第二章　体系结构

- 体系结构样式
- 系统体系结构
- 体系结构与中间件
- 分布式系统的自我管理

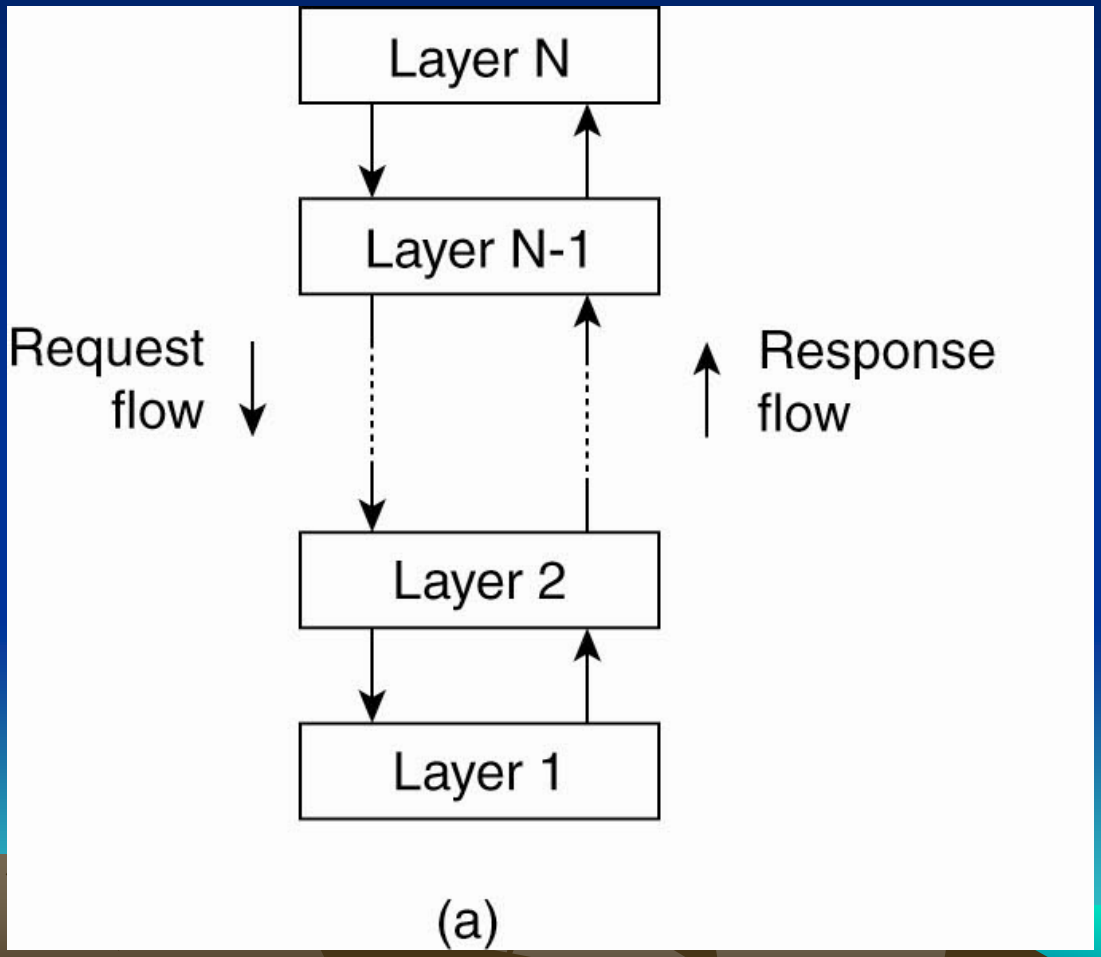
2.1 体系结构样式
Architectural Styles

体系结构样式：软件体系结构，根据组件、组件之间的连接方式、数据交换以及这些元素如何集成到一个系统来定义：

- 分层体系结构（Layered architectures）
- 基于对象的体系结构（Object-based architectures）
- 以数据为中心的体系结构（Data-centered architectures）
- 基于事件的体系结构（Event-based architectures）

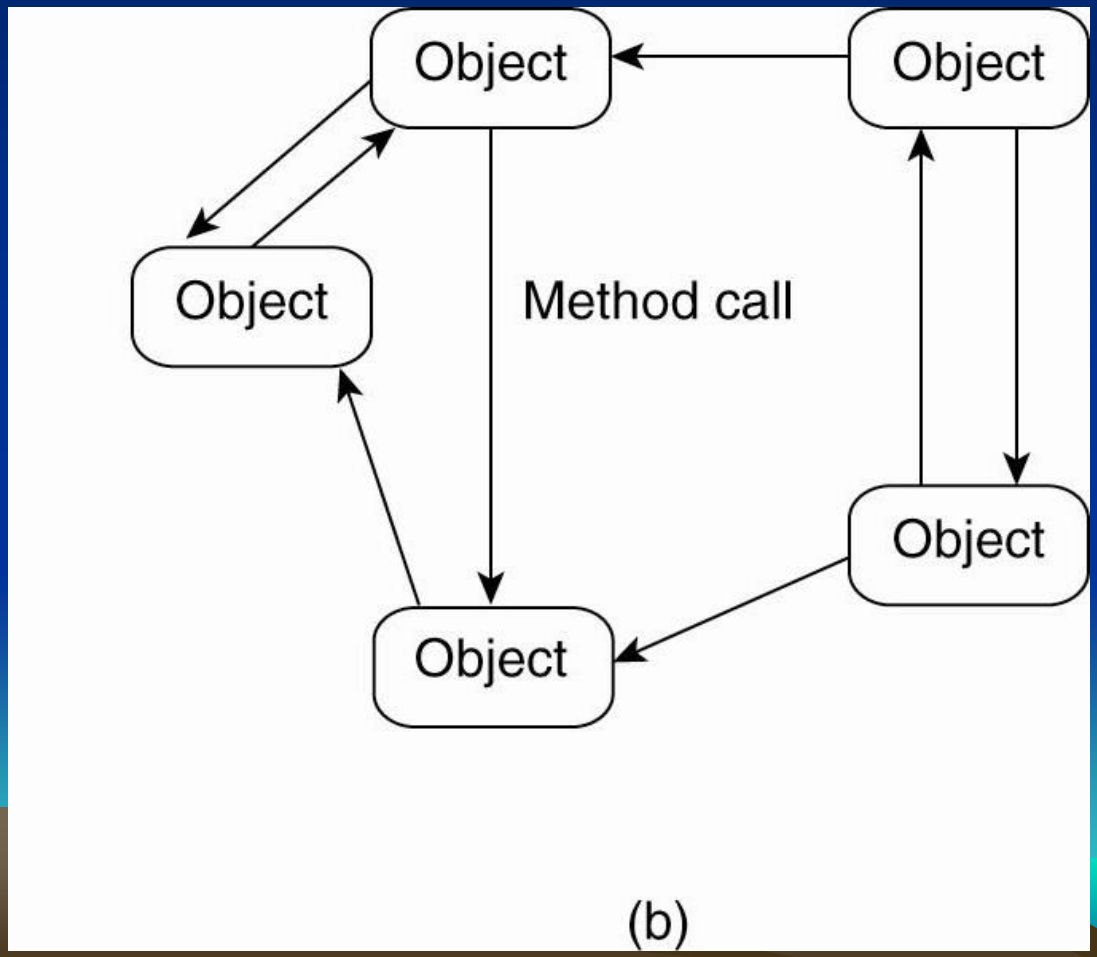
分层体系结构

网络通信广泛使用



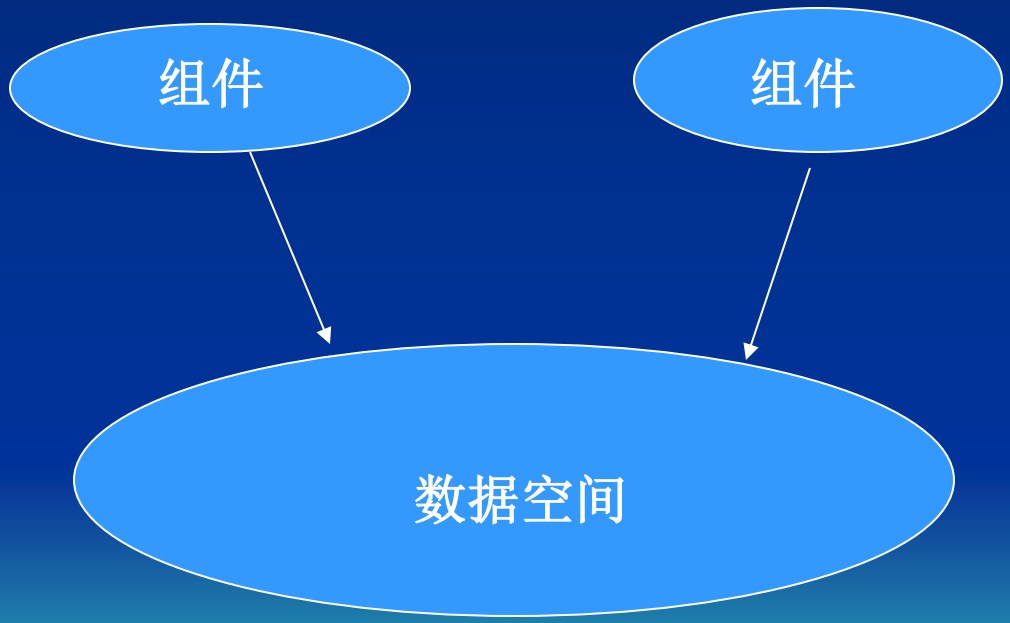
基于对象的体系结构

- 松散的组织结构
- 通过远程过程调用进行通信



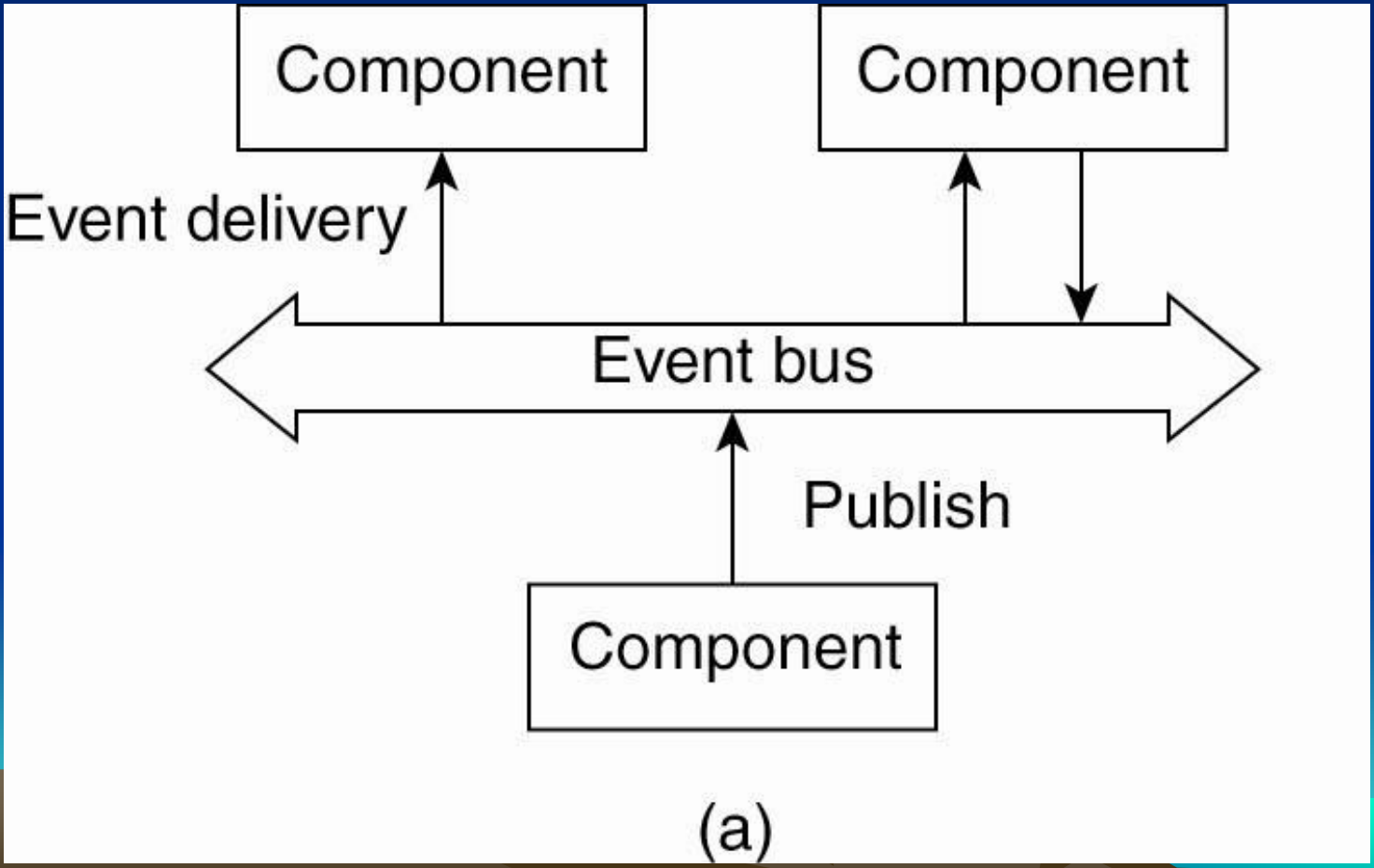
以数据为中心的体系结构

进程通信借助一个公用的数据仓库：分布式文件系统



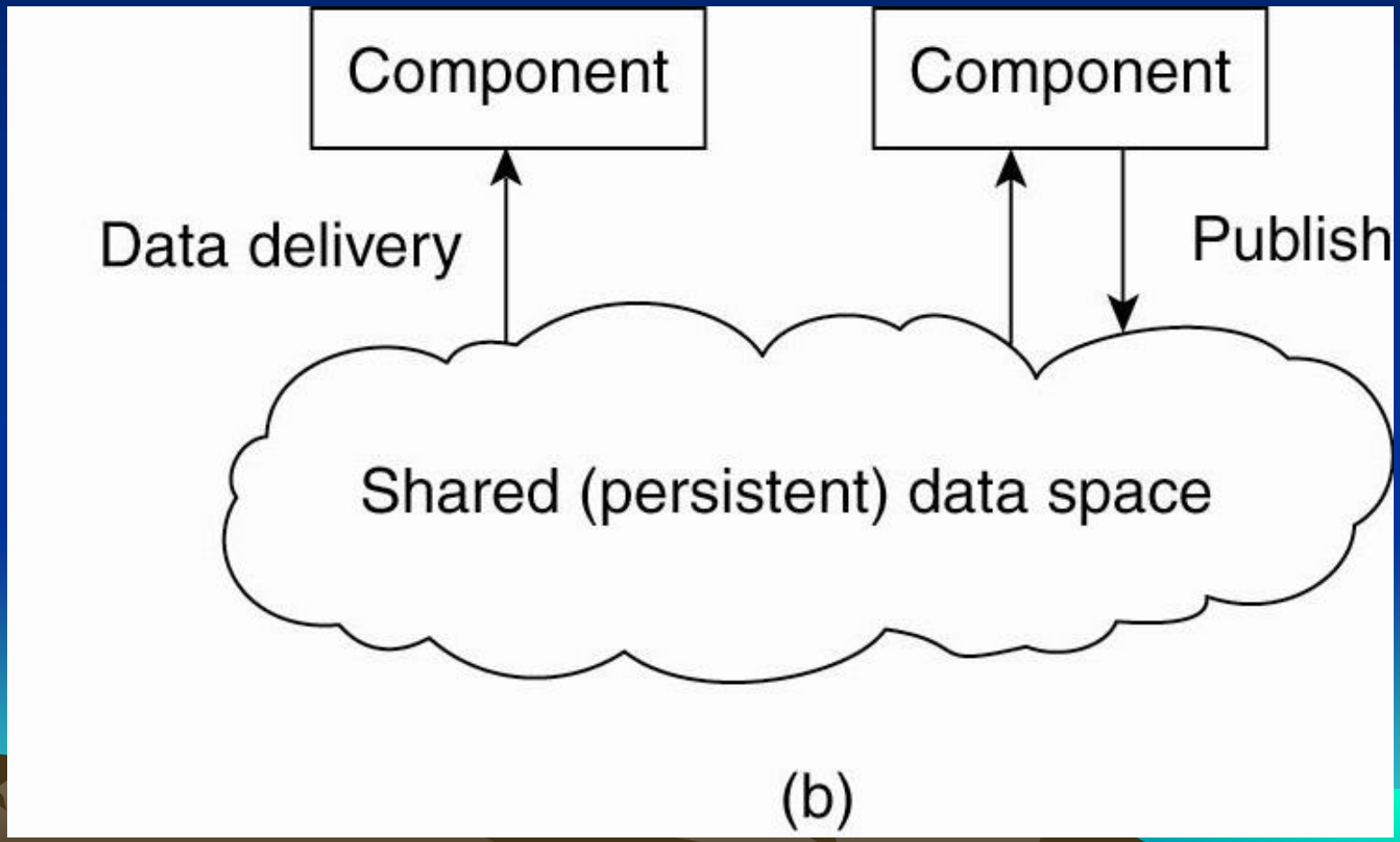
基于事件的体系结构

进程通过事件的传播来通信（可以携带数据）：松散耦合、发布和订阅



共享数据空间

基于事件的体系结构和以数据为中心的体系结构组合



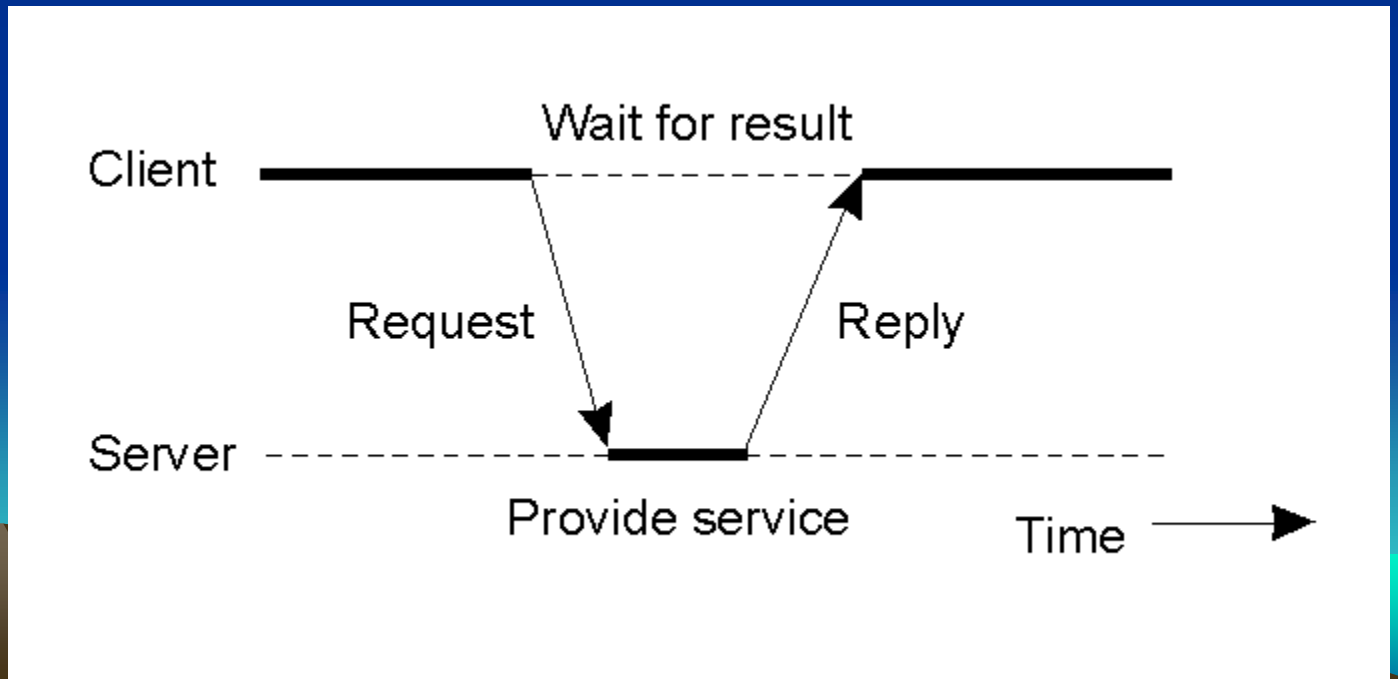
2.2 系统体系结构

系统体系结构：软件体系结构的实例，确定软件组件、组件的交互以及它们的位置

- 集中式体系结构
- 非集中式体系结构
- 混合体系结构

集中式体系结构
客户端-服务器模型

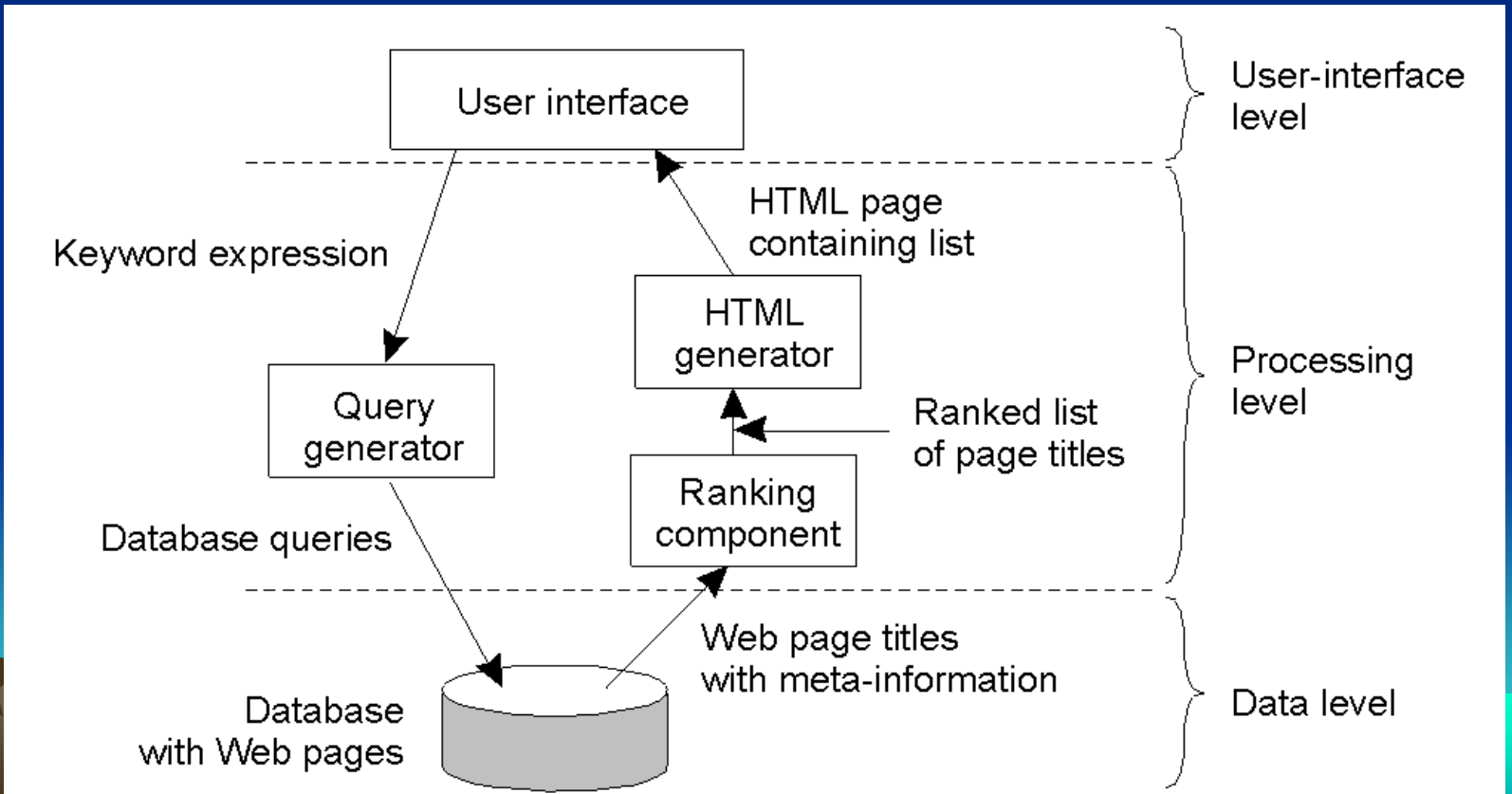
- 服务器（server）：实现某个特定服务的进程
- 客户（client）：向服务器请求服务的进程
- 客户端-服务器之间的一般交互：请求/回复
- 无连接的协议：高效，受传输故障的影响，适合局域网
- 基于连接的协议：性能相对较低，适合广域网（TCP/IP）



应用程序的分层

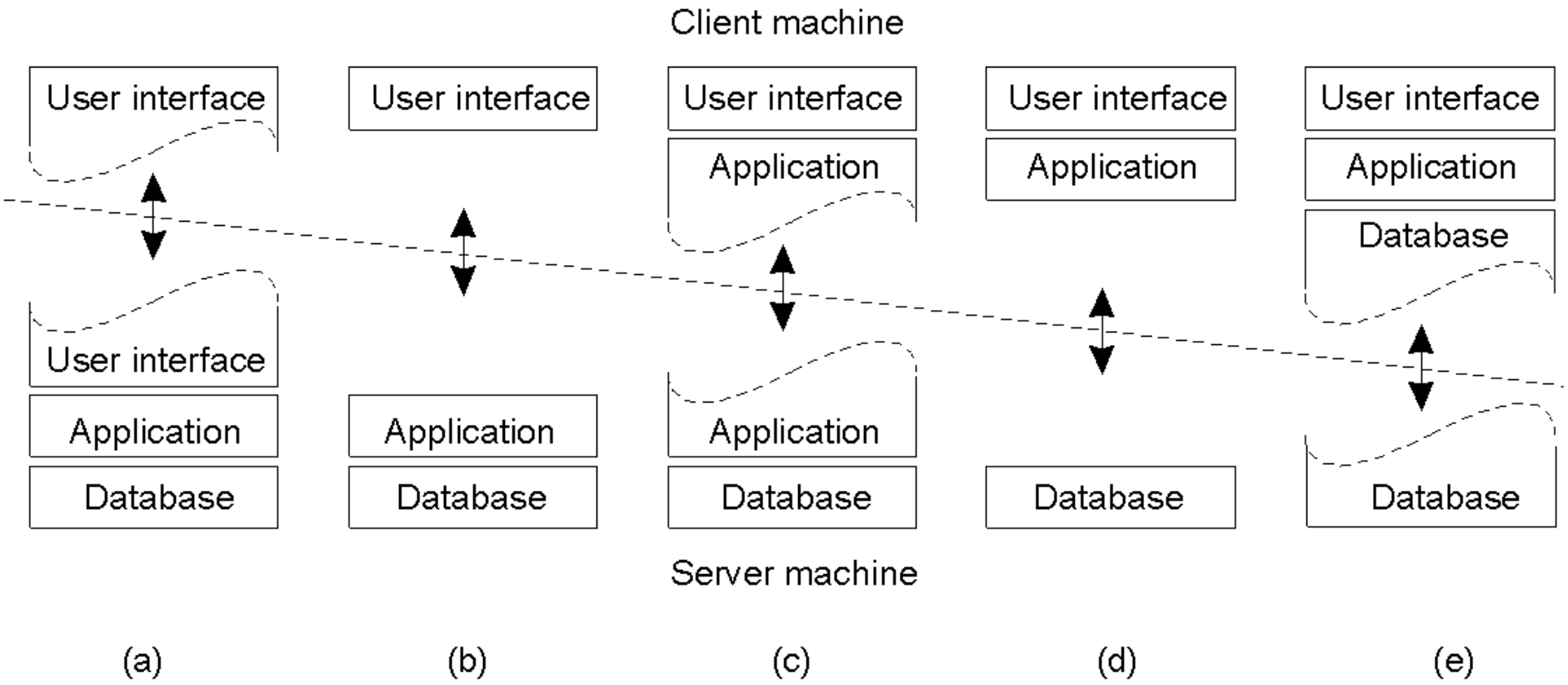
客户服务器应用程序通常组织为三个层次：

- 用户界面层：用户交互所需的一切
- 处理层：应用程序核心功能
- 数据层：操作数据或文件系统，保持一致性



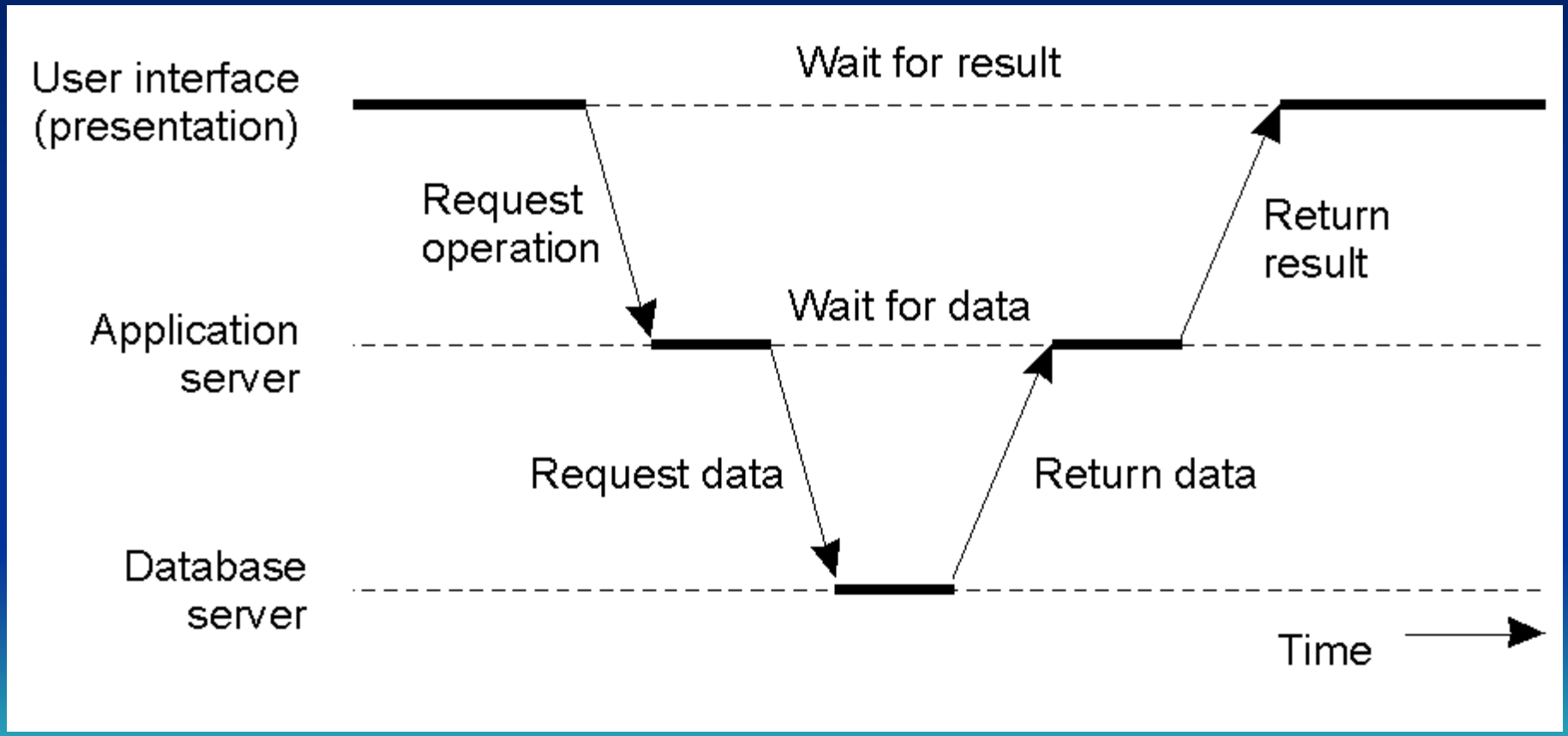
多层体系结构 (1)

- 客户端-服务器模型可能的组织结构 **(a) – (e).**



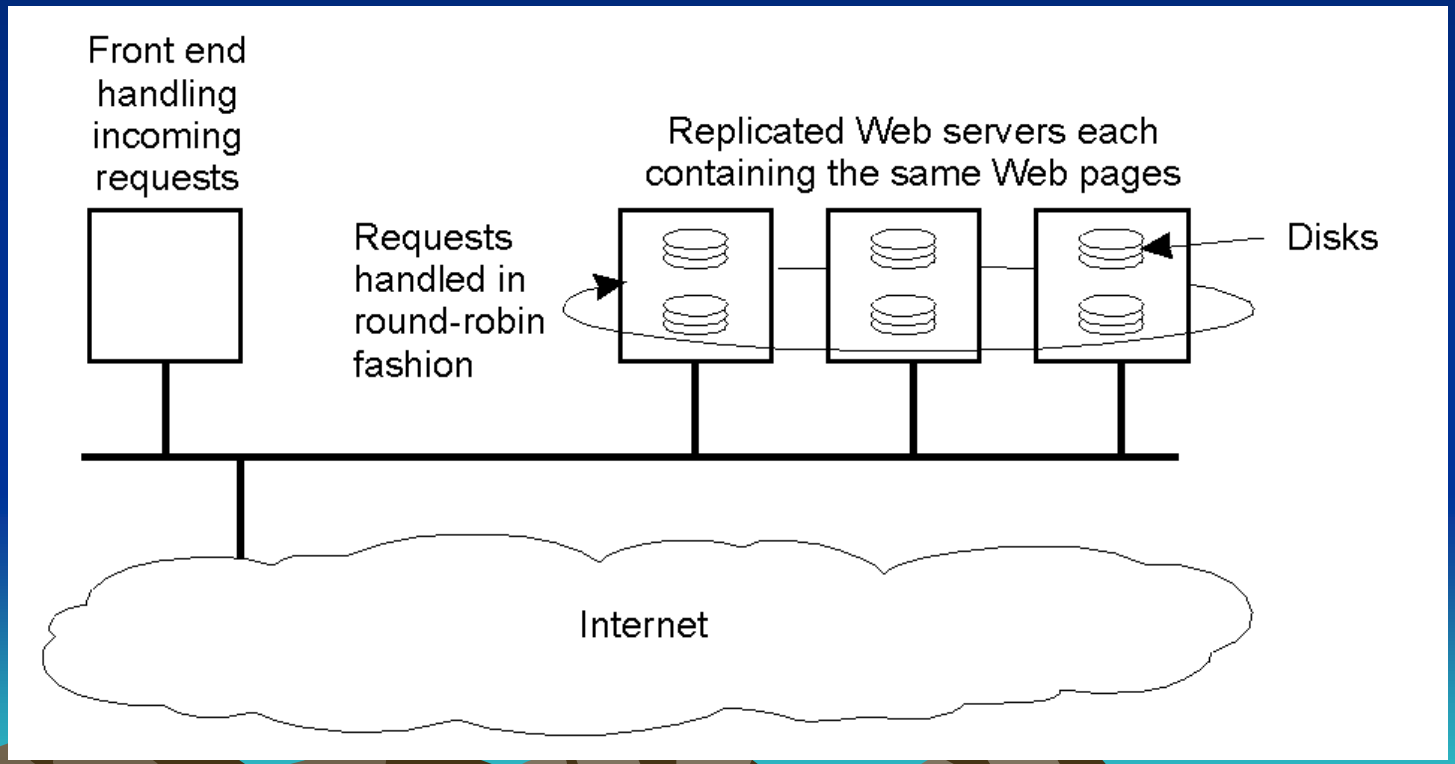
多层体系结构 (2)

- 服务器充当客户端角色的例子



非集中式体系结构

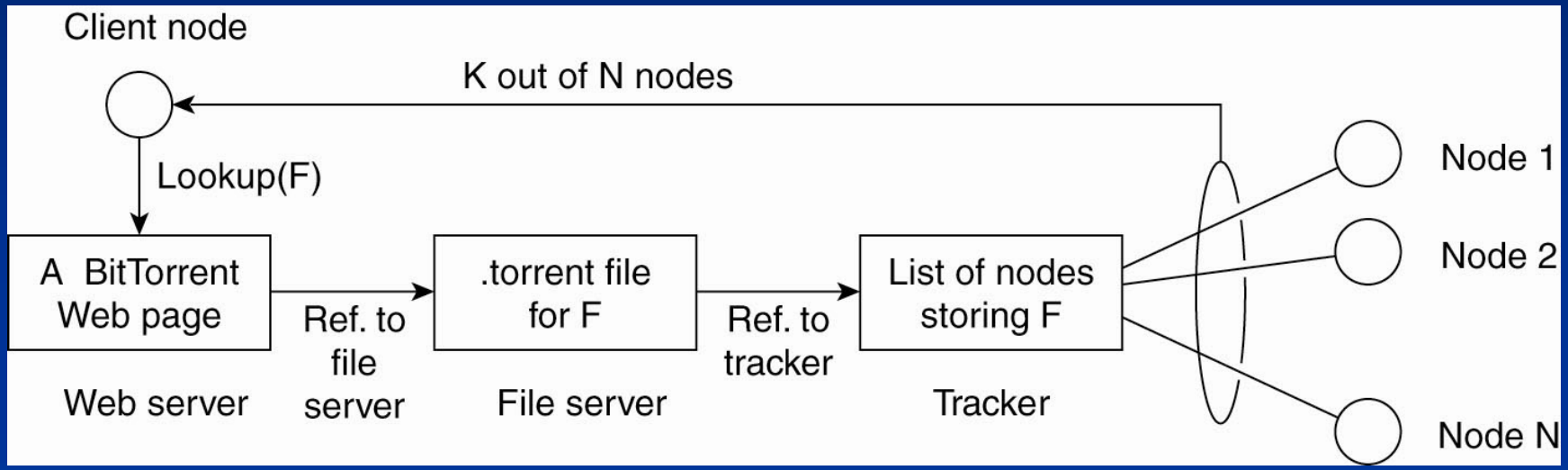
- 垂直分布性：按逻辑把不同的组件放在不同的机器上。
- 水平分布性：客户或服务器按照在物理上被分割成逻辑上相同的几部分:点对点系统。



Web 服务水平分布示例

混合体系结构-协作分布式系统

•协作分布式系统：以C/S方式启动，一旦节点加入，则以完全非集中式协作- **BitTorrent** 。

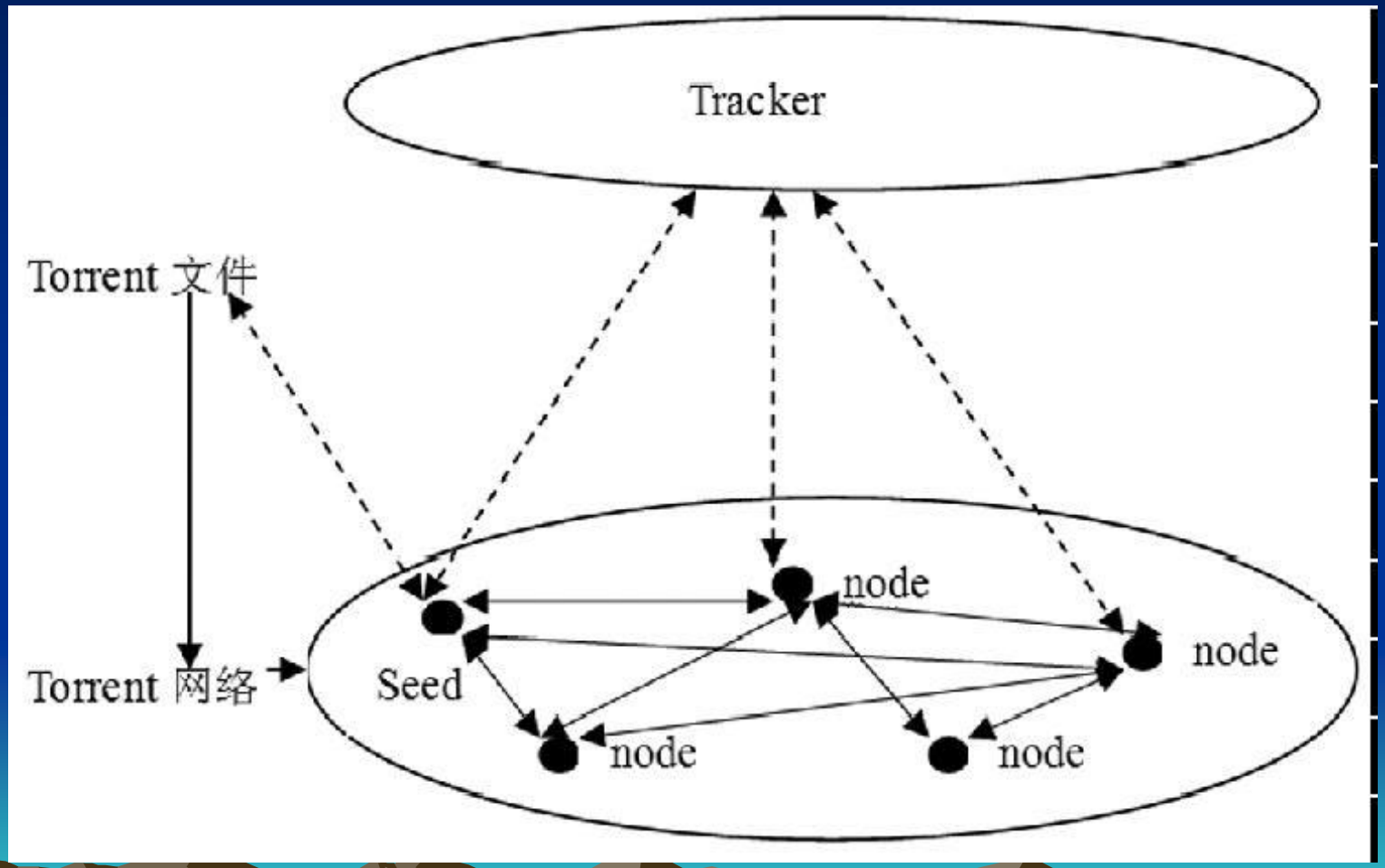


BitTorrent 工作原理

BT 下载网络有三个关键静态组件：

- 跟踪器（Tracker）：Tracker 跟踪器是一个中央服务器，它主要跟踪系统中所有的参与结点，收集和统计这些结点的状态，帮助参与结点间互相发现并进行文件块的交换；
- 种子节点（Seed）:Seed 种子节点是指拥有完整文件的节点，提供上载服务；
- 下载节点（Downloader）。相对于Seed 的节点称为下载节点，一个下载节点完成下载后，可以成为种子节点

BitTorrent系统结构



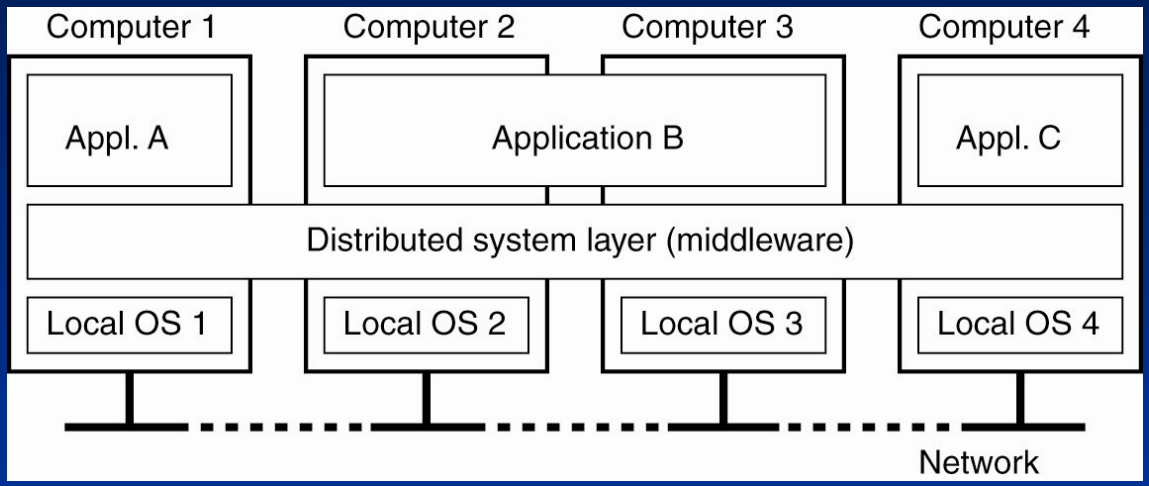
动态流程

- 第一个用户通过BT工具制作要共享文件的Torrent 文件（Torrent 文件包含共享文件的下载信息）并发布此Torrent 文件到WWW中。
- 其他用户从WEB服务器上下载此Torrent 文件并通过节点跟踪器协议（如TrackerHTTP）去访问Tracker 跟踪器，参与到此Torrent 网络中。
- Tracker跟踪器接收到一个新加入节点的下载请求后，随机选择部分此Torrent网络中的节点发送给新加入者作为邻居节点，并记录新节点。
- 新加入节点通过一定的算法同邻居节点连接进行文件的下载和上载直到文件下载完成，这一过程会根据一定的策略重复。如果继续上载，Tracker 服务器将此节点看作种子节点。
- 所有参与的节点将周期地报告自己的状态和进程给Tracker 跟踪器.

BT 关键技术

- BT 文件发布系统采用针锋相对（Tit_for_Tat）的方法来达到帕累托（pareto）有效，与当前其他的P2P技术相比，它达到了更高层次的鲁棒性和资源利用。
- 帕累托最优:指资源配置已达到这样一种境地，即任何重新改变资源配置的方式，都不可能使一部分人在没有其他人受损的情况下受益。
- 最少优先原则:对一个下载者来说，在选择下一个被下载的片断时，通常选择的是它的Peers 所拥有的最少的那个片断，也就是所谓的“最少优先”。

2.3 体系结构与中间件



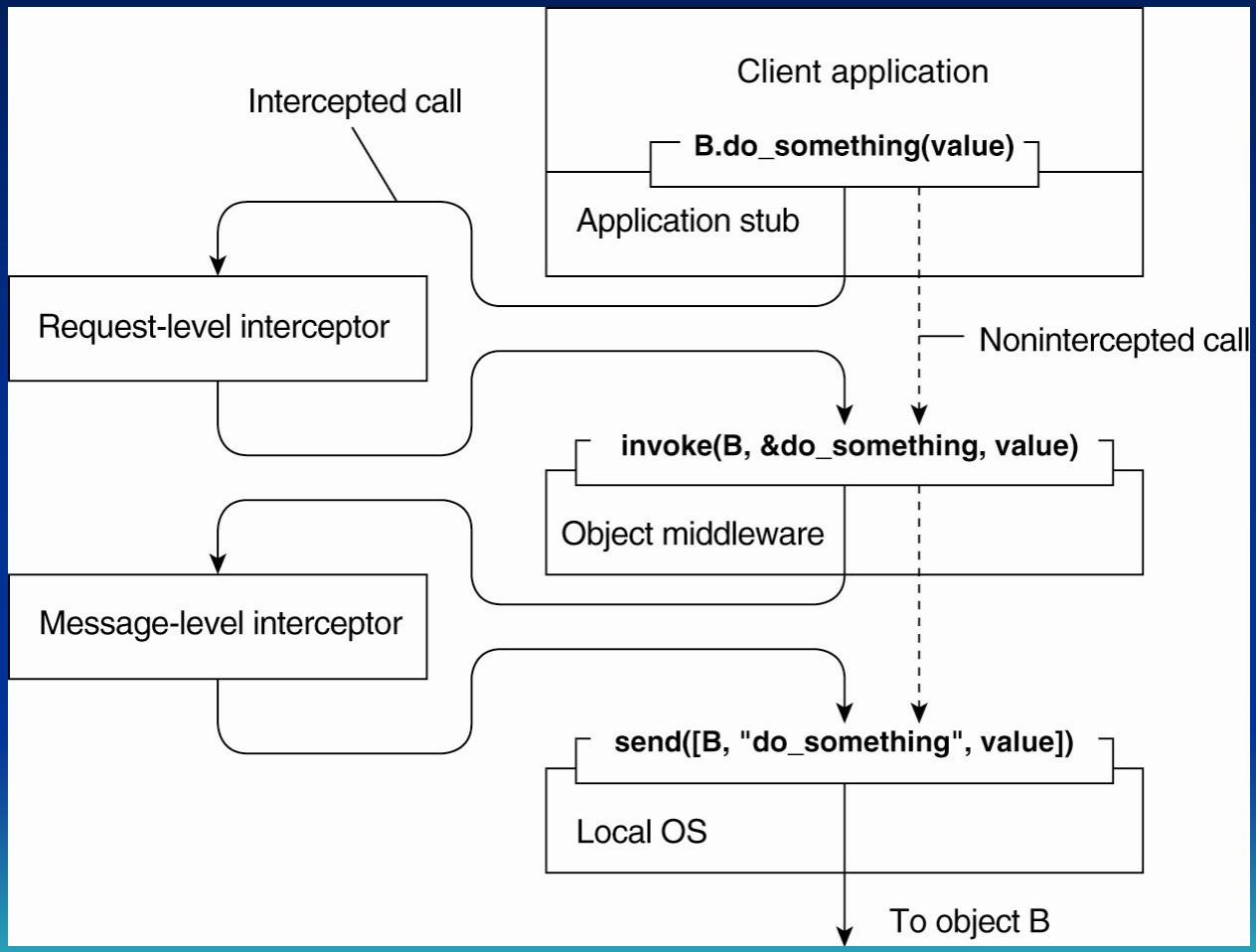
中间件一般要遵循一定的体系结构风格：

- 基于对象的体系结构
- 基于事件的体系结构
- 中间件可以按照应用程序的需求方便的进行配置、适应和定制：
  - 拦截器
  - 自适应软件的常见方法

拦截器（intercepter）

- Interceptor是CORBA规范提出的一种重要思想。它允许扩充中间件的功能而无须改变中间件的核心构造。
- Interceptor pattern是一种设计模式,它允许将服务透明地添加到框架中, 当某一事件发生时, 可以自动触发。这种模式可以保护组件免受运行环境的影响, 应用程序开发者可以集中精力于业务逻辑的开发。

使用拦截器处理远程对象激活



- 请求级中断器使得对象A不需要知道B的副本。
- 消息级中断器帮助提高性能和可靠性。

自适应软件的常见方法

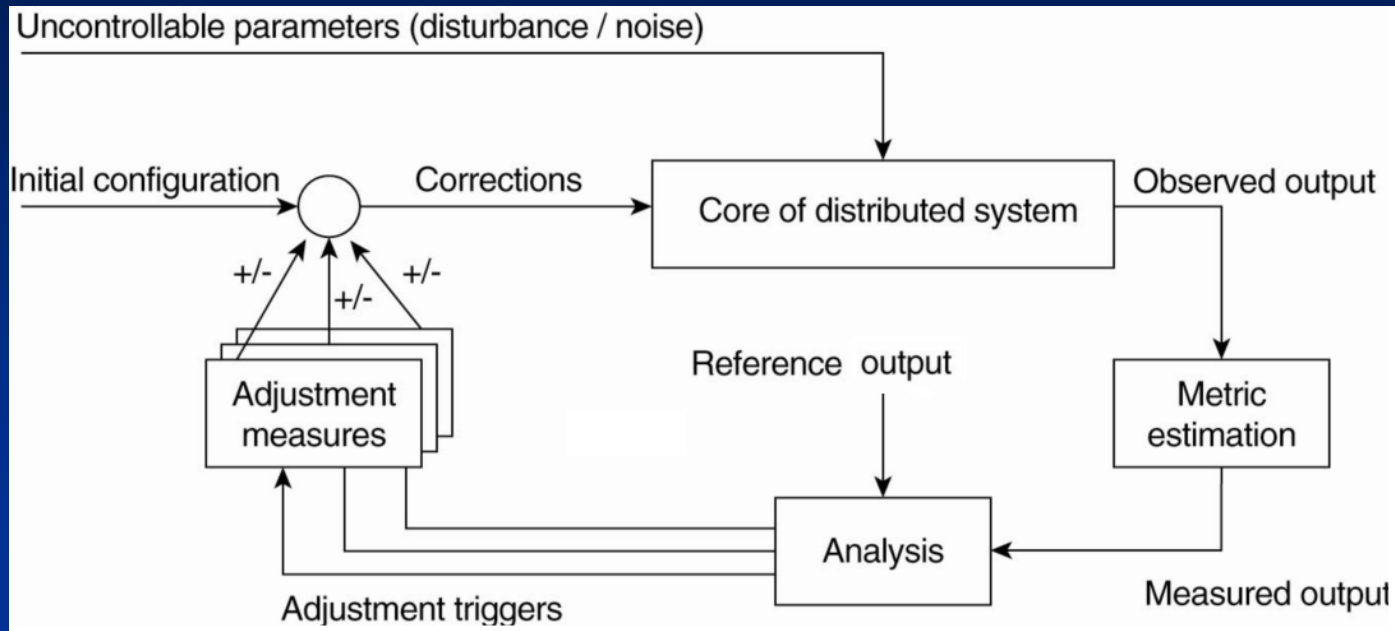
实现软件自适应性的基本技术:

- 要点分离(Separation of concerns)：实现功能的部分与负责可靠性、性能和安全的部分分开
- 计算反射(Computational reflection)：程序检查自己，调整其行为的能力
- 基于组件的设计(Component-based design)：通过组件的不同组合来支持自适应

2.4 分布式系统的自我管理

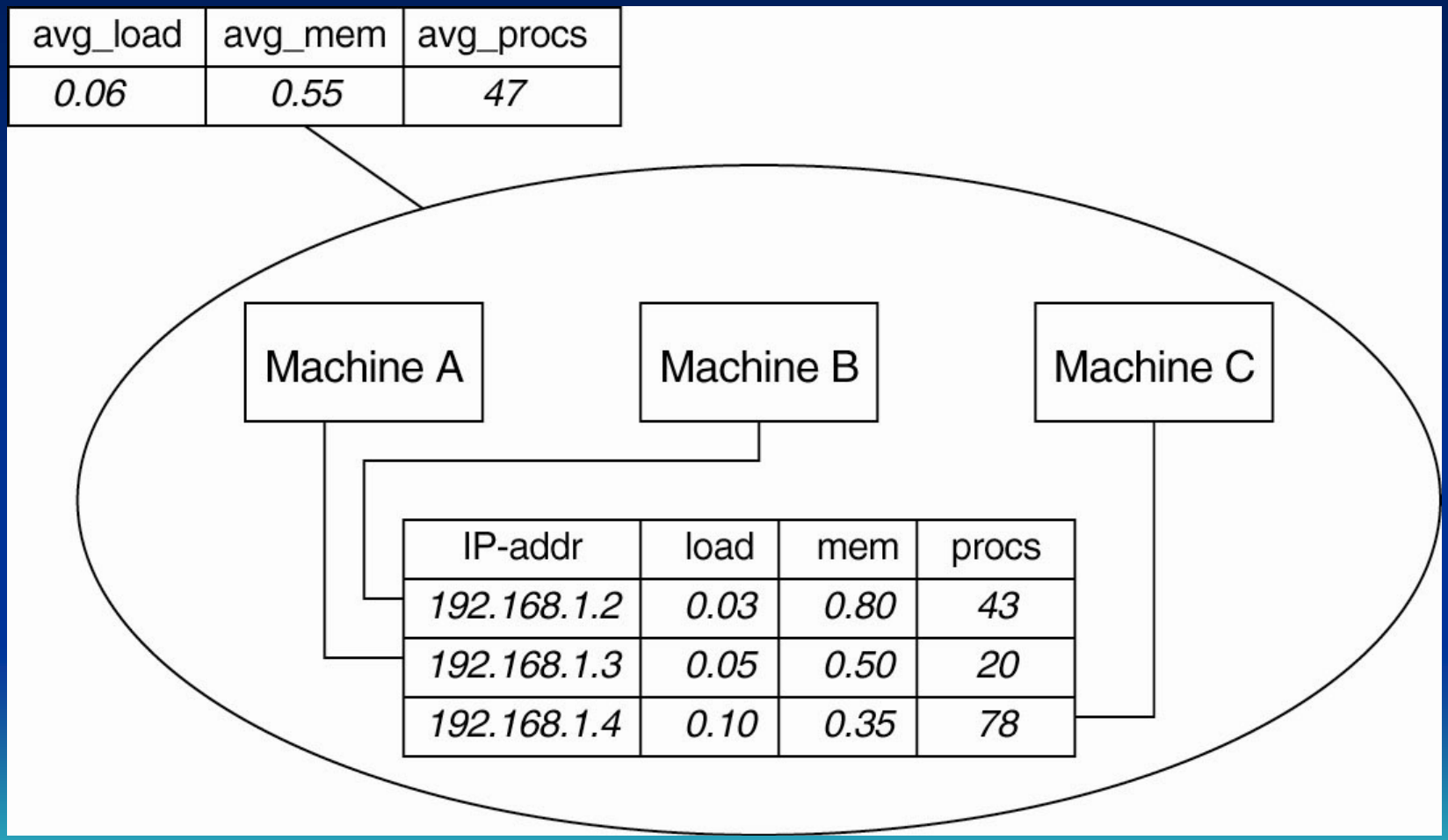
- 以高级反馈控制系统的形式来组织分布式系统，允许自动适应变化
- 自治计算或自主系统：自我管理、自我恢复、自我配置和自我优化
- 反馈控制模型
  - Astrolabe监视系统
  - Globule中的差分复制策略
  - Jade的自动组件修复管理

反馈控制模型



反馈控制系统的逻辑组织

示例:Astrolabe监视系统

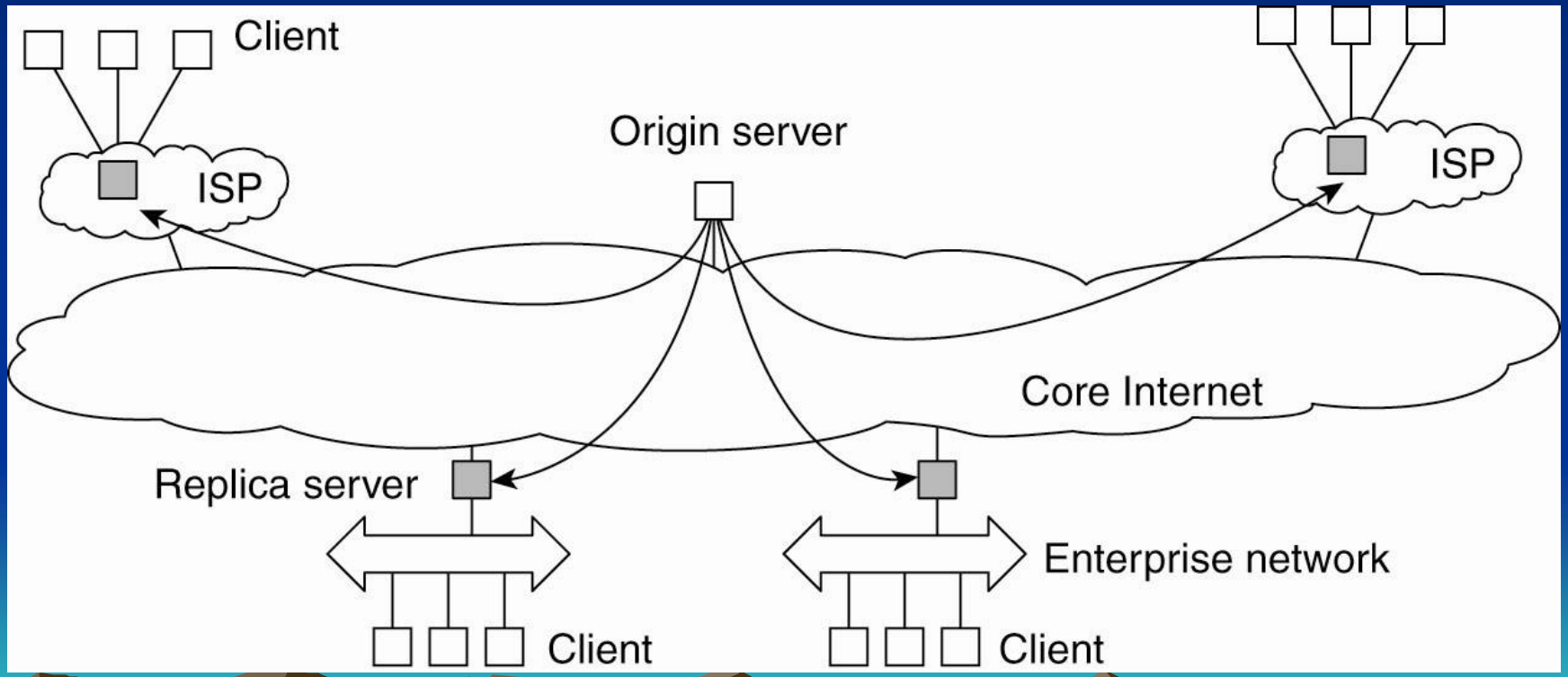


Astrolabe中的数据采集和信息集成

示例:Globule中的差分复制策略

初始服务器搜集对页面的请求信息，确定复制策略：特定页面复制位置、如何保持页面的一致性

复制策略的成本cost=$(w_1*m_1)+(w_2*m_2)+\ldots+(w_n*m_n)$



Globule中的边界服务器模型

示例:Jade的自动组件修复管理

- 对于基于组件的服务器，可以检测组件故障，并自动代替它们
- 修复管理域
- 结点管理器:添加和删除结点
- 故障检测器:每个结点一个
- 修复策略的执行步骤:
  - 终止无故障结点组件与已发生故障结点组件之间的所有绑定
  - 请求节点管理器启动和添加新结点到该域
  - 配置新节点使之具有与已崩溃结点相同的组件
  - 重新建立前面已终止的绑定

小 结

- 体系结构类型
- 系统体系结构
- 体系结构与中间件
- 分布式系统的自我管理

习 题

- 考虑一个进程链，该进程由进程P_1, P_2,… P_n构成，实现了一个多层客户-服务器体系结构。进程P_i是P_{i+1}的客户， P_i只有得到P_{i+1}的应答之后才能向P_{i-1}发出应答。如果考虑到进程P_1的请求-应答性能，这种组织结构主要存在什么问题？