CompSci 131

Parallel and Distributed Systems

Prof. A. Veidenbaum

Today's topics

- Last Lecture Summary
- Software Architectures

- Reading assignment:
  - Today's: Sec. 2.1-2.2
  - Next time: 2.2-2.4
  - And the lecture after: 3.1-3.2
    - Complete the assignment *before* next class

Review of last lecture

- Types of Distributed Software
  - DCS
  - DIS
  - Pervasive systems

DS Architectures

- A *software* architecture defines how the sftw works
  - Described in terms of components
    - Their organization, connection, interaction and data exchange
- A *system* architecture defines software instantiated on a real system
  - After final configuration choices were made
- Components are modules with interfaces that are
  - Required to be provided
  - Well defined
- Components are *replaceable*
  - *Possibly while the system is running*

Architectures

- Connectors are mechanism for mediating interaction
- Components and connectors form an architecture
- There are many architectural styles, some of them are
  - Layered architectures
  - Object-based architectures
  - Resource-centered architectures
  - Event-based architectures
- Styles may be combined in a given architecture

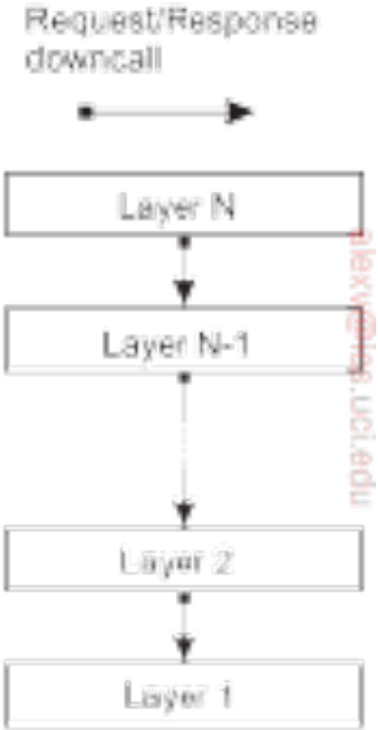
Layered Architectures

- Components are organized into layers

- Layer L_i can interact *only* with layers L_{i-1} and L_{i+1}
  - – I can call components of l-1, but not the other way around

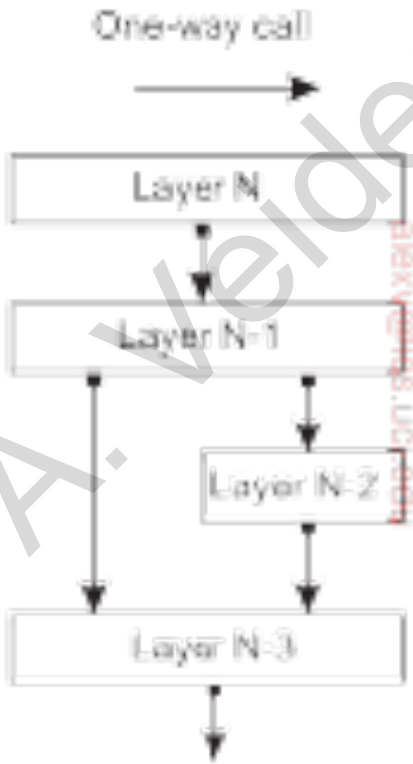
- Requests flow down, responses up
  - – Downcall/upcall

- An example: network protocol stack
  - – The OSI model



Layered Architectures

- There are more complex variants
  - Allowing different paths through components/layers
  - Allowing lower layers to make upcalls



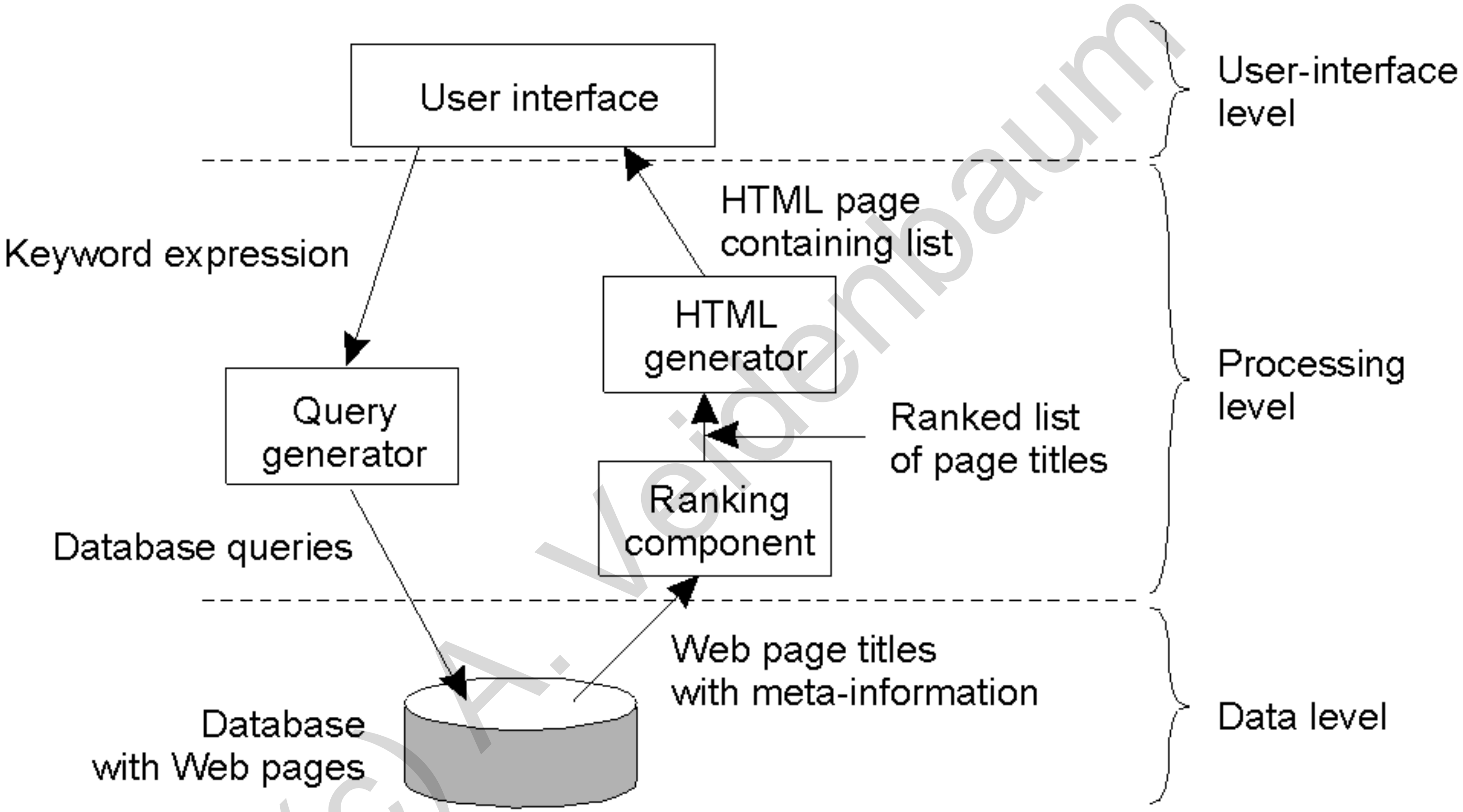
Application Layering

- Many distributed apps support access to DBs
  - By a user or by another app

- A layered approach proposed for this is to have three logical levels
  - The application-interface level
  - The processing level
  - The data level

- One example – an Internet search engine

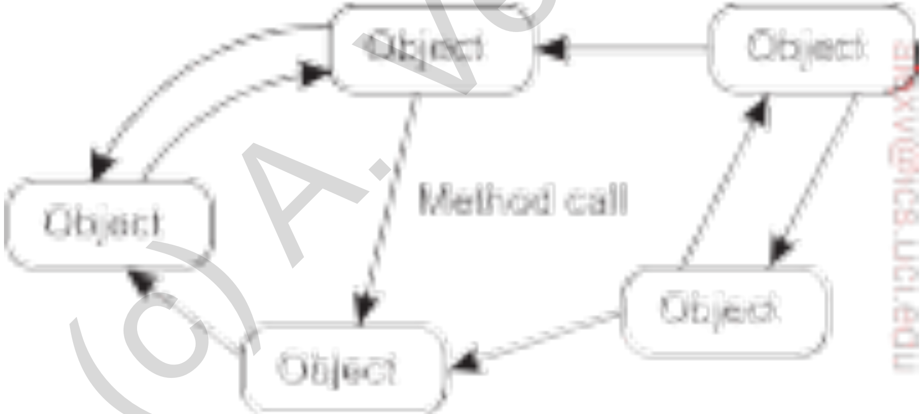
Application Layering Example



The internet search engine (simplified)

Object Oriented Architectures

- Components are objects in OOA
  - Encapsulating data, defining interface
- They export functions or methods
  - Thus separating interface from implementation
    - » This allows distributed implementations
- Connected through a remote procedure call (RPC)
  - RMI for methods

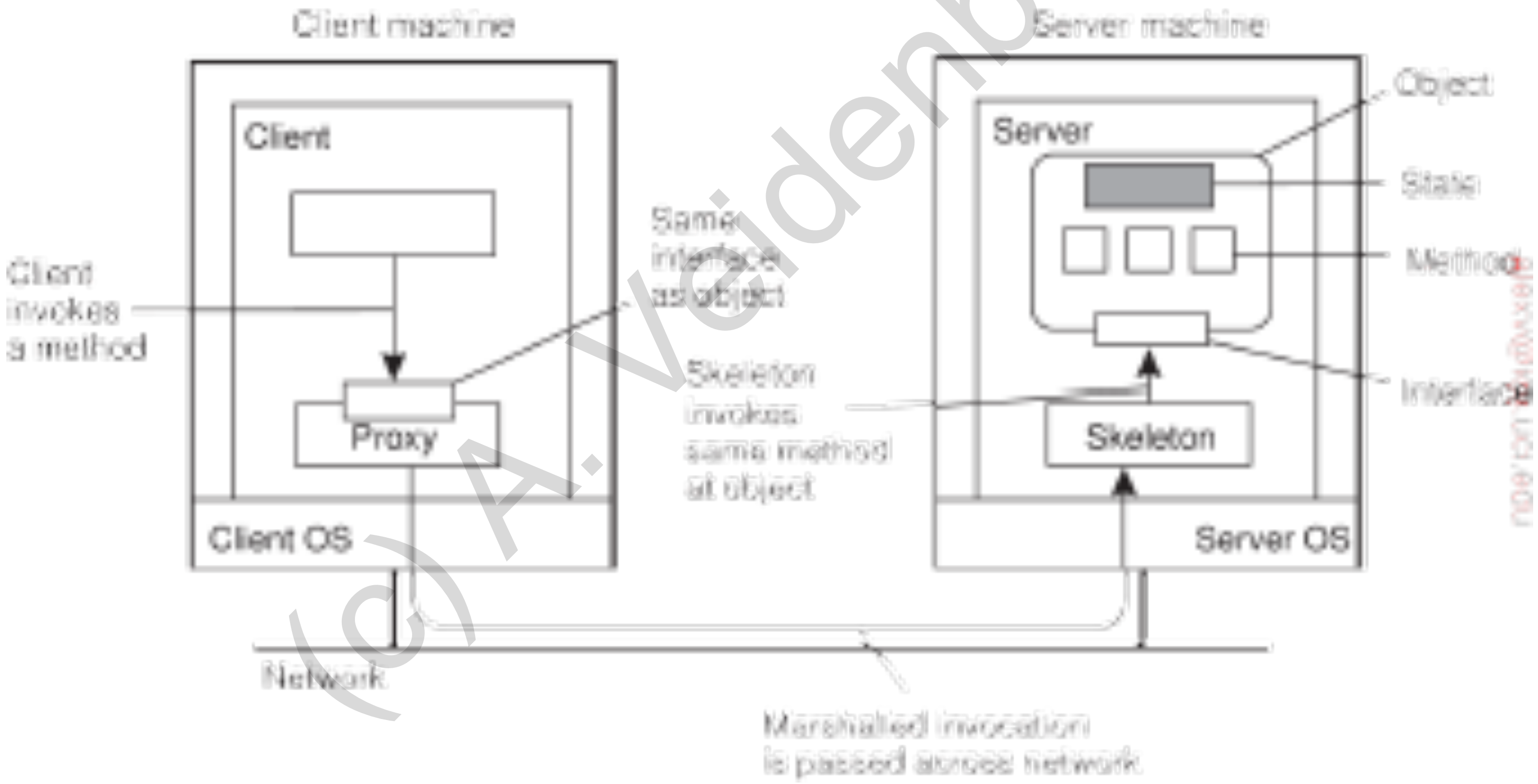


Object Oriented Architectures

- Allows more flexible connections than in layered
- One of the common styles for building large systems
- Objects themselves can be distributed!
  - Data partitioned among nodes
  - Functions/methods on different nodes

Distributed Objects

- **Interface on a client system, object on a server**
  - **Client system installs a proxy, server a skeleton (stub)**
  - **Really just a *remote* object**



Service Oriented Architectures

- Objects can be thought of as providing a way to encapsulate services
- An SOA is just a composition of different services
- Composition of services becomes harder as the number of services grows
  - A problem similar to enterprise application integration

Resource-based Architectures

- The Web has too many services to be an SOA
- Instead, one can think of a DS as providing resources
  - Managed by components
- REpresentative State Transfer (REST) is one such approach
- A RESTful architecture has a number of unique features

RESTful architecture features

- Resources are identified through a single naming scheme
- All services offer the same interface
  - consisting of at most four operations
- Messages sent to or from a service are fully self-described
- A component forgets everything about the caller after executing an operation at a service
  - Aka stateless execution

RESTful operations

1. PUT
   - Create a new resource
2. GET
   - Retrieve the state of the resource
3. DELETE
   - Remove the resource
4. POST
   - Modify a resource by transferring a new state

Example – Amazon S3

- Has two resources:
  - Objects, which are files
  - Buckets, which are directories (non-nested)
- Uses URIs (via http)
  - http://BucketName.s3.amazonaws.com/ObjectName
- Has equivalents of PUT, GET for buckets, objects
- Has another, more traditional interface – SOAP
  - 16 operations, with variants of REST operations

Publish-subscribe architectures

- A system is a collection of processes
  - Operating autonomously, joining dynamically
- For scalability reasons, many such systems separate processing from coordination
  - Minimizes dependencies between processes
- Coordination is communication and cooperation
  - binds processes together
- Coordination models have two aspects
  - Referential, coupled by explicit references
  - Temporal, coupled by simultaneous operation

Coordination taxonomy

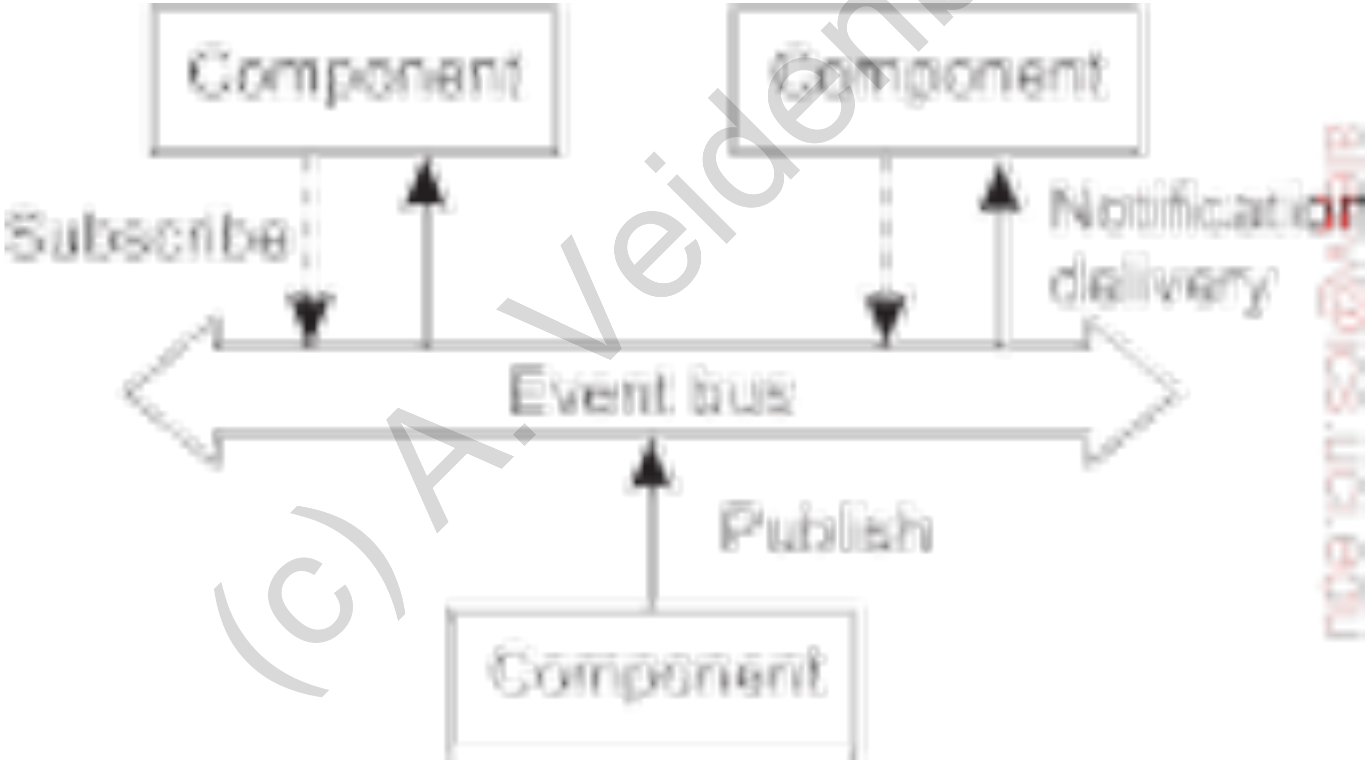
	Temporally coupled	**Temporally decoupled**
Referentially coupled	Direct	Mailbox
Referentially decoupled	Event based	Shared data Space

- **Direct – mobile telephony**
- **Mailbox – exchange data**
- **Event based – no direct identification**
- **Shared data spaces – tuple space access**

- **The last two are publish subscribe architectures**

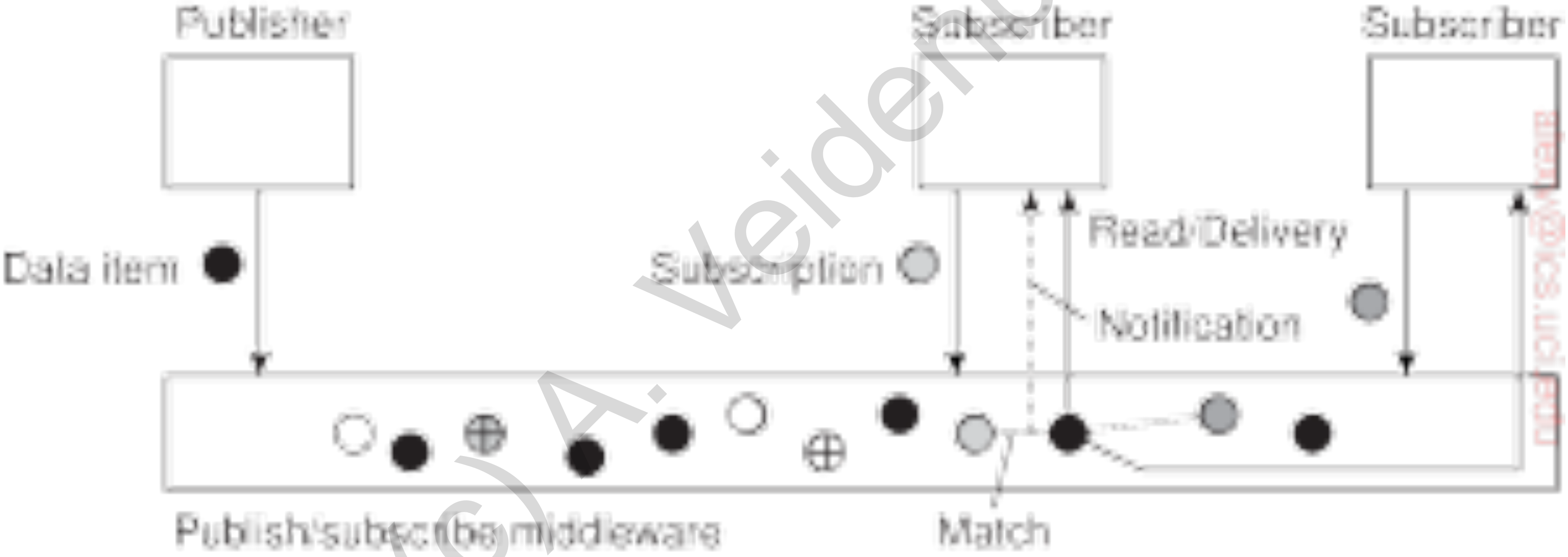
Publish-subscribe architectures

- An "event bus" is a mechanism for matching publishers and subscribers
  - A process publishes a notification – makes it known to all
  - A process subscribes to events it is interested in



Publish-subscribe middleware

- It keeps track of subscriptions and publications
- Forwards published data to waiting subscribers


Publisher
Subscriber
Subscriber
Data item
Subscription
Read/Delivery
Notification
Publish/subscribe middleware
Match