

# 第四章 分布式系统通信

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顶层细节：

- 0, 1 的电压表示

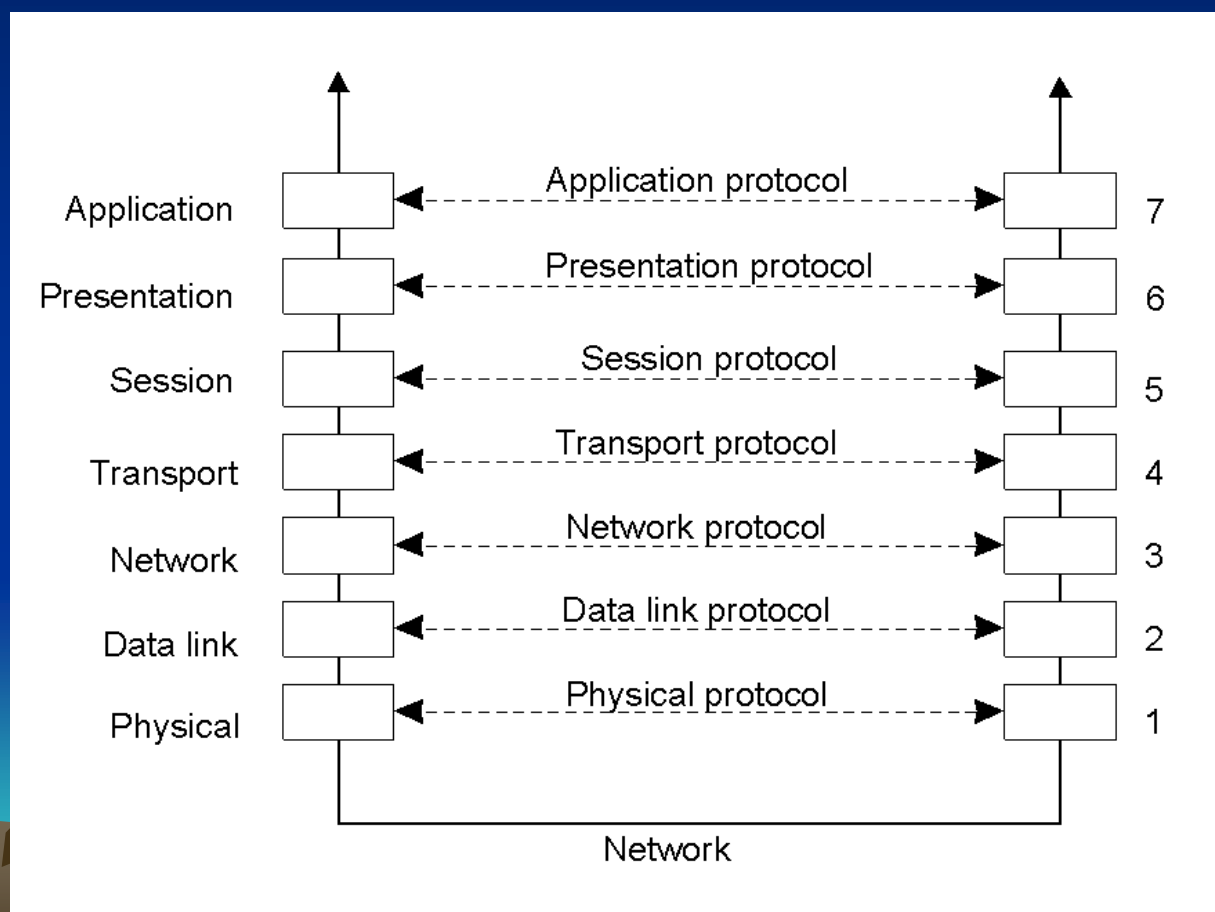
- 消息的结束位

- 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

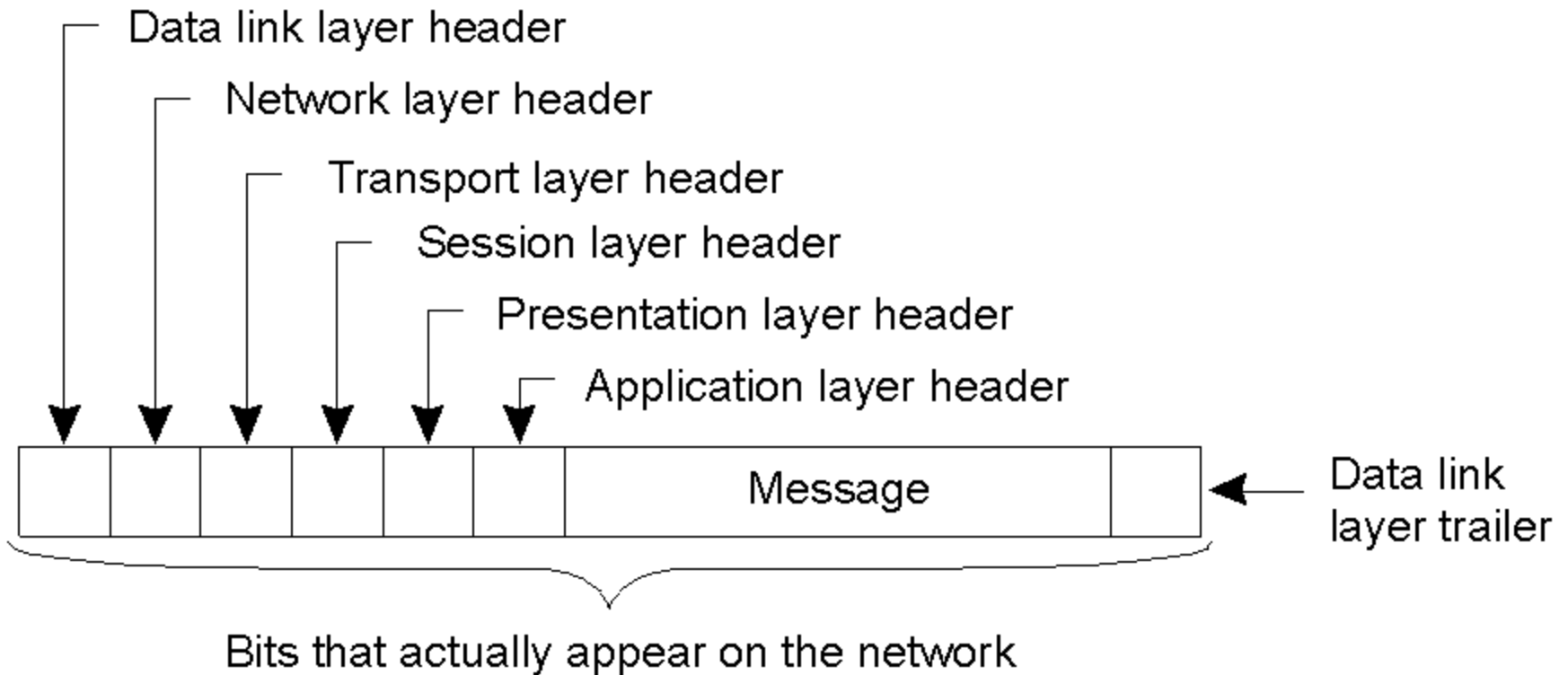
- 面向连接的协议：电话

- 无连接的协议：邮箱



# 层次协议(2)

- 在网络上传输的典型消息



# 远程过程调用

## ( Remote Procedure Call )

**RPC**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。

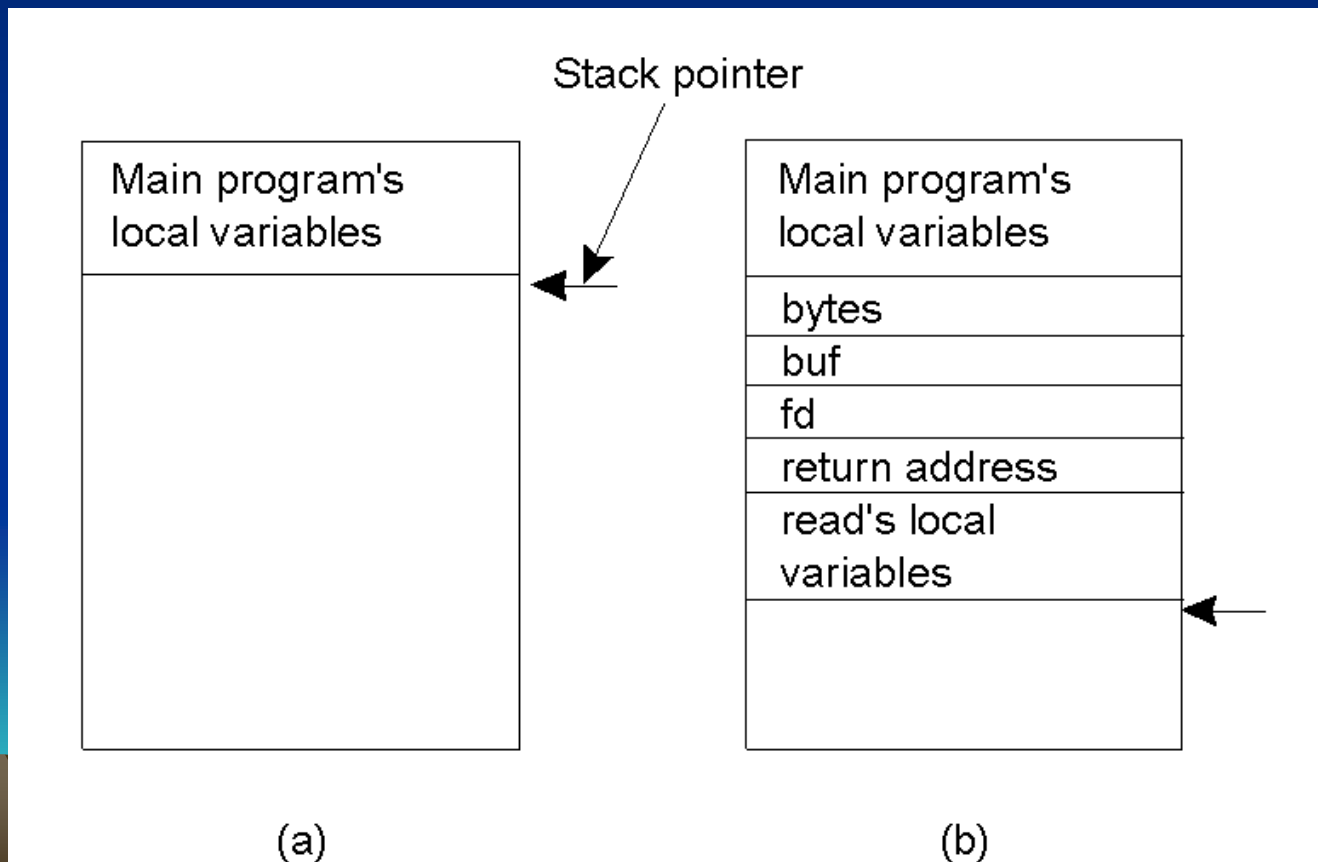
- 常规过程调用
- 客户存根和服务器存根
- 参数传递



# 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



# 客户存根和服务端存根

- 客户和服务端间的RPC原理

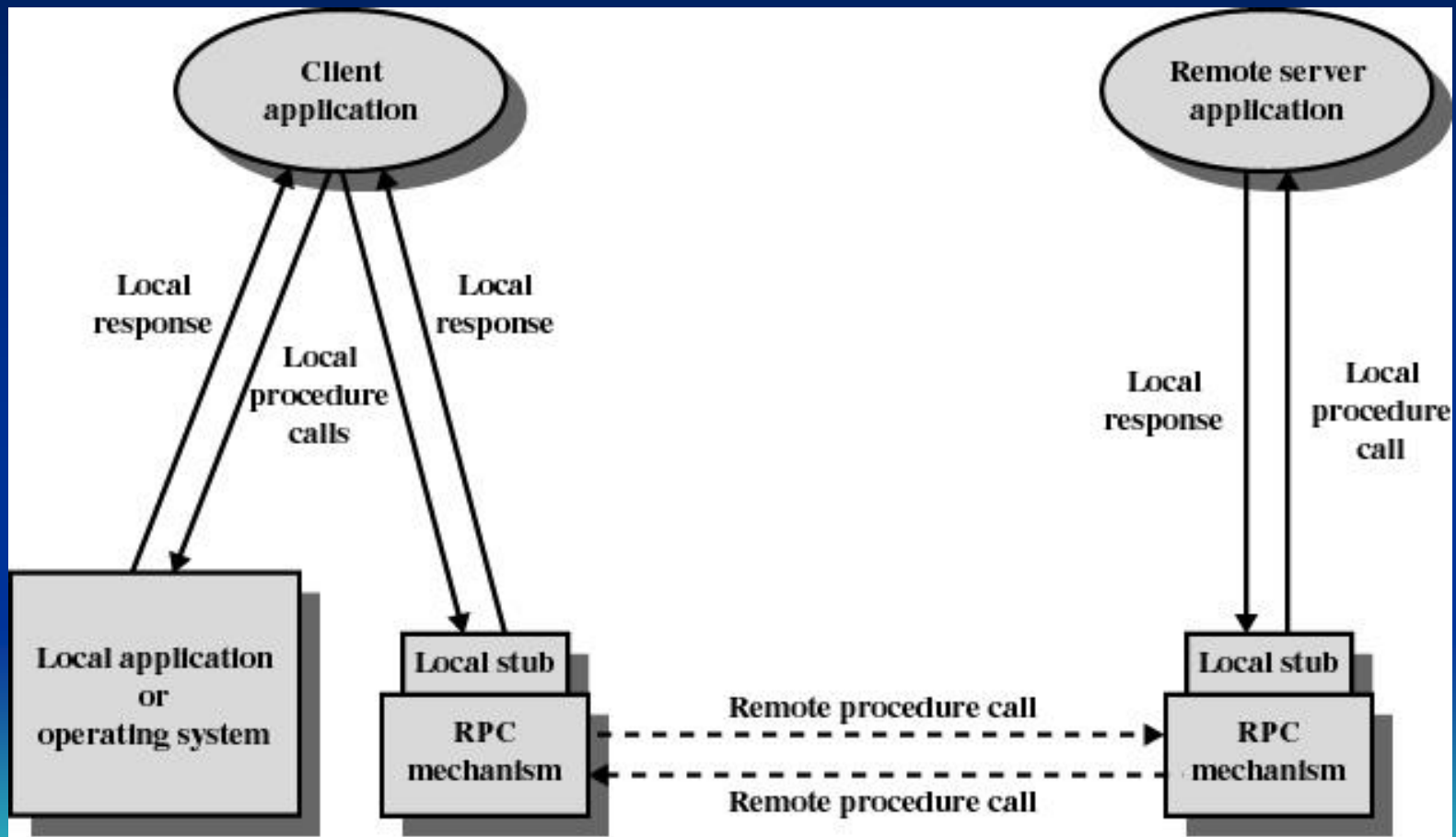



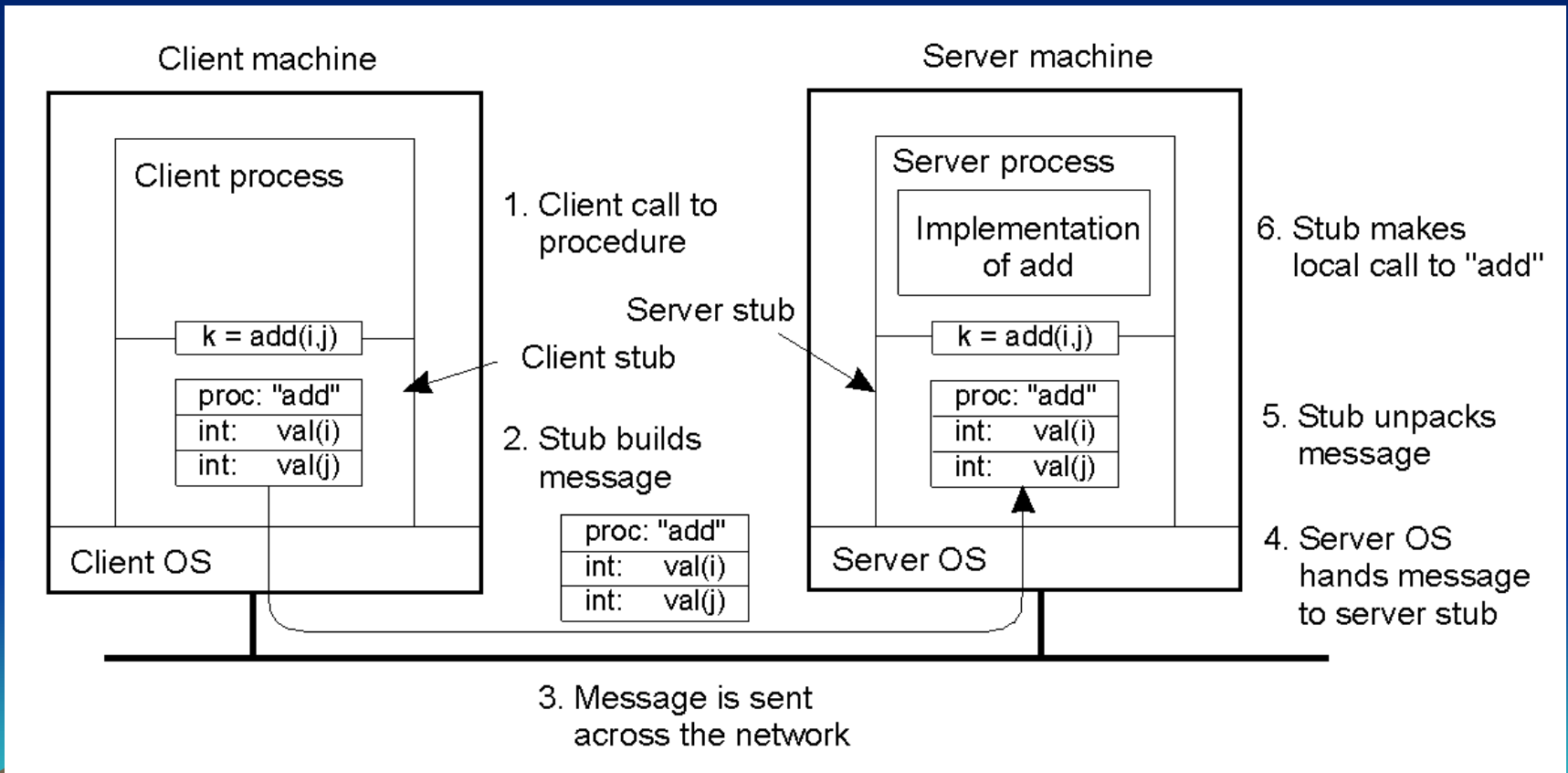
Figure 13.13 Remote Procedure Call Mechanism

# 远程过程调用步骤

1. 客户过程以正常的方式调用客户存根
  2. 客户存根生成一个消息，然后调用本地操作系统
  3. 客户端操作系统将消息发送给远程操作系统
  4. 远程操作系统将消息交给服务器存根
  5. 服务器存根将参数提取出来，然后调用服务器
  6. 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
  7. 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
  8. 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
  9. 客户端操作系统将消息交给客户存根
  10. 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
- 

# 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)

3	2	1	0
0	0	0	5
7	6	5	4
L	L	I	J

(a)

0	1	2	3
5	0	0	0
4	5	6	7
J	I	L	L

(b)

0	1	2	3
0	0	0	5
4	5	6	7
L	L	I	J

(c)

**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

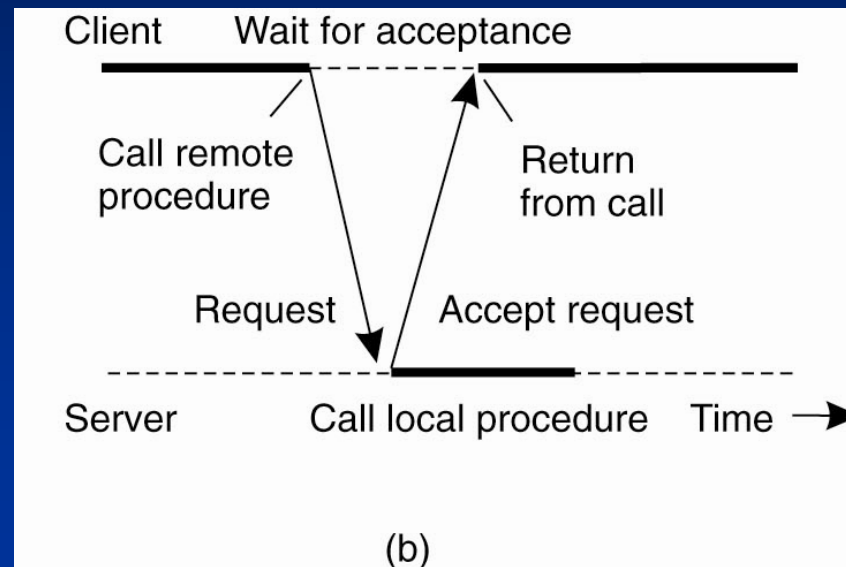
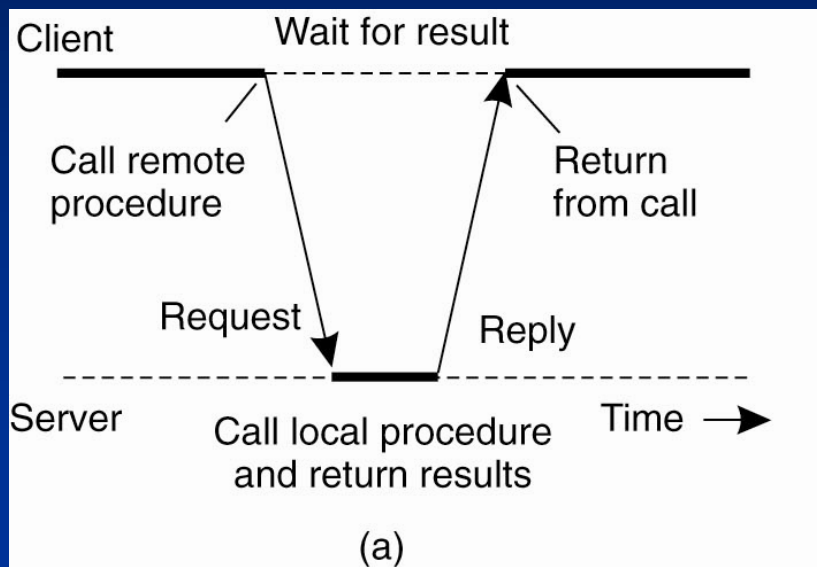
```
foobar( char x; float y; int z[5] )  
{  
    ....  
}
```

(a)

foobar's local variables	
	x
y	
5	
z[0]	
z[1]	
z[2]	
z[3]	
z[4]	

(b)

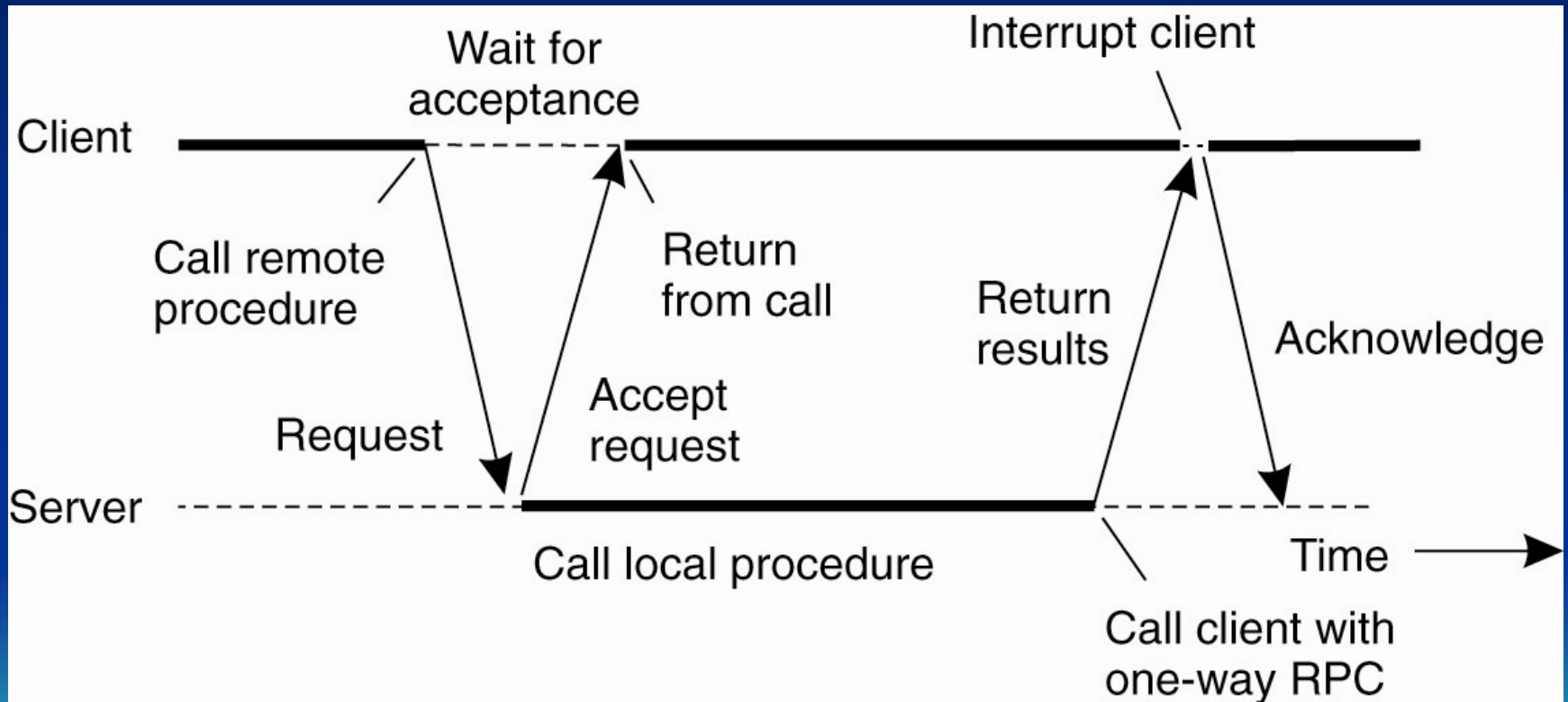
# 异步RPC



传统RPC交互过程

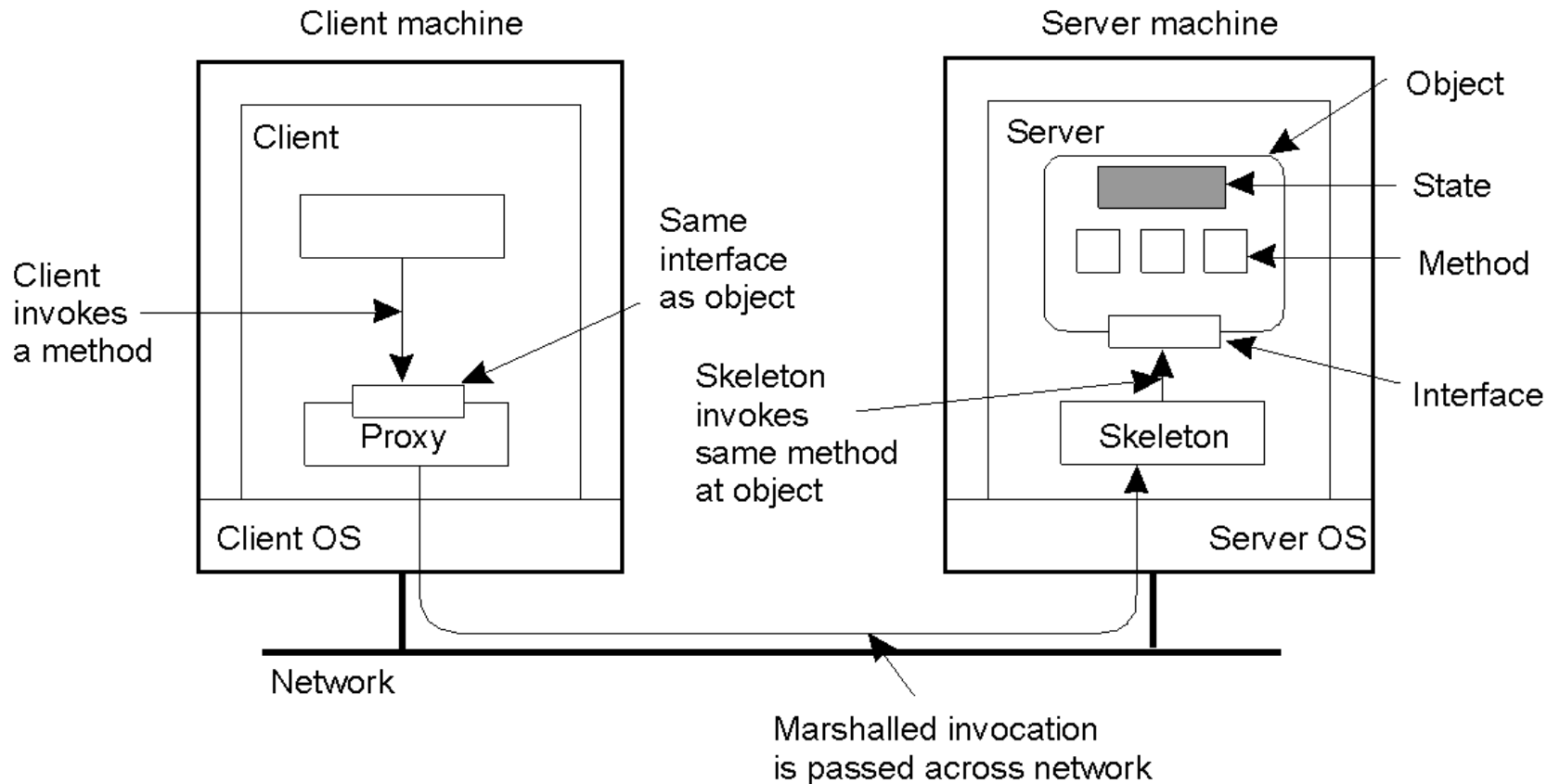
异步RPC交互过程

# 延迟的同步RPC



# 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



# 面向消息的通信

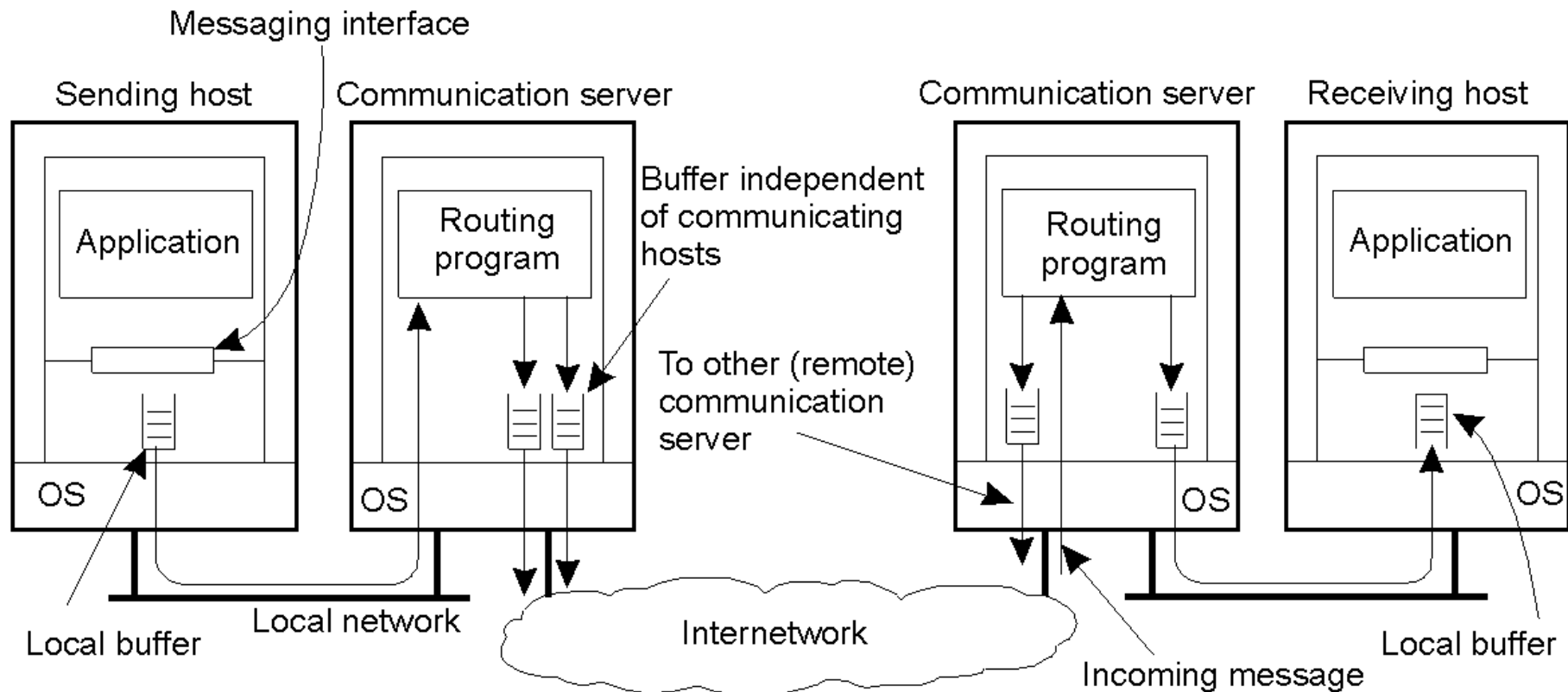
当远程过程调用和远程对象调用不适用时，需要面向消息的通信。

- 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)

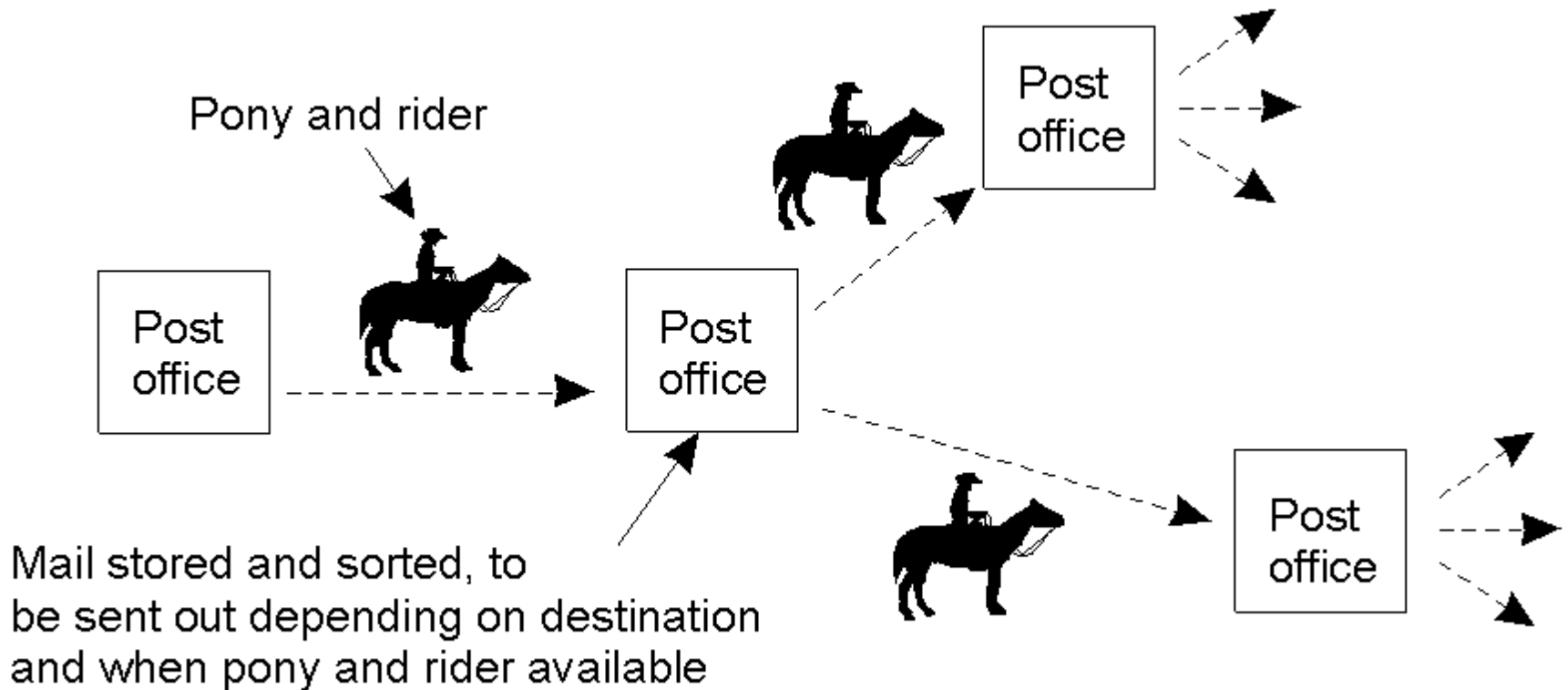
- 通信系统的通用结构





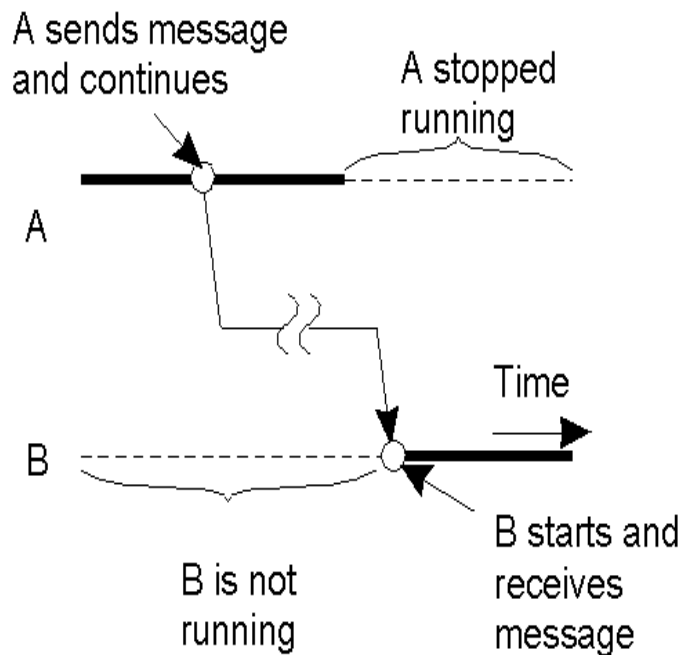
## 消息中的持久性和同步性(2)

- 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

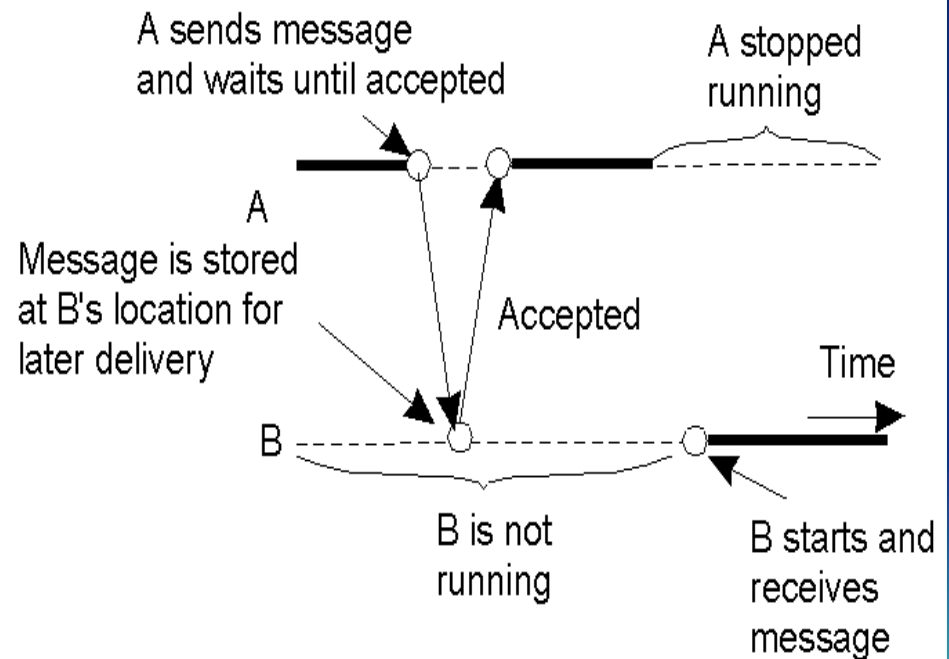


# 消息中的持久性和同步性(3)

- a) 持久异步通信: 提交消息后立即执行其他程序, 电子邮件
- b) 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

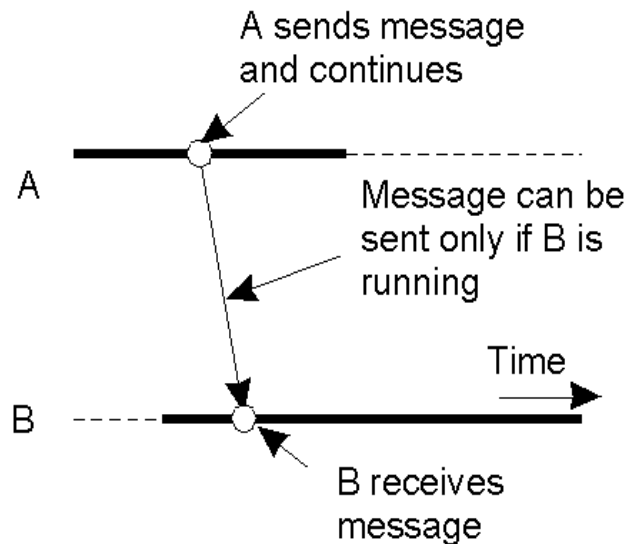


(a)

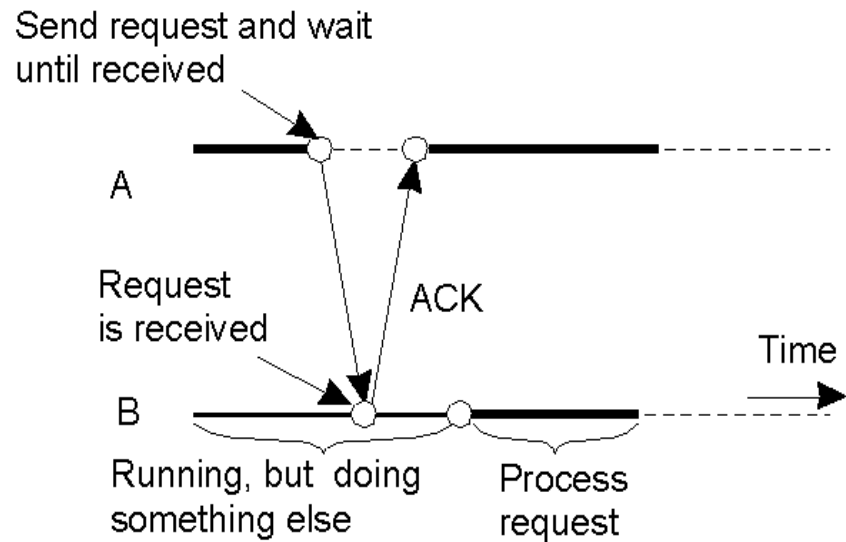


(b)

# 消息中的持久性和同步性(4)



(c)

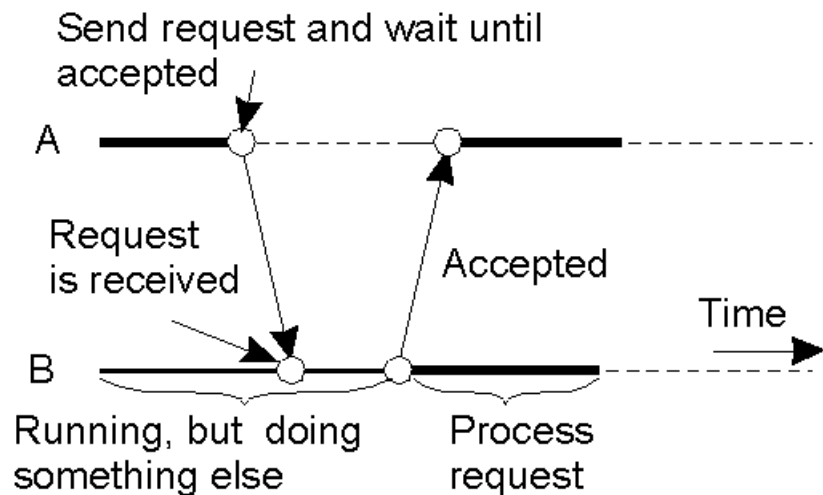


(d)

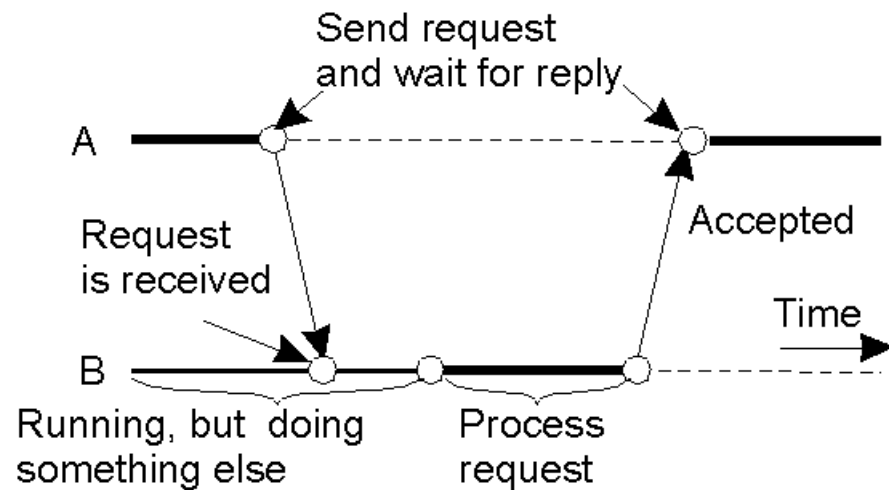
c) 暂时异步通信

d) 基于接收的暂时同步通信

# 消息中的持久性和同步性(5)



(e)



(f)

e) 基于交付的暂时同步通信

f) 基于响应的暂时同步通信

# 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**

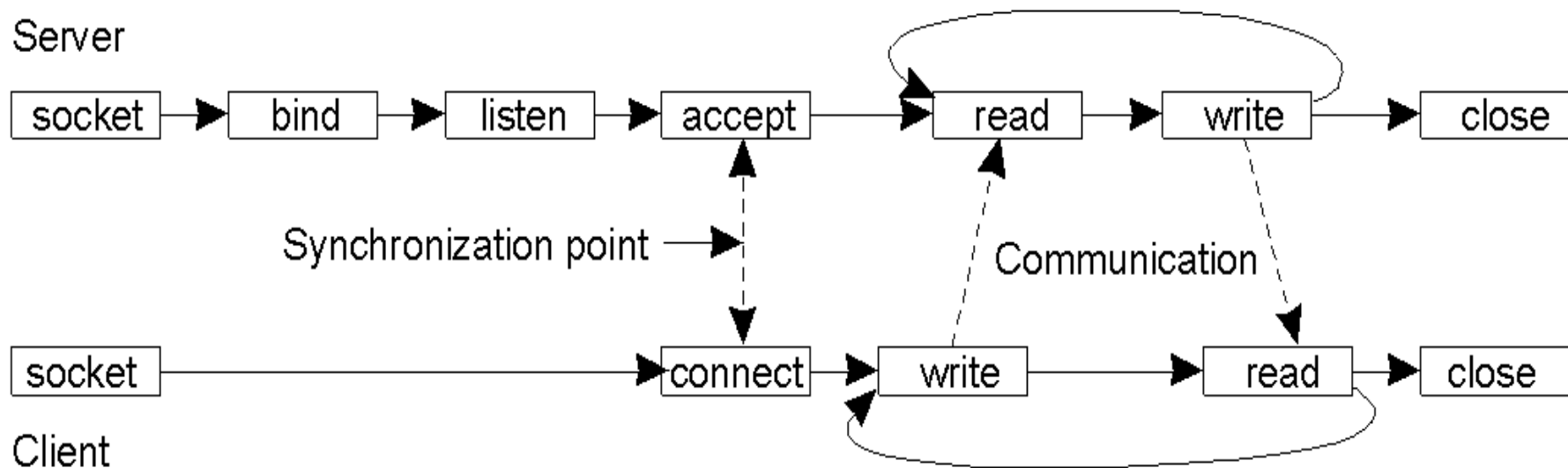


# Berkeley Sockets (1)

- TCP/IP套接字原语

原语	含义
<b>Socket</b>	创建新的通信端点
<b>Bind</b>	将本地地址附加（ <b>attach</b> ）到套接字上
<b>Listen</b>	宣布已准备好接受连接
<b>Accept</b>	在收到连接请求前阻塞调用方
<b>Connect</b>	主动尝试建立连接
<b>Send</b>	通过连接发送数据
<b>Receive</b>	通过连接接收数据
<b>Close</b>	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
<b>MPI_bsend</b>	将消息追加到本地发送缓冲区中
<b>MPI_send</b>	发送消息，并等待到消息复制到本地或远程缓冲区为止
<b>MPI_ssend</b>	发送消息，并等待到对方开始接收为止
<b>MPI_sendrecv</b>	发送消息，并等待到收到应答消息为止
<b>MPI_issend</b>	传递要发送消息的引用，并等待到对方开始接收为止
<b>MPI_irecv</b>	检查是否有输入的消息，但无论有无消息都不会阻塞
<b>MPI_recv</b>	接受消息，如果不存在等待的消息则阻塞
<b>MPI_isend</b>	传递要发送消息的引用，然后继续执行



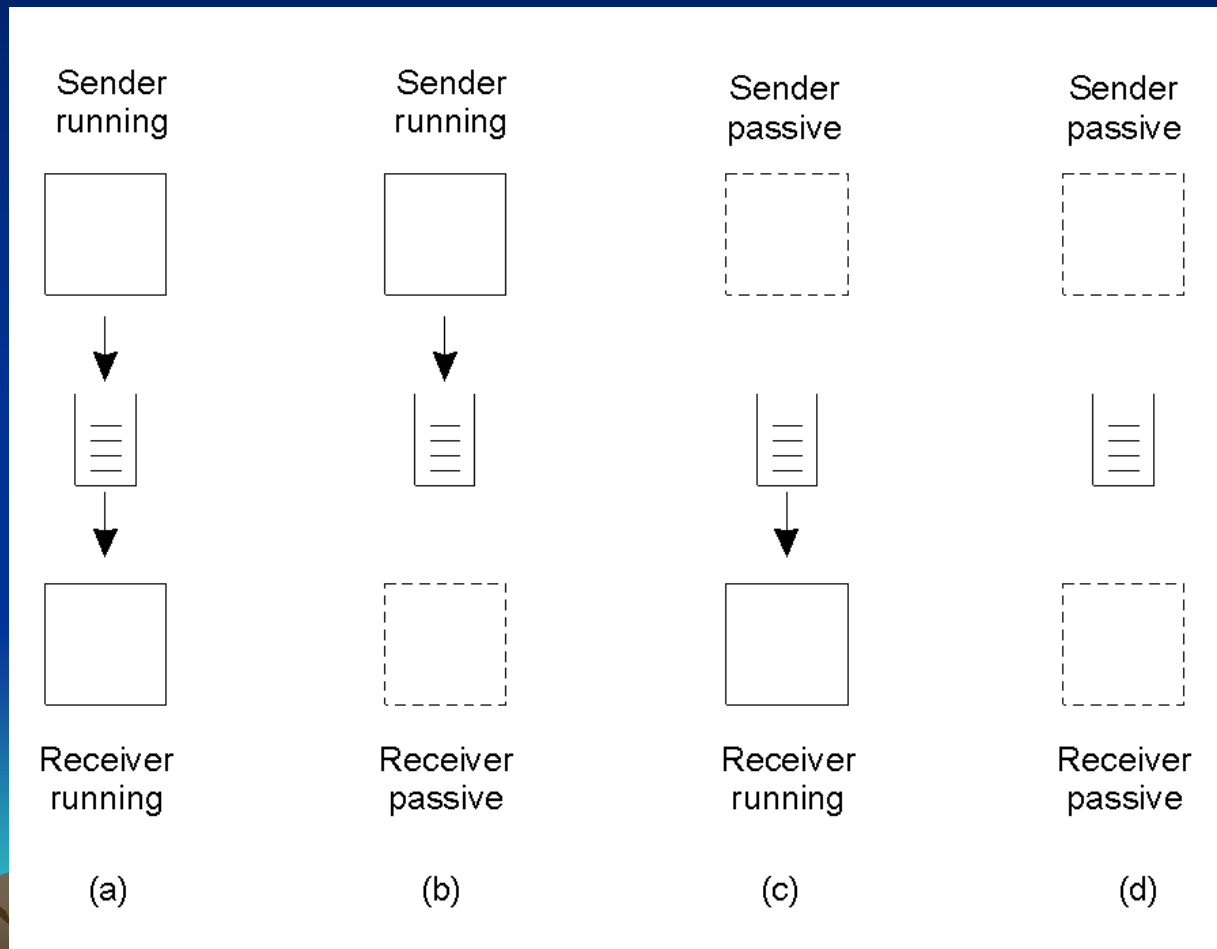
# 面向消息的持久通信

- 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



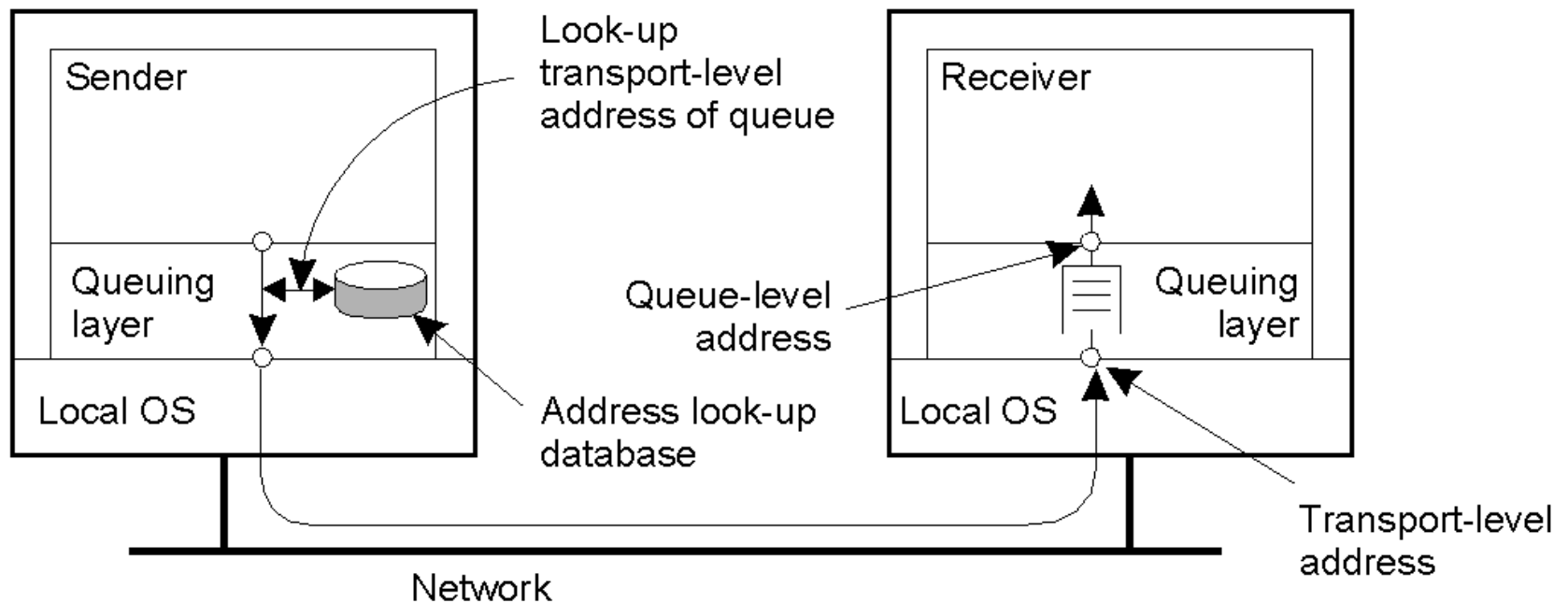
# Message-Queuing Model (2)

- 消息队列系统中队列的基本接口

原语	含义
<b>Put</b>	将消息追加到指定队列
<b>Get</b>	调用进程阻塞，直到指定队列非空，取出第一个消息
<b>Poll</b>	察看指定队列的消息，取出第一个消息，不阻塞调用进程
<b>Notify</b>	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

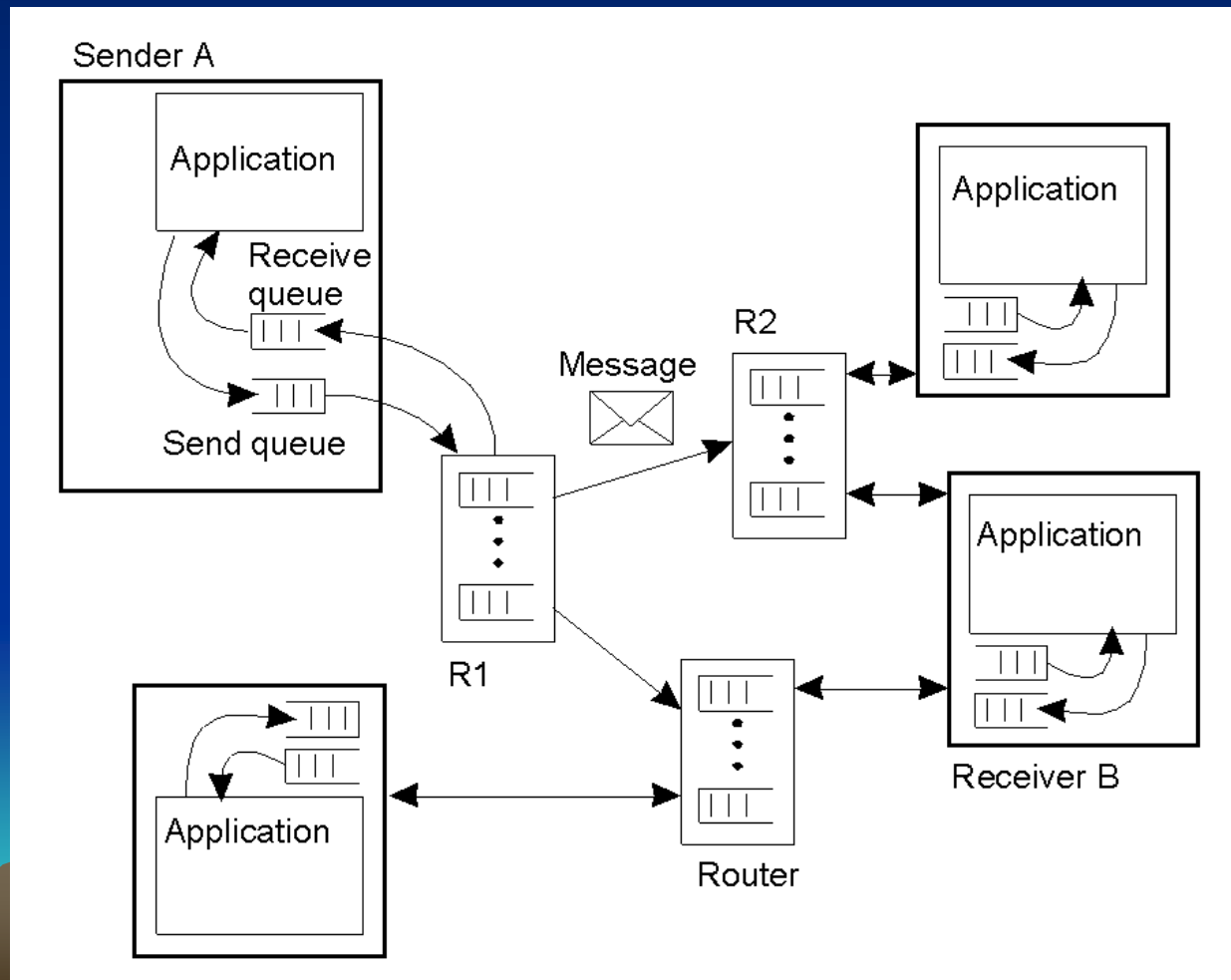
# 消息队列系统的通用体系结构(1)

- 源队列，目的队列
- 队列级编址与网络级编址间的关系

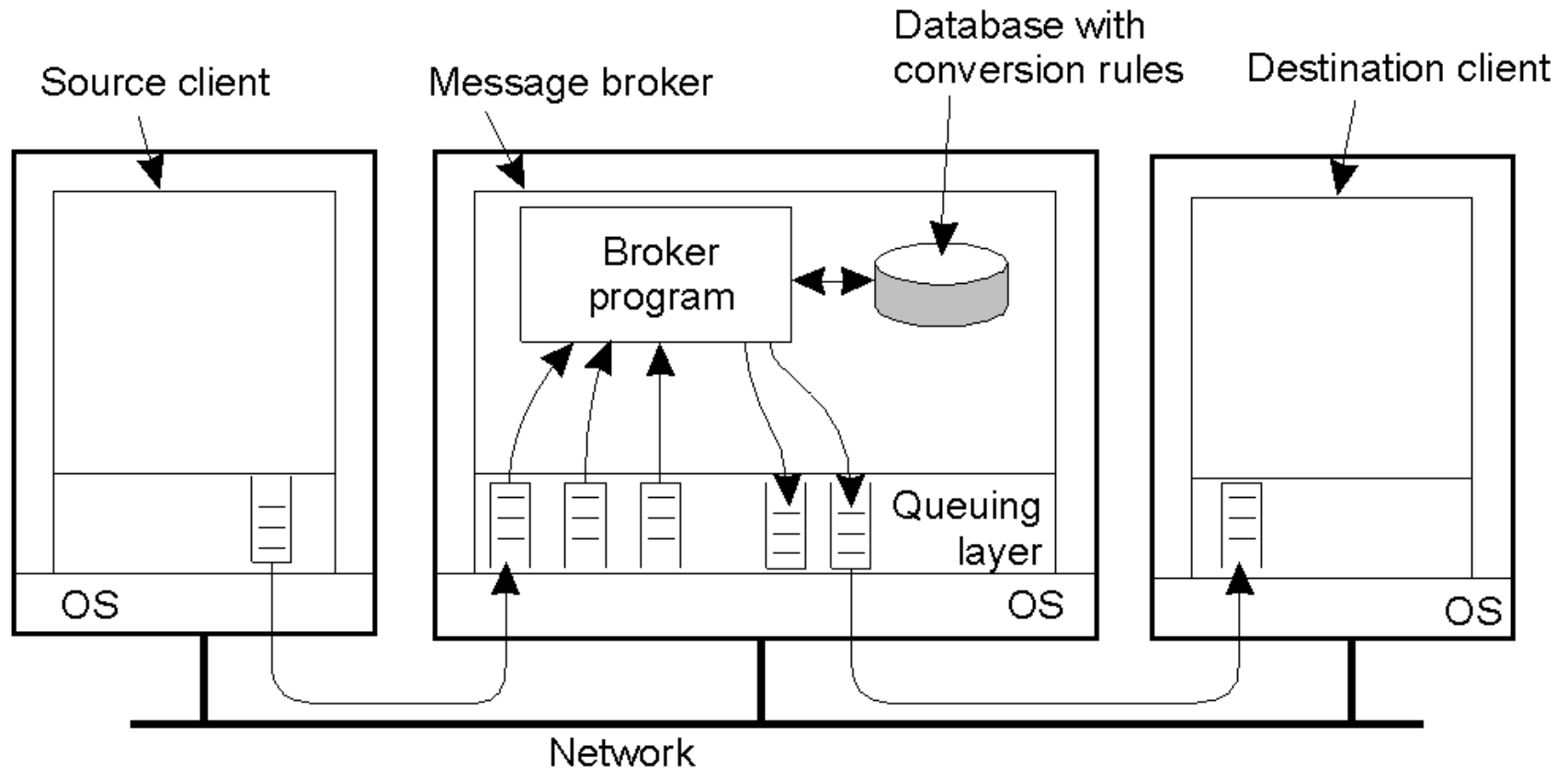


# 消息队列系统的通用体系结构(2)

- 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



# 多播通信

- 多播：服务器向其他N台服务器发送更新时，底层的网络负责向多个接收者发送一个消息，高效
- 网络层多播
- 应用层多播：将节点组织成一个覆盖网络，用它来传播信息给其成员
- 覆盖网络的构建：
  - 树状网络：路径唯一
  - 网状网络：多个路径



# 基于gossiping的数据通信

- **Epidemic**协议使用本地信息在大型节点集中快速地传播信息
- 提供**最终一致性**：保证所有的副本最终是一致的
- 一个服务器可以是：
  - 传染性的：持有愿意向其他服务器散布的更新
  - 易感的：尚未更新的服务器
  - 隔离的：已更新的服务器如果不愿意或不能扩散其更新
- **反熵**传播模型：服务器P周期的随机选取一台服务器Q交换更新，方式包括：
  - P只把自己的更新推入Q：较差的选择？
  - P只从Q拉出新的更新
  - P和Q相互发送更新
  - 可以证明：如果初始只有一台服务器具有传染性，无论采用哪种形式，更新最终将被传播到所有服务器上： $O(\log(N))$ , N为系统结点数



# Gossiping模型

- 思想：
  - 如果服务器P刚刚因为数据项x而被更新，那么它联系任意一个其他服务器Q，并试图将更新推入Q。
  - 如果Q已经被其他服务器更新了，P可能会失去继续扩散的兴趣，变成隔离的（这种可能性是 $1/k$ ）
- 快速传播更新的方法
- 但不能保证所有的服务器都被更新了
- $s = e^{-(k+1)(1-s)}$ ,  $k=3$ 时， $s$ 小于2%； $k=4$ 时， $s$ 小于0.7%
- 可以将gossiping和反熵模型结合

# 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
- 多播通信

