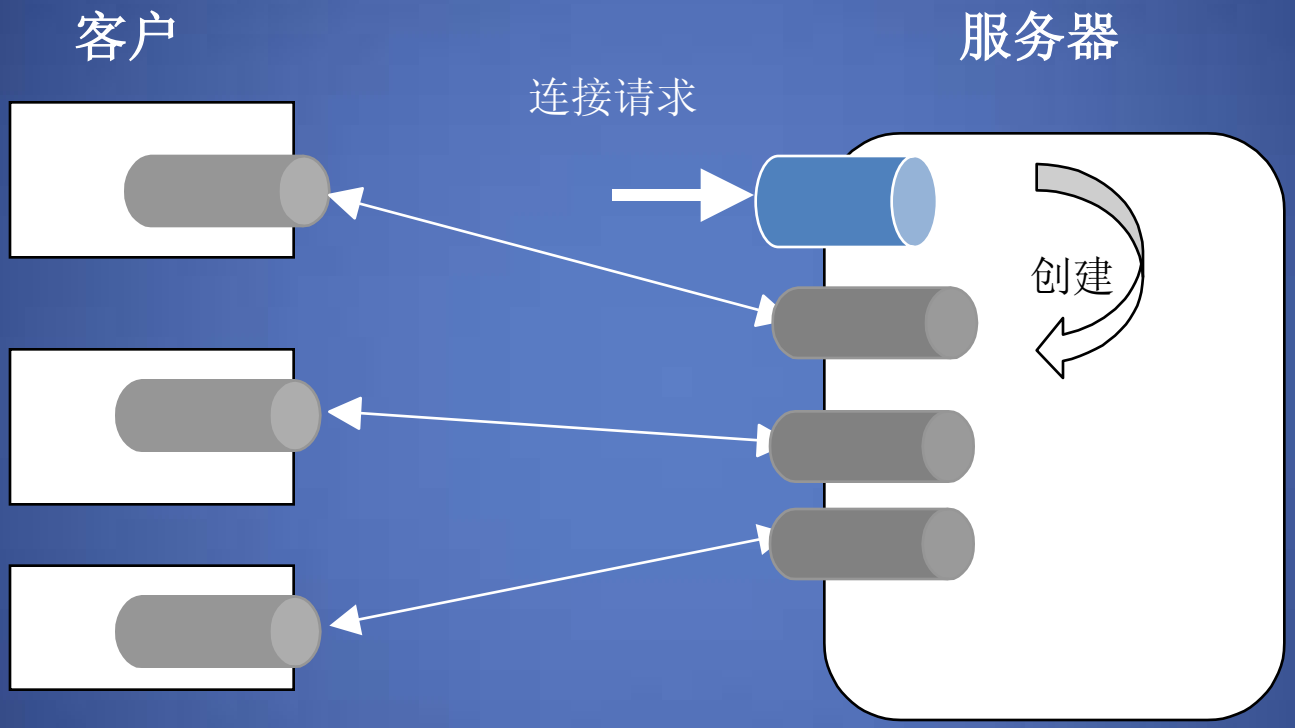
2、基于Socket的编程

# 一、网络协议和SOCKET编程

- Socket接口是TCP/IP网络的API，Socket接口定义了许多函数或例程，程序员可以用它们来开发 TCP/IP网络上的应用程序 --百度知道

- **Berkeley套接字**应用程序接口（API）包括了一个用C语言写成的应用程序开发库，主要用于实现进程间通讯

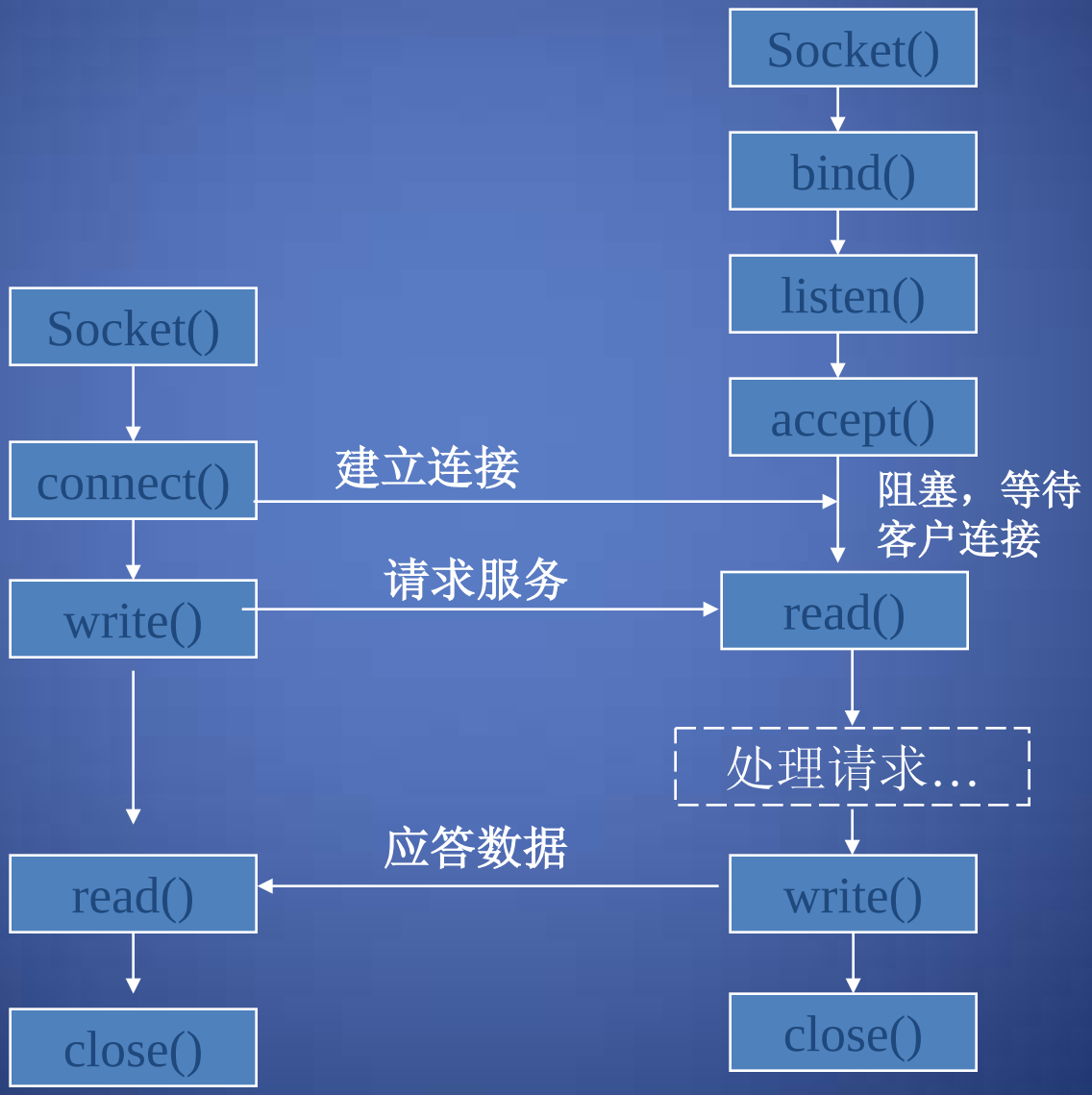
--维基百科



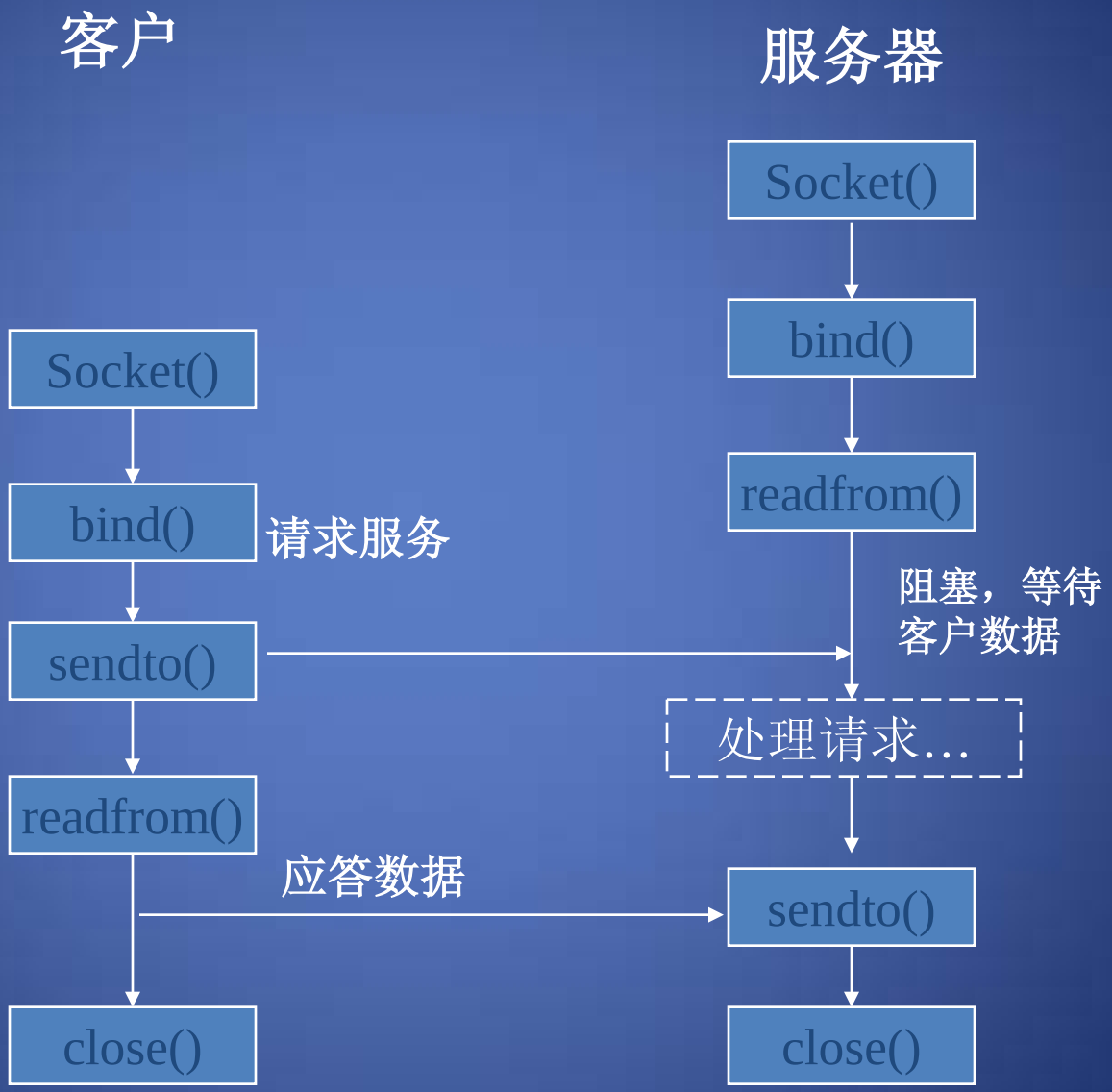

等待连接套接字
通常对应于一个固定端口

通信套接字
通常通过随机分配得到端口

# 面向连接：


客户
服务器
Socket()
bind()
listen()
accept()
Socket()
connect()
建立连接
阻塞，等待
客户连接
write()
请求服务
read()
处理请求…
应答数据
read()
write()
close()
close()

面向非连接：


客户
服务器
Socket()
bind()
readfrom()
阻塞，等待
客户数据
处理请求…
sendto()
close()
Socket()
bind()
请求服务
sendto()
readfrom()
应答数据
close()

# 基于Socket编程的“不足”

- 套接字的出现
  - 促进了软件从单机环境向网络环境的发展
  - 扩展了软件的应用范围
- 人们很快不满足于直接基于套接字的开发过程
  - 基于套接字的开发方式较为繁琐
  - 这种软件的排错十分困难
    - 特别是当服务器端需要根据用户的不同请求内容区分不同的处理过程时

根本原因在于基于套接字的交互层次较低
不同软件之间需要约定专门的消息格式、数据类型等

# 内　容

一、网络协议和Socket编程

二、互操作框架和中间件

三、面向服务体系结构SOA

1、软件互操作的基本框架
2、CORBA
3、中间件类型

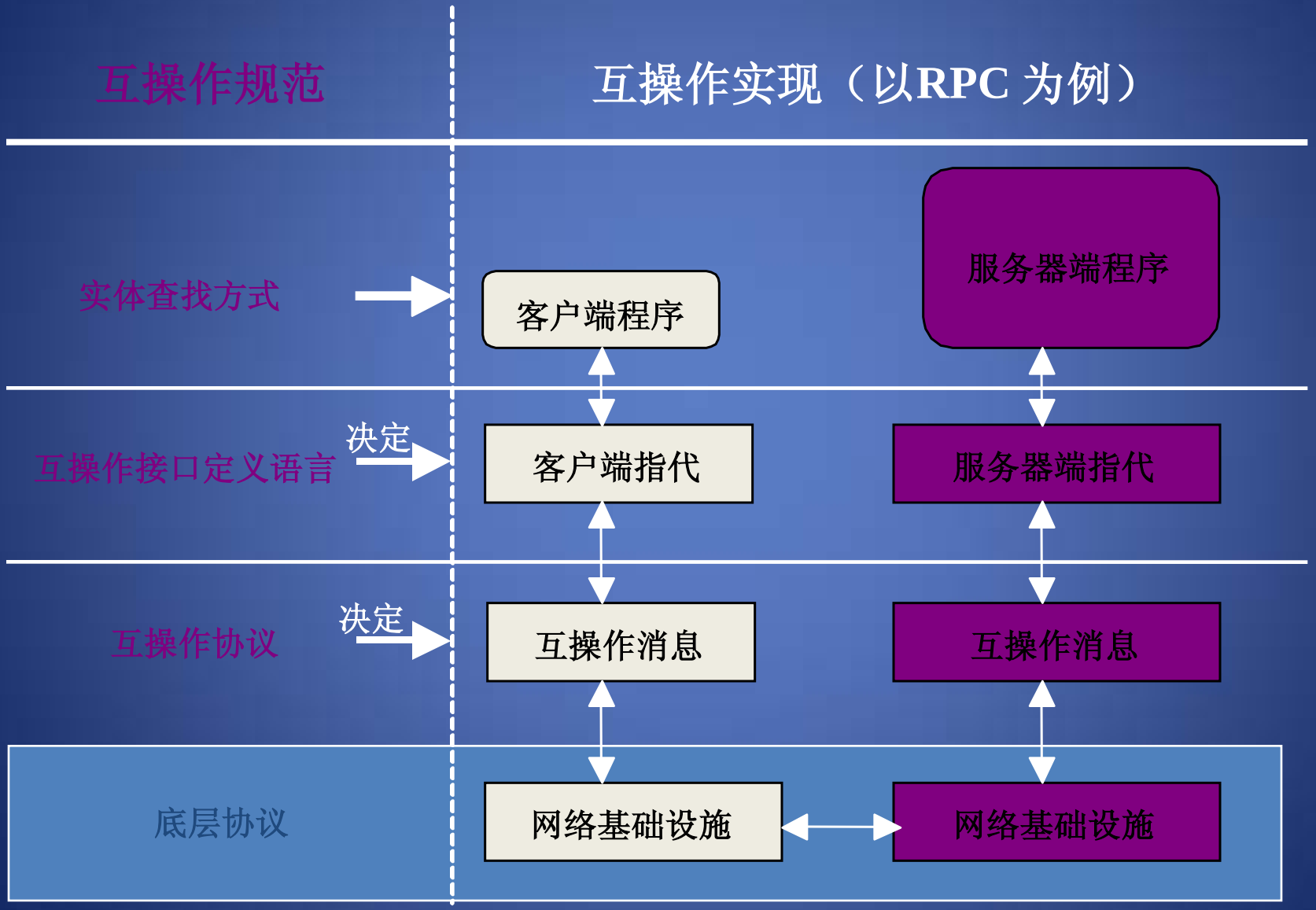
# 二、互操作框架和中间件

# 1、软件互操作的基本架构

为支持

应用层的某一实体使用另一实体

而制定的一套技术规范


互操作规范
互操作实现（以RPC 为例）
实体查找方式
互操作接口定义语言
决定
互操作协议
决定
底层协议
客户端程序
服务器端程序
客户端指代
服务器端指代
互操作消息
互操作消息
网络基础设施
网络基础设施

## (1) 底层协议

底层协议是互操作协议所依赖的底层通信机制

目前最通用的协议　是前面讲过的　TCP/IP协议
也可以是 ATM等其它协议

# (1) 底层协议

- 底层协议是互操作协议所依赖的底层通信机制

- 目前最通用的协议是前面讲过的TCP/IP协议，也可以是 ATM等其它协议

# (2) 互操作协议

- 消息中关于字节序、数据表示等问题的解决方法体现了通信双方之间关于消息的 **数据格式** 和 **消息的类型** 等的约定，类似的约定还有 服务器的管理 等等，所有这些约定 共同构成了互操作的高层协议
- 互操作协议一般独立于底层协议，即互操作协议可以向不同的底层协议进行映射，从而由不同的底层协议进行支持
- 互操作协议一般都比较烦琐，其实现由，例如应用基础设施提供商，承担应用系统的开发者在开发具体系统时不必关心协议的实现问题

# 互操作协议要素

所谓互操作协议要素是指

设计一个互操作协议时必须考虑的主要问题

这些问题主要包括：

- 数据表示
- 消息格式
- 引用表示　等等

其它需要考虑的问题包括：

- 如何将协议映射到底层协议之上
- 如何管理连接过程
- 如何提高协议的效率　等等

## 数据表示

在单机环境内，数据表示属于硬件层、语言层
而在网络环境下，调用参数必须经由网络进行传输
这意味着这些数据将变成一种字节流的形式
以便于参数（数据）在网络上传输
就产生了如何在网络传输上表示程序中的数据问题

数据表示是一种传输语法
描述各种数据类型在传输线路上
以字节流的形式表示出来的格式
**ONC-RPC中使用的格式为外部数据表示**
**（XDR：eXternal Data Representation）**
**而CORBA中使用的格式为公共数据表示**
**（CDR：Common Data Representation）**

# 消息格式

底层协议 主要解决通信的 可行性
以及部分 可靠性 等问题

高层协议不同于底层协议的一个明显特征在于
高层协议带有一定的语义信息

几乎每种高层协议 都对消息进行分类
定义了多种不同类型的消息格式

对于互操作协议而言

一次调用通常至少包含请求与应答两种消息

消息的种类较为类似

但消息格式差异较大

## 向底层协议的映射

互操作协议必须映射到底层协议上

方能得到实现

目前的互操作协议都定义协议一到多种向传输层的映射

例如 **GIOP定义了向TCP/IP的映射：IIOP**

**SOAP主要定义了向HTTP的映射**

# (3) 互操作接口定义语言

- 客户指代主要完成上层代码（客户程序、服务器程序）与底层代码（RPC API等）之间的“映射”
  - 其参数传递、编排、服务器定位等功能完全与应用系统具体的业务逻辑实现细节无关
  - 因此只要系统明确定义了服务器的接口，即可以产生与该接口对应的指代
- 接口定义语言（Interface Definition Language）描述了 客户与服务器之间的接口

# 含义解释

模块通常由接口和实现两部分组成

模块的接口部分　刻画了各个模块是如何耦合的
其他模块的设计者和使用者需要知道

模块的实现部分　是各个模块的内部事务
其他模块的设计者和使用者不需要知道

应用编程接口（API）
对象接口
构件接口
抽象？实在？

包含什么具体内容？（功能性、约束性）
利用什么形式描述？（接口定义语言）

# 功能性（Functional）

单机环境下的不同软件模块之间
主要定义模块的功能性（**Functional**）内容
类似于一个函数的映射过程

接口的功能性定义是对接口中各个操作调用方式的描述

操作　是由操作符标识的实体
指明了一个不可再分的服务原语
请求一个操作的动作被称为调用一个操作

对一个操作功能的描述由输入、输出两部分组成
也被称为一个基调（**signature**）
用于描述操作的输入、输出参数名称及类型

# 约束性（Constraint）

网络环境下的不同软件模块的合作需要考虑的因素
不仅仅包含功能方面
还涉及分布性、可靠性、安全性等方面的因素

网络环境下的接口　除需要定义模块的功能性内容外
还需要定义模块的约束性内容

接口的约束性定义是指对功能以外特征的描述
简单的包括：例外处理、执行语义等
复杂一些的包括：
（1）行为特征:用于描述操作的输出
通过对操作增加前置与后置条件而实现
（2）同步特征:用于描述分布性与并发性

# (4) 实体查找方式

- 实体查找方式主要是指服务器的定位

  根据自己掌握的信息

  客户如何才能查找到具体服务器

- 这实际上涉及　服务器端服务信息的发布，服务信息的管理等问题

# 现有互操作架构比较

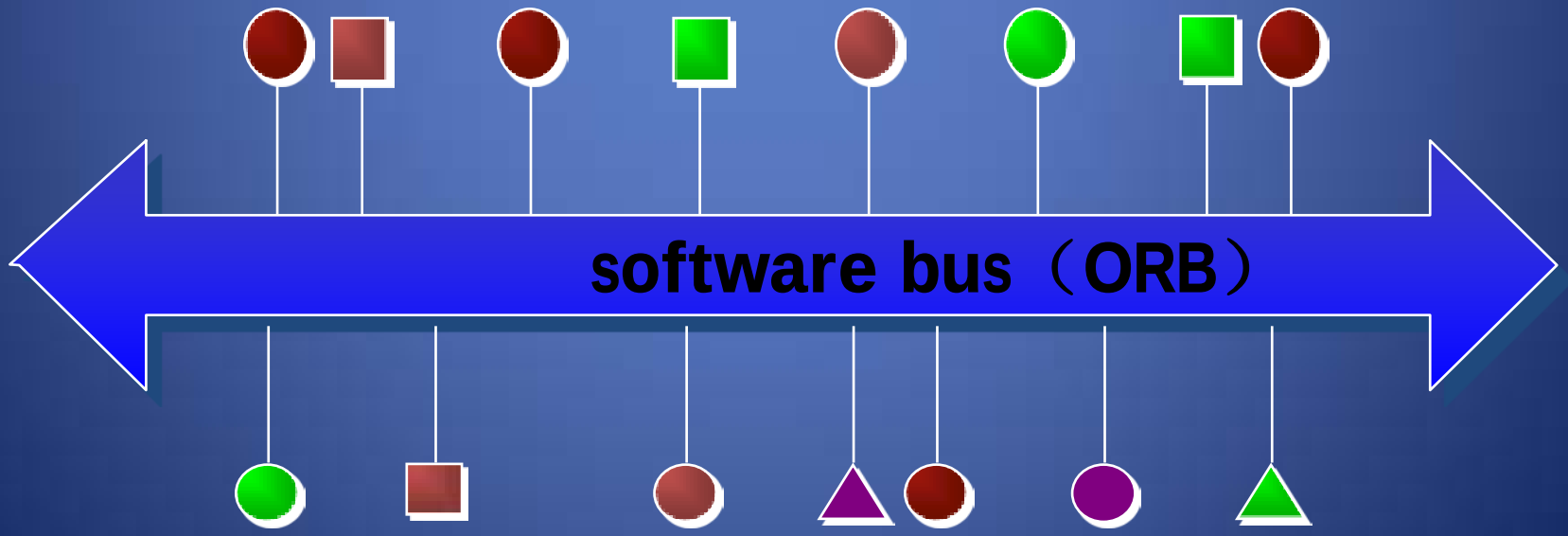
| 互操作体系名称 | 互操作协议 | 接口定义语言 | 实体查找 |
|---|---|---|---|
| RPC | RPC | IDL | RPC注册 |
| DCOM | ORPC | MIDL | 系统注册 |
| CORBA | GIOP（IIOP） | IDL | 命名服务 |
| EJB | JRMP | Java Interface | JNDI |
| Web Service | SOAP | WSDL | UDDI |

1、软件互操作的基本框架
2、CORBA
3、中间件类型

# 二、互操作框架和中间件

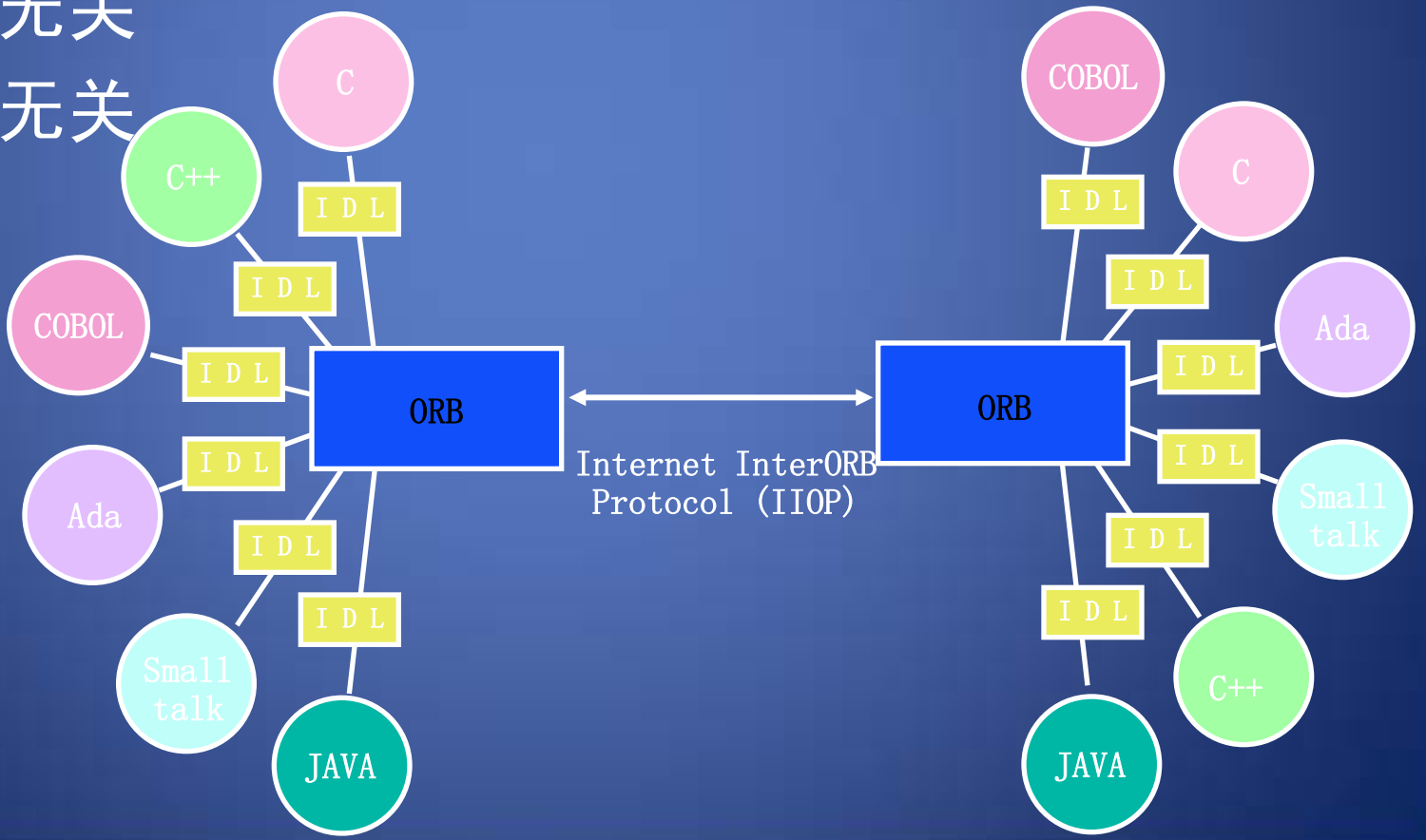
# CORBA：软总线+软构件

- OMA：Object Management Architecture
- 试图成为普遍接受的规范

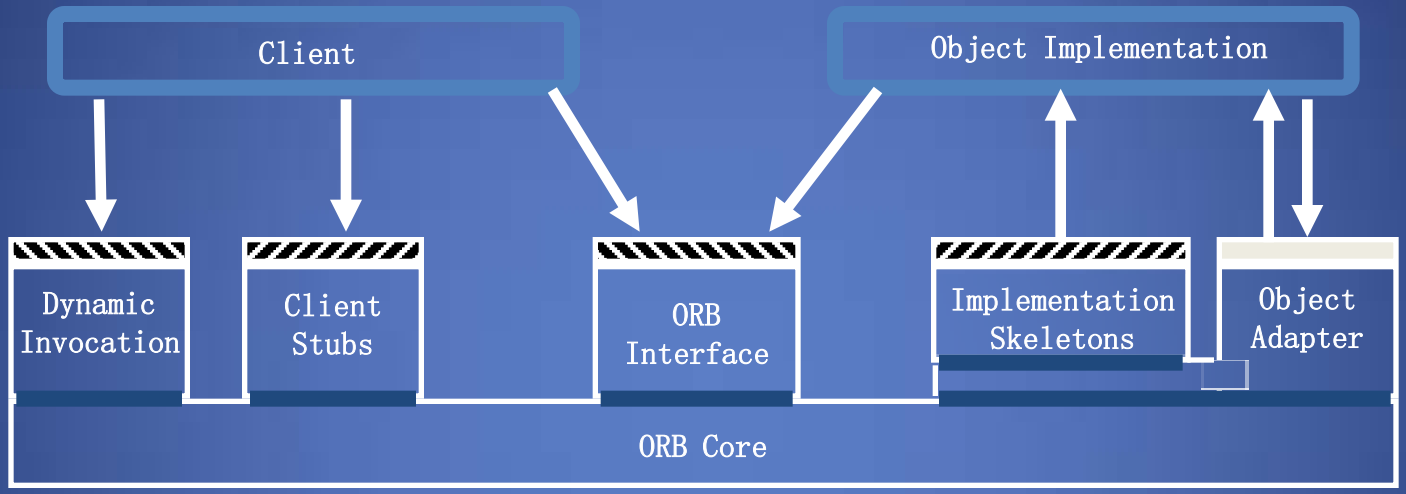


# CORBA

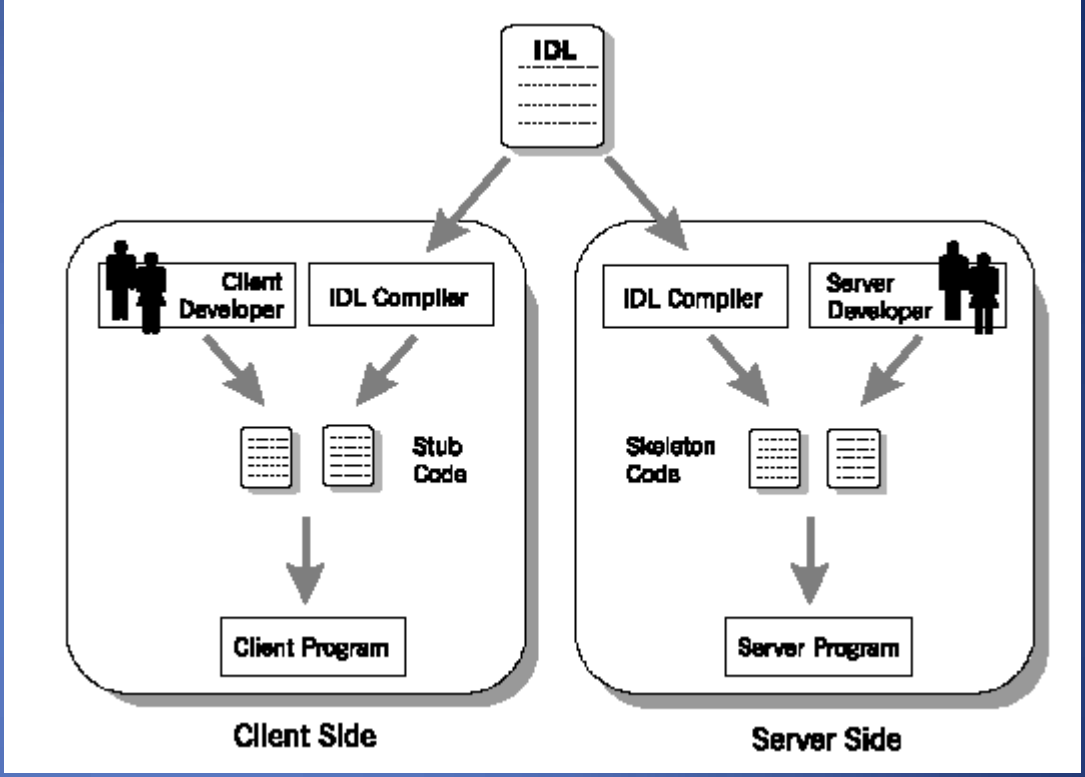
- 互操作interoperability
  - 语言无关
  - 平台无关
  - 厂商无关

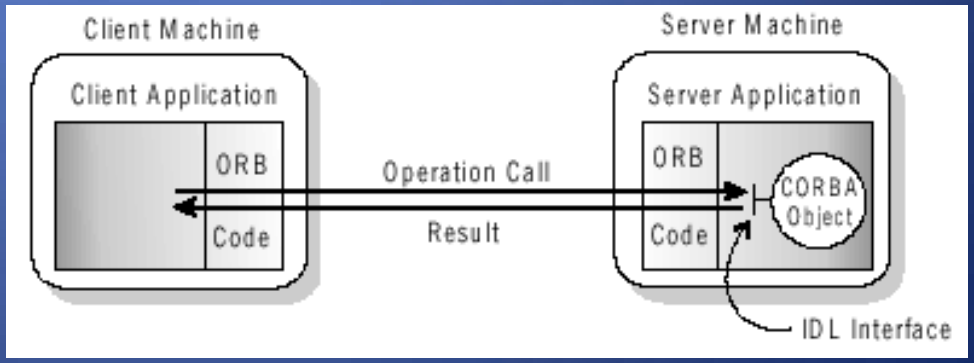


# CORBA 体系结构


Client
Object Implementation
Dynamic Invocation
Client Stubs
ORB Interface
Implementation Skeletons
Object Adapter
ORB Core
standard interface
Proprietary ORB interface
One interface per object adaptor
Normal call interface
Up call interface
One interface per object operation

# 接口定义语言


IDL
Client Developer
IDL Compiler
Stub Code
Client Program
Client Side
IDL Compiler
Server Developer
Skeleton Code
Server Program
Server Side


Client Machine
Client Application
ORB
Code
Operation Call
Result
Server Machine
Server Application
ORB
Code
CORBA Object
IDL Interface

## IDL: Interface Definition Language

```
interface Hello
{
    void say_hello();
};
```

运行idl hello.idl
生成hello.h和hello.cpp

```
…
void Hello::say_hello()
{
}
```

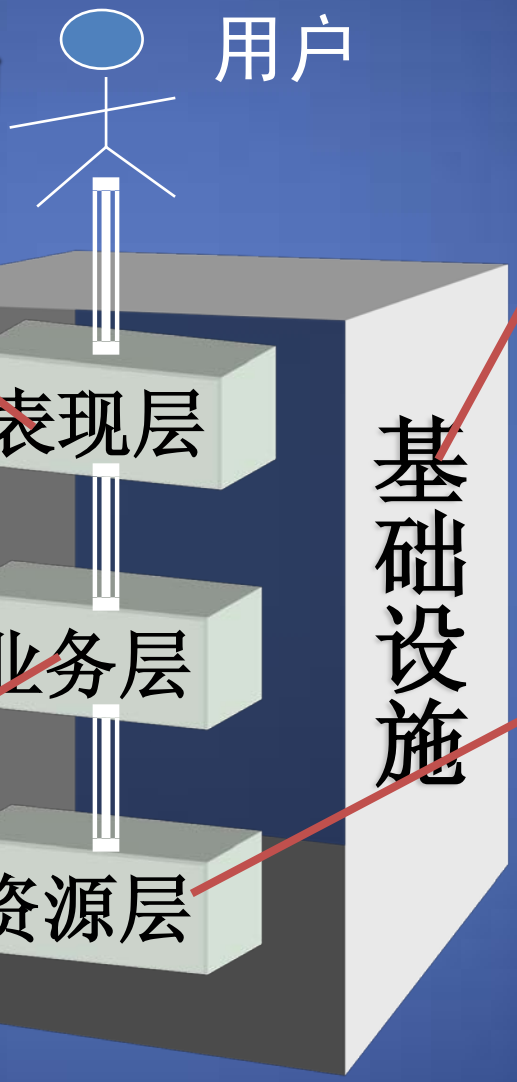
1、软件互操作的基本框架
2、CORBA
3、中间件类型

# 二、互操作框架和中间件

# 中间件标准体系


“英联邦”
模式
“单极”
模式
“联合国”
模式
OBJECT MANAGEMENT GROUP
CORBA/COSS/MDA
Sun
microsystems
Java/RMI/J2EE
Microsoft
COM/DCOM/.NET

# 信息系统基本结构




以多媒体等直观形式采集信息，展现结果

实现应用业务逻辑及其流程的程序代码

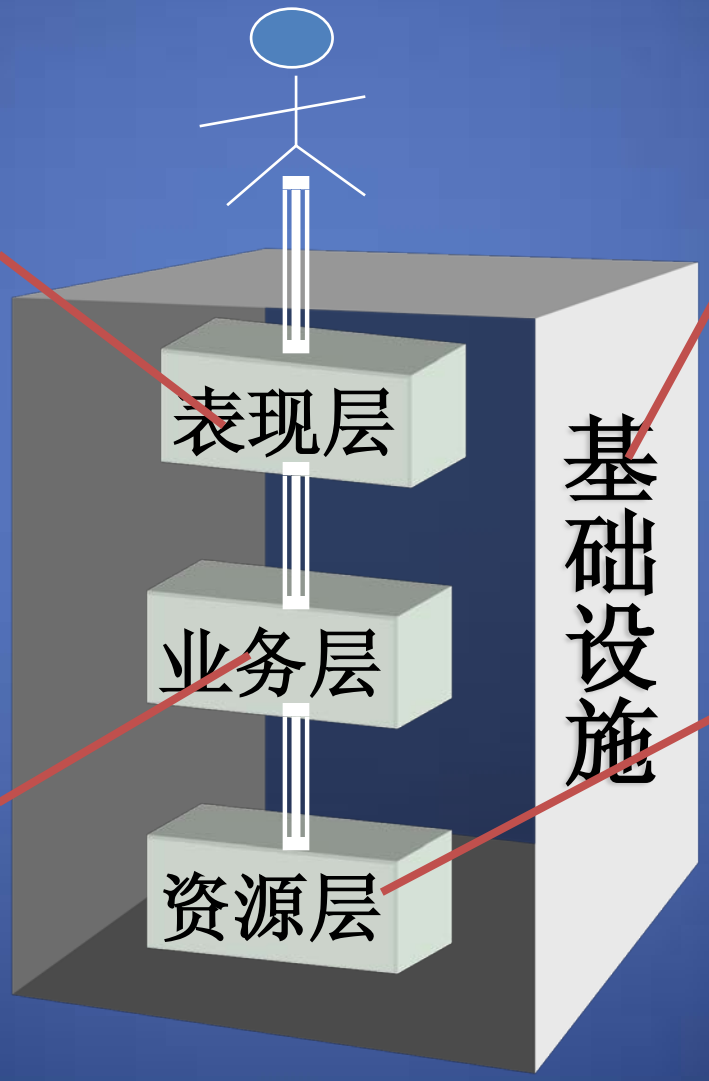
硬件、网络、操作系统

数据库等资源库及其相关的资源管理系统，如数据库管理系统，目录服务系统

# 系统集成

表现资源集成，实现界面的自动生成、统一管理与集成

应用资源集成，实现应用计算资源和业务流程的集成




通信资源集成，实现互联互通互操作

数据资源集成，实现不同数据源的数据交换和数据的语义级集成

# 中间件的分类

- 基础中间件
- 应用中间件
- 领域应用框架

# 基础中间件

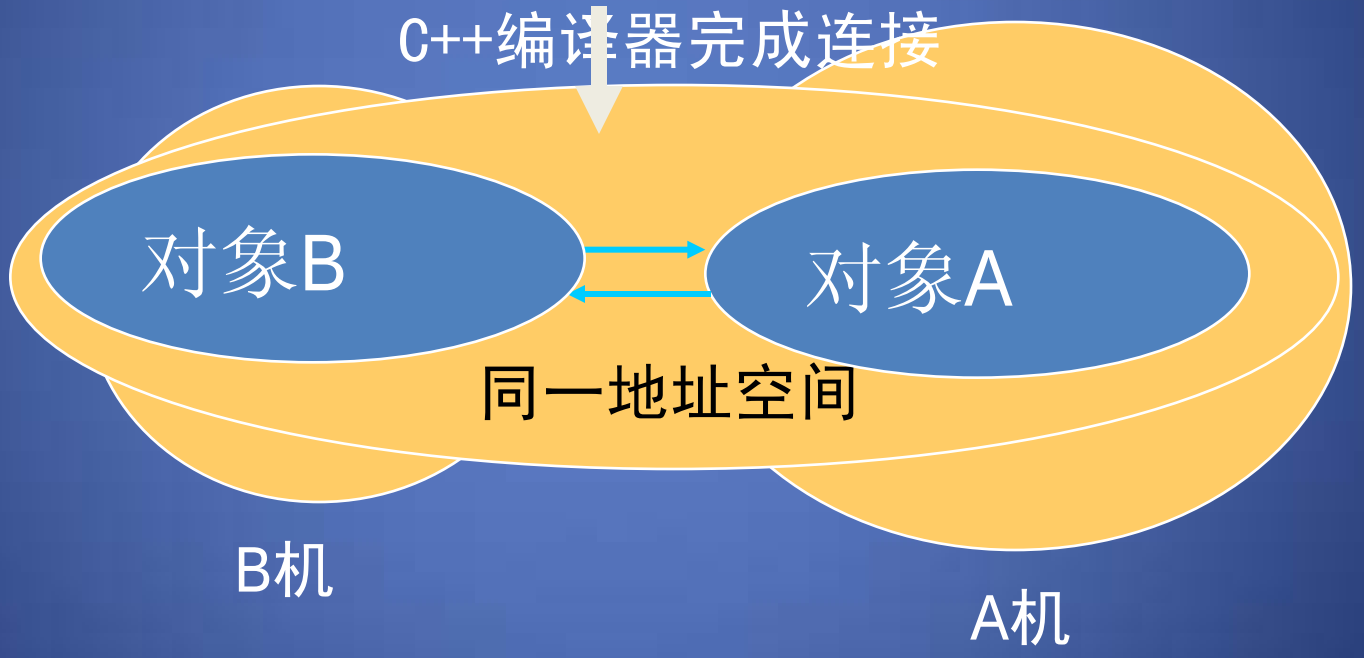
- 基础中间件是在信息系统基础设施层之上构建的
- 底层通信平台＋构件化应用服务器
  - 底层通信平台相当于操作系统的内核
  - 应用服务器相当于操作系统运行环境

# 底层通信平台

- 作用：屏蔽底层各种异构的网络和操作系统，在物理位置透明的情况下，实现异地对象之间的通信和互访。
- 底层通信平台的核心是“软总线”。

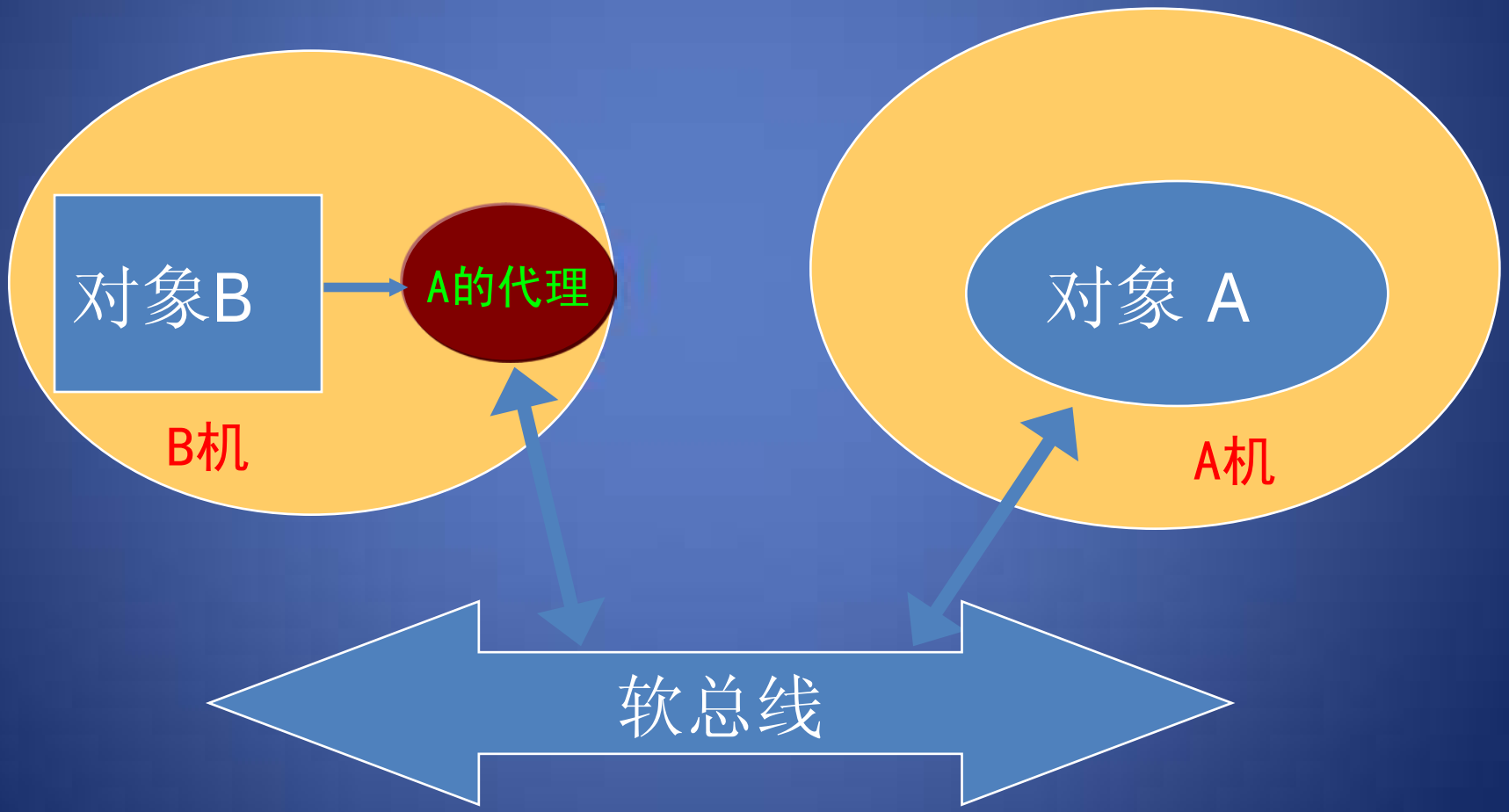
# 软总线

底层通信平台要解决的问题




就是支持访问异地对象

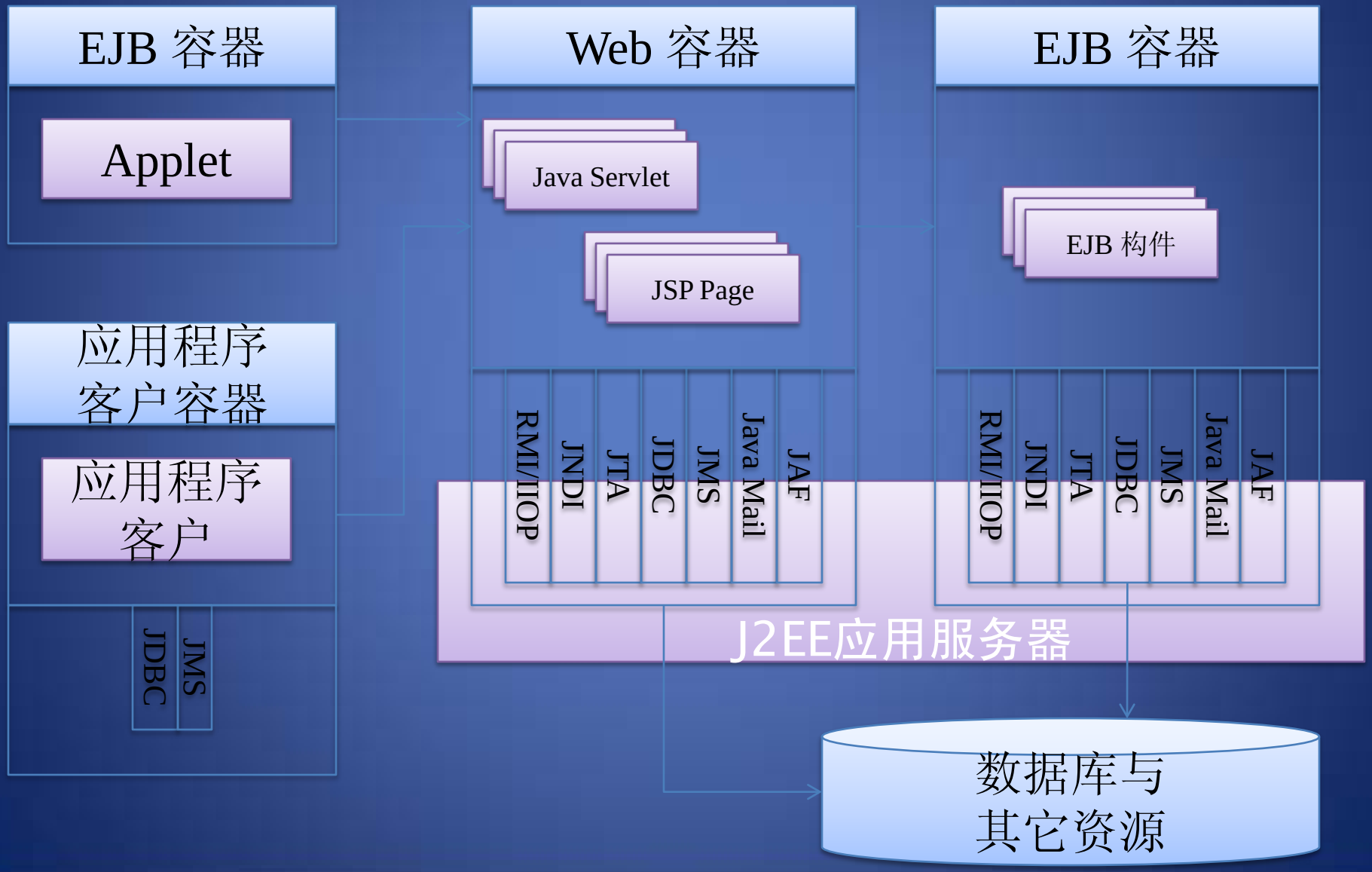
# 软总线



# 构件化应用服务器

- 应用服务器是在底层通信平台的基础上搭建的，是当前中间件中最活跃的部分。
- 主要作用是对网络上各种软硬件资源进行调度和管理，为分布式网络应用提供数据库资源连接、事务处理和安全管理等各类共性服务以及构件化的开发、部署与运行维护环境。

# J2EE体系结构


EJB 容器
Applet
Web 容器
Java Servlet
JSP Page
EJB 容器
EJB 构件
应用程序
客户容器
应用程序
客户
RMI/IIOP
JNDI
JTA
JDBC
JMS
Java Mail
JAF
JDBC
JMS
J2EE应用服务器
数据库与
其它资源

# 应用中间件

- 数据集成中间件
- 应用集成中间件
- 业务流程集成中间件
- 门户集成中间件
- ……

# 数据集成中间件

- 作用：支持信息系统资源层的开发与运行管理，实现不同来源、格式、性质的数据的转换与包装，从而把各种异构数据源集成在一起，并提供一个统一的高层访问服务。
- 原理：数据集成中间件向下需协调各数据库系统，向上应为集成数据的应用提供统一数据模式，以及数据访问的通用接口。
- 关键技术：如何解决好数据的异构性、完整性和语义冲突的问题，是该中间件技术的关键。

# 应用集成中间件

- 利用适配机制把各种新建和遗留应用代码中的各类方法统一成标准的应用接口，并包装为消息的形式
- 通过类似跨国邮政系统所提供的服务机制，即消息代理机制，实现信息系统业务逻辑层应用代码之间跨网络的互连、互通和互操作

# 业务流程集成中间件

- 作用：对信息系统业务逻辑层中的业务流程的整个生命周期进行管理和控制，以协调参与流程的各应用资源代码之间的动态执行关系，并监控和分析流程的执行状况。
- 提供可视化的开发方法，以简化业务流程的描述，适应业务流程的变化。

# 门户集成中间件

- 作用：根据不同的应用需求，调用相应的信息系统业务层、资源层和基础设施层软件，向不同角色的用户提供个性化的服务，为信息系统展现层软件的开发与运行提供支持。
- 主要功能：集中的门户管理和开发，个性化的内容组织与管理，单点认证登录，以及统一而直观的用户界面等。

# 领域应用框架

- 指建立在上述各层中间件之上、面向具体领域应用的信息系统平台
  - 电子政务、电子商务
  - 电信、金融、卫生、教育、交通、……
  - ERP
  - SCM
  - CRM
  - ……

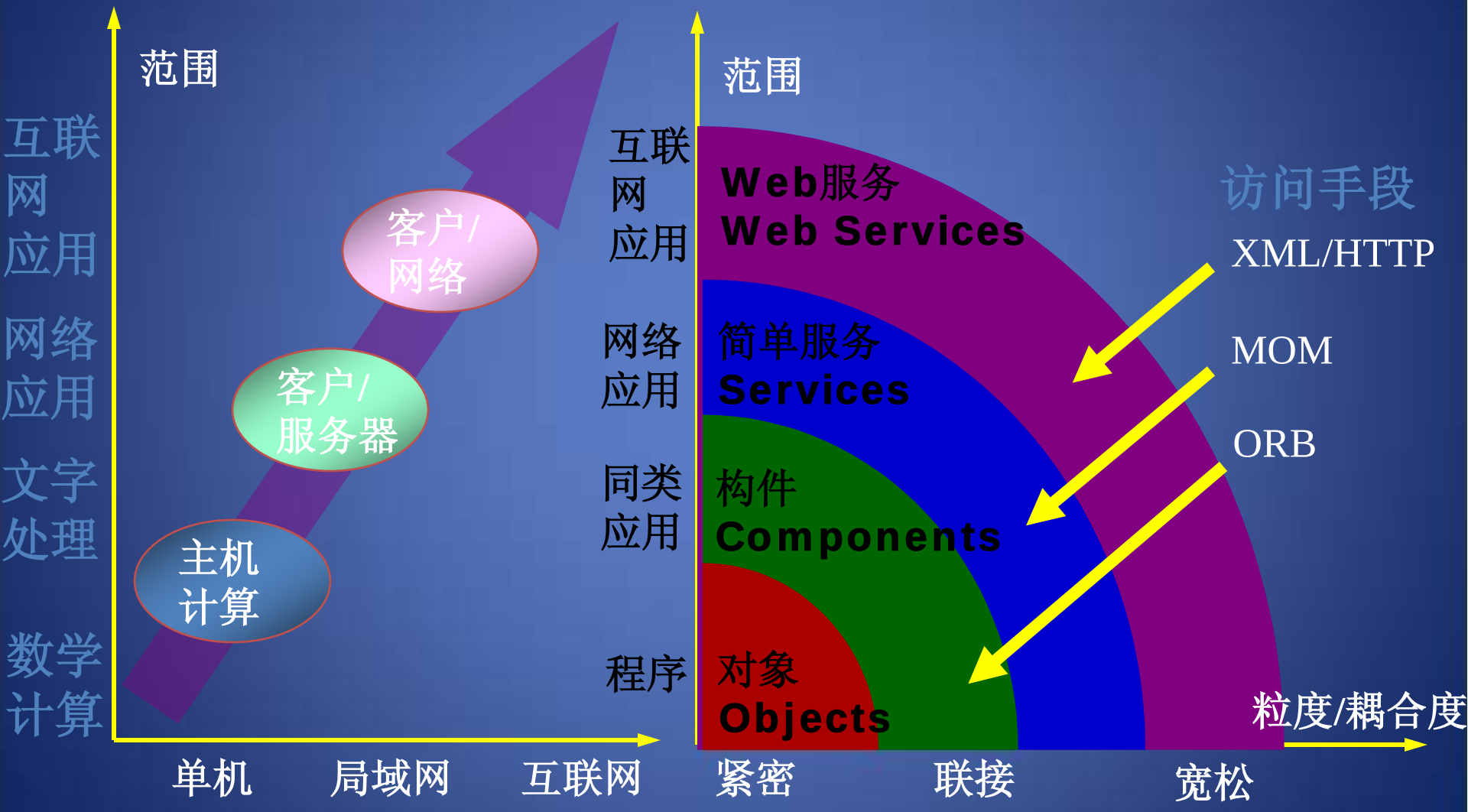
# 中间件技术的难点

- 从计算环境来看
  - 中间件面对的是一个复杂、不断变化的计算环境，要求中间件技术具有足够的灵活性和可成长性；
- 从资源管理的角度来看
  - 操作系统和数据库管理系统管理的是有限资源，资源种类有限，资源量也有限，而中间件需要管理的资源类型（数据、服务、应用）更丰富，且资源扩展的边界是发散的；
- 从应用支撑角度来看
  - 中间件需要提供分布应用开发、集成、部署和运行管理的整个生命周期的总体运行模型
- 从应用的角度来看
  - 利用中间件完成的往往是复杂、大范围的企业级应用，其关系错综复杂，流程交织（例如物流、供应链管理等企业之间应用集成）

# 中间件总体走向

- 中间件技术正在将关注重点从解决局域网资源共享上升到解决广域网、Internet或大型企业Intranet资源共享的转移。
- 中间件的整体技术走向将呈现“收敛”和“普适”两大特色
  - 各类技术通过MDA、具有丰富语义的模型和XML技术而不断聚合；
  - 聚合的技术又可以普遍适用于各类计算平台，从而减轻企业应用集成和互操作的代价。

# 计算模式和中间件发展趋势



# 内　容

一、网络协议和Socket编程

二、互操作和中间件

三、面向服务的体系结构SOA

# SOA的提出

- 1996年，Gartner的研究报告中提出SOA
  - SSA Research Note SPA-401-068, 12 April 1996, 'Service Oriented' Architectures, Part 1

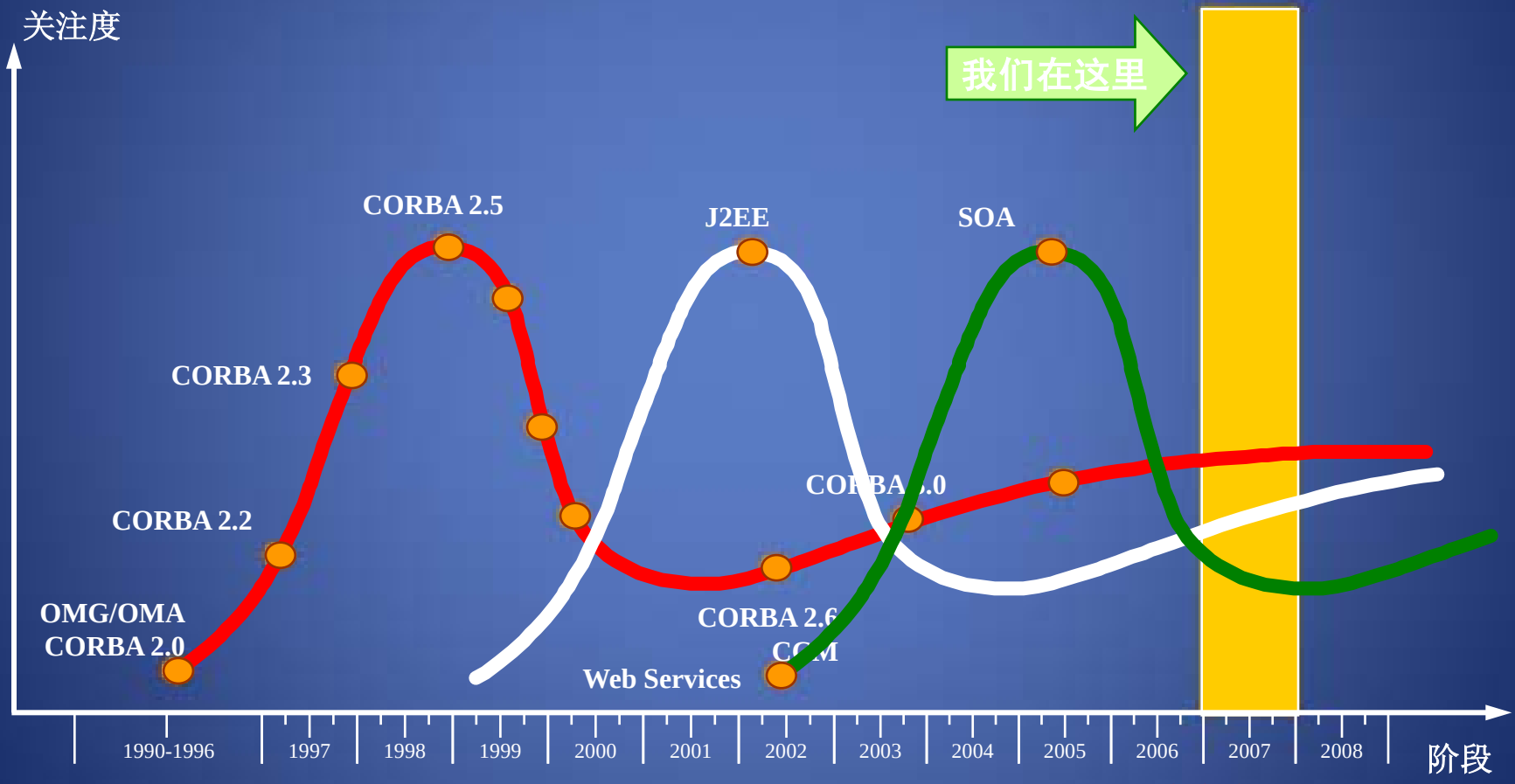
**A service-oriented architecture is a style of multitier computing that helps organizations share logic and data among multiple applications and usage modes.**

**--by Gartner,12 April 1996**

# 关于SOA的预测

- Gartner
  - 到2007年，SOA将成为全球公司的主流
  - 到2008年，SOA将成为占有绝对优势的软件工程实践方法
  - 它将结束传统软件体系架构长达40年的统治地位
  - 将有60％的商业公司在进行商业IT建设时会转向SOA
- IDC
  - 预测到 2007年，包括软件、服务和硬件在内的SOA市场将达到210亿美元
  - 其中商业企业方面的市场将达到120亿美元

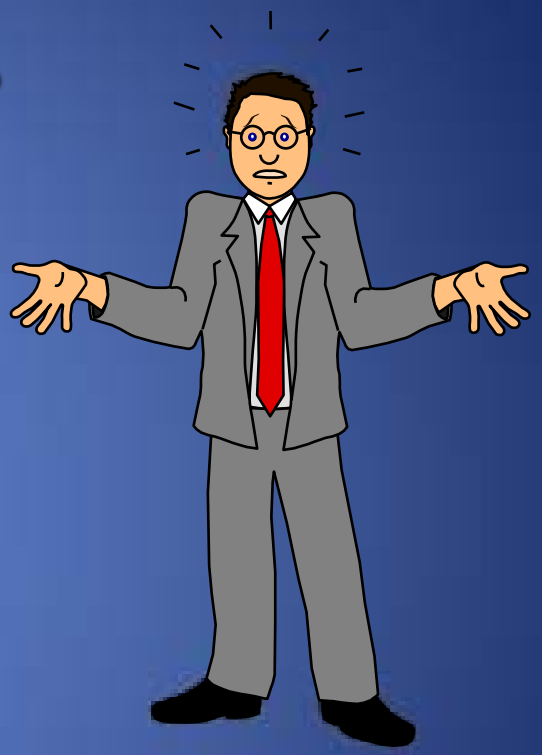
# 技术发展趋势


关注度
我们在这里
CORBA 2.5
J2EE
SOA
CORBA 2.3
CORBA 2.2
CORBA 3.0
OMG/OMA
CORBA 2.0
CORBA 2.6
CCM
Web Services
1990-1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
阶段
启动
阶段
被夸大的
预期峰值
幻灭的
低谷
启蒙
斜坡
生产力
平原

# SOA是什么

- CORBA、J2EE、WebService都可以从技术或者架构上给出较为清晰的解释
  - CORBA是分布对象技术，是“软总线+软构件”架构
  - J2EE是基于Java的企业计算技术，是“构件+容器”架构
  - WebService是基于Web的集成技术

# SOA不是什么



- SOA不是一种语言
  - Java/C/C++
- SOA不是一种具体的技术
  - CORBA/J2EE/WebService
- SOA更不是一种产品
  - StarBus/Orbix/PKUAS/Webspher/Web logic

# SOA是什么

- Gartner
  - 客户端/服务器的软件设计方法，一项应用由软件服务和软件服务使用者组成……SOA与大多数通用的客户端/服务器模型的不同之处，在于它着重强调软件组件的松散耦合，并使用独立的标准接口。
- Service-architecture.com
  - 本质上是服务的集合。服务间彼此通信，这种通信可能是简单的数据传送，也可能是两个或更多的服务协调进行某些活动。服务间需要某些方法进行连接。
  - 所谓服务就是精确定义、封装完善、独立于其他服务所处环境和状态的函数。
- Looselycoupled.com
  - 按需连接资源的系统。在SOA中，资源被作为可通过标准方式访问的独立服务，提供给网络中的其他成员。与传统的系统结构相比，SOA规定了资源间更为灵活的松散耦合关系。

# SOA是什么

- 为了解决在Internet环境下业务集成的需要
- 通过连接能完成特定任务的独立功能实体实现
- 一种软件系统架构

# SOA的目的

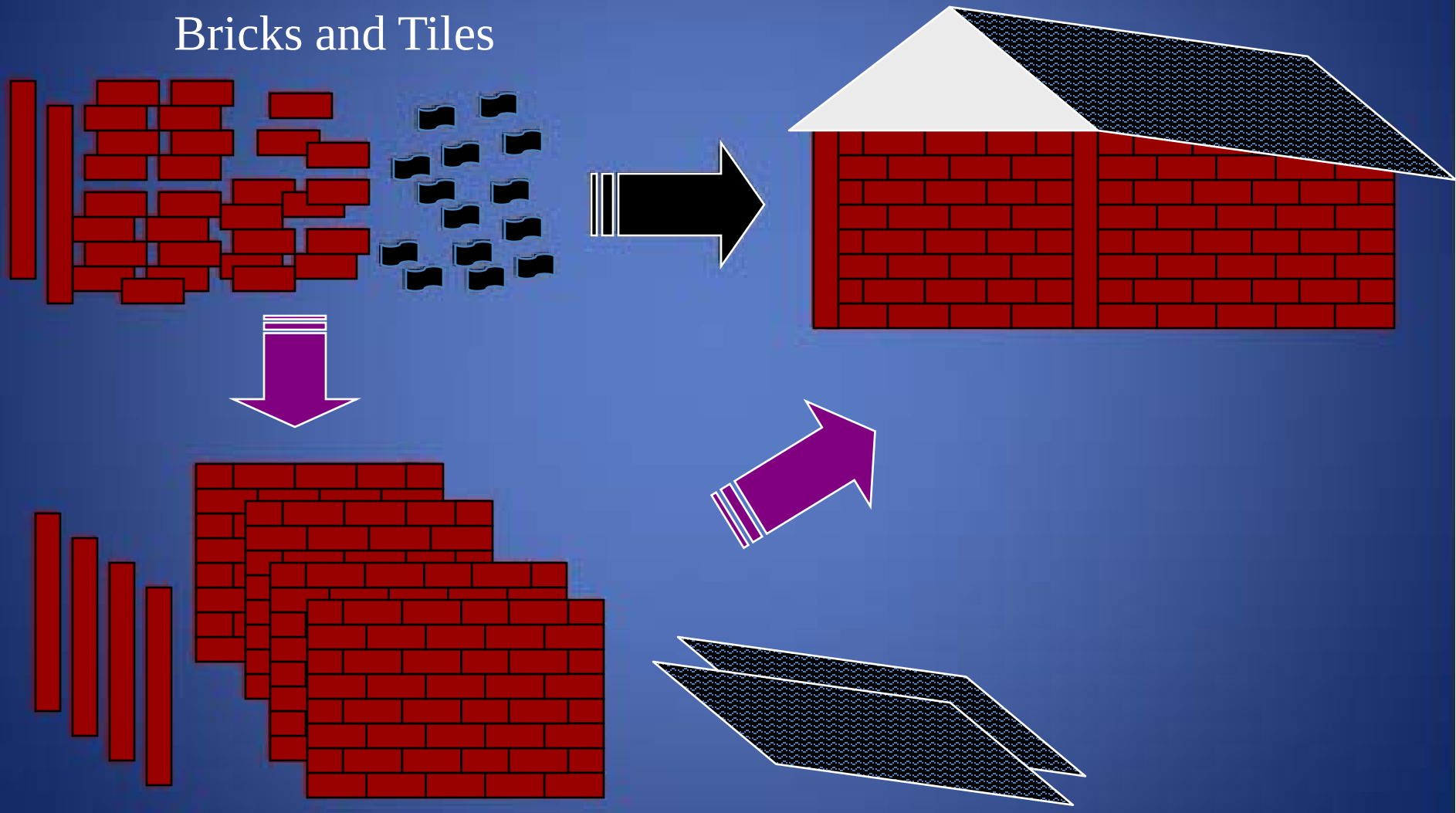
- 为了解决在Internet环境下业务集成的需要
  - 区别于传统的局域网和企业内部网
    - 范围、规模
  - 有保障的TCP/IP会话已不再占据主导
  - 任何访问请求都有可能出错，因此任何企图通过Internet进行控制的结构都会面临严重的稳定性问题

# SOA的手段

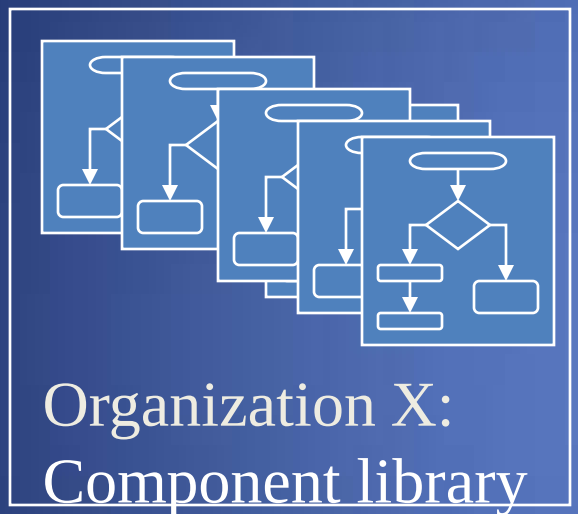
- 通过连接能完成特定任务的独立功能实体实现
  - 每一个功能实体代表一个实际的业务
  - 可独立存在
  - 通过之间定义良好的接口和契约联系起来
    - 接口采用中立的方式进行定义
      - 独立于实现服务的硬件平台、操作系统和编程语言
  - 使得构建在各种这样的系统中的服务可以以一种统一和通用的方式进行交互

# Component-Based Development

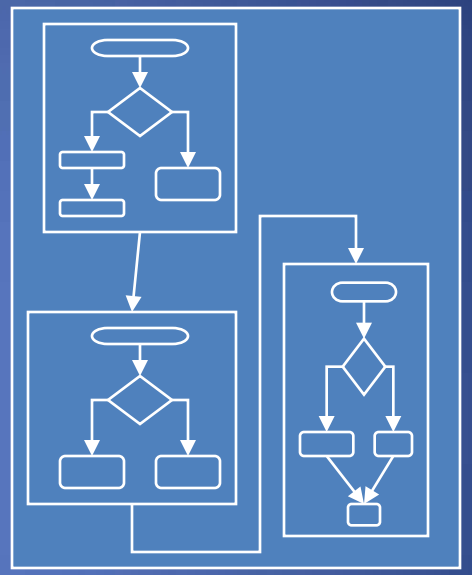
Bricks and Tiles

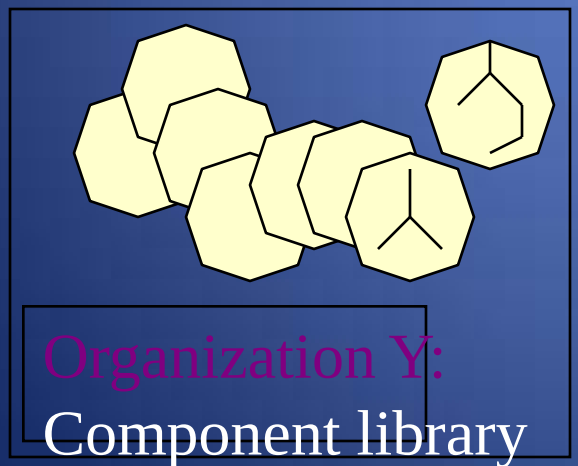


# Component-Based Software Development

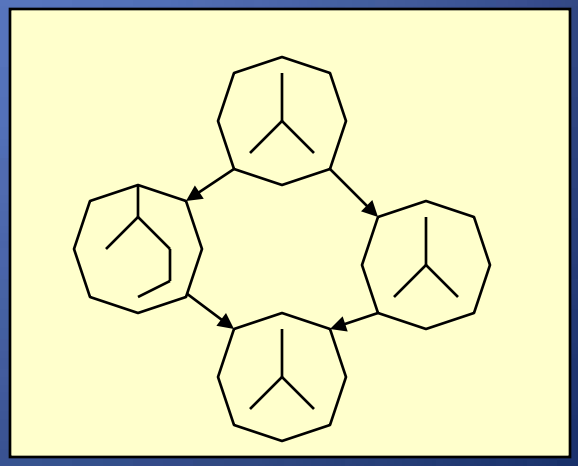


Organization X:
Component library

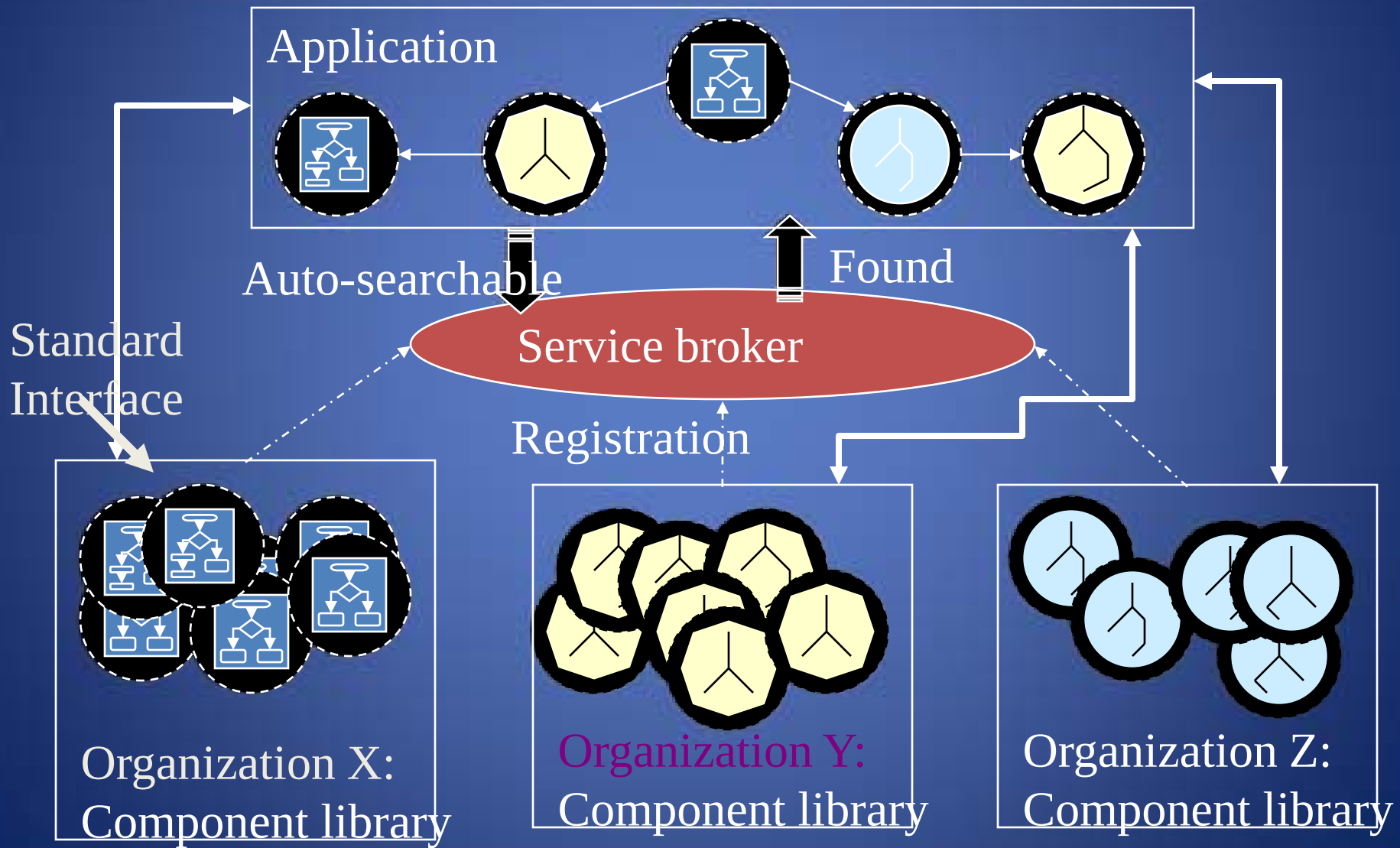




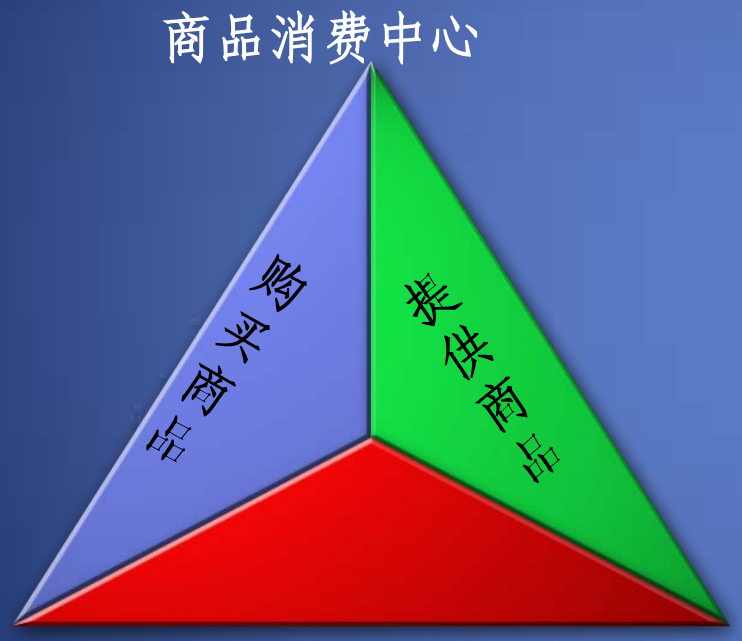
Organization Y:
Component library

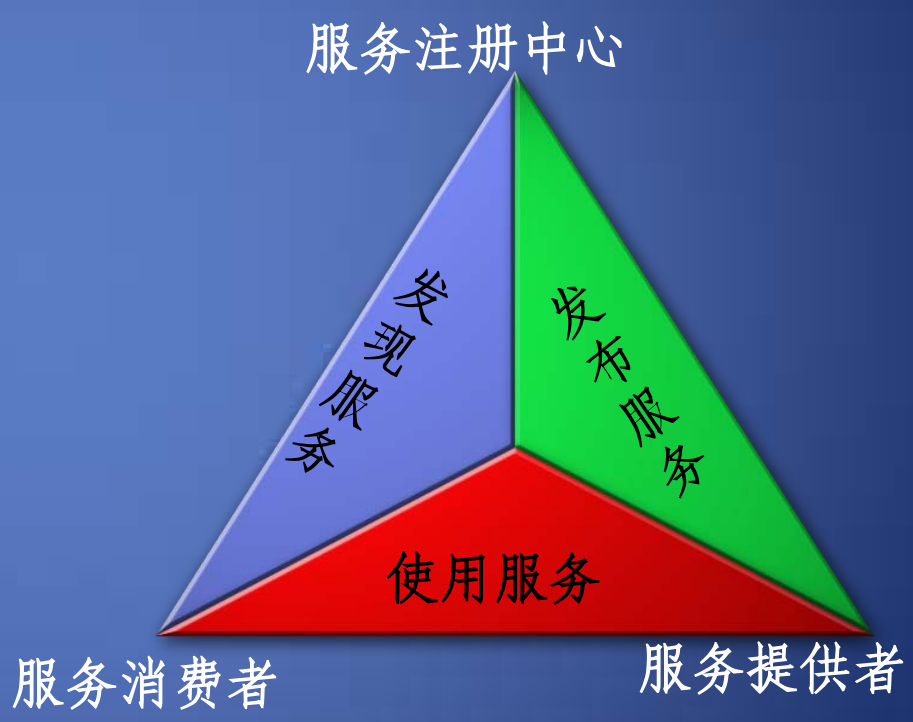


# Service-Oriented Software Development



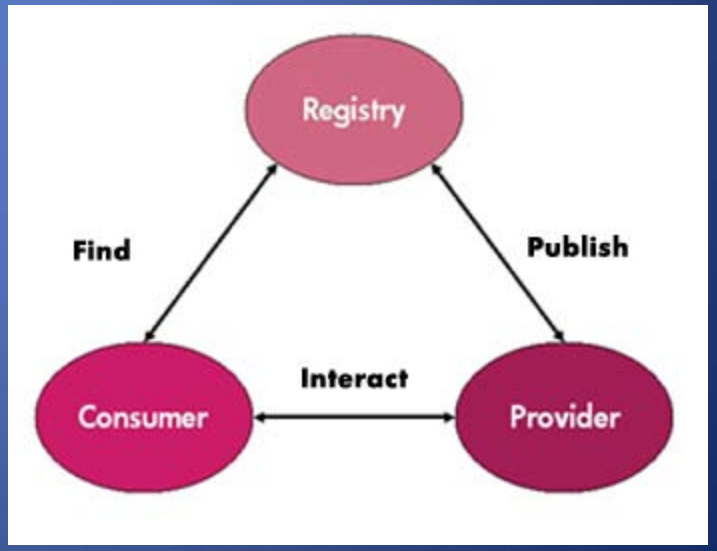
# 商品消费—软件服务


商品消费中心
购买商品
提供商品
商品消费者
商品提供者

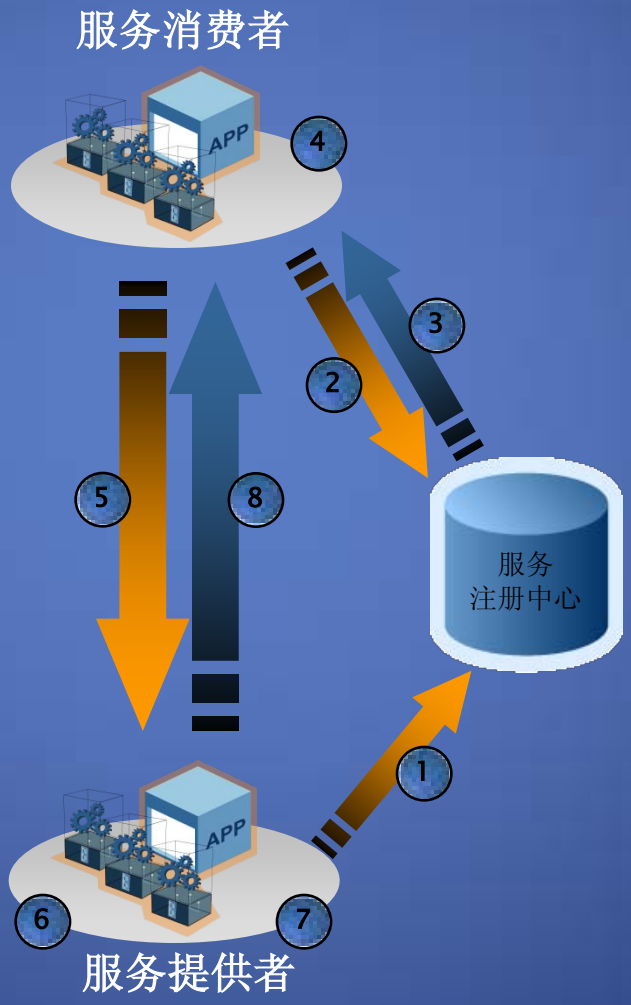

服务注册中心
发现服务
发布服务
使用服务
服务消费者
服务提供者

# SOA的组成

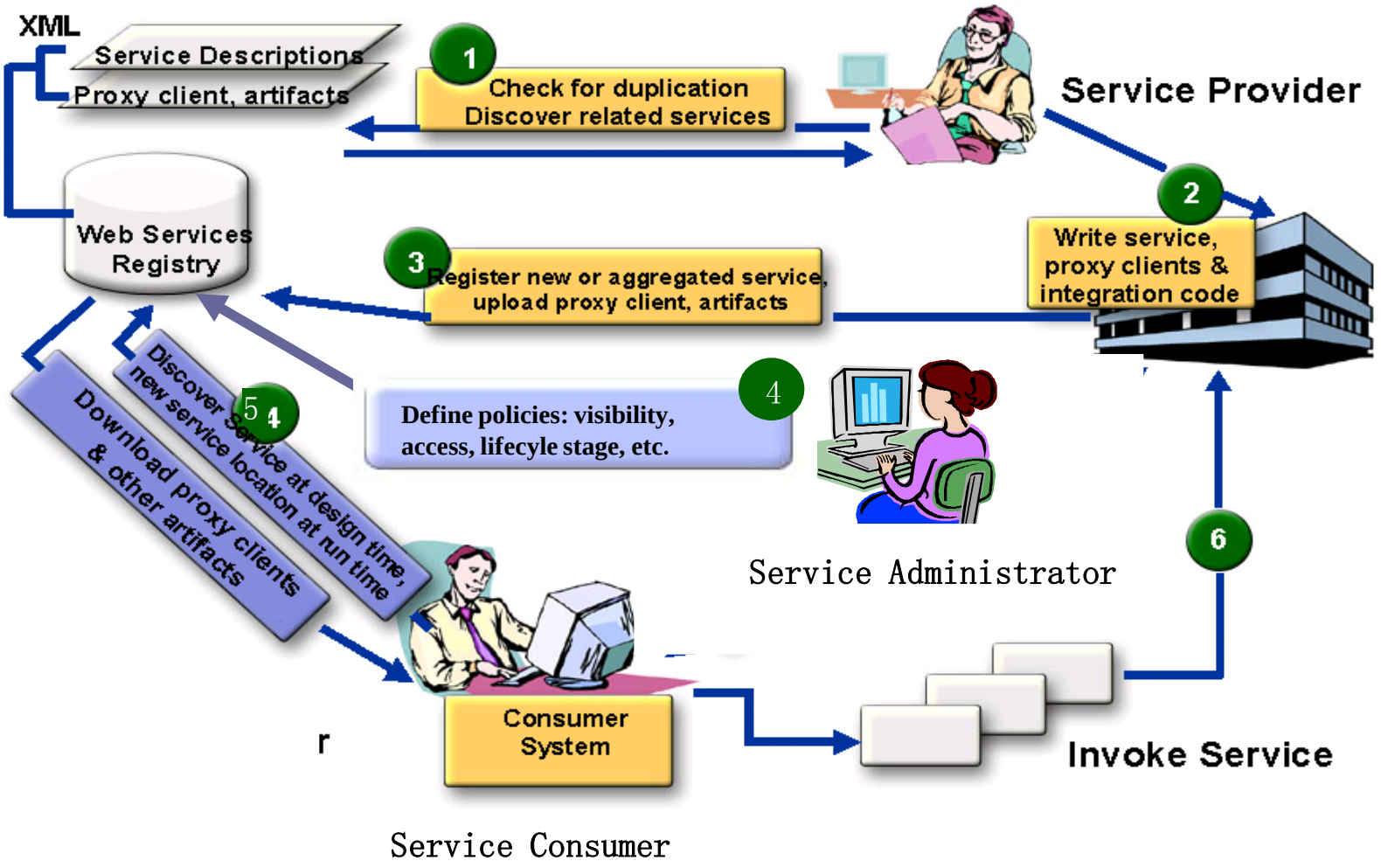
- 服务提供者：
  - 一个可通过网络寻址的实体，接受和执行来自使用者的请求。
  - 将自己的服务和接口契约发布到服务注册中心，以便服务使用者可以发现和访问该服务。
- 服务使用者：
  - 一个应用程序、一个软件模块或需要一个服务的另一个服务。
  - 它发起对注册中心中的服务的查询，通过传输绑定服务，并且执行服务功能。
  - 服务使用者根据接口契约来执行服务。
- 服务注册中心：
  - 服务发现的支持者；
  - 它包含一个可用服务的存储库
  - 允许感兴趣的服务
    使用者查找服务提供者接口。



# 动态交互视图



# SOA 场景举例


XML
Service Descriptions
Proxy client, artifacts
1
Check for duplication
Discover related services
Service Provider
Web Services Registry
2
Write service, proxy clients & integration code
3
Register new or aggregated service, upload proxy client, artifacts
4
Define policies: visibility, access, lifecyle stage, etc.
5 4
Discover Service at design time, new service location at run time
Download proxy clients & other artifacts
Service Administrator
6
Consumer System
Invoke Service
Service Consumer

# 关键服务特征

- 主要特征：
  - 松耦合的服务
    - 接口耦合
    - 技术耦合
    - 流程耦合
  - 明确定义的服务契约
  - 对服务请求者有意义
  - 开放的、基于标准的
- 次要特征：
  - 预期服务水平协议
  - 动态的、可发现的、元数据驱动的
  - 设计服务契约时考虑到相关服务
  - 独立于其他服务的实现
  - 可考虑到对补偿事务的需要
  - 设计多种调用方式
  - 无状态的
  - 设计服务时考虑到效率

# SOA的基本特征

- 独立的功能实体
- 可从企业外部访问
- 大数据量低频率访问
- 粗粒度的服务接口
- 松散耦合
- 标准化的服务接口
- 基于文本的消息传递
- 支持各种消息模式
- 精确定义的服务契约

# 独立的功能实体

- Internet环境
- 强调架构中提供服务的功能实体的完全独立自主的能力
- 强调实体自我管理和恢复能力
  - 事务处理(Transaction)
  - 消息队列(Message Queue)
  - 冗余部署(Redundant Deployment)
  - 集群系统(Cluster)

# 可从企业外部访问

- 通常被称为业务伙伴的外部用户也能像企业内部用户一样访问相同的服务。
  - 业务伙伴采用先进的B2B协议（ebXML或RosettaNet）相互合作。
  - 当业务伙伴基于业务目的交换业务信息时，他们就参与了一次会话。会话是业务伙伴间一系列的一条或多条业务信息的交换。会话类型（会话复杂或简单、长或短等）取决于业务目的。
  - 除了B2B协议外，外部用户还可以访问以Web服务方式提供的企业服务。

# 大数据量低频率访问

- 传统的分布式计算模型
  - 服务提供是通过函数调用的方式进行的
  - 一个功能的完成往往需要通过客户端和服务器来回很多次函数调用才能完成
  - Intranet的环境下，这些调用给系统的响应速度和稳定性带来的影响都可以忽略不计
  - 在 Internet环境下这些因素往往是决定整个系统是否能正常工作的一个关键决定因素
- SOA系统推荐采用大数据量的方式一次性进行信息交换

# 粗粒度服务接口

- 粗粒度服务提供一项特定的业务功能，而细粒度服务代表了技术构件方法。
- 采用粗粒度服务接口的优点在于使用者和服务层之间不必再进行多次的往复，一次往复就足够。Internet环境中有保障的TCP/IP会话已不再占据主导、建立连接的成本也过高，因此在该环境中进行应用开发时粗粒度服务接口的优点更为明显。
- 除去基本的往复效率，事务稳定性问题也很重要。在一个单独事务中包含的多段细粒度请求可能使事务处理时间过长、导致后台服务超时，从而中止。与此相反，从事务的角度来看，向后台服务请求大块数据可能是获取反馈的唯一途径。

# 松散耦合

- SOA具有“松散耦合”构件服务，这一点区别于大多数其他的构件架构。该方法旨在将服务使用者和服务提供者在服务实现和客户如何使用服务方面隔离开来
- 服务提供者和服务使用者间松散耦合背后的关键点是服务接口作为与服务实现分离的实体而存在。这是服务实现能够在完全不影响服务使用者的情况下进行修改
- 大多数松散耦合方法都依靠基于服务接口的消息。基于消息的接口能够兼容多种传输方式（如HTTP、JMS、TCP/IP、MOM等）。基于消息的接口可以采用同步和异步协议实现，Web服务对于SOA服务接口来讲是一个重要的标准
- 当使用者调用一个Web服务，被调用的对象可以是CICS事务、DCOM或CORBA对象、J2EE EJB或TUXEDO服务等，但这与服务使用者无关
- 消息类Web服务通常是松散耦合和文档驱动的，这要优于与服务特定接口的连接。当客户调用消息类Web服务时，客户通常会发送一个完整的文档（如采购订单），而非一组离散的参数。Web服务接收整个文档、进行处理、而后会或不会返回结果。由于客户和Web服务间不存在紧密耦合请求响应，消息类Web服务在客户和服务器间提供更为松散的耦合

# 标准化的接口

- 两个重要标准XML和Web服务将SOA推向更高的层面，并大大提升了SOA的价值。尽管以往的SOA产品都是专有的、并且要求IT部门在其特定环境中开发所有应用，但XML和Web服务标准化的开放性使企业能够在所部署的所有技术和应用中采用SOA。这具有巨大的意义！

- Web服务使应用功能得以通过标准化接口（WSDL）提供，并可基于标准化传输方式（HTTP和JMS）、采用标准化协议（SOAP）进行调用。例如，开发人员可以采用最适于门户开发的工具轻松创建一个新的门户应用，并可以重用ERP系统和定制化J2EE应用中的现有服务，而完全无须了解这些应用的内部工作原理。采用XML，门户开发人员无须了解特定的数据表示格式，便能够在这些应用间轻松地交换数据。

- 你也可以不采用Web服务或XML来创建SOA应用，但是这两种标准的重要性日益增加、应用日趋普遍。尽管目前只有几种服务使用者支持该标准，但未来大多数的服务使用者都会将其作为企业的服务访问方法。

# 基于文本的消息传递

- 传统的构件模型
  - 从服务器端传往客户端的是一个二进制编码的对象，客户端通过调用这个对象的方法来完成某些功能
- 在Internet环境下
  - 不同语言，不同平台对数据、甚至是一些基本数据类型定义不同
  - 基于文本的消息本身不包含任何处理逻辑和数据类型的
  - 服务间只传递文本
  - 对数据的处理依赖于接收端的方式可以帮忙绕过兼容性这个的大泥坑
- 在Internet环境下
  - 版本管理极其困难
  - 采用基于文本的消息传递方式，数据处理端可以只选择性的处理自己理解的那部分数据，而忽略其它的数据，从而得到非常理想的兼容性

# 支持各种消息模式

- SOA中可能存在以下消息模式。在一个SOA实现中，常会出现混合采用不同消息模式的服务。
- 无状态的消息。使用者向提供者发送的每条消息都必须包含提供者处理该消息所需的全部信息。这一限定使服务提供者无须存储使用者的状态信息，从而更易扩展。
- 有状态的消息。使用者与提供者共享使用者的特定环境信息，此信息包含在提供者和使用者交换的消息中。这一限定使提供者与使用者间的通信更加灵活，但由于服务提供者必须存储每个使用者的共享环境信息，因此其整体可扩展性明显减弱。该限定增强了服务提供者和使用者的耦合关系，提高了交换服务提供者的服务难度。
- 等幂消息。向软件代理发送多次重复消息的效果和发送单条消息相同。这一限定使提供者和消费者能够在出现故障时简单的复制消息，从而改进服务可靠性。

# 精确定义的服务接口

- 服务是由提供者和使用者间的契约定义的。契约规定了服务使用方法及使用者期望的最终结果。此外，还可以在其中规定服务质量。此处需要注意的关键点是，服务契约必须进行精确定义。
- META将SOA定义为：“一种以通用为目的、可扩展、具有联合协作性的架构，所有流程都被定义为服务，服务通过基于类封装的服务接口委托给服务提供者，服务接口根据可扩展标识符、格式和协议单独描述。”该定义的最后部分表明在服务接口和其实现之间有明确的分界。

# SOA和其他技术看问题的角度不同