第2讲 分布式系统体系结构

§2.1 体系结构样式

§2.2 不同类型体系结构

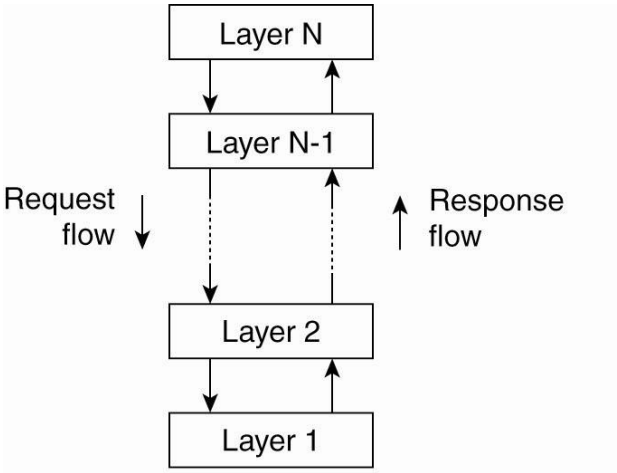
§2.3 体系结构与中间件

§2.1 体系结构样式

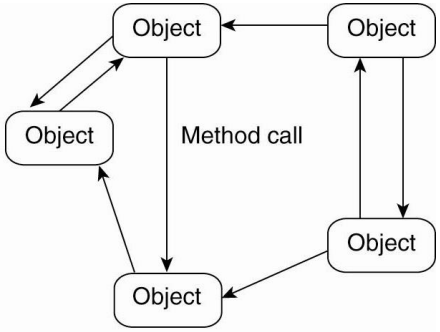
- *Software architecture*
  - How various software components are organized and how they interact.
- *System architecture*
  - An instance of a software architecture after deciding on the software components, their interaction and their placement.

系统体系结构确定软件组件、这些组件的交互以及它们的位置。

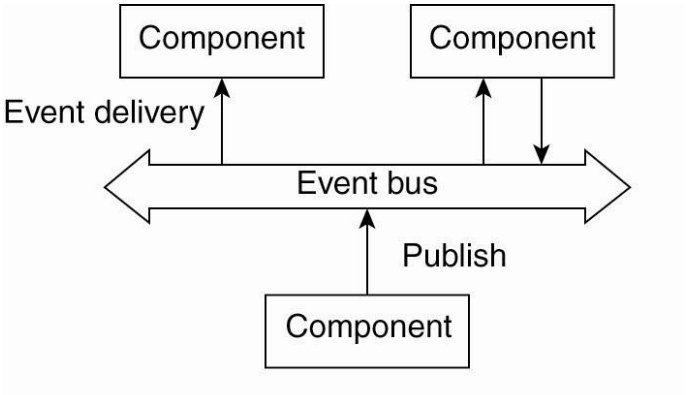
Architectural Styles （样式）



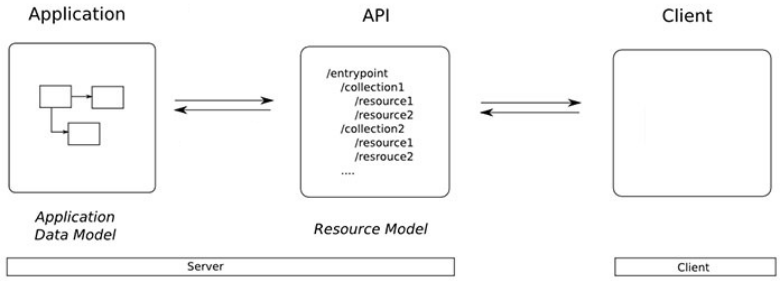
Layered architectures



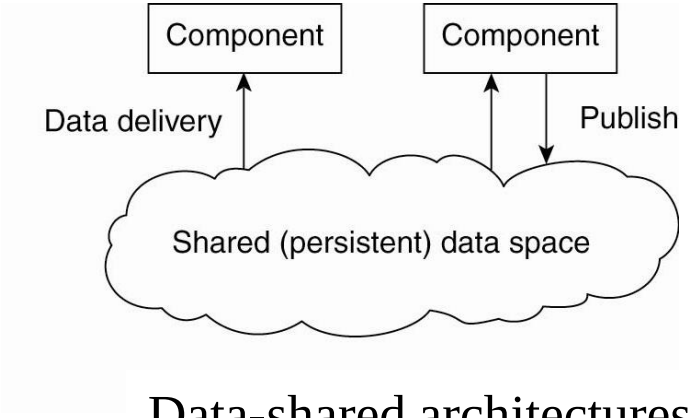
Object-based architectures



Event-based architectures



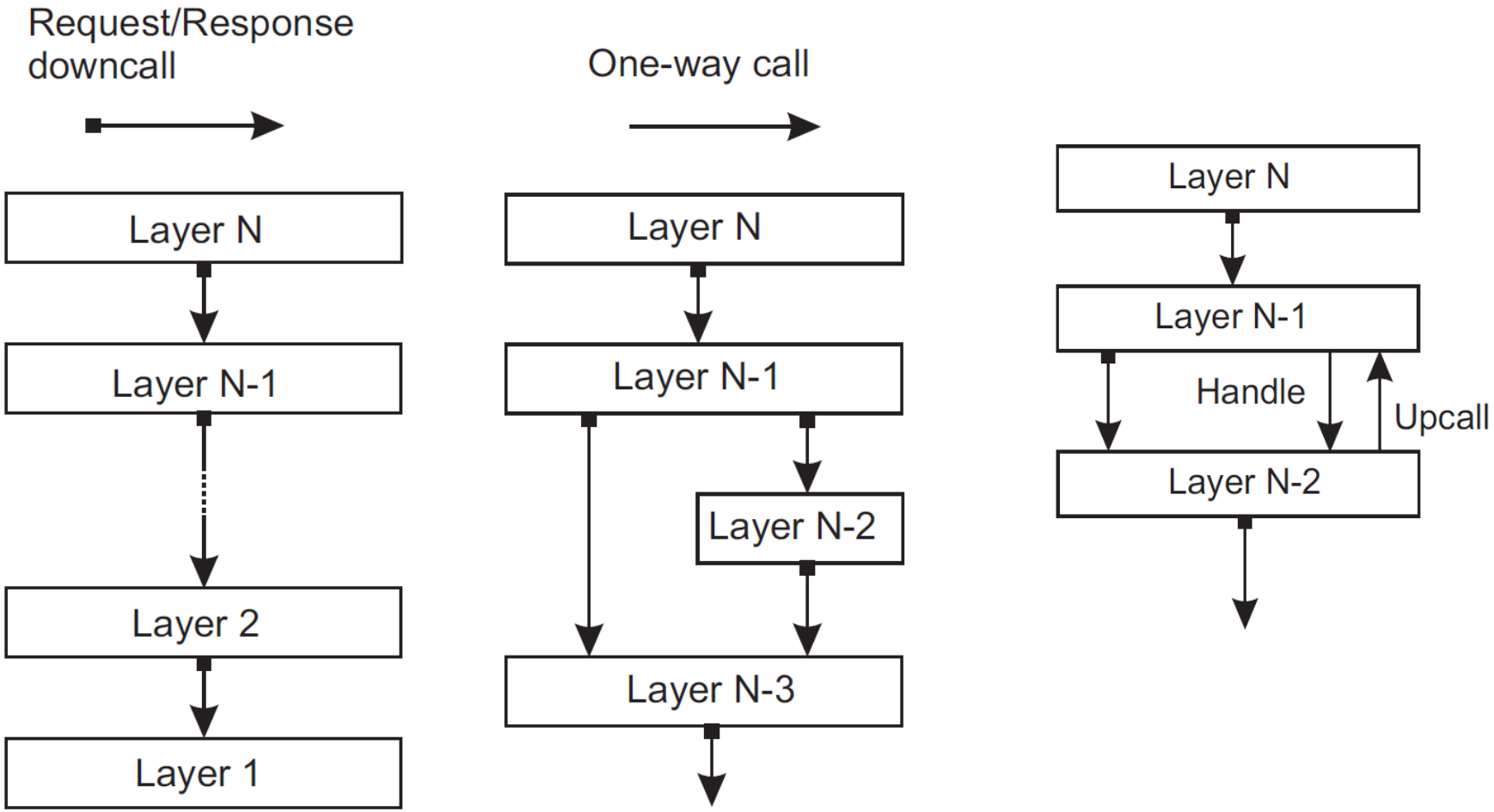
Resource-centered architectures



Data-shared architectures

分层架构样式

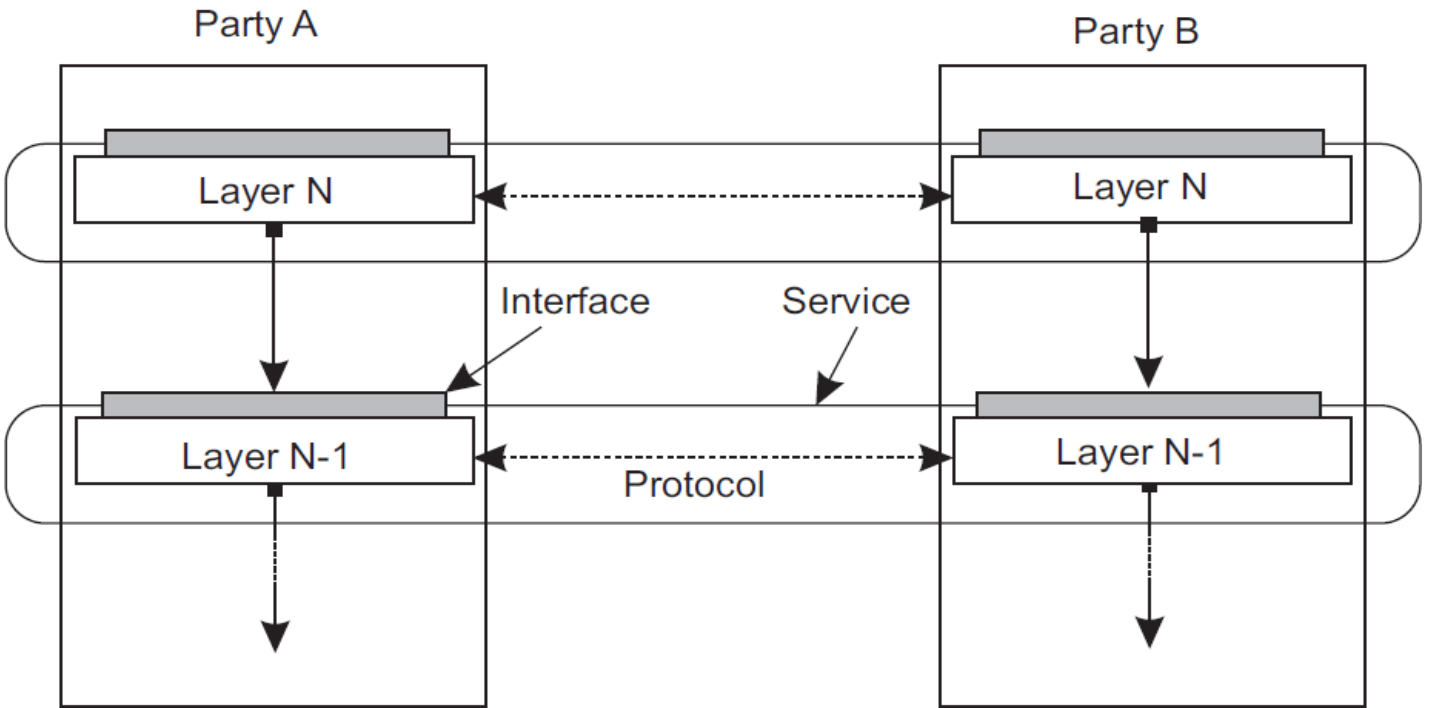
- 主要用于客户端-服务器模型



(a) Pure layered organization. (b) Mixed layered organization. (c) Layered organization with upcalls

分层通讯协议

- **Service:** functions allowing data to be sent.
- **Protocol:** describes the rules that parties will follow in order to exchange information.
- **Interface:** specifying the functions that can be called (hide implementation of service).

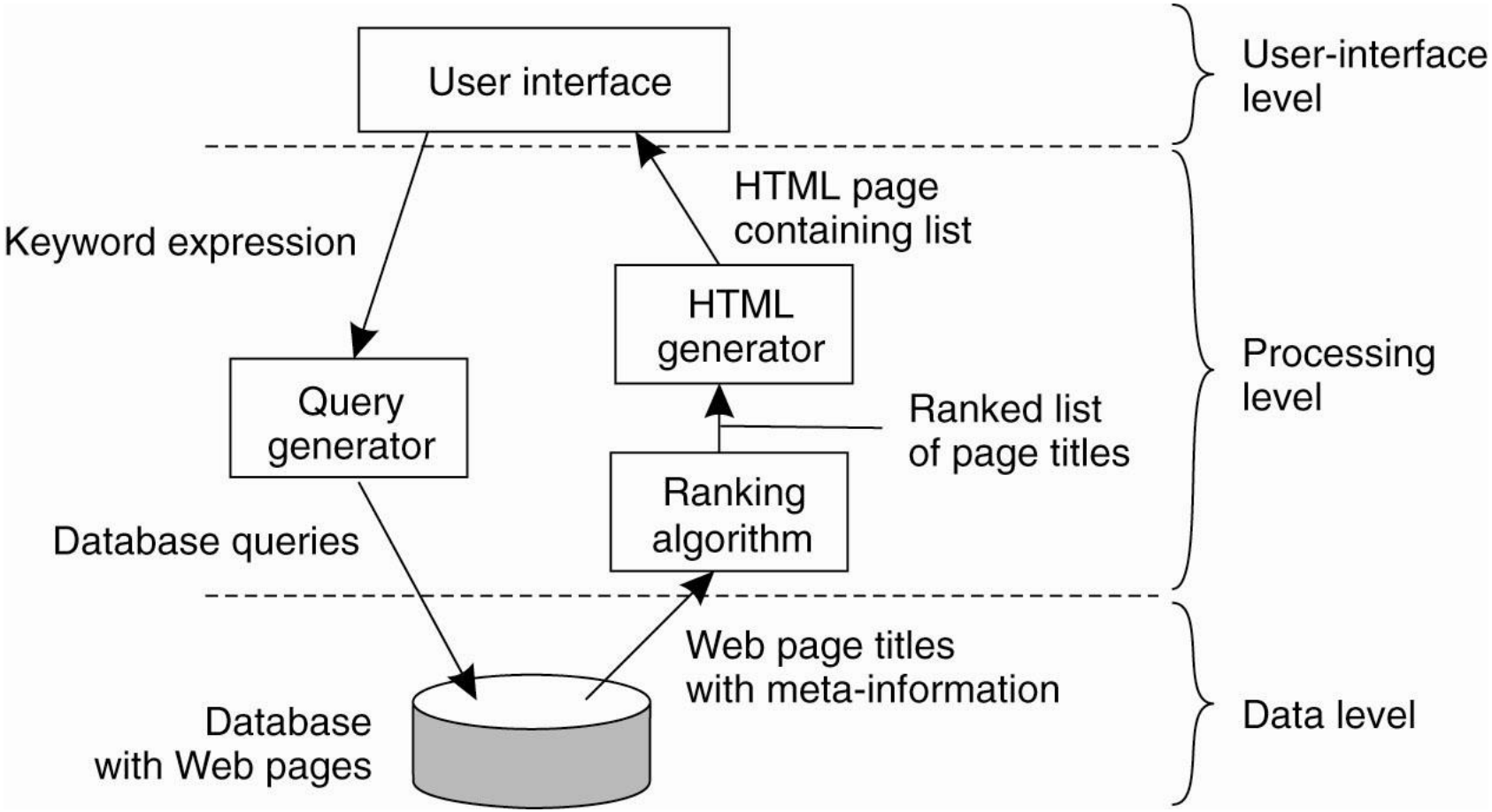


应用分层

- 应用程序本身的三个典型层
- 用户接口层
  - 用户接口层包含系统与用户直接交互的单元；例如：显示管理等；
- 应用处理层
  - 包含应用的主要函数，但是，不与具体的数据绑定；
- 数据层
  - 数据层管理应用使用的实际数据。

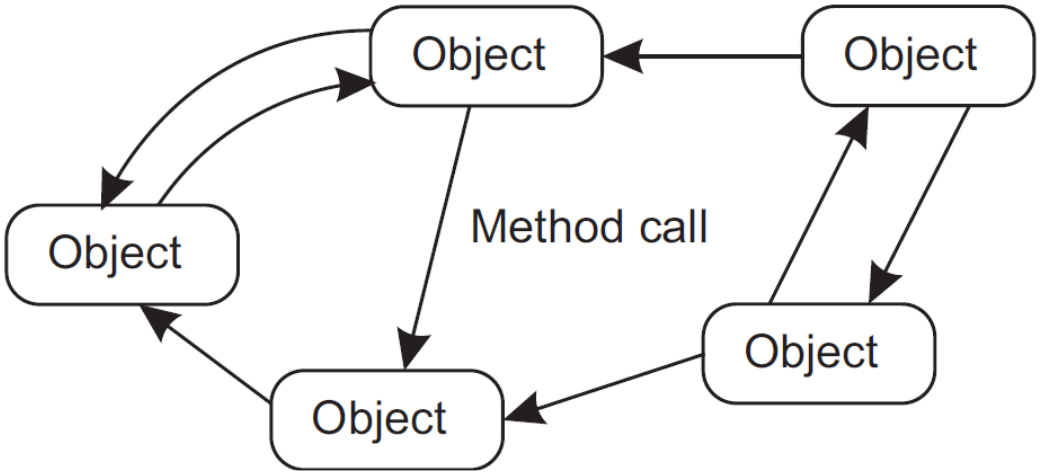
一个分层结构例子

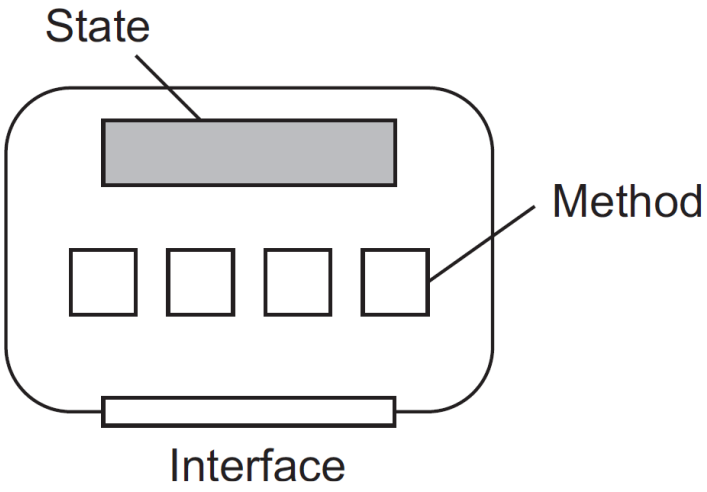
The simplified organization of an Internet search engine



基于对象的架构样式

- 主要用于分布式对象系统。
- 组件是由通过过程调用相互连接的对象构成。
- 对象可以放置在不同的机器上，调用可以跨网络执行。
- 对象封装了数据，提供面向数据的方法（屏蔽内部实现）。





基于对象的体系结构

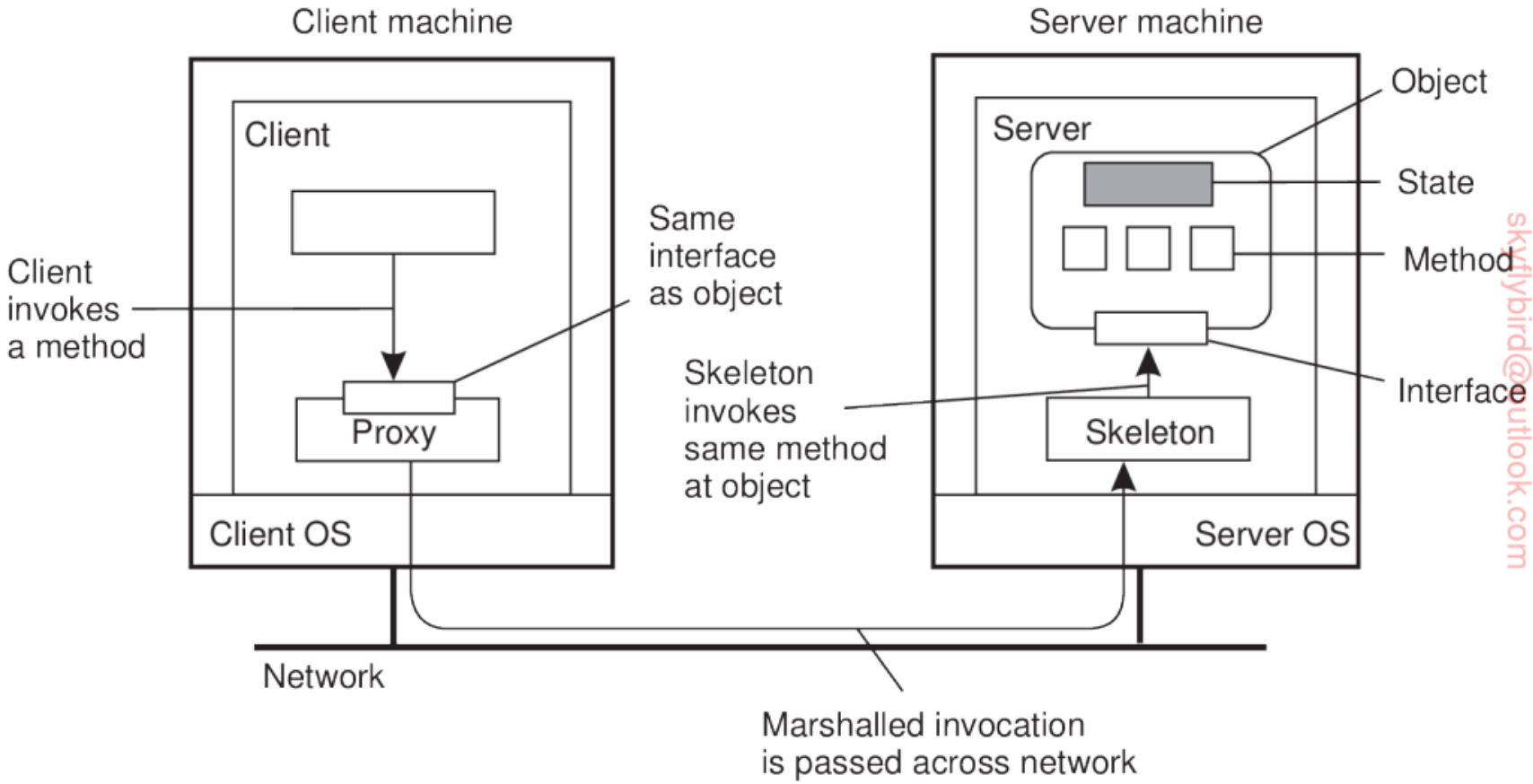
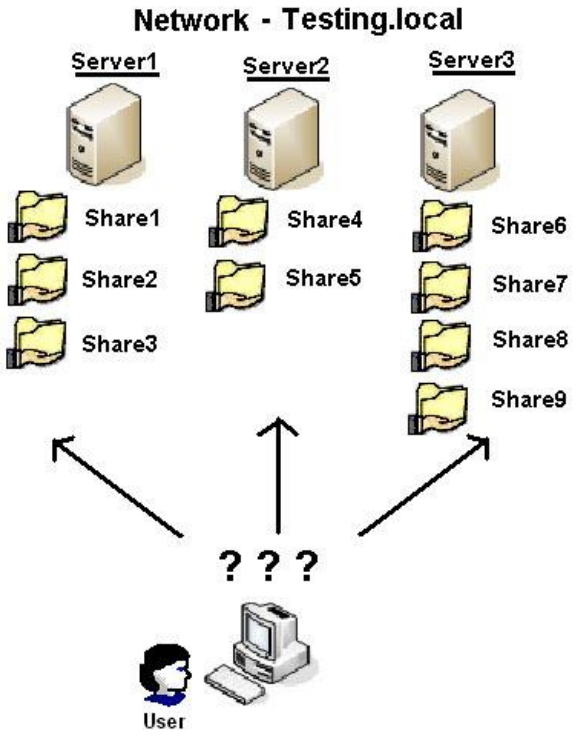


Figure 2.6: Common organization of a remote object with client-side proxy.

基于资源的架构样式（直接API访问）

- 广泛用于Web系统。
- 将分布式系统看做资源的集合， 由组件独立管理。
- 这些资源可以由应用程序增删查改。



Hardware sharing example

基于资源的架构样式（直接API访问）

- **REST (Representational State Transfer) 风格**
- RESTful API
  - 资源可以通过命名机制标识；
  - 所有的服务提供相同的接口；
  - 从一个服务发出或者传入的消息是完全自描述的；
  - 执行完操作后，执行组件不再记录调用者的信息。

Basic operations

Operation	Description
PUT	Create a new resource
GET	Retrieve the state of a resource in some representation
DELETE	Delete a resource
POST	Modify a resource by transferring a new state

RESTful API vs. Traditional API

- 传统API方法接口数量多

Amazon S3 SOAP interface

Bucket operations	Object operations
ListAllMyBuckets	PutObjectInline
CreateBucket	PutObject
DeleteBucket	CopyObject
ListBucket	GetObject
GetBucketAccessControlPolicy	GetObjectExtended
SetBucketAccessControlPolicy	DeleteObject
GetBucketLoggingStatus	GetObjectAccessControlPolicy
SetBucketLoggingStatus	SetObjectAccessControlPolicy

- RESTful API，接口少、标准化
  - 代价：与资源相关的操作的参数空间设计复杂
  - 如：上下文有关的操作，需要参数信息弥补

Pub-Sub架构样式

	Temporally coupled	Temporally decoupled
Referentially coupled	Direct	Mailbox
Referentially decoupled	Event-based	Shared data space

- 分布式协同的不同模式
- 时间解耦：异步处理，不指定操作时间
- 引用解耦：访问没有具体指定目的节点
- Pub/Sub:引用解耦访问

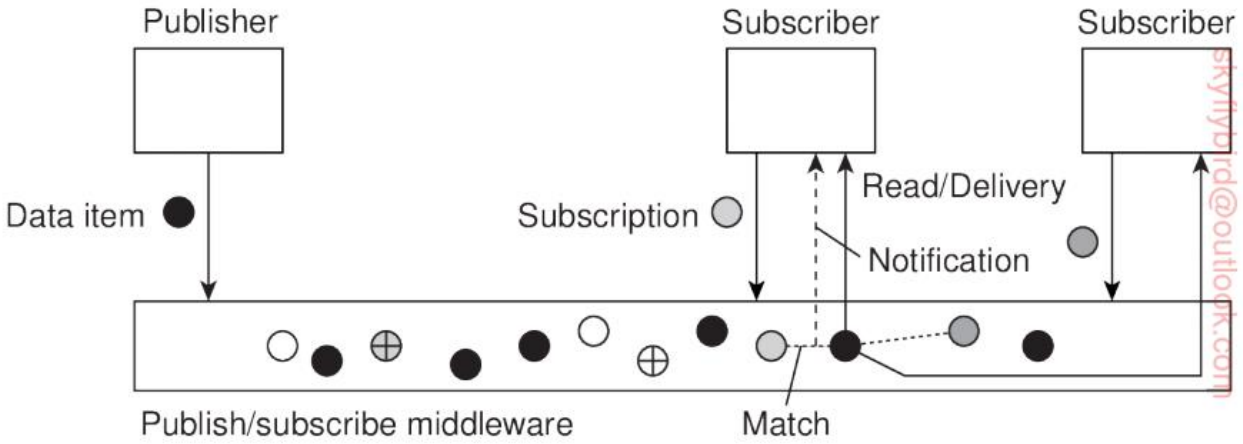


Figure 2.12: The principle of exchanging data items between publishers and subscribers.

Pub-Sub架构样式

- 采用Pub-Sub机制
- 信息异步交换、共享
  - 事件+数据

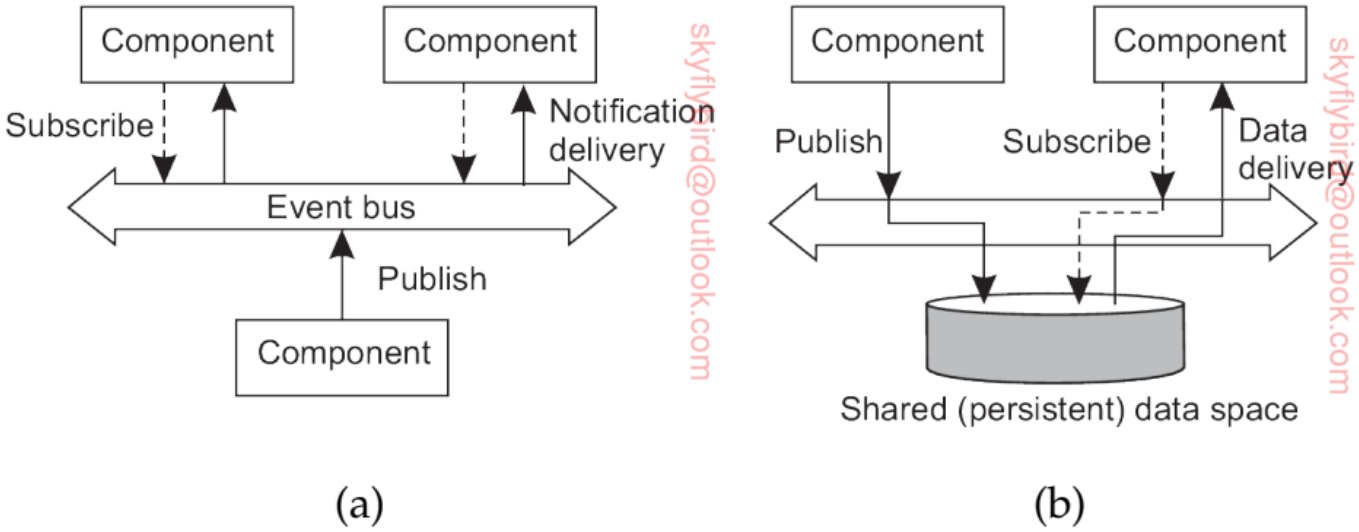
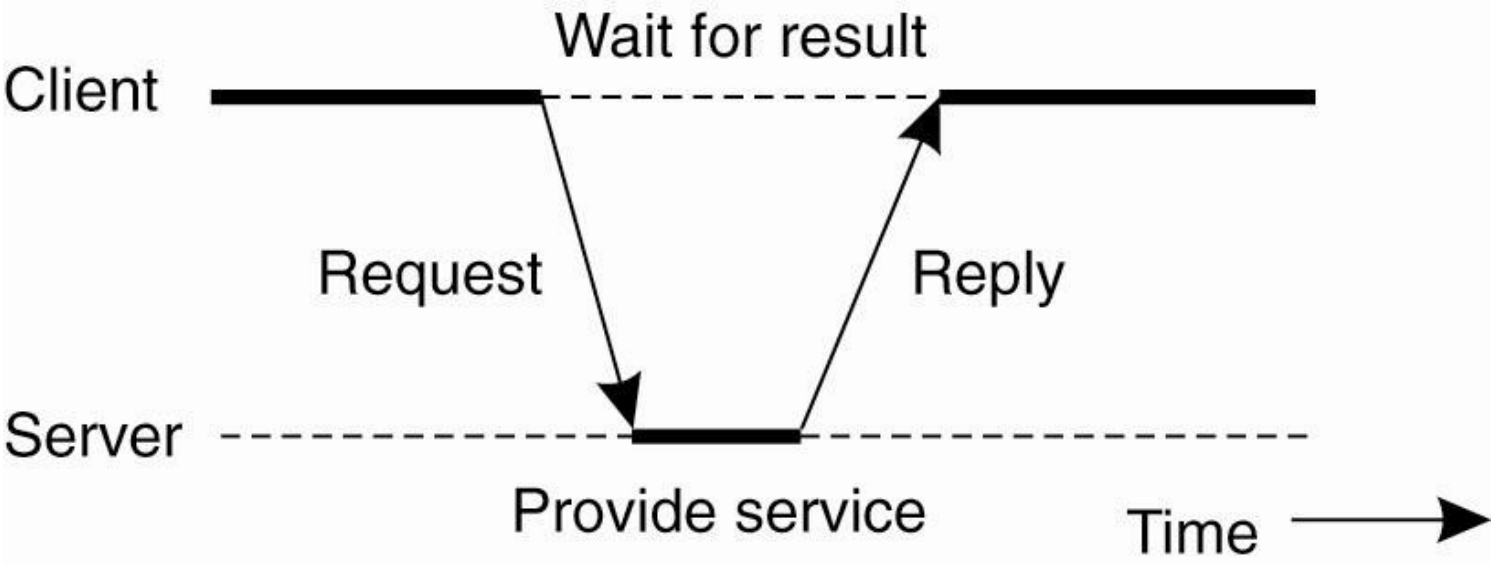


Figure 2.10: The (a) event-based and (b) shared data-space architectural style.

§2.2 不同类型体系结构

- Centralized architectures
  - Application layering
  - Multitiered architectures
- Decentralized architectures
  - Structured peer-to-peer
  - Unstructured peer-to-peer
  - Hierarchically organized peer-to-peer
- Hybrid architectures
  - Edge-server systems
  - Collaborative distributed systems

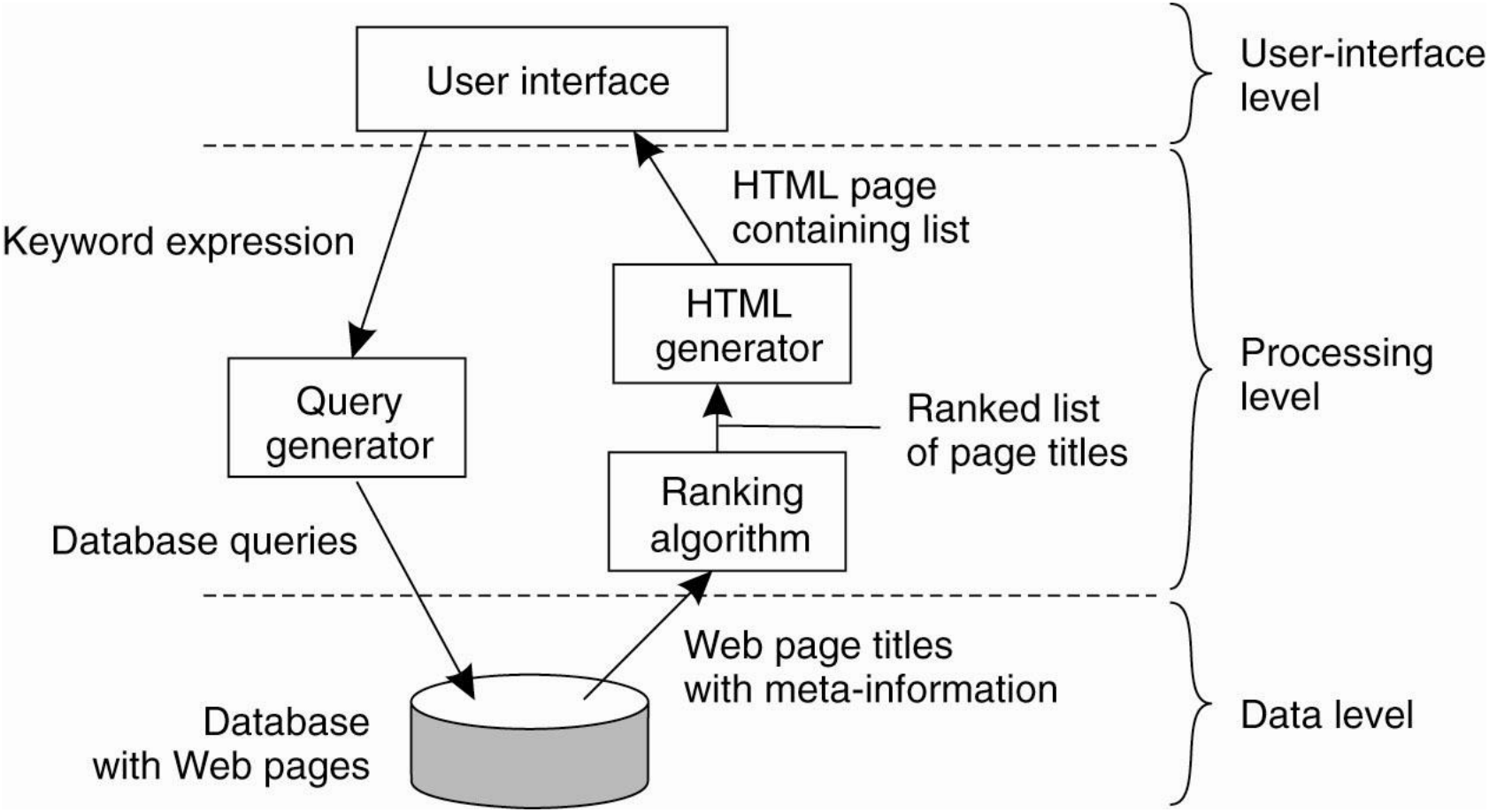
Centralized Architectures



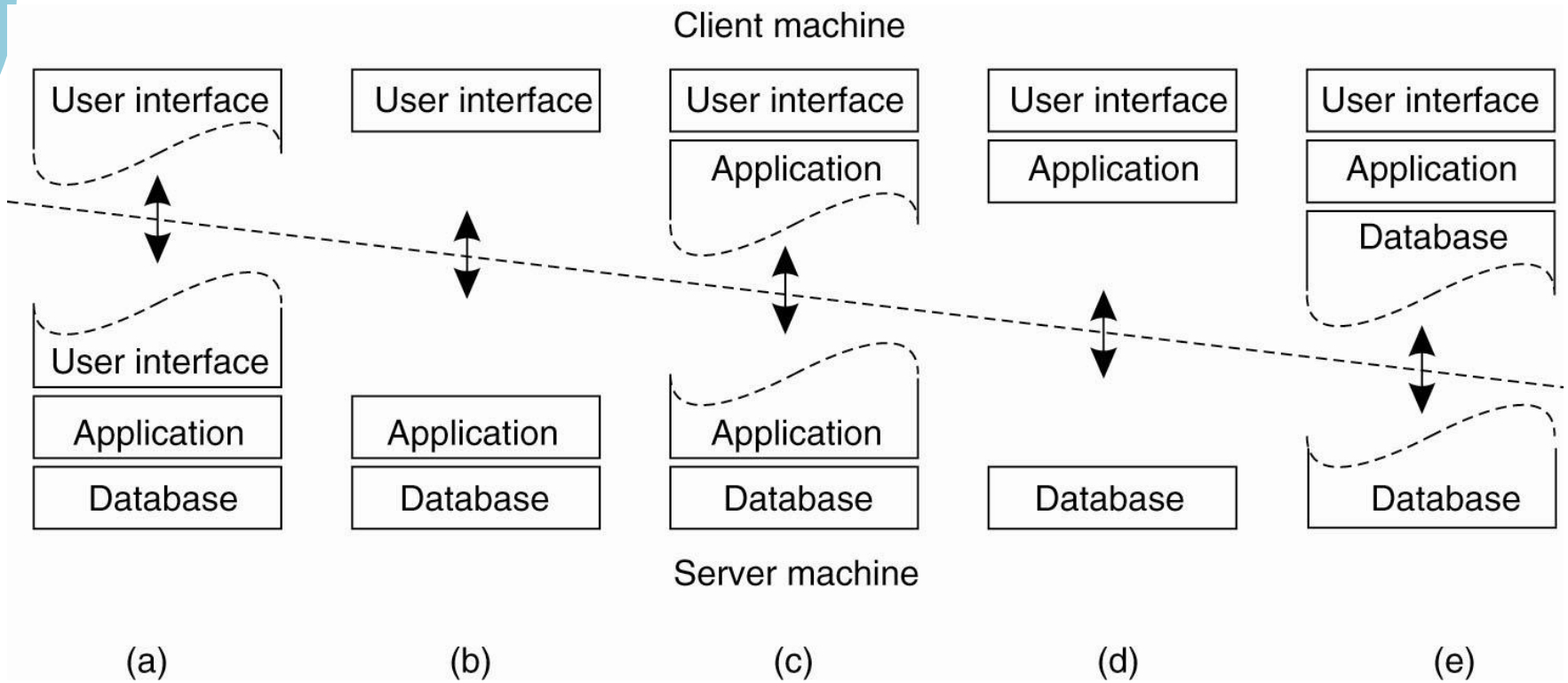
- 主流通信协议：
  - **HTTP、HTTPS： REST API (GET、POST…);**
  - **AJAX ：异步请求;**
  - **RPC： XMLRPC、SOAP、gRPC，Web服务通过API调用;**

Application Layering

- 经典架构：三层



Application Layering



Alternative client-server organizations (a)–(e).

Multitiered Architectures

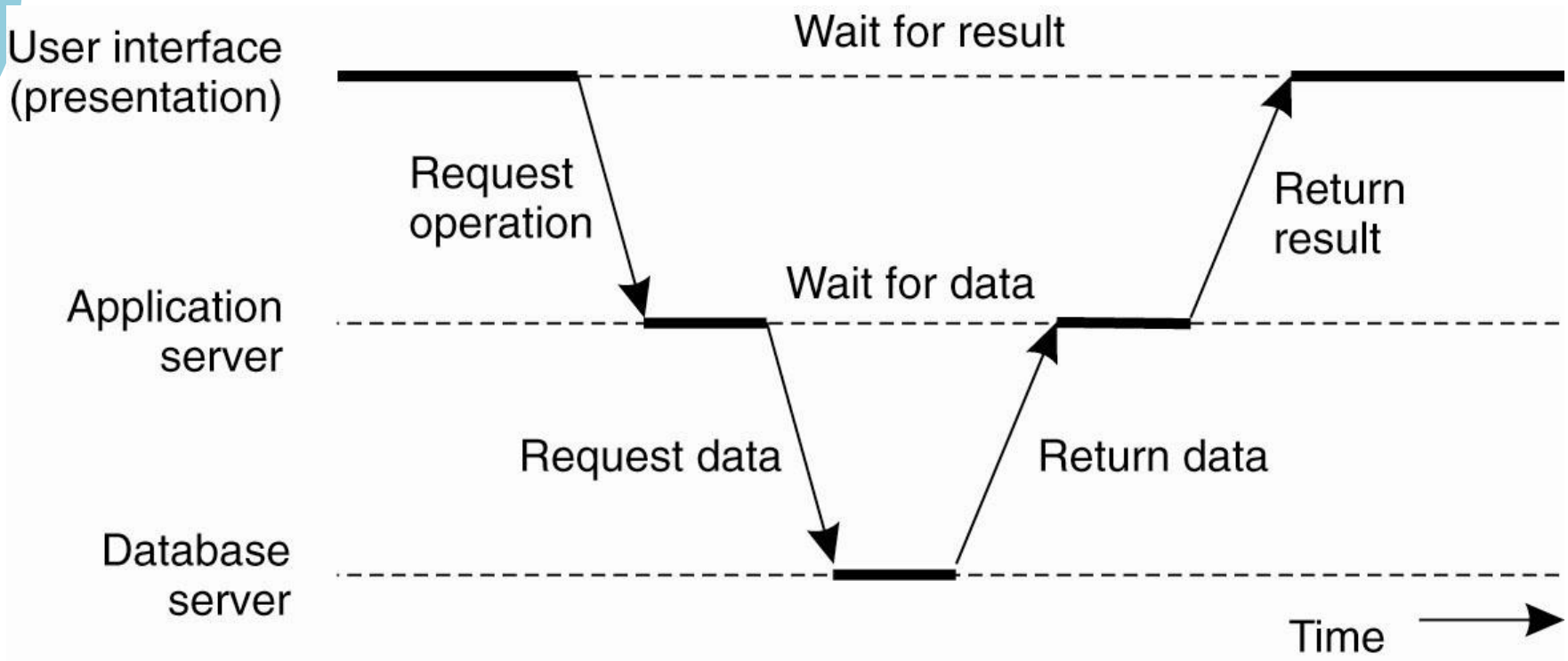
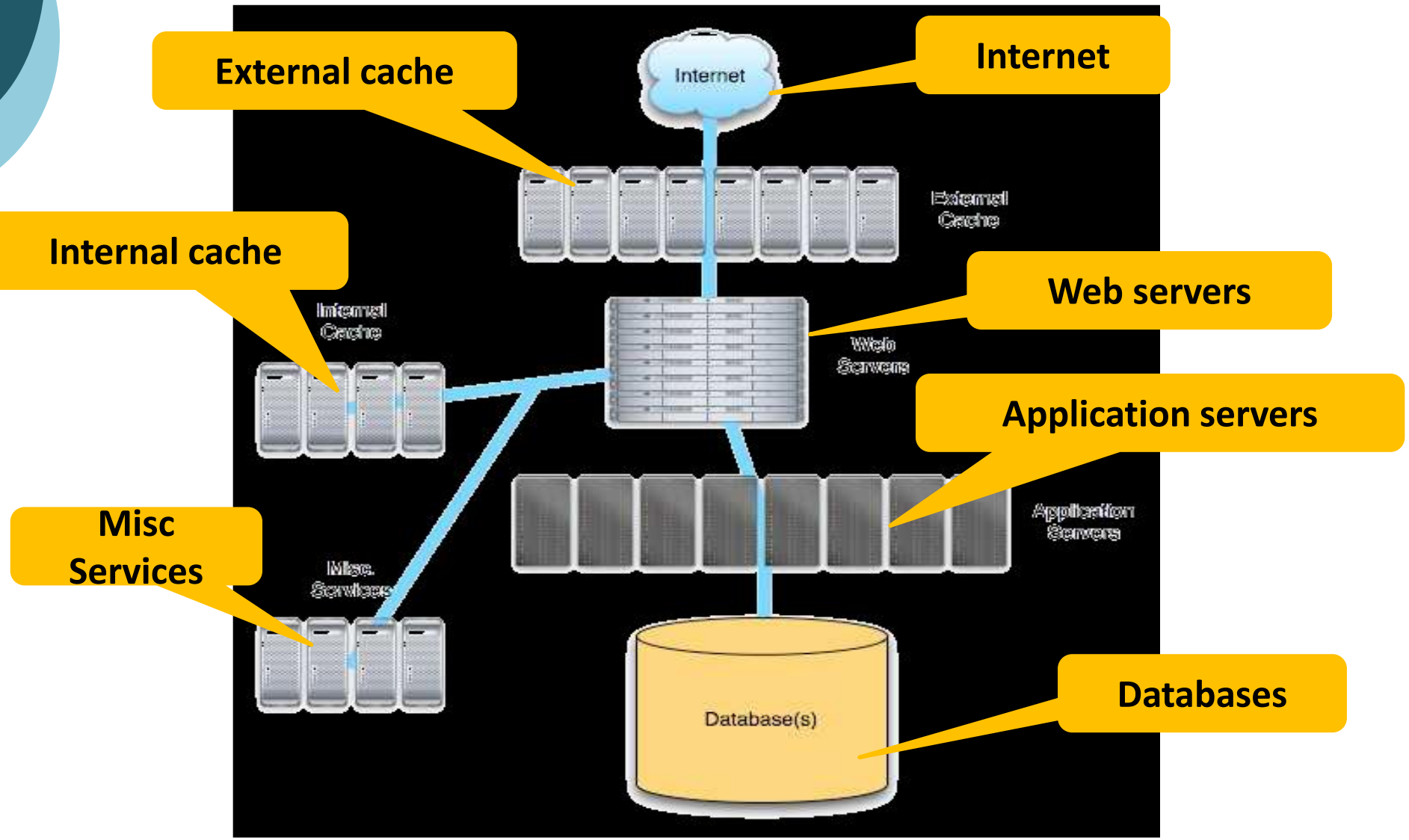


Figure 2-6. An example of a server acting as client.

Web服务系统主流架构



Decentralized Architecture

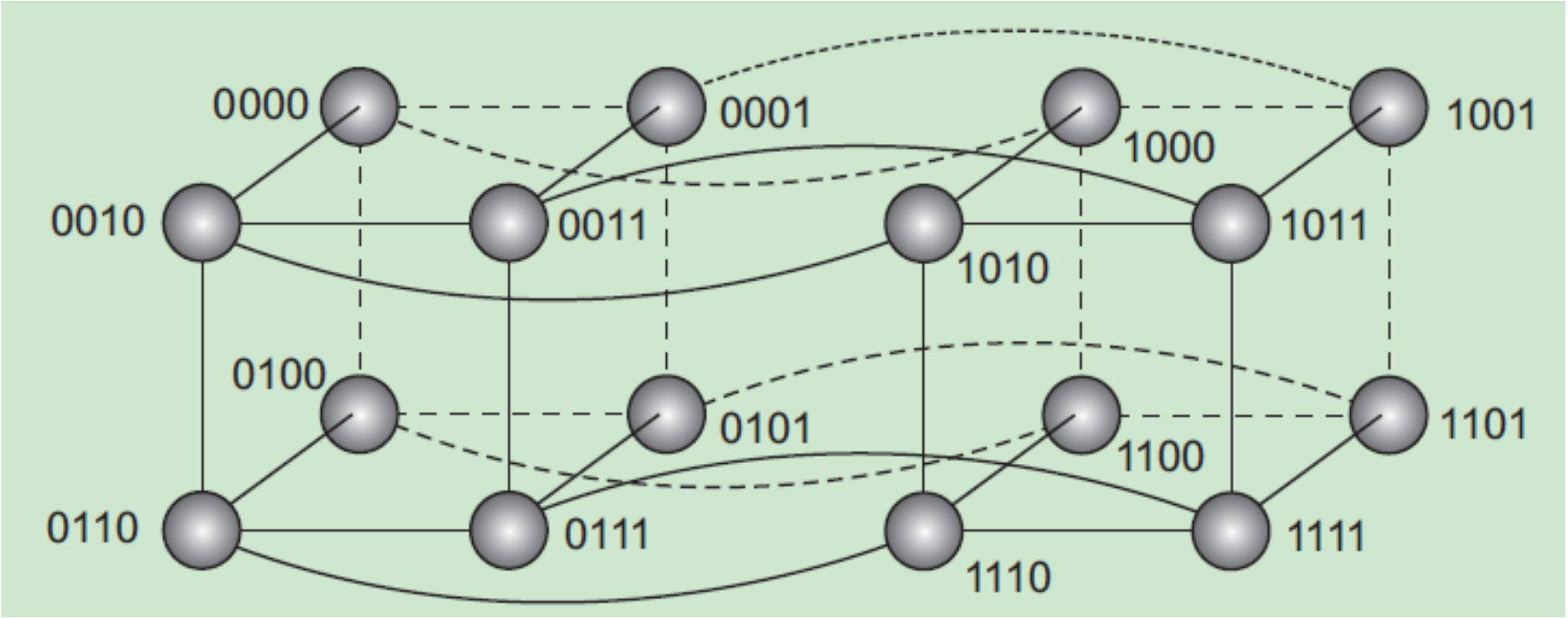
- Structured peer-to-peer
- Unstructured peer-to-peer
- Hierarchically organized peer-to-peer

Structured P2P Architectures

- 将节点组织在一个特定结构的覆盖网络中
  - 如环形结构
- 数据-健值-节点形成对应关系
- 语义无关的索引
- 最佳实践：利用Hash函数
- key(数据项) = hash(数据项的值)
- P2P系统用于存储键值对<key, value>数据

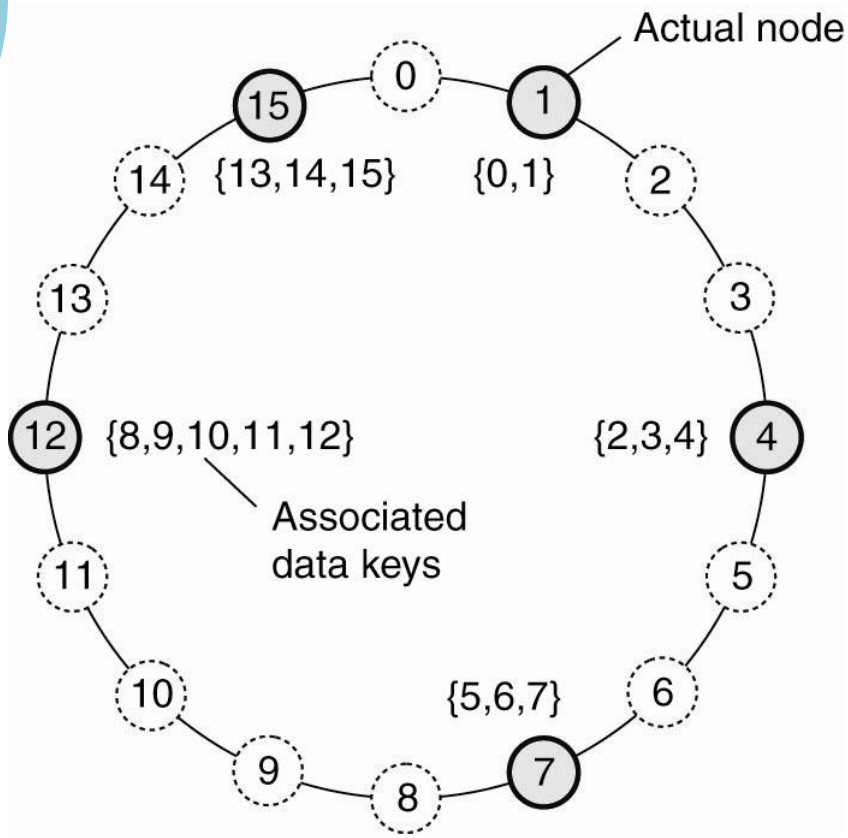
Structured P2P Architectures

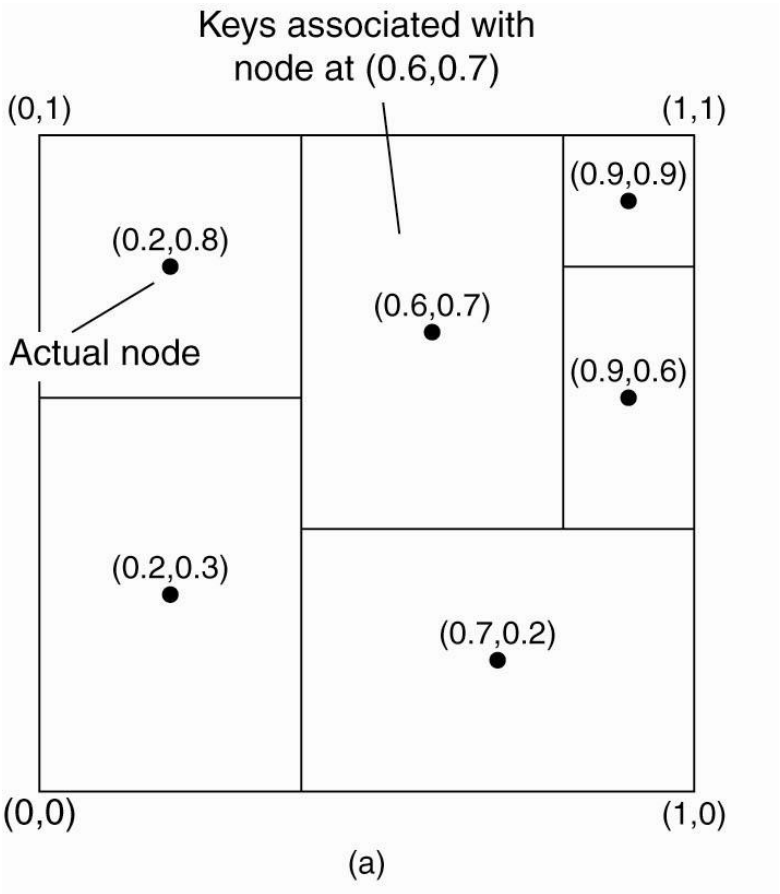
- 超立方体



查找数据键值为 **k** 的数据 **d**，意味着将请求路由到节点标识符为 **k** 的节点

Structured P2P Architectures

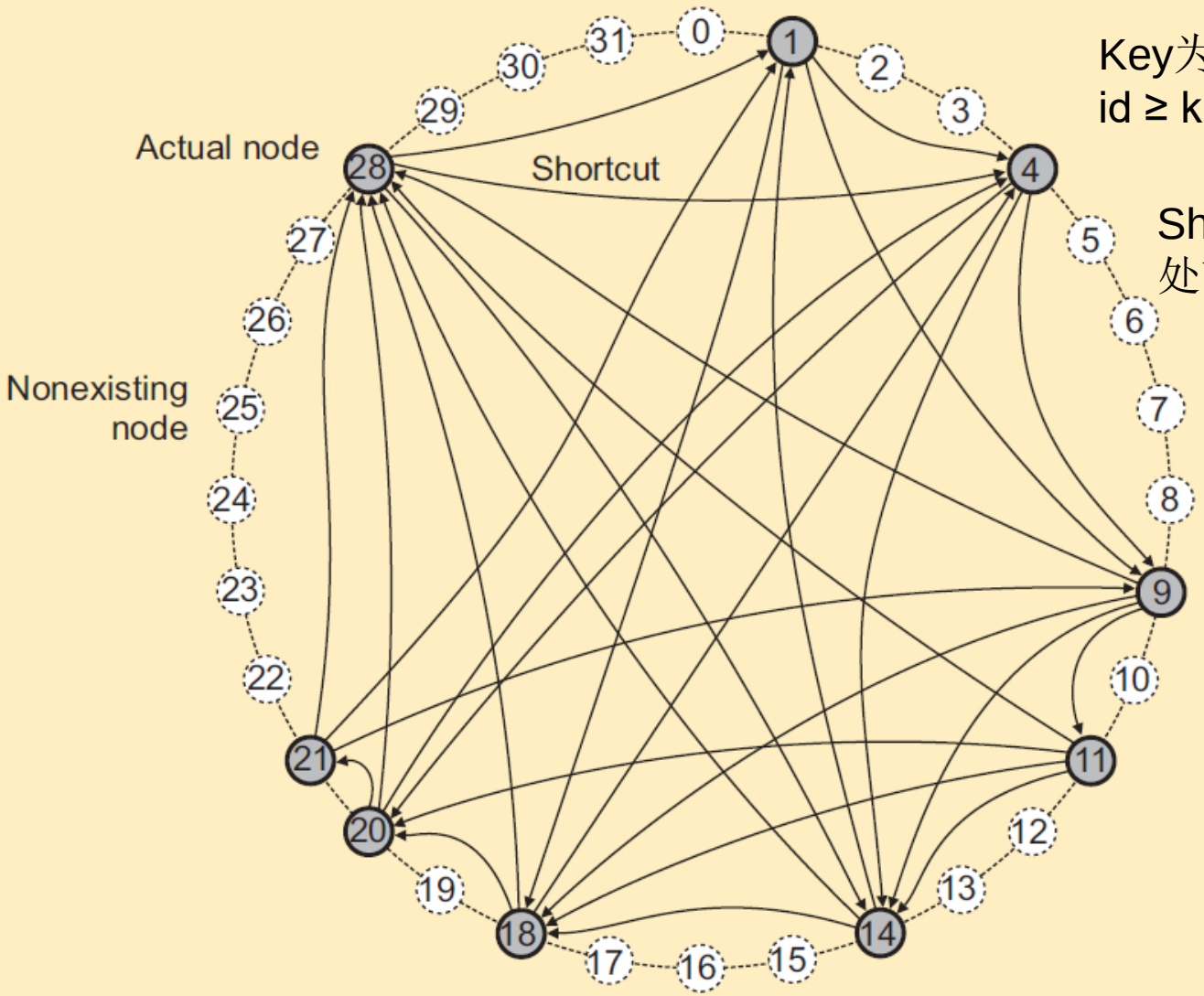




Data mapping in Chord and CAN.

Chord结构

Chord ring, with 5-bit key



Key为 k 的数据项→
id ≥ k的最小节点 succ(k)

Shortcuts: 跳跃指向远处节点，如succ(k+2^i)

Unstructured P2P Architectures

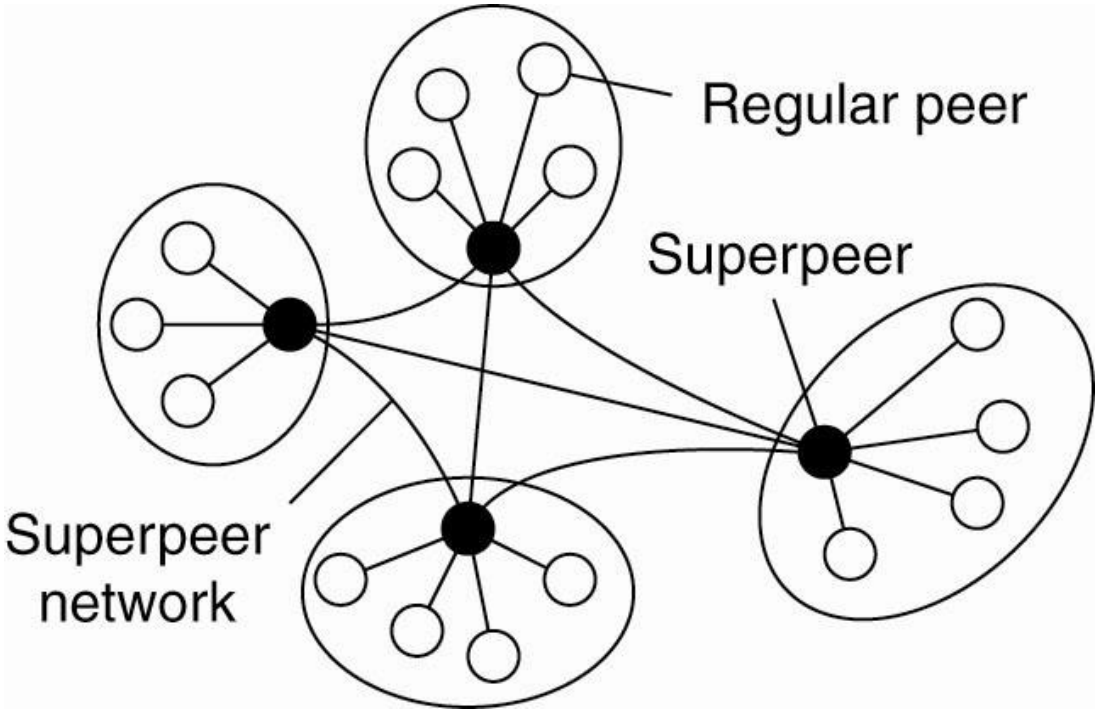
- The overlay network resembles a *random graph*.
- Each node maintains a list of *c* neighbors
  - each of which represents a randomly chosen *live* node from the current set of nodes
  - The list of neighbors is referred to as a *partial view*
- How to maintain a partial view?
  - Nodes are in *push* or *pull* mode. Using only one mode leads to isolated sub-networks so most nodes will do both (*exchange* mode)
  - To add to the group, simply contact any node. To leave, simply leave.

Unstructured P2P Architectures

- 非结构化P2P系统数据搜索方式
- 泛洪方式
  - 请求发出节点 u 会其向其所有邻居节点发出数据搜索请求。
  - 如果之前已收到节点请求，则请求会被忽略。否则，节点会在本查找数据项。
  - 请求会迭代传递下去。（有TTL限制、通信代价高）
- 随机游走
  - 请求发出节点 u 从其邻居节点中随机选择一个节点，然后进行本地搜索。
  - 如果没有完成，继续从其邻居节点中选择一个随机节点向下进行。（需要一种停止机制）

分层组织的P2P （Superpeers）

- 作用：
  - 提高数据项定位效率；
  - 打破点对点网络的对称性；
  - 设置代理程序；



Superpeers in Skype

Skype中 A 连接 B 的原理

➢A 和 B 均在公共网络上

- A 和 B 之间建立TCP连接用于控制报文的传输；
- A 和 B之间实际通过 UDP 协议在商定的端口之间通信；

➢A 在防火墙内，而B在公网上

- A 和 超级对等节点 S 之间建立TCP连接用于控制报文的传输；
- S 与 B节点建立TCP连接用于中继报文的传输；
- A 和 B之间直接通过UDP协议传输数据；

➢A 和 B 都在防火墙后面

- A 与在线的超级对等节点 S 通过TCP连接；
- S 通过 TCP协议与 B 连接 （用于控制报文交换）；
- 对于实际的呼叫，另外一个超级对等节点 R 作为中继节点：A 和 B 均会和 R 建立连接；
- 所有的音频数据通过 R 以及两个 TCP连接转发；

Hybrid Architectures: edge-server systems

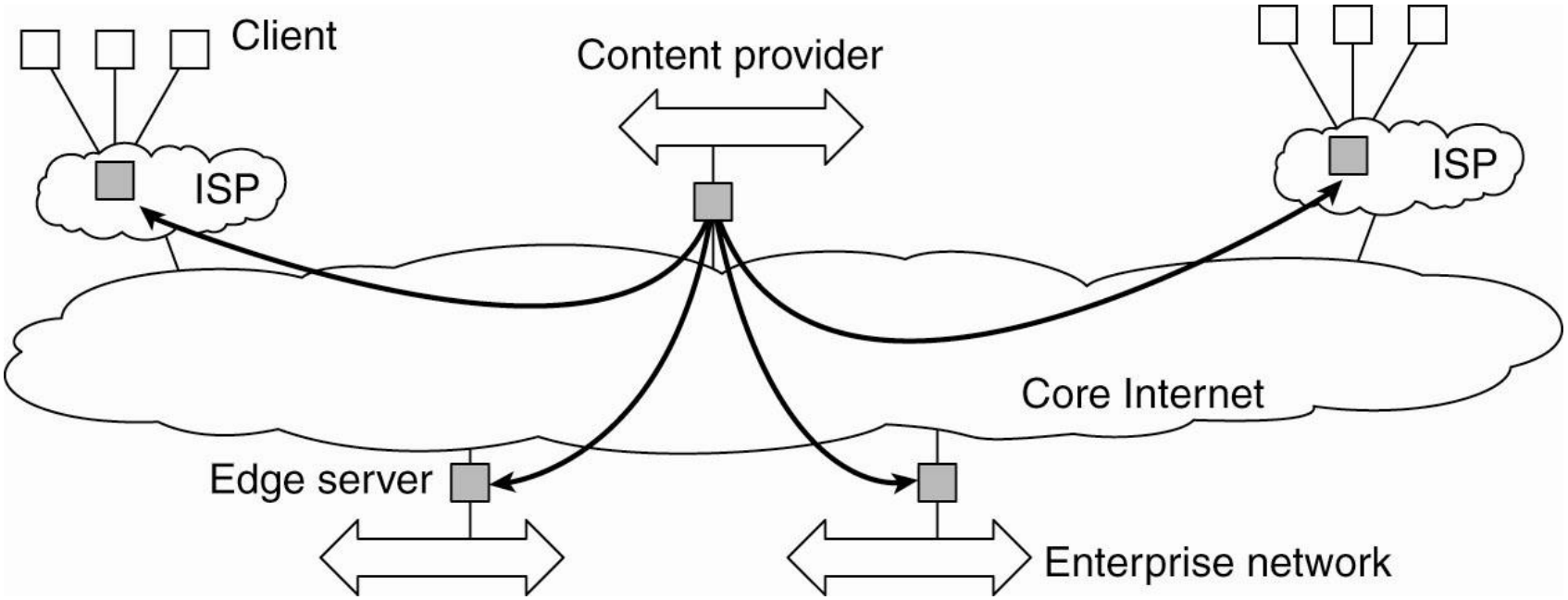
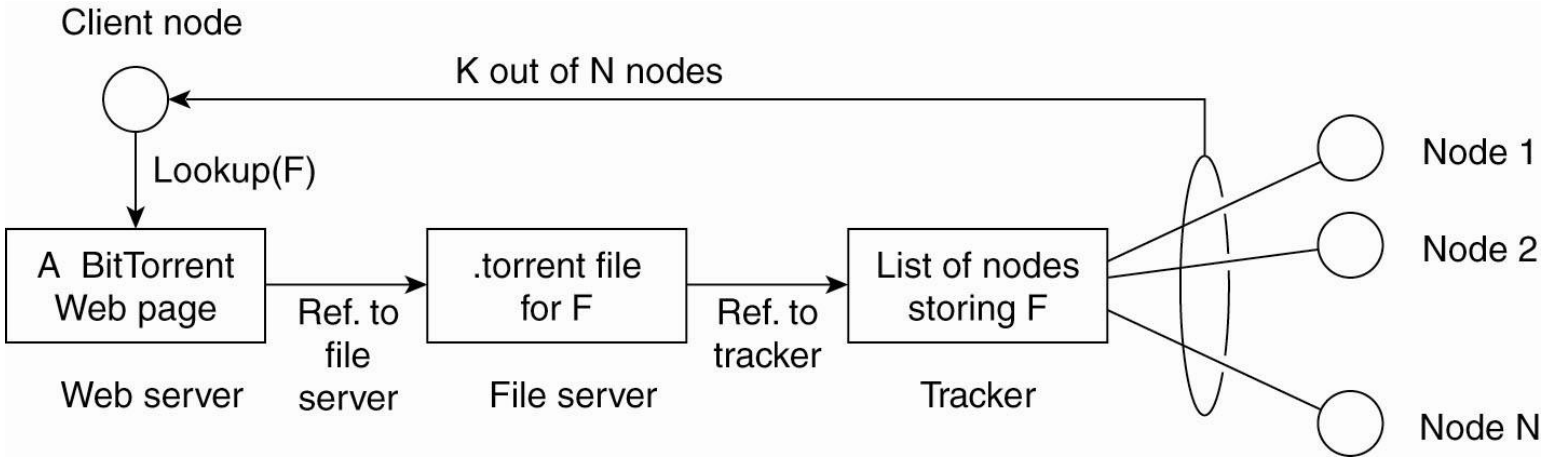


Figure 2-13. Viewing the Internet as consisting of a collection of edge servers.

在很多场景中，客户端-服务器架构是融合有非集中式模式的。
边缘服务器主要的目的是进行过滤和编码转换后提供内容服务。

Hybrid Architectures: collaborative sys.

- Client-Server + Decentralized
- To download a file, a user needs to access a global directory
  - one of a few well-known Web sites
  - Stores: a link to info. of active nodes.
- Then the user can download chunks from peer nodes



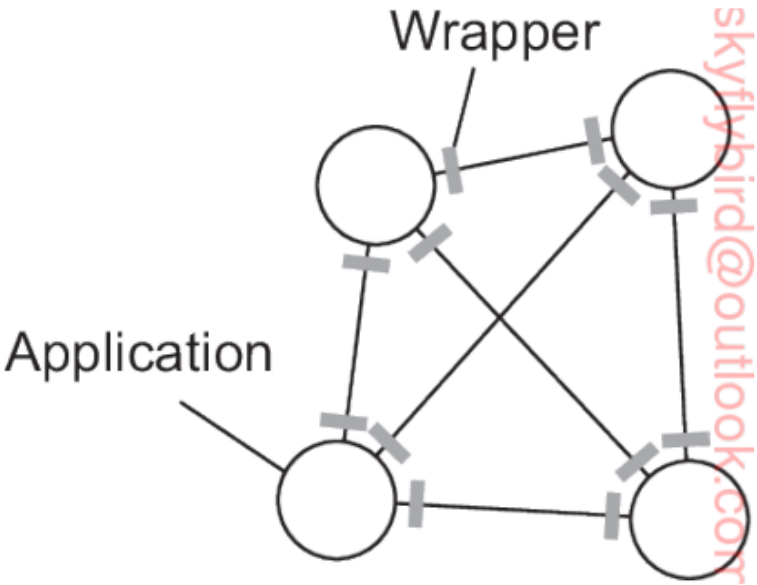
§2.3 体系结构与中间件

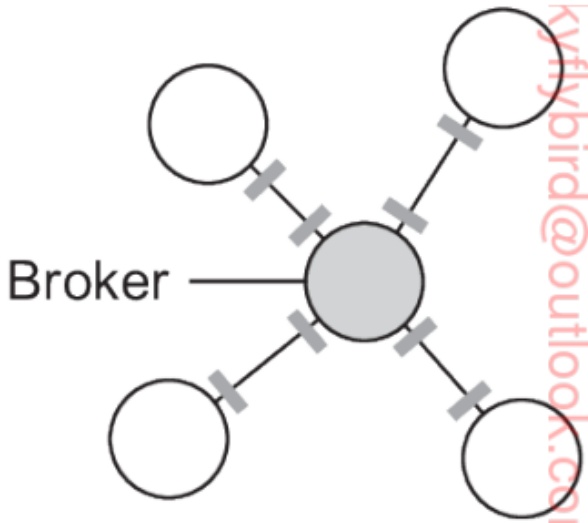
- 用于组织、粘合分布式系统不同部分的软件
- 实现Openness的重要手段
- Two major design patterns
  - Wrappers
  - Interceptors
- Modifiable middleware
  - Composing middleware at runtime

Adapters/Wrappers

- 适配器（装饰器、包裹器...）
  - 面向客户程序的特殊接口组件
  - 主要是解决接口不兼容的问题
- 例子：Amazon’s S3 storage
  - 由Web server接收处理用户的RESTful requests并转交给内部的S3服务器
- Specialized Wrapper
  - 面向特定client，扩展性差
- 中间件化的Wrapper
  - 通用性好，可扩展
  - 实际实现形式：代理（Broker），如消息代理

Wrapper vs. Broker





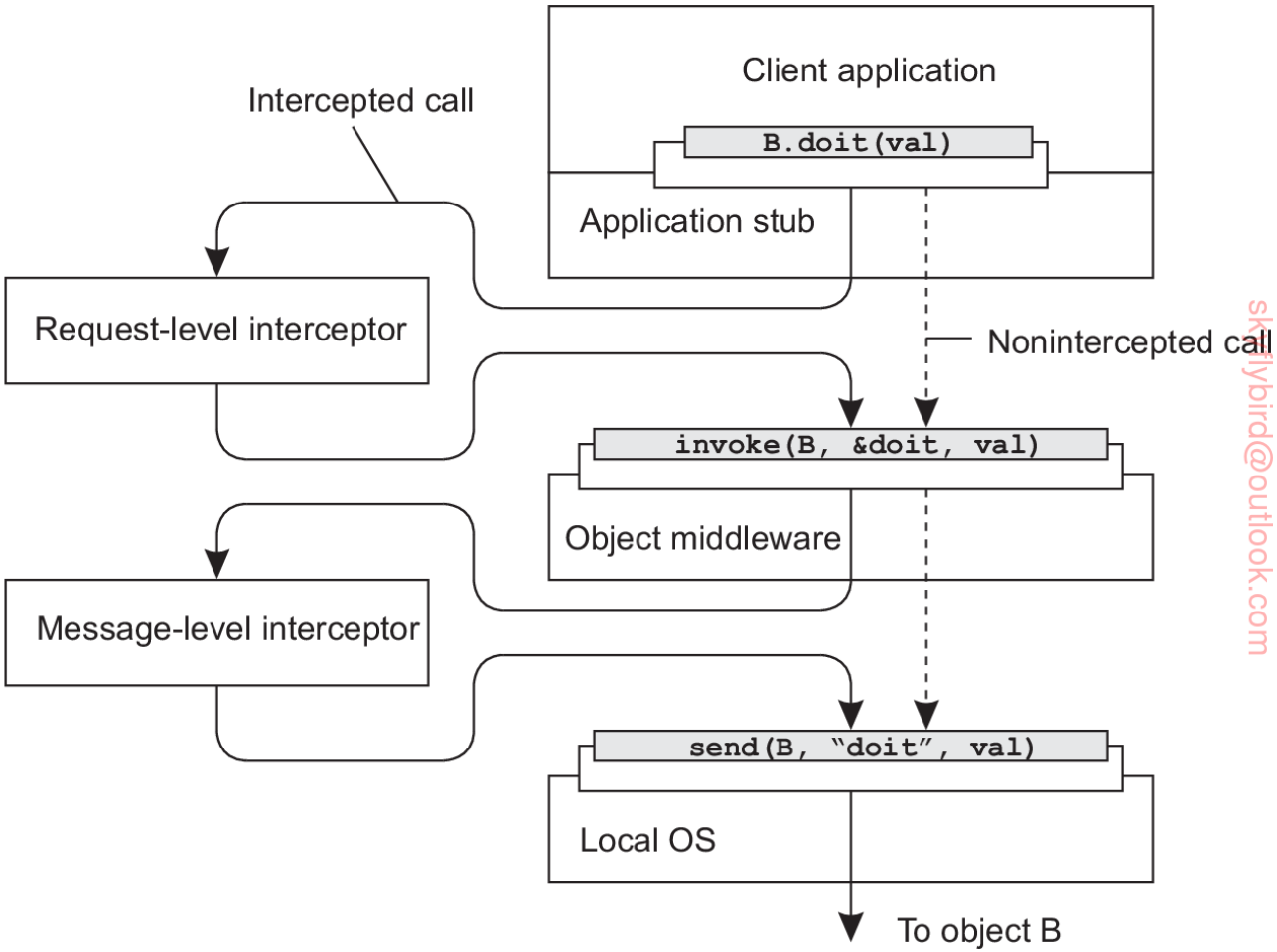
skyflybird@outlook.com

Interceptors

- 中断器：分布式系统级的“中断”
  - A software construct that will break the usual flow of control and allow other (application specific) code to be executed
- 实现通用性中间件执行应用程序的个性化操作
- 通常用于分布式对象系统

Interceptors

- Interceptors handle remote-object invocations
- Request level + message level



skflybird@outlook.com

Modifiable中间件

- 自适应中间件
- 要点：
  - 把要实现功能的部分与非功能部分如可靠性等分开如面向方面的编程，annotation （注解）
- 实现技术：计算反射
  - 让程序在运行时刻检查自己的行为，如果有必要，调整其行为
  - Java、Python、C#等编程语言均支持这一功能
- 基于组件的设计
  - 通过组件的不同组合来支持自适应。 系统可以在设计时静态配置，或者是在运行时动态配置

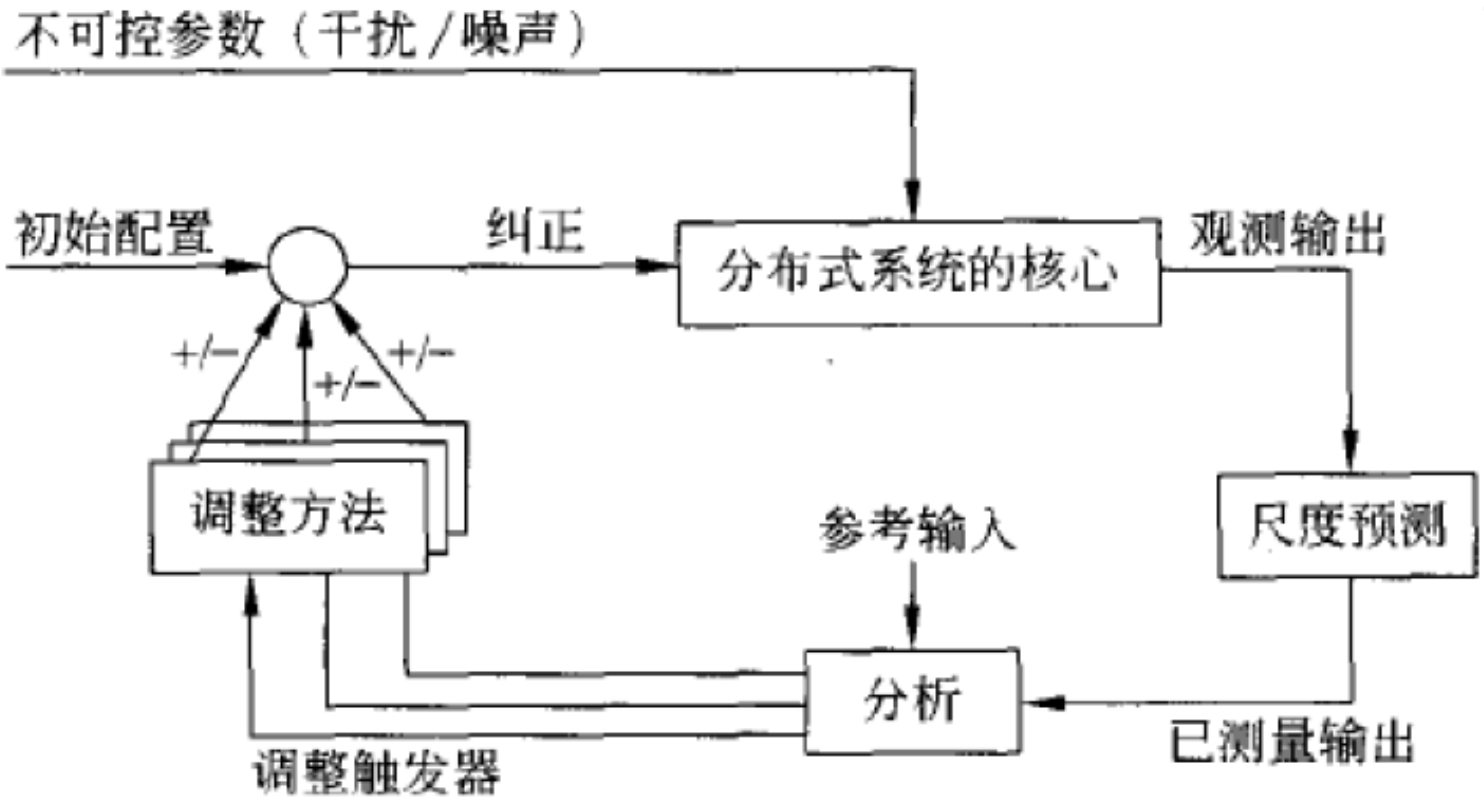
分布式系统自我管理

- 目标：自主计算
  - 自我管理；
  - 自我恢复；
  - 自我配置；
  - 自我优化；
  - 自我…
- 在需要完成自适应功能时， 系统架构和软件架构之间的界线逐渐模糊

分布式系统自我管理

- **反馈控制循环**

– 自主计算往往通过反馈控制循环实现自适应性



Summary

- 分布式系统体系结构的样式
  - 分层、面向对象、资源直接访问、事件驱动、数据共享
- 不同类型的分布式体系结构
  - Centralized：C-S、分层、多层
  - Decentralized：结构化P2P、非结构化P2P、混合P2P
  - Hybrid：C-S+Edge Svr、C-S+P2P

谢谢！

wuweig@mail.sysu.edu.cn