高级操作系统

Advanced Operating System

Distributed Systems

Concepts and design

朱 青 Qing Zhu,

Department of Computer Science, Renmin University of China

zqruc2012@aliyun.com

Chapter 4
Remote Procedure Call 远程过程调用RPC

Teacher: Qing Zhu 朱 青

Department of Computer Science,
Information School,
Renmin University of China
zqruc2012@aliyun.com

Textbook

- **Distributed Systems – Concepts and Design, George Couloris, Jean Dollimore, and Tim Kindberg Edition 3,4, © Pearson Education 2001, 2005**
- **The lecture is based on this textbook.**

第4章 远程过程调用RPC

- 4.1 分层协议
- 4.2 远程过程调用
- 4.3 远程对象调用
- 4.4 面向消息的通信

第4章 远程过程调用RPC

进程间的通信是一切分布式系统的基础，它基于底层网络提供的底层消息传递机制

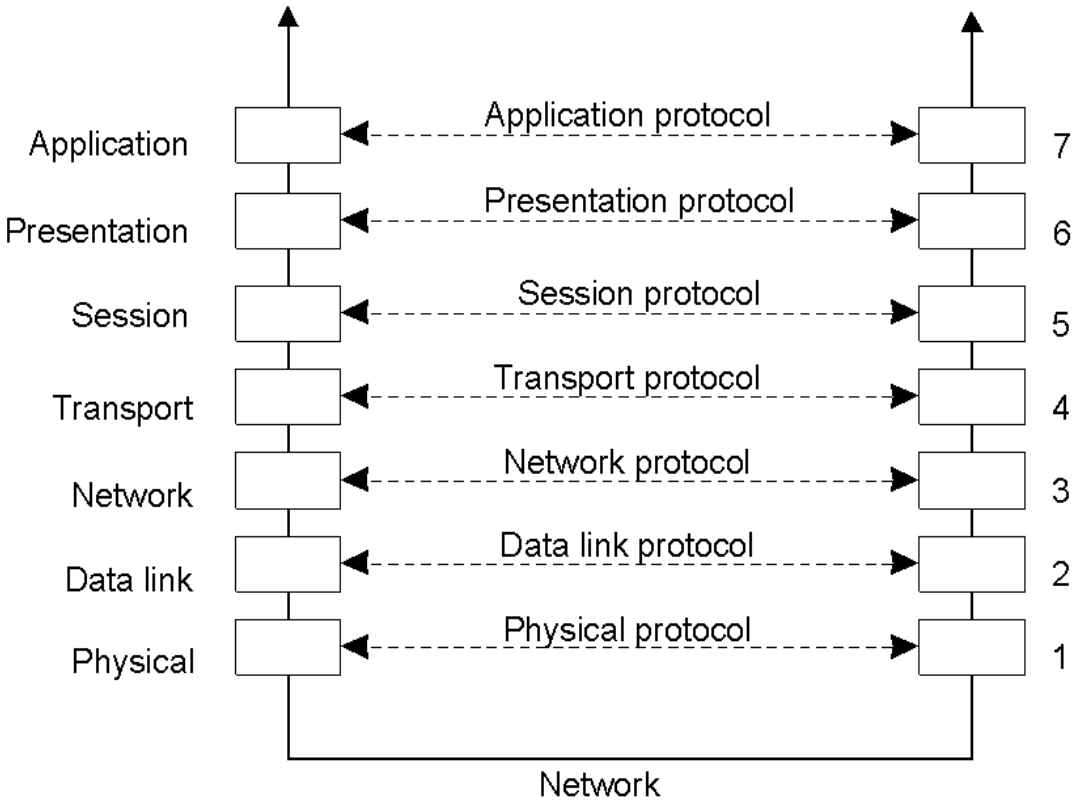
- 分层协议
- 远程过程调用
- 远程对象调用
- 面向消息的通信
- 多播通信

层次协议（1）

- OSI 模型中的层、接口和协议

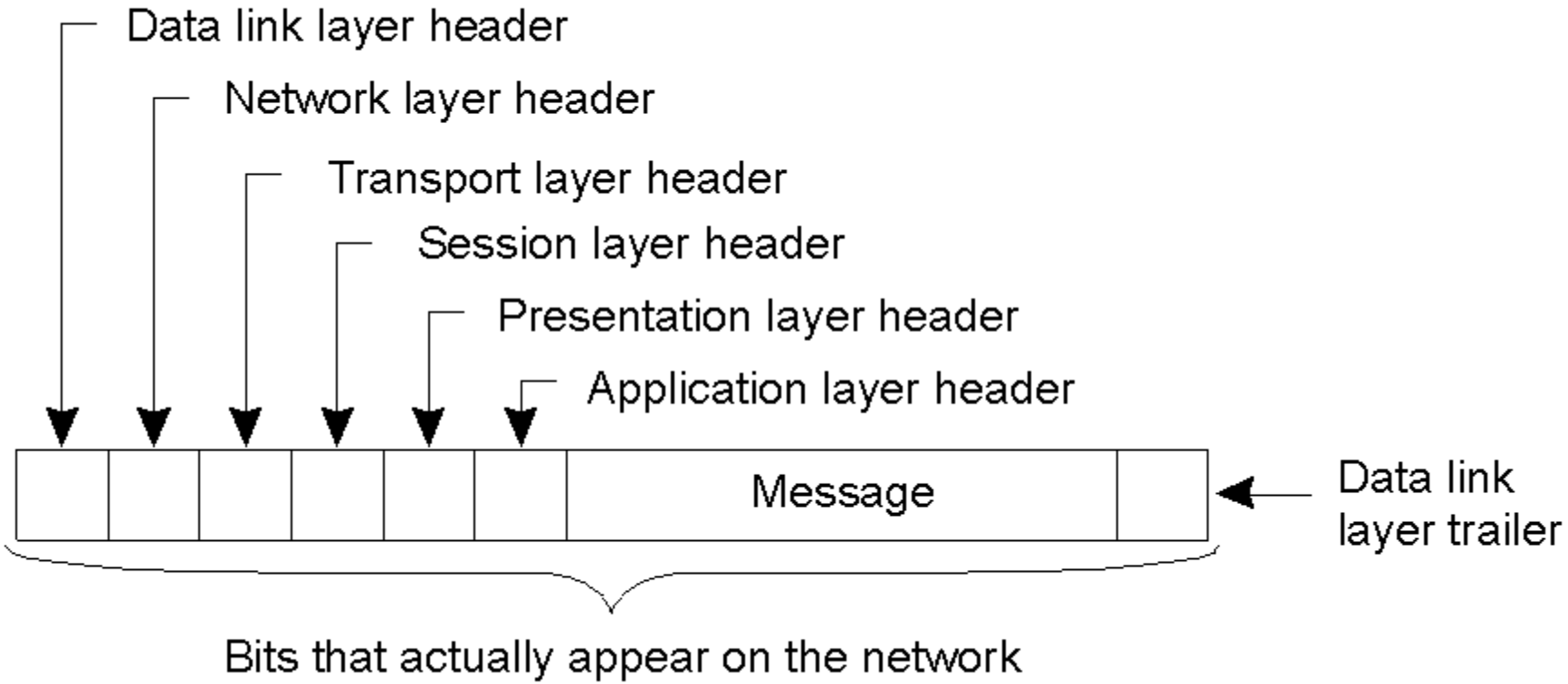
必须在不同层次制订多种协议，包括从位传输的底层细节到信息表示的高层细节：

- 0，1的电压表示
- 消息的结束位
- 检测消息的丢失或损坏及其处理
- 数值、字符串及其它数据项的长度和表示方法
- 面向连接的协议：电话
- 无连接的协议：邮箱



层次协议(2)

- 在网络上传输的典型消息



第4章　远程过程调用RPC

- 4.1 分层协议
- 4.2 远程过程调用
- 4.3 远程对象调用
- 4.4 面向消息的通信

4.2 远程过程调用

RPC（Remote Procedure Call）是分布式系统通信处理的事实标准,实现消息传输的透明性。

- 常规过程调用
- 客户存根和服务器存根
- 参数传递

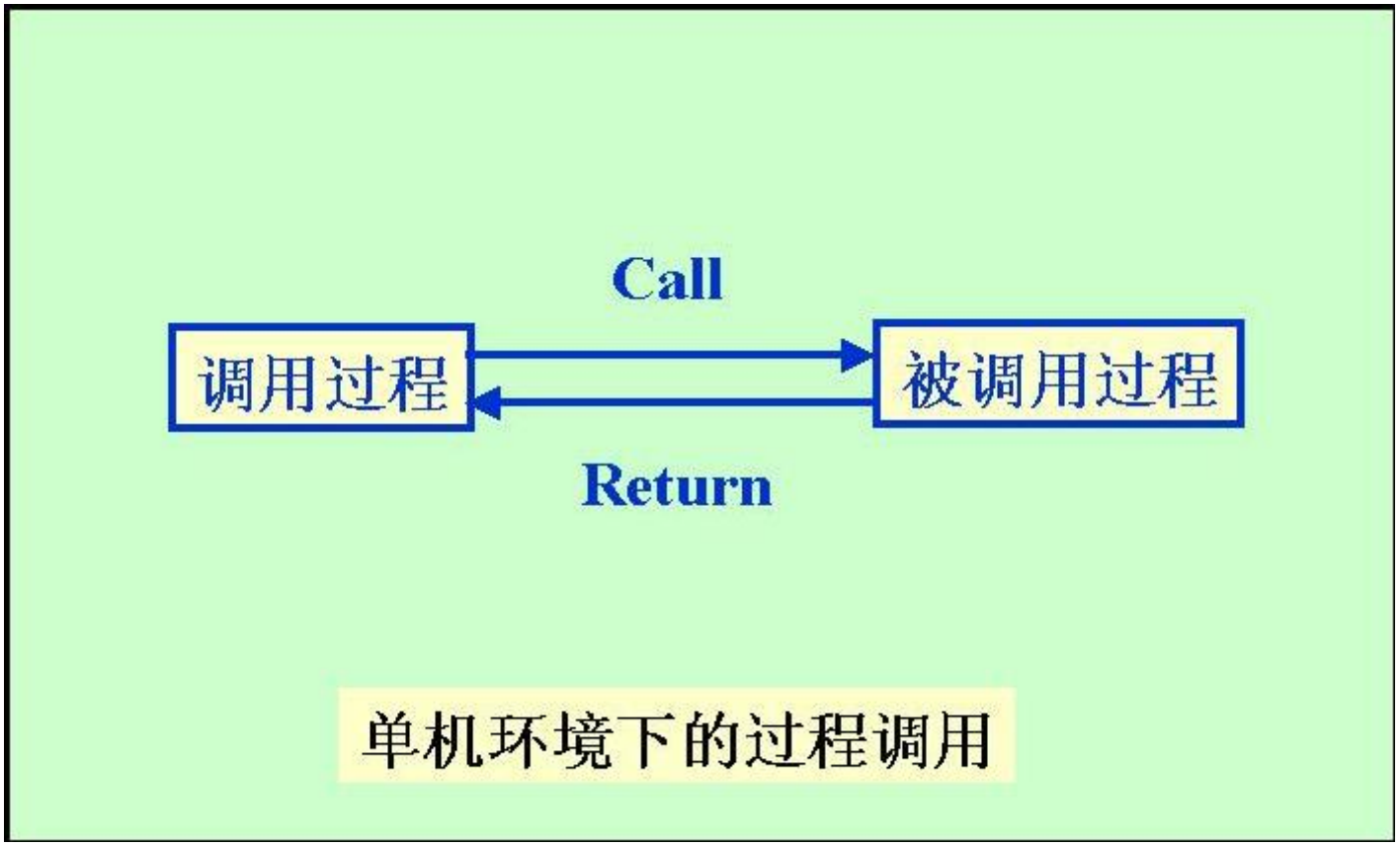
远程过程调用（RPC）

将单机环境下的过程调用延伸到分布式系统环境

ISO将RPC作为计算机网络、分布式计算机系统的国际标准化草案

一种信息交换的标准规程

单机环境下的过程调用



单机环境下的过程调用

传统的过程调用

- 传统的过程调用

  count=read(fd, buf, nbytes)

  fd：整数，buf：字符数组，nbytes：整数

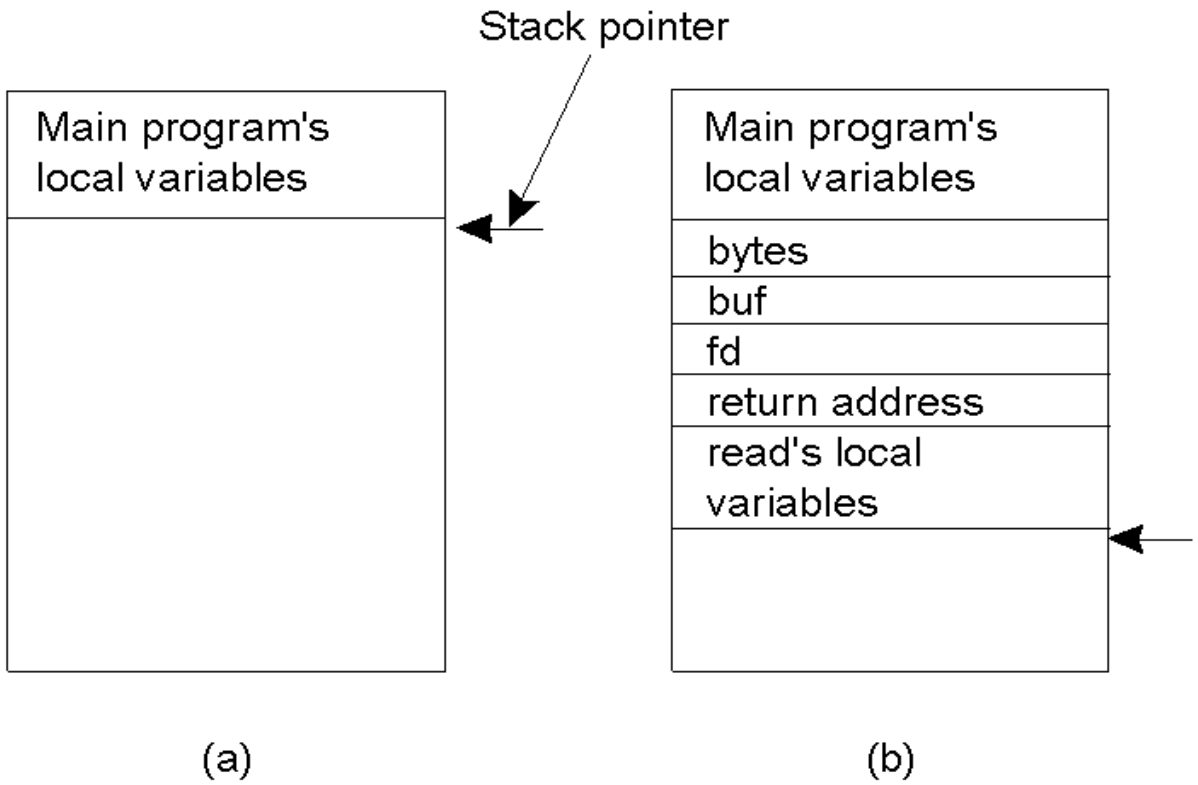
  如果是由主程序进行的调用，则调用之前的栈如图a所示。然后，调用者将参数顺序推入栈，最后一个参数首先入栈，如图b所示。

  read运行完毕以后，它把返回值放入一个寄存器，从栈里移走返回地址，将控制权返回给调用者。调用者再从栈里移去参数，回到先前的状态。

常规过程调用

Count=read(fd,buf,nbyte),本地过程调用中的参数传递：

- 调用read前的堆栈状态
- 过程调用执行时的堆栈状态

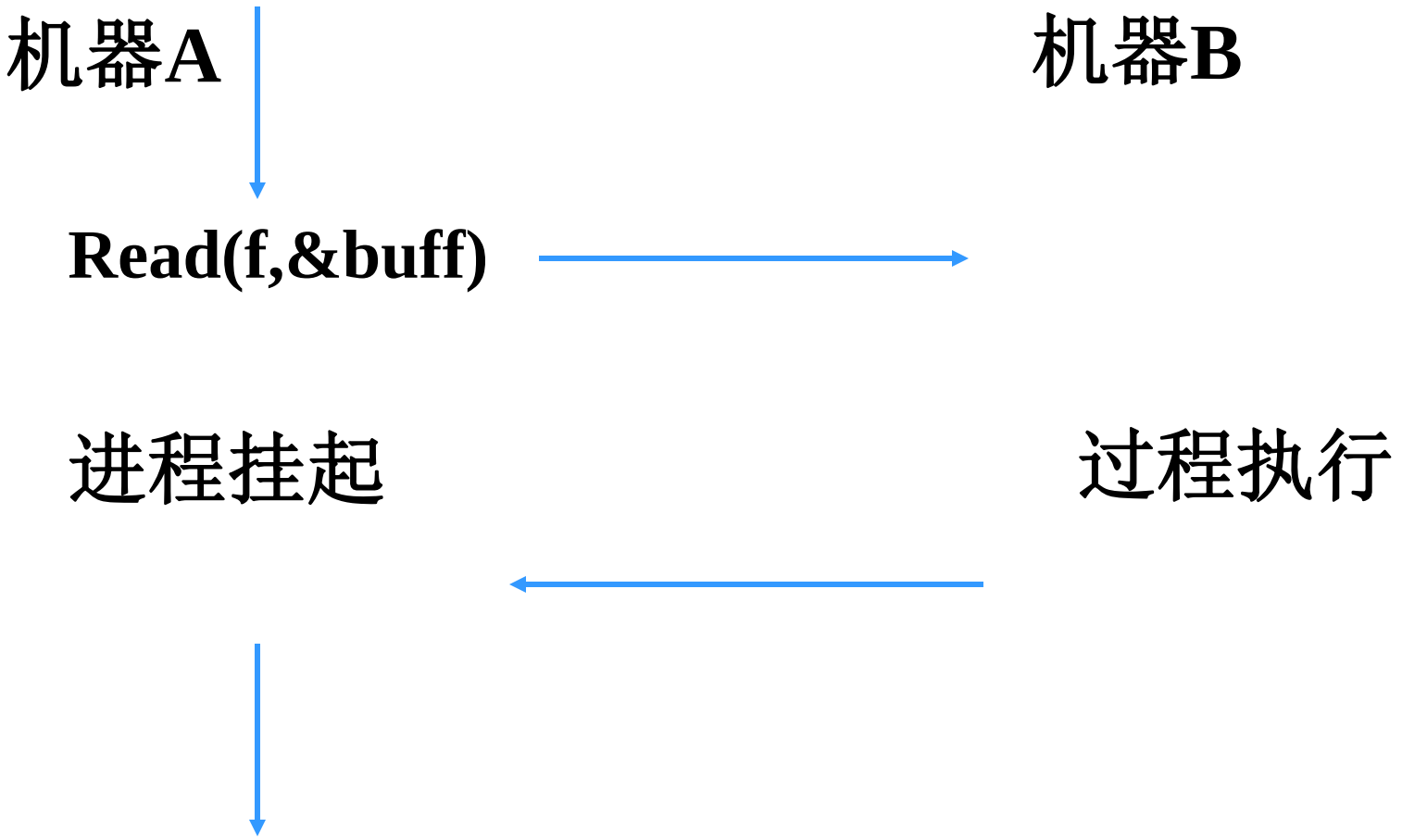


远程过程调用（Remote Procedure Call）

指用户可以向调用本地过程一样调用不同地域的不同计算机上的过程，从而使得应用程序设计人员不必设计和开发有关发送和接收信息的实现细节

没有区别，透明

远程过程调用（RPC）



远程过程调用（RPC）

- 对调用者透明：
- 　被调用过程是在远程机器上执行

- 透明性是通过在客户端插入一个称为client stub，在服务器端插入一个称为server stub完成的

客户存根和服务器存根

- 客户和服务器间的RPC原理

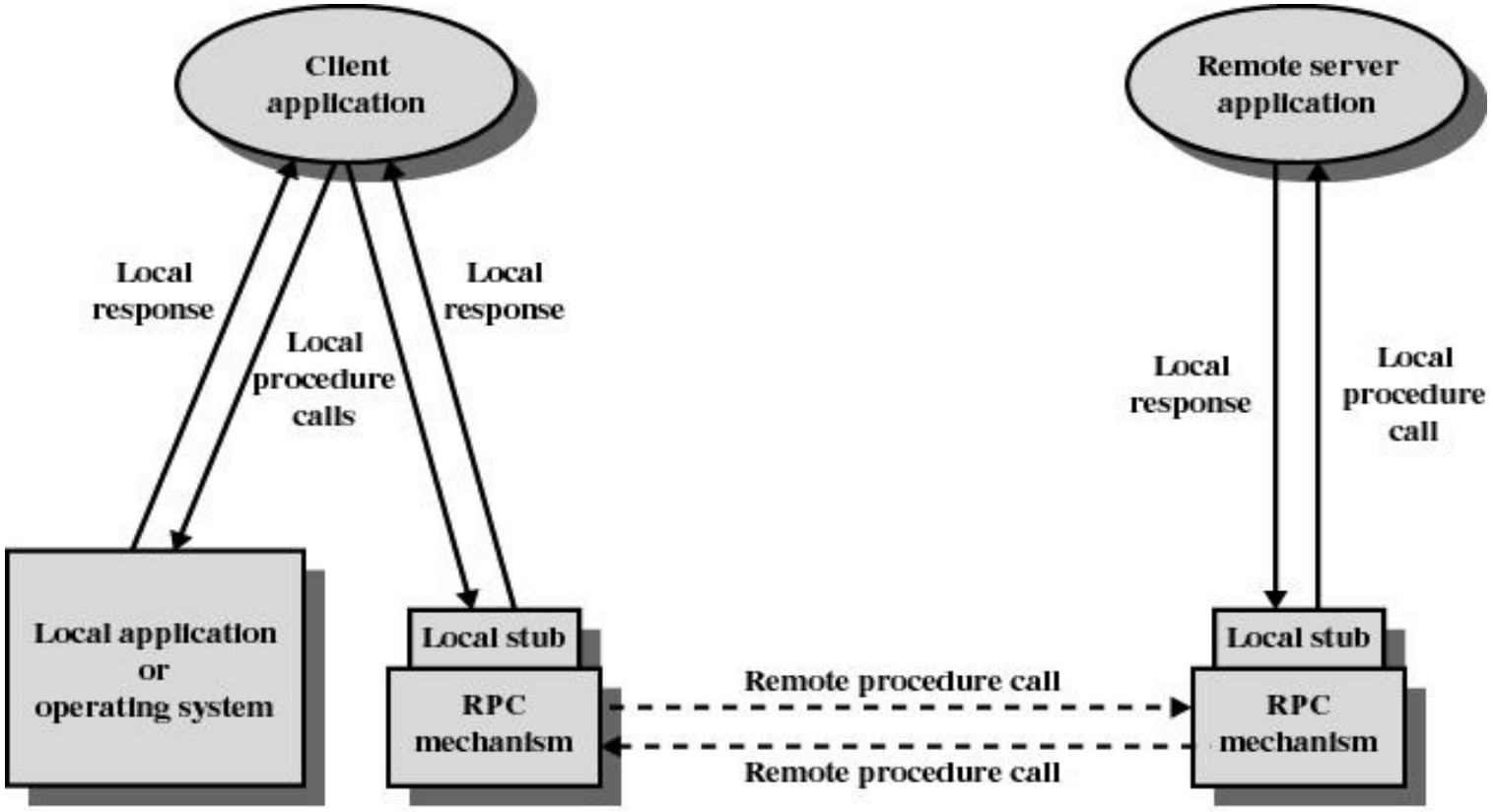
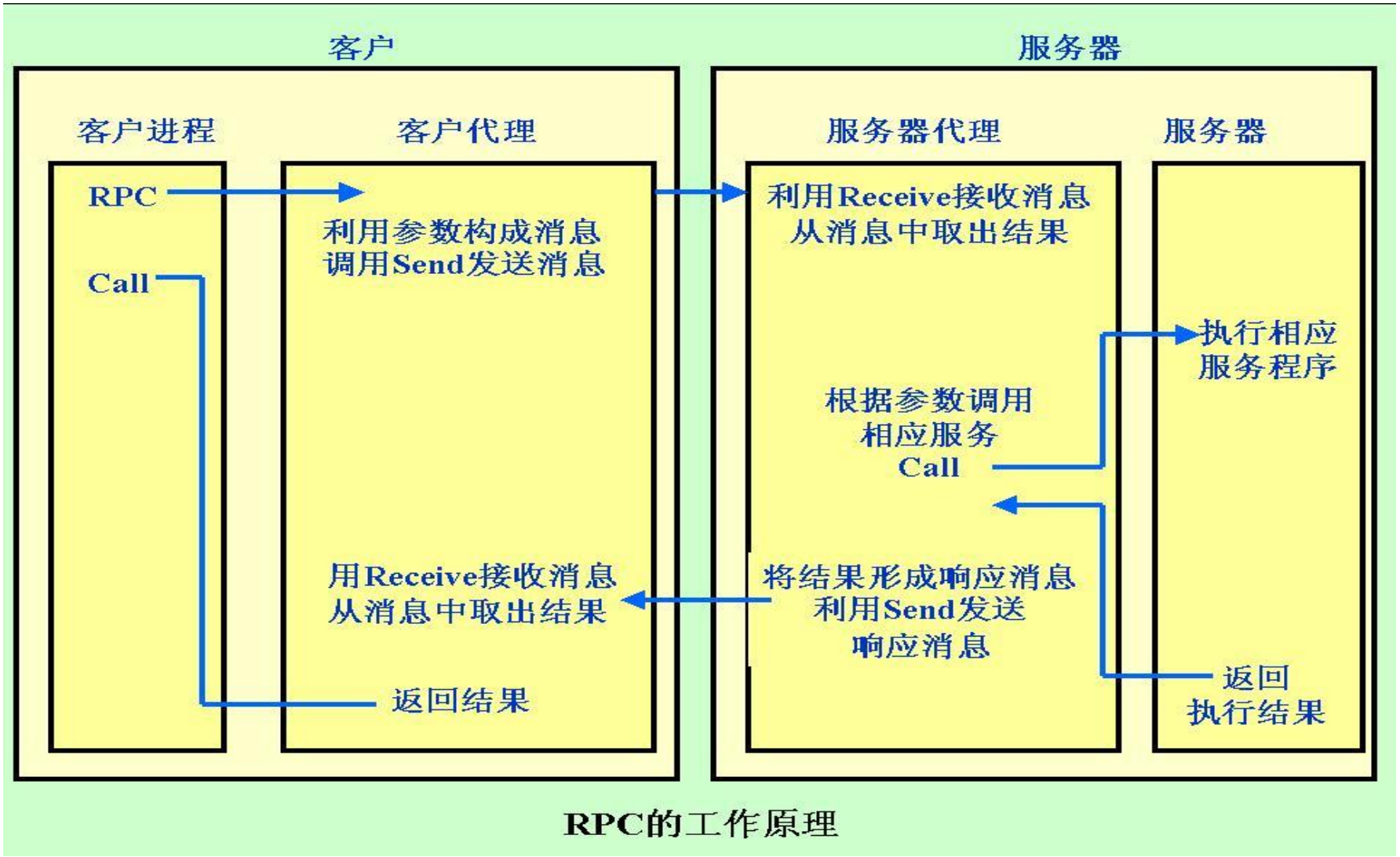
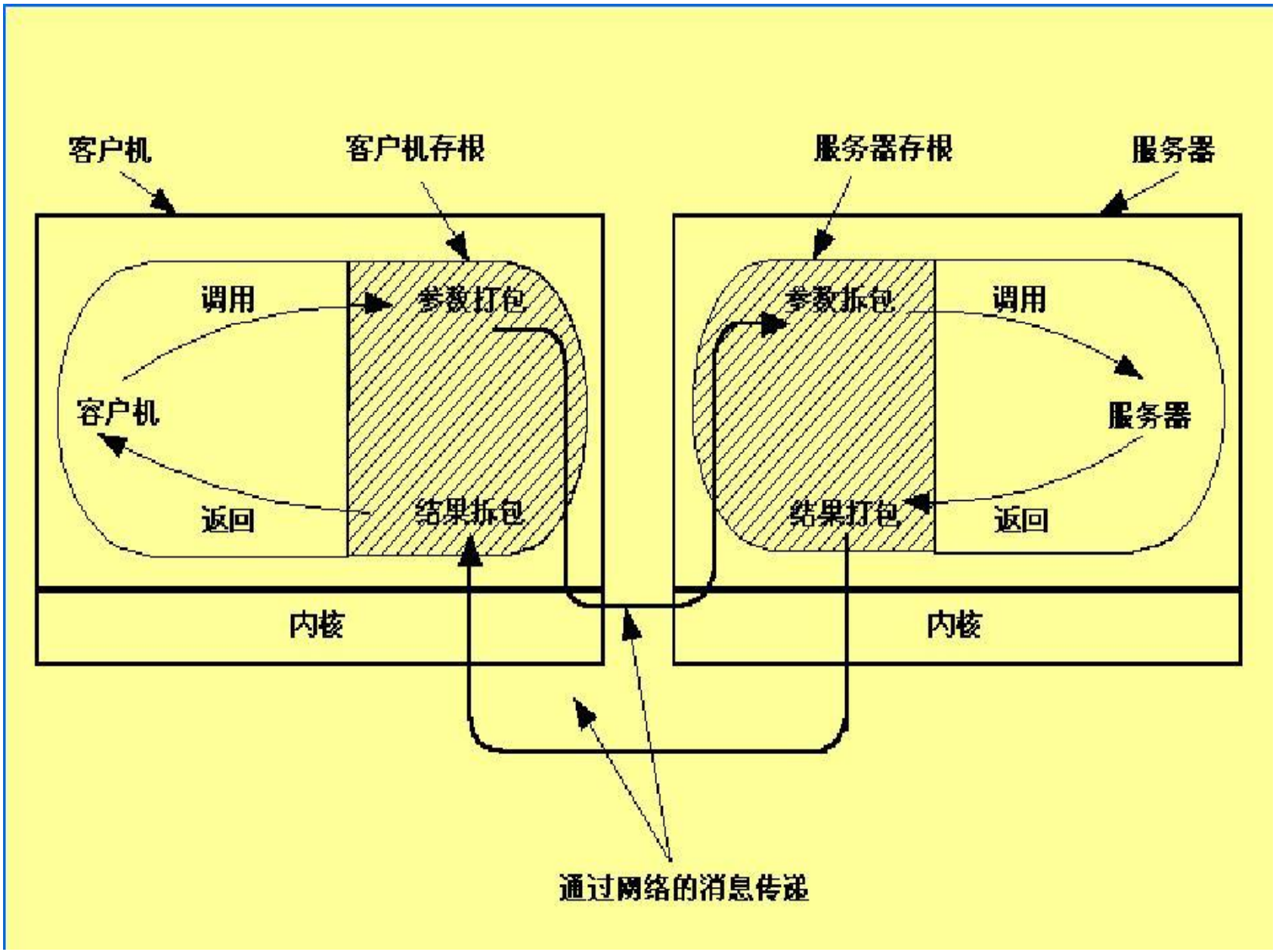


Figure 13.13 Remote Procedure Call Mechanism

远程过程调用通信模型



RPC的工作原理


客户机
客户机存根
服务器存根
服务器
调用
参数打包
参数拆包
调用
客户机
服务器
返回
结果拆包
结果打包
返回
内核
内核
通过网络的消息传递

远程过程调用的结构

RPC机制组成

- **Stub**

  客户、服务器各一个

- **binding**

  使客户能定位到相应的服务器

- 控制部分：为跟踪RPC的调用状态设置
- 传送部分：确定如何将信息从一个节点传送到另一个节点

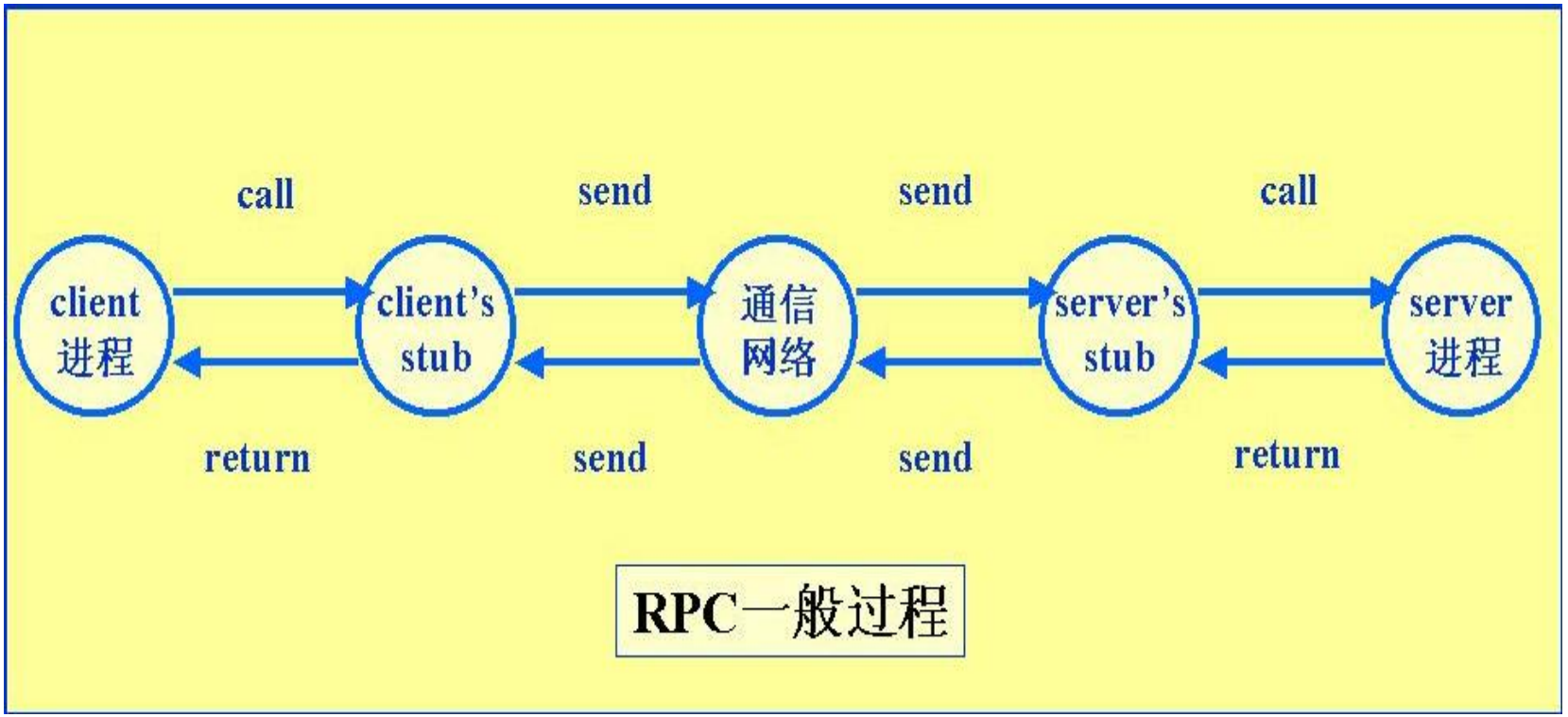
远程过程调用步骤

- 客户过程以正常的方式调用客户存根
- 客户存根生成一个消息，然后调用本地操作系统
- 客户端操作系统将消息发送给远程操作系统
- 远程操作系统将消息交给服务器存根
- 服务器存根将参数提取出来，然后调用服务器

远程过程调用步骤

- 服务器执行要求的操作，操作完成后将结果返回给服务器存根
- 服务器存根将结果打包成一个消息，然后调用本地操作系统
- 服务器操作系统将含有结果的消息发送回客户端操作系统
- 客户端操作系统将消息交给客户存根
- 客户存根将结果从消息中提取出来，返回给调用它的客户过程

远程过程调用通信模型


call
send
send
call
client
进程
client's
stub
通信
网络
server's
stub
server
进程
return
send
send
return
RPC一般过程

关键问题——参数传递方式

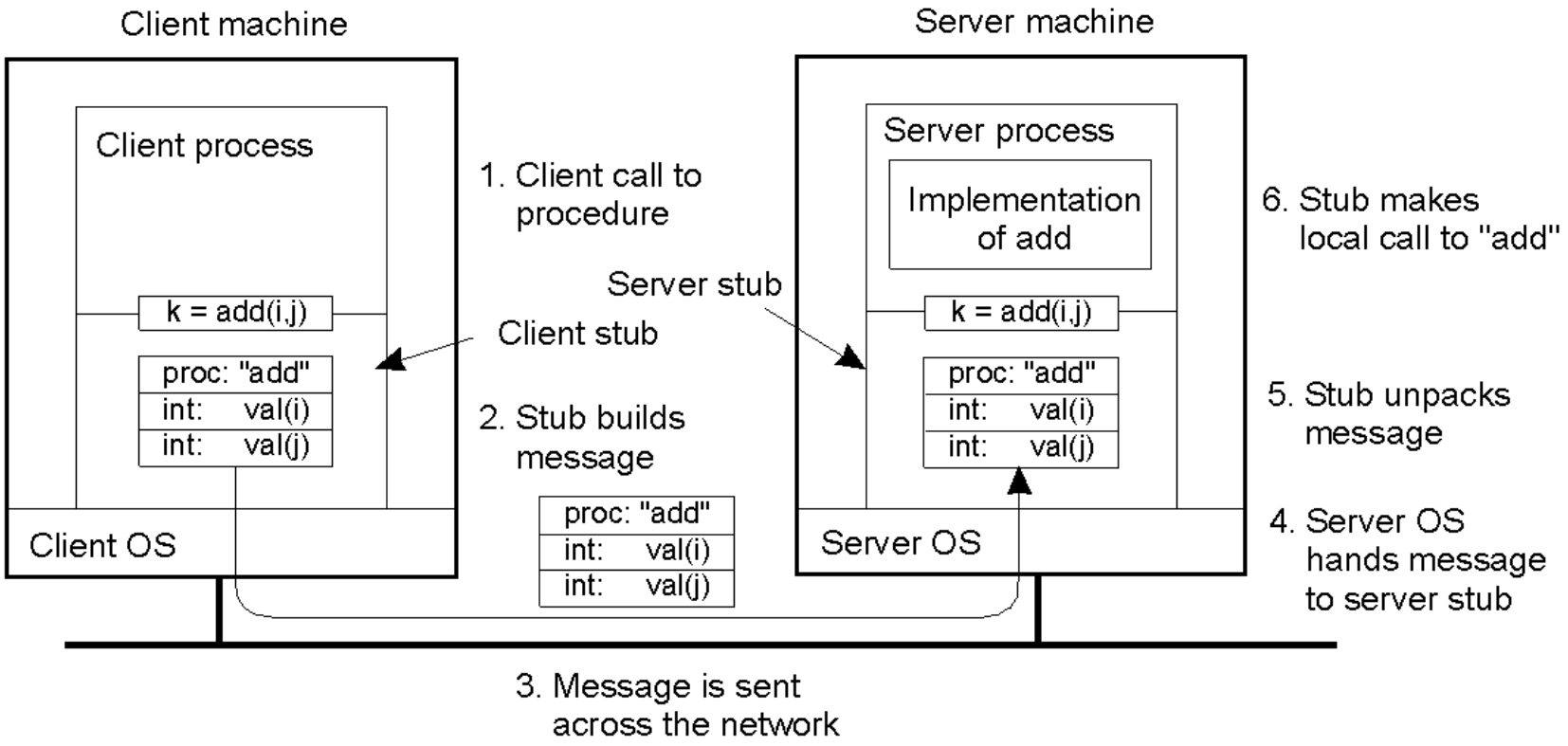
- 参数传递方式
  由语言本身的特点及语言设计者决定
- 值参和变参：传值和传递引用
- 复制/恢复参数传递机制

例子：**sum(i,j)，sum(4,7)**

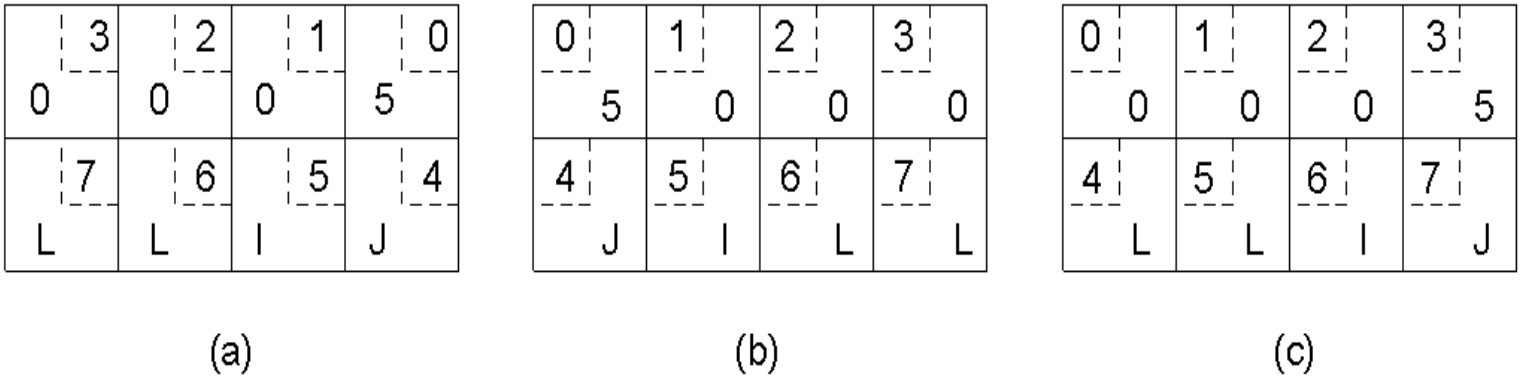
- 问题：客户机与服务器的机器类型不同
  - 数据表示不同
    Intel 386, SUN SPARC

参数传递-传递值参(1)

- 通过RPC进行远程计算的步骤



传递值参（2）



(a) Pentium上的原始消息

(b) SPARC收到的消息

(c) 进行逆转后的消息

传递引用参数

- 对于简单数组和结构：使用复制-还原代替引用调用
- 很难传递一般意义的指针：如复杂图形的指针

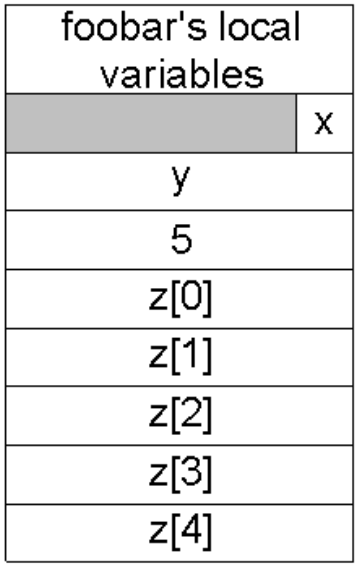
参数说明

RPC双方必须就交换的格式达成一致

- 一个过程
- 相应的消息

```
foobar( char x; float y; int z[5] )
{
   ....
}
```

(a)



(b)

关键问题——参数传递方式（续1）

- 例子：整数和字符串类型
  消息在网络上是按字节传送（位）
  参数类型（哪些是整数，哪些是字符串）

- 解决
  在消息里加参数和过程标识
  对应于有n个参数的远端过程，消息里就有n+1个域，其中一个域记录了过程标识，其他n个域记录n个参数。一旦在客户和服务器之间建立起一个关于基本数据类型表示法的协议，给定某个参数列表和消息，就可以推断出哪个字节属于哪个参数

关键问题——参数传递方式（续2）

- 问题：信息在消息包中如何表示？

- 解决方案：2种

  - 规范形式（网络标准）
  - 标识格式

关键问题——参数传递方式（续3）

- 存根过程的来源

  自动生成

给定一个服务器程序的描述和编码规则，消息的格式是唯一确定的。所以，可以让编译器自己读取服务器程序描述，生成一个客户存根过程，该过程将参数打包进正式的同一格式的消息。同样，编译器也可以生成一个服务器存根过程，对消息拆包，并调用服务器程序

作用：可以减轻程序员的负担，还可以减少出错的可能，使得系统在数据的内部表示的差异方面更加透明

关键问题——参数传递方式（续4）

- 指针传递问题

  采用复制/恢复传递机制（代替引用调用）

  对该策略的优化

关键问题——动态联编

- 所解决的问题

  客户方如何定位服务器

- 一种办法是将服务器的网络地址硬编码进客户程序

  程序的灵活性差：如果服务器移动了，或是服务器被备份了，或是接口改变了，那么无数的程序将不得不重新编译
- 为了避免这样的问题，一些分布式系统使用“动态联编”来保持客户程序和服务器程序的映射

关键问题——出现差错时的RPC语义

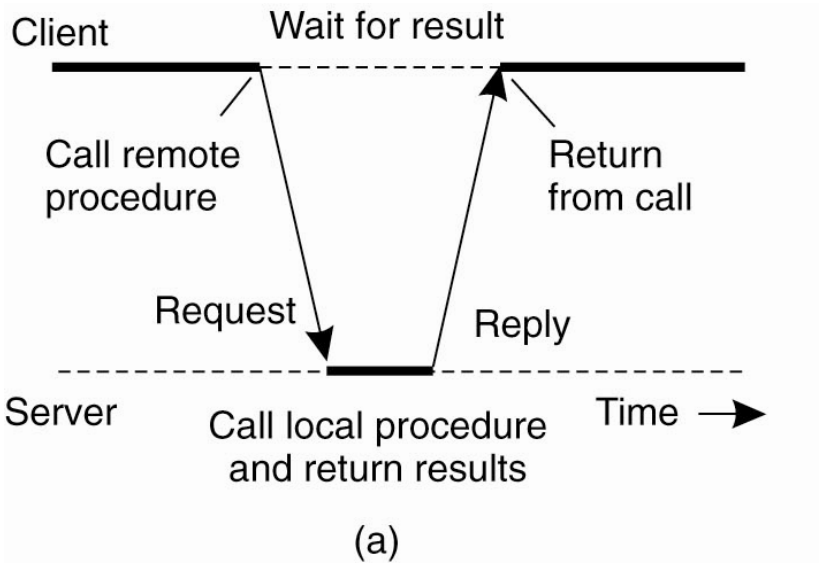
- 可能出现的错误

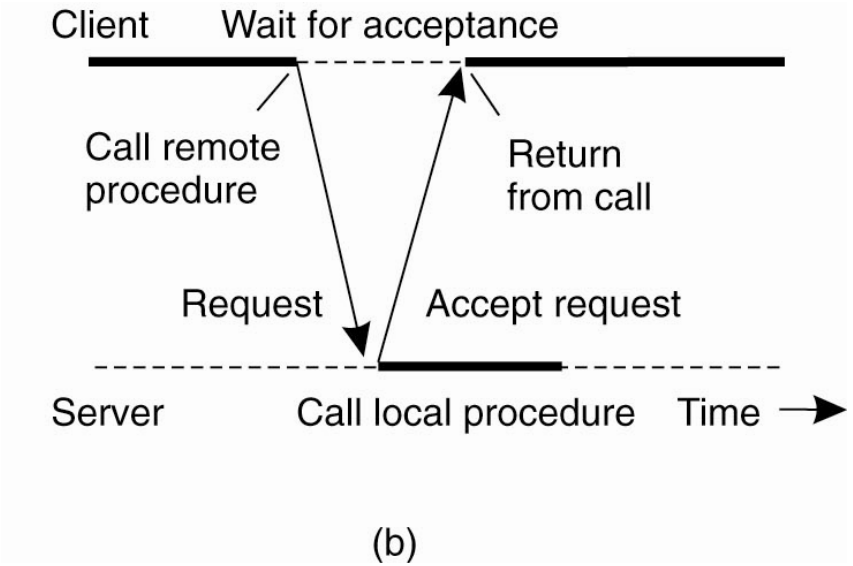
- 客户找不到服务器
- 从客户方到服务器的请求消息丢失
- 从服务器发回给客户方的应答消息丢失
- 服务器在接收了一个请求消息后突然崩溃
- 客户方在发送了一个请求消息后突然崩溃

实现的问题

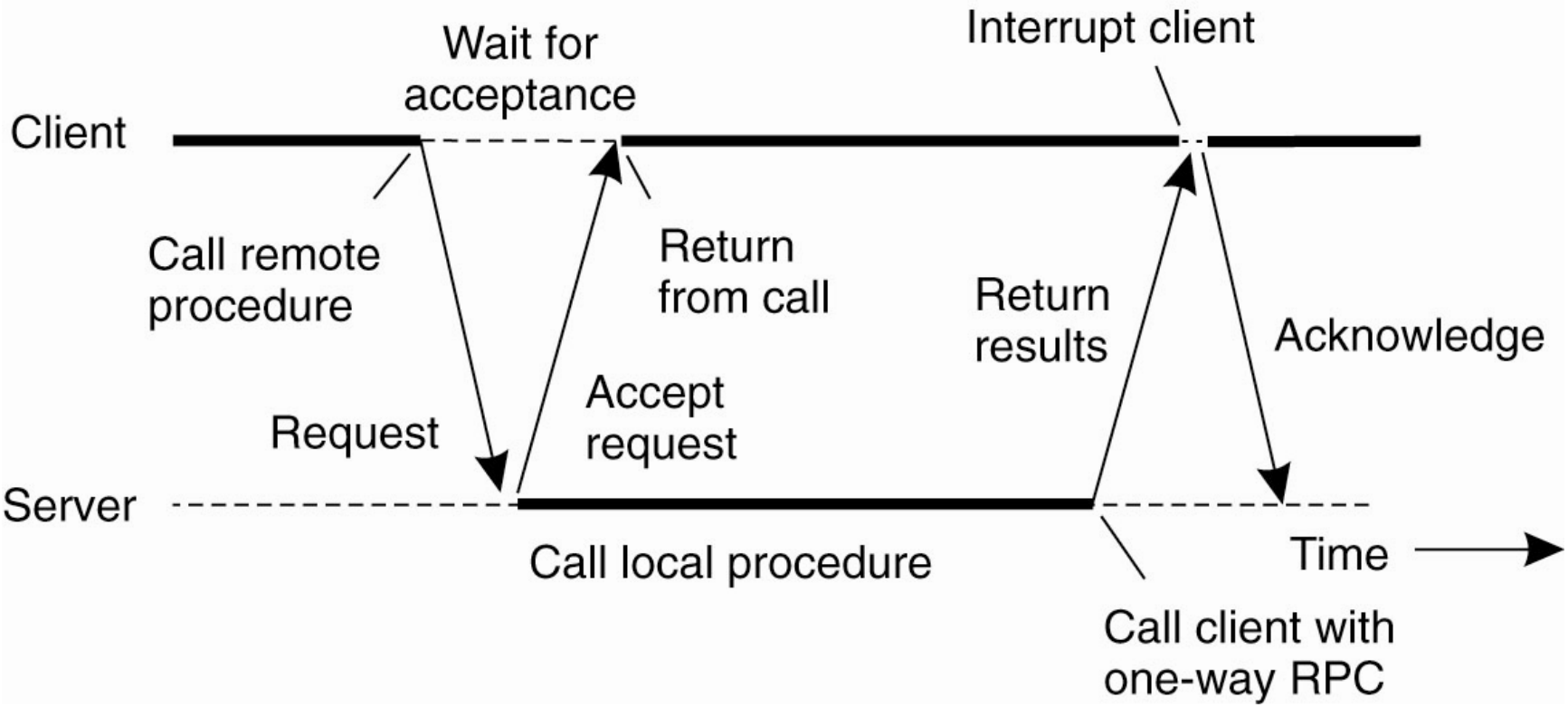
- RPC协议
- 消息确认
- 关键路径
- 拷贝
- 计时器管理

异步RPC


Client
Wait for result
Call remote procedure
Return from call
Request
Reply
Server
Call local procedure and return results
Time
(a)


Client
Wait for acceptance
Call remote procedure
Return from call
Request
Accept request
Server
Call local procedure
Time
(b)

延迟的同步RPC


Wait for
acceptance
Interrupt client
Client
Call remote
procedure
Return
from call
Return
results
Acknowledge
Accept
request
Request
Server
Call local procedure
Time
Call client with
one-way RPC

第4章　远程过程调用RPC

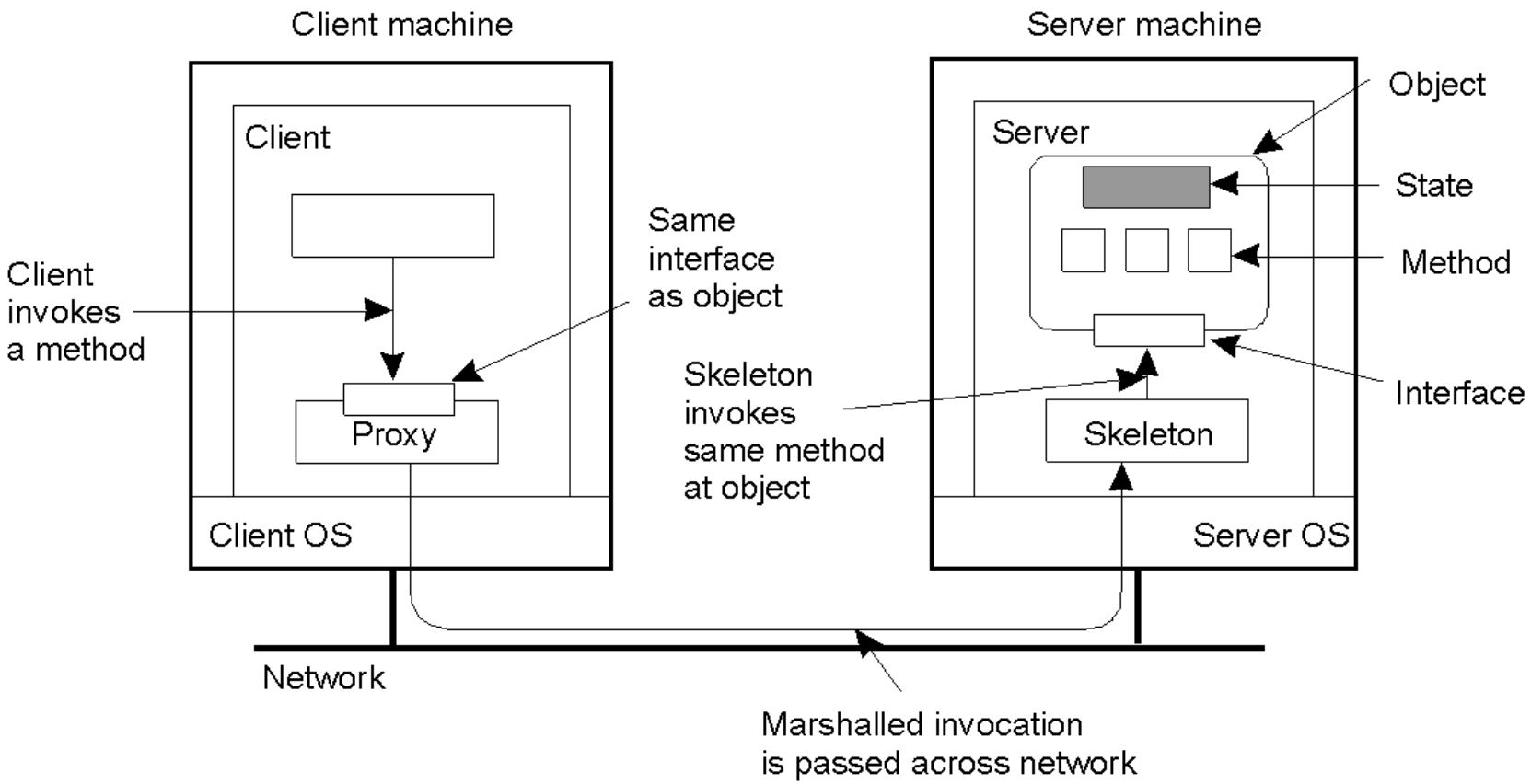
- 4.1 分层协议
- 4.2 远程过程调用
- 4.3 远程对象调用
- 4.4 面向消息的通信

4.3 远程对象调用

- 分布式对象：实现对象的接口和对象本身的分离，即接口放在一台机器上，但对象本身却驻留在另一台机器上。
- 当一个客户绑定到一个分布式对象的时候，该对象接口的一种实现——代理（proxy）被加载到客户的地址空间中。代理与RPC中的客户存根相类似。
- 代理的作用是将对对象的方法调用编组成消息，并对应答消息解编。与之相对应，在服务器上也有存根，也称为骨架（skeleton），负责调用请求解编，以及应答消息的编组。

4.3 远程对象调用

- 使用客户端代理的远程对象的一般组织结构



分布式对象分类

- 编译时对象和运行时对象
  编译时对象：直接与语言级对象相关联
  运行时对象：运行时显式生成对象
- 持久对象和暂时对象
  持久对象不依赖于服务器是否运行，服务器退出后，对象的状态存储到辅助存储器上，服务器再次启动后，对象可以接续以前状态工作。暂时对象则相反，服务器退出后，对象就销毁了。

远程对象方法调用

- 在将客户绑定到对象之后，就可以通过代理来调用对象的方法，这种远程对象方法的调用称为RMI（Remote Method Invocation）。
- 静态远程方法调用：使用预定义接口。即，在开发客户程序的时候需要知道对象的调用接口。
- 动态远程方法调用：运行时建立对象方法的调用。最常见的形式如下：

  invoke(object, method, input_parameters, output_parameters);

远程对象调用

Remote Object Reference: Identifier of a remote object that is valid throughout the system.

远程对象引用：通过系统确认的有效的远程对象标识。

- Remote object references is passed in an invocation message.
- 远程对象引用是通过调用消息传递。
- Remote object references must be unique in space and time.
- 远程对象引用是单独的时间和空间。
  - A remote object reference cannot be reused even if the corresponding object is deleted.
  - 远程对象引用即使相应对象被删除，也不能被重用。

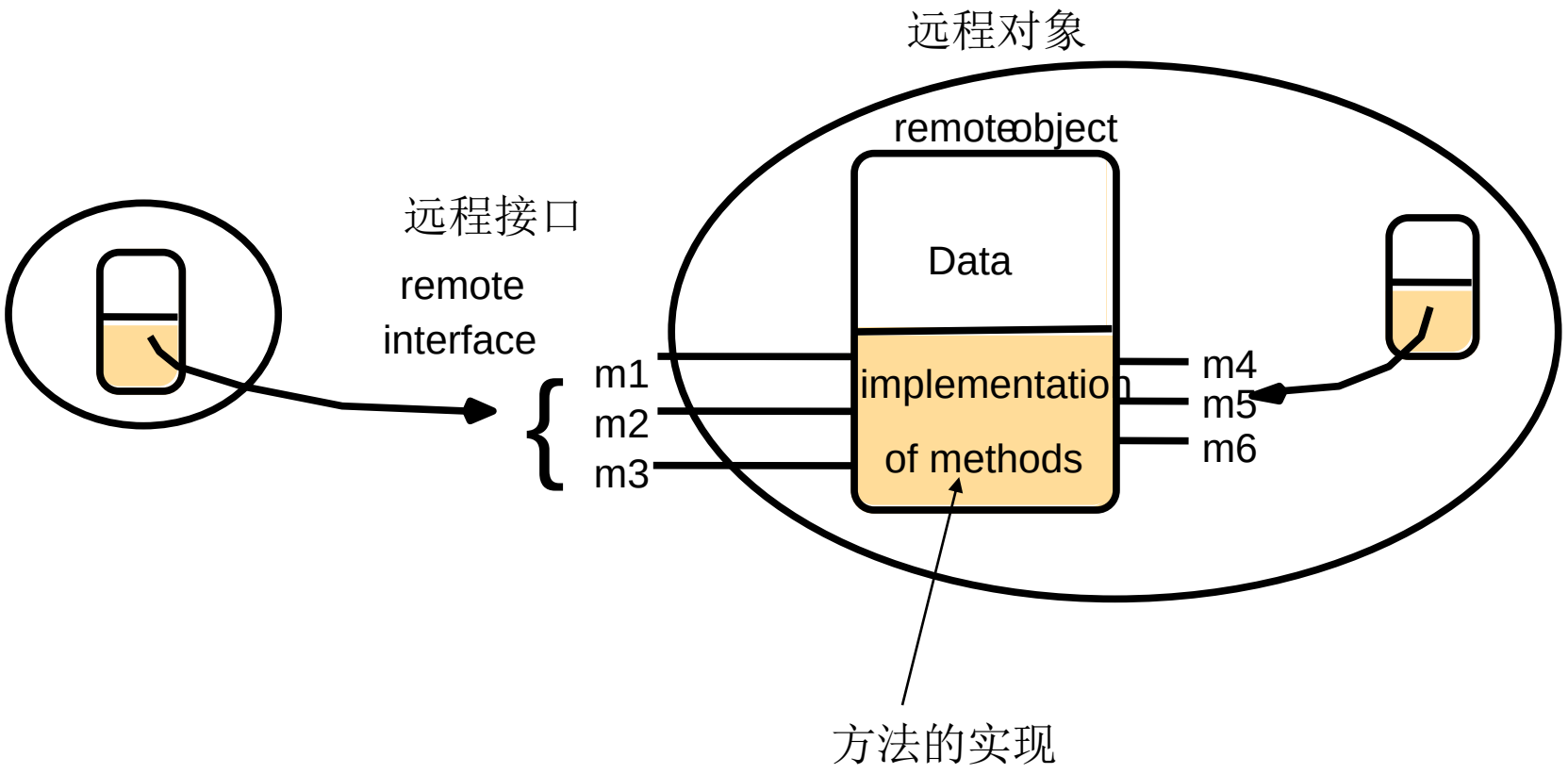
Remote Object Reference (1)

- The reference contains the internet address, port number, time of creation and local object number.
- 引用包含internet地址，端口号，创建时间和本地对象号。
- The local object number is incremented each time an object is created.
- 当每次对象被创建的时候，本地对象号是增量。
- It may be necessary to relocate a remote object.
- 重新部署远程对象是必须的。
- A remote object reference should not be used as the address of the remote object.
- 远程对象引用不能被作为远程对象地址使用。

Remote Object Reference (2)
远程对象引用

32 bits	*32 bits*	*32 bits*	*32 bits*	
Internet address	port number	time	object number	interface of remote object

Figure 5.4
A remote object and its remote interface
远程对象及远程接口



The Object Model (1)对象模块

In a distributed object system, an object‘s data are accessible *only* via its methods.

分布式对象系统，一个目标的数据只能经过这种方式被访问。

- **Encapsulation** of the data数据的封装
- This is not completely enforced by some object oriented languages like Java and C++.
- 面向对象的语言Java和C++并不完全被使用。

- Objects are accessed via **object references**.
- 对象通过对象引用被访问。
  - Object references can be assigned to variables and passed as arguments.
  - 对象引用可以被赋予变量和传递参数。
  - A **remote object reference** is an identifier that can be used throughout the distributed system to refer to a particular unique remote object.
  - 远程对象引用：是一个可以用于整个分布式系统的标识符，用于指向某个唯一的远程对象。
  - The representation of a remote object reference is different from that of local object references.
  - 远程对象引用的表示不同于本地对象的引用。
  - The object whose method is invoked is called **receiver** or **target** of an invocation.
  - 调用的对象被叫做调用的接收者或目标。

The Object Model (2)

- An **action** is a chain of related method invocations, each of which eventually returns. It is initiated by an object invoking another object.
  动作是一个相关方法调用的链，每一个最终被返回。他被一个目标调用另一个目标而启动。
- An invocation of a method may result in
  - a change of the state of the receiver
  - further invocations of methods in other objects.
  - 调用方式会导致：
    - 接收者状态的改变
    - 在其它对象中，有更远的调用方法。

- **Exceptions** provide a clean way to handle error conditions without complicating the code.
- 异常：提供一个清晰的方法处理错误条件，无须使编码复杂化。
  - Upon an error or an unexpected condition a block of code *throws an exception*. 错误或不期望的条件，代码块会除去异常。
  - Control passes to another block of code that *catches the exception*. 控制传递到另一个代码段可以捕捉异常。

- Control does not return to the first code block.
- 控制不能返回到第一代码段。
- In a distributed system, additional exceptions are necessary.
- 在分布式系统中，附加的异常处理是需要的。
  - Timeout of a network connection, remote process crash, …
  - 例如，网络连接的间隙，远程调用的失败，。。。

Access Transparency访问的透明性

There are several necessary conditions to achieve access transparency for RPC and RMI.有一些必须的条件可以成功地透明的访问RPC和RMI。

- All calls to marshalling and message passing must be hidden from the programmer of a procedure.信号编组和信息传递的调用必须被过程程序隐藏。
- In RMI, the task of locating and contacting a remote object must be hidden as well.
- 在远程方法调用中，定位和联系远程对象的工作也必须被隐藏。

- The semantics of a remote procedure call and of a local procedure call must be identical.远程过程调用和本地过程调用的语义必须是同样的。

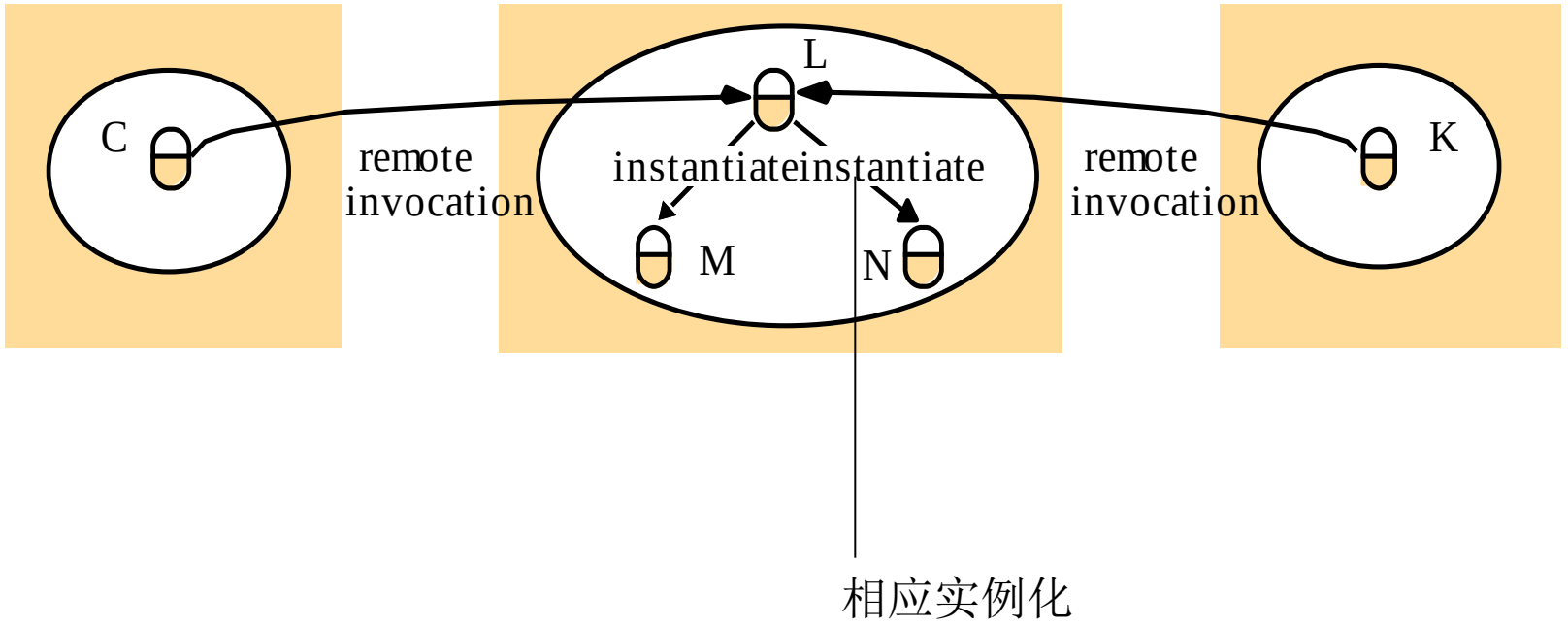
However, there are some differences that cannot be hidden.然而，有一些不同点不能被隐藏。

- Differences in the invocation latency调用反应时间的不同。
- Differences in failure types错误类型的不同。

Those differences between remote and local invocations are expressed in the interface

远程和本地调用的不同在接口应被表示。

Figure 5.5 Instantiation of remote objects
远程对象的实例化



第4章　远程过程调用RPC

- 4.1 分层协议
- 4.2 远程过程调用
- 4.3 远程对象调用
- 4.4 面向消息的通信

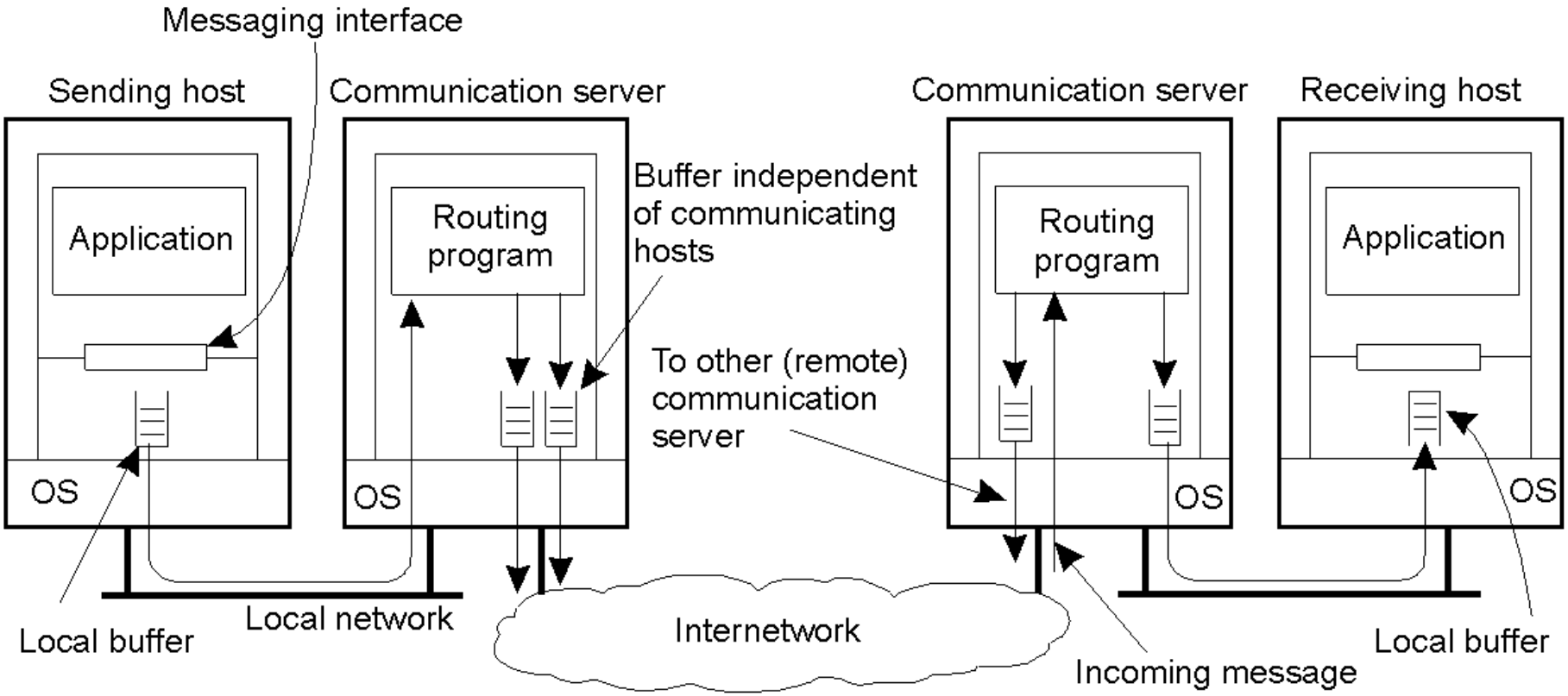
4.4 面向消息的通信

当远程过程调用和远程对象调用不适用时，需要面向消息的通信。

- 消息中的持久性和同步性
- 面向消息的暂时通信
- 面向消息的持久通信

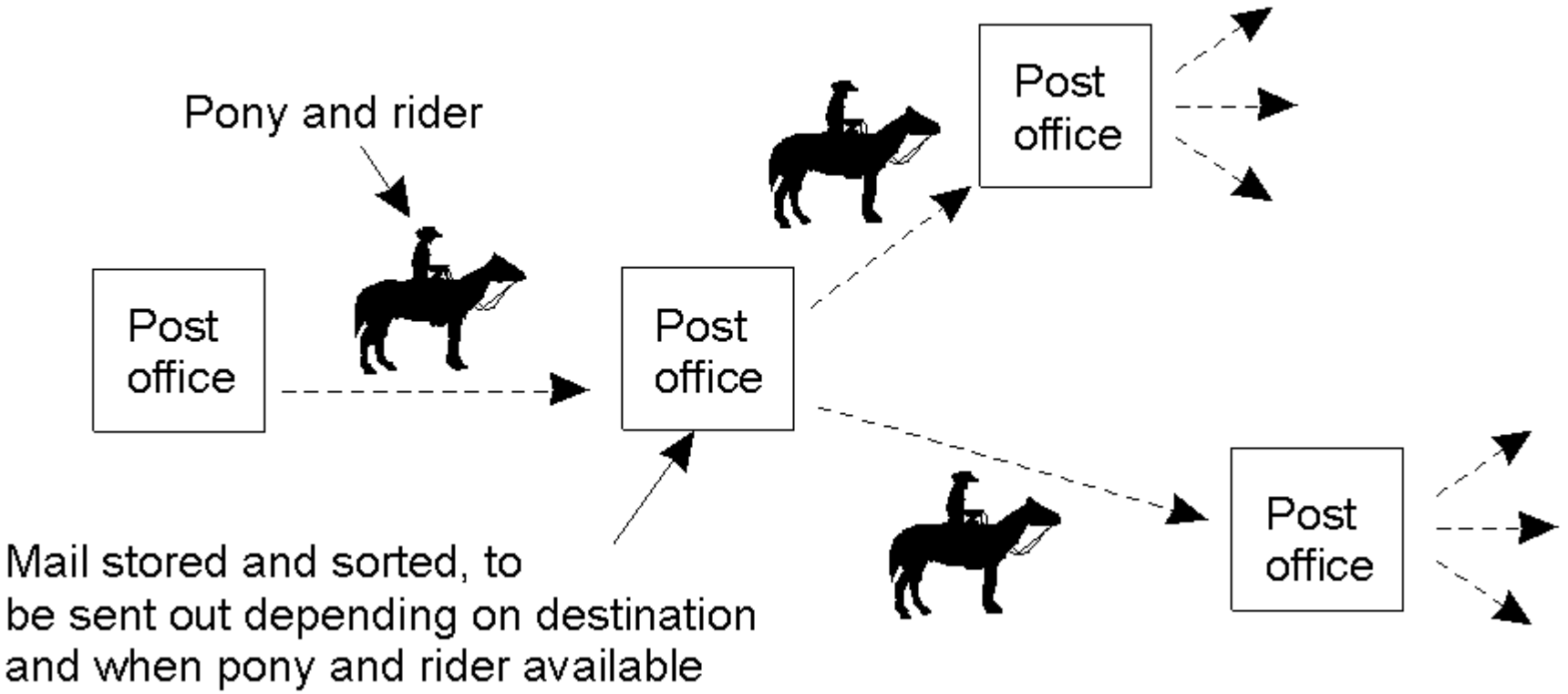
消息中的持久性和同步性(1)

- 通信系统的通用结构



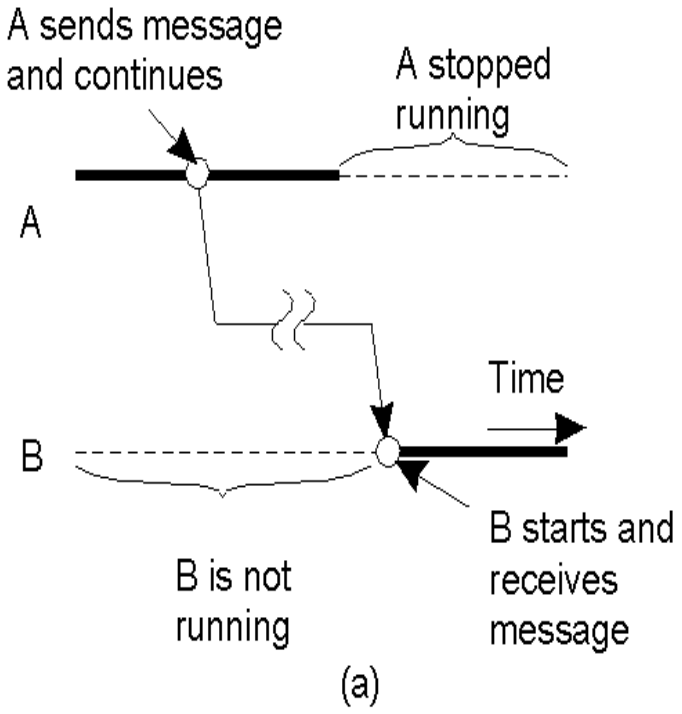
消息中的持久性和同步性(2)

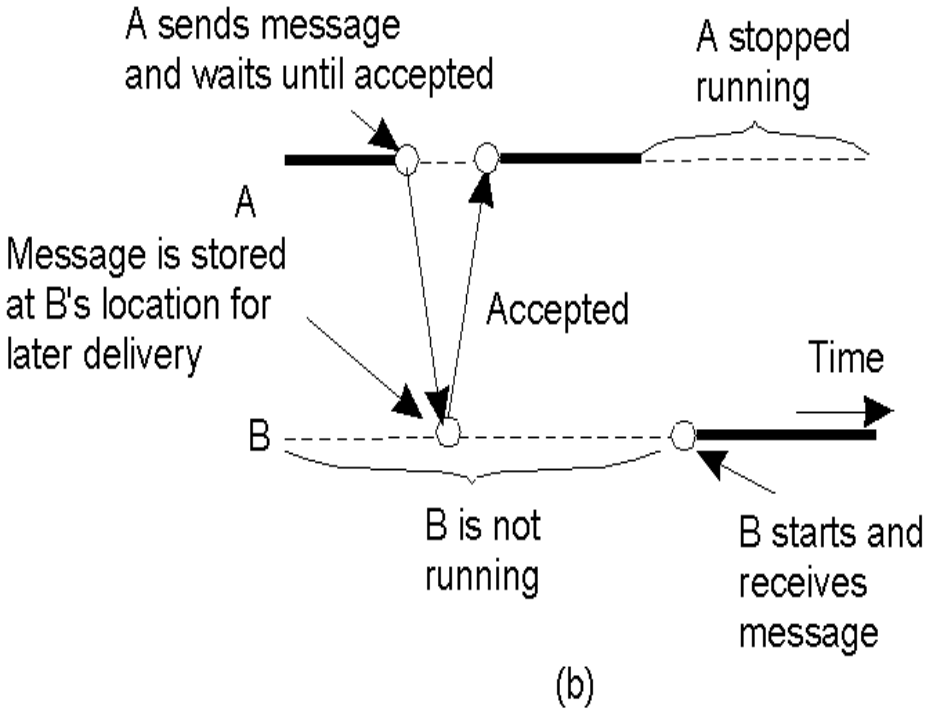
- 驿马快递时代使用的持久通信：通信双方不必保持运行
- 在暂时通信中，通信系统只在发送者和接收者运行时存储消息



消息中的持久性和同步性(3)

1. 持久异步通信：提交消息后立即执行其他程序，电子邮件
2. 持久同步通信：提交消息后会被阻塞，直到消息已到达并存储在接收主机





面向消息的暂时通信

不提供消息的中介存储，实时性要求高（几秒甚至几毫秒）

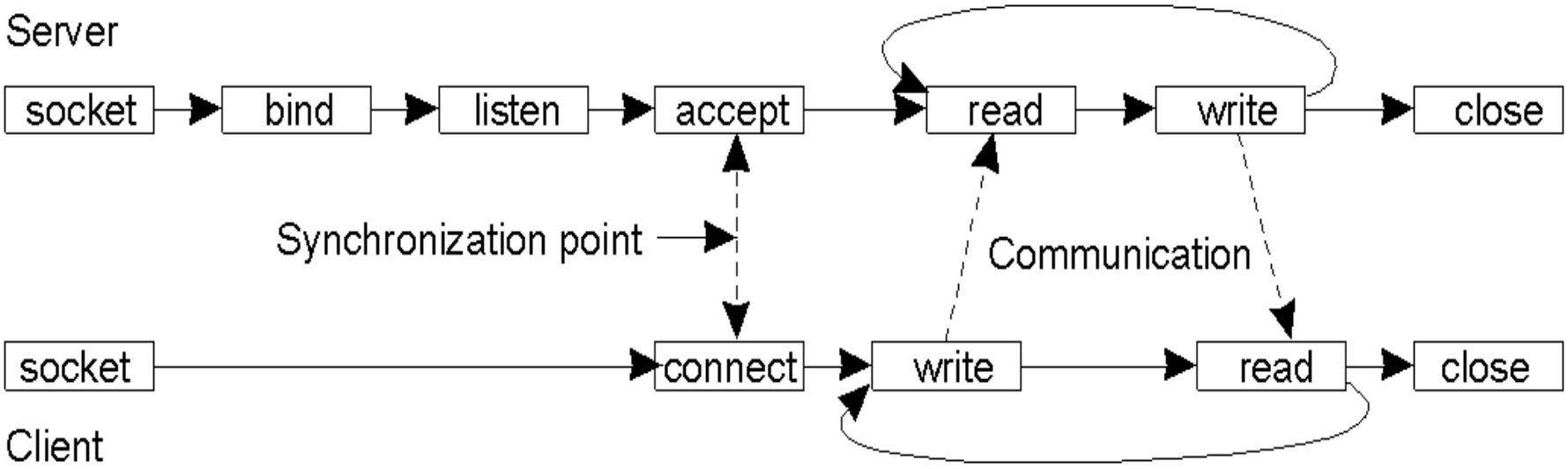
- **Berkeley Sockets**
- **Message-Passing Interface**

Berkeley Sockets (1)

- TCP/IP套接字原语

原语	含义
Socket	创建新的通信端点
Bind	将本地地址附加（attach）到套接字上
Listen	宣布已准备好接受连接
Accept	在收到连接请求前阻塞调用方
Connect	主动尝试建立连接
Send	通过连接发送数据
Receive	通过连接接收数据
Close	释放连接

Berkeley Sockets (2)



- 使用套接字的面向连接通信模式

The Message-Passing Interface (MPI)

- 适用于高速互联网络，为高性能并行应用程序设计，有更高级的特性：缓冲、同步
- MPI中的基本消息传递原语

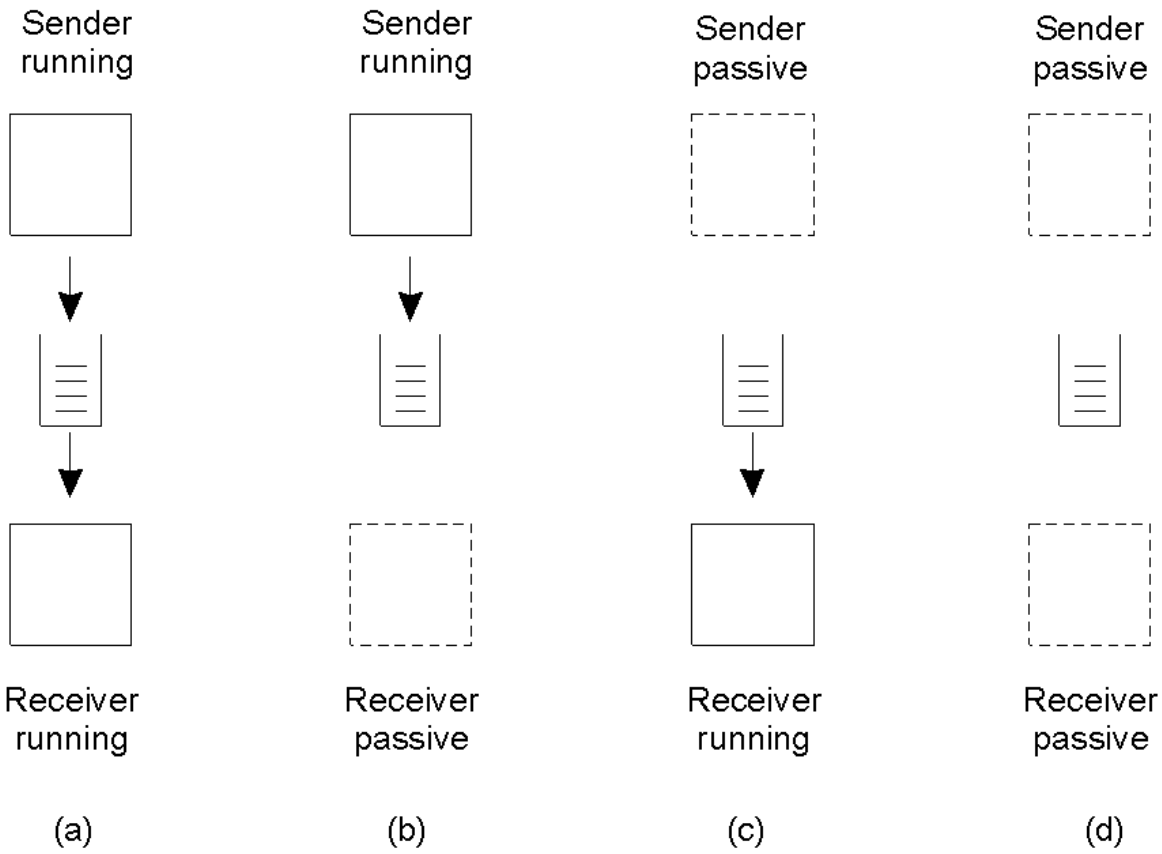
原语	含义
MPI_bsend	将消息追加到本地发送缓冲区中
MPI_send	发送消息，并等待到消息复制到本地或远程缓冲区为止
MPI_ssend	发送消息，并等待到对方开始接收为止
MPI_sendrecv	发送消息，并等待到受到应答消息为止
MPI_isend	传递要发送消息的引用，然后继续执行
MPI_issend	传递要发送消息的引用，并等待到对方开始接收为止
MPI_recv	接受消息，如果不存在的等待的消息则阻塞
MPI_irecv	检查是否有输入的消息，但无论有无消息都不会阻塞

面向消息的持久通信

- 提供消息的中介存储，实时性要求低（几分钟）
- **Message-Queuing system** 消息队列系统
  - 应用程序通过在特定队列中插入消息来进行通信
  - 只保证发送者的消息最终被放置到接收者的队列中，并不保证时间，也不保证消息被读取。

Message-Queuing Model (1)

- 使用队列的松耦合通信的四种组合方式



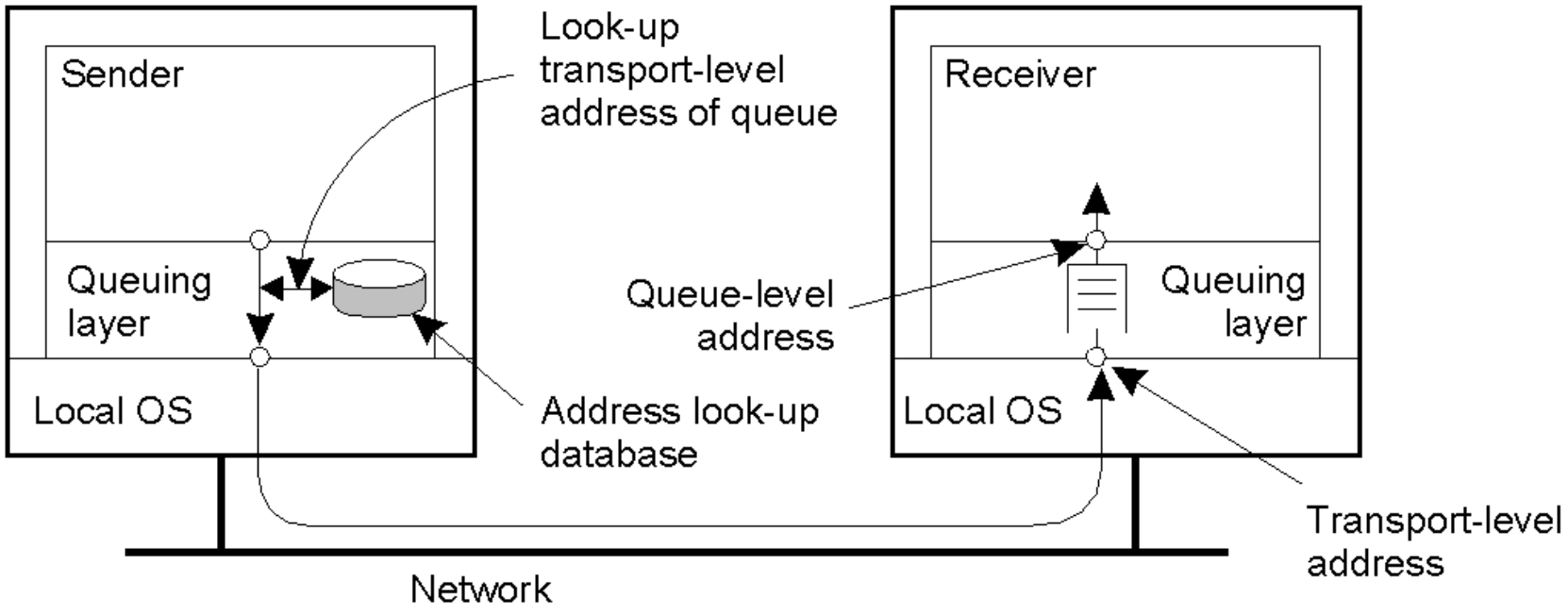
Message-Queuing Model (2)

- 消息队列系统中队列的基本接口

原语	含义
Put	将消息追加到指定队列
Get	调用进程阻塞，直到指定队列非空，取出第一个消息
Poll	察看指定队列的消息，取出第一个消息，不阻塞调用进程
Notify	注册一个处理程序，在有消息进入指定队列时调用该处理程序

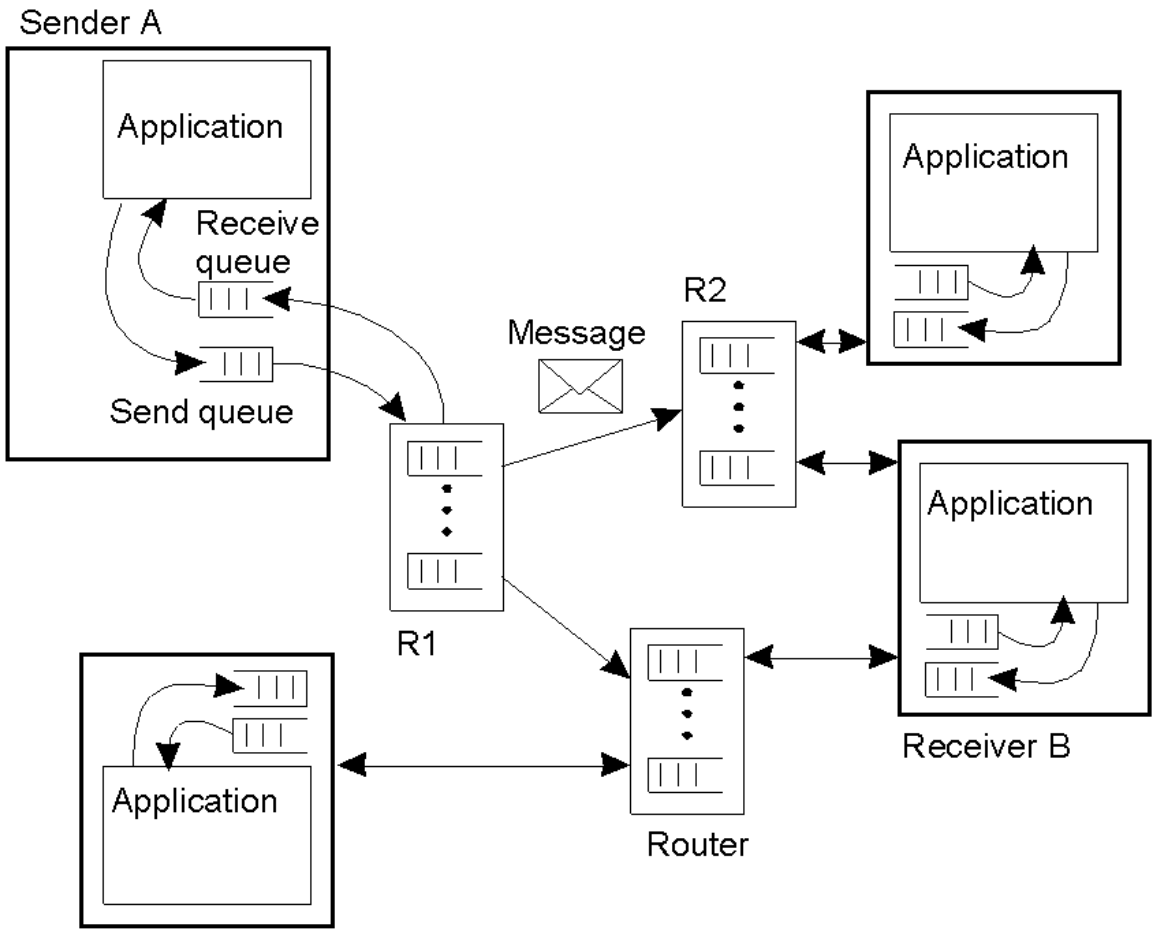
消息队列系统的通用体系结构(1)

- 源队列，目的队列
- 队列级编址与网络级编址间的关系

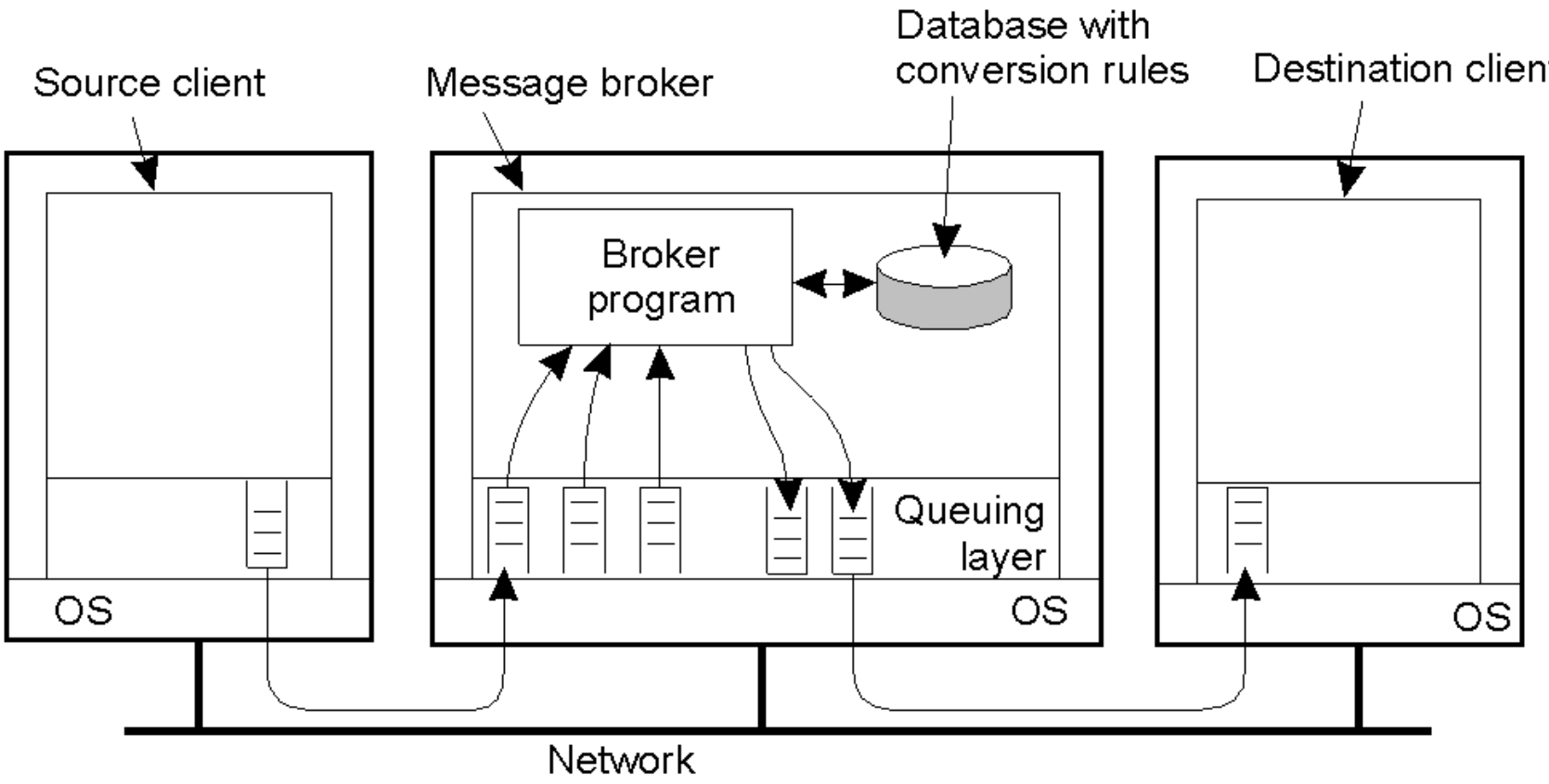


消息队列系统的通用体系结构(2)

- 含有路由器的消息队列系统的通用体系结构



（消息转换器）Message Brokers



小 结

- 分层协议
- 远程过程调用
- 远程对象调用
- 面向消息的通信

高级操作系统—通信思考题

- 任务

  使用socket编程实现一个简单的文件服务器。客户端程序实现put功能(将一个文件从本地传到文件服务器)和get功能(从文件服务器取一远程文件存为本地文件)。客户端和文件服务器不在同一台机器上。

put [-h *hostname*] [-p *portname*] *local_filename remote_filename*

get [-h *hostname*] [-p *portname*] *remote_filename local_filename*

- 编程语言： C, C++, or Java, or ~~~