Introduction Distributed Systems Architecture

Today

- Software and system architecture
- Adaptation and self-adaptation

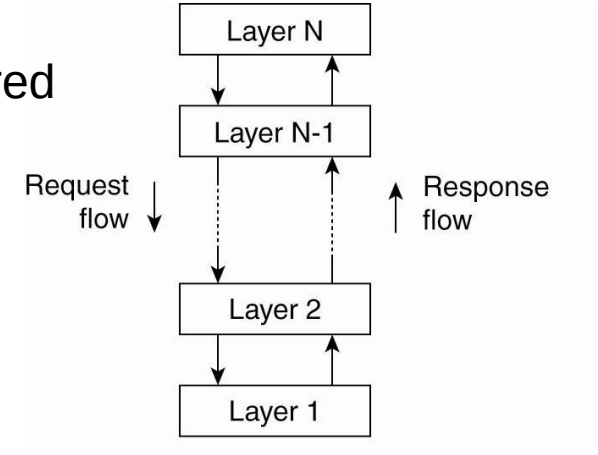
Dealing with complexity

- Distributed systems are complex pieces of software – to master complexity: good organization
- Different ways to look at organization of distributed systems – two obvious ones
  - Software architecture – logical organization of software components and interconnections
  - System architecture – their physical realization – the instantiation of software components on real machines

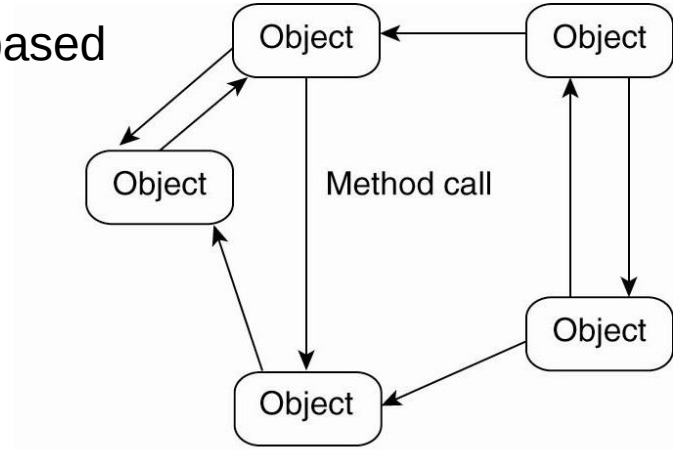
Architectural styles

- Organize into logically different components, and distribute those components over the various machines
  - Component: modular, replaceable unit with well defined I/F
  - Connector: a mechanism that mediates communication, coordination or cooperation among components
- Using components and connectors, different architectural styles

Layered



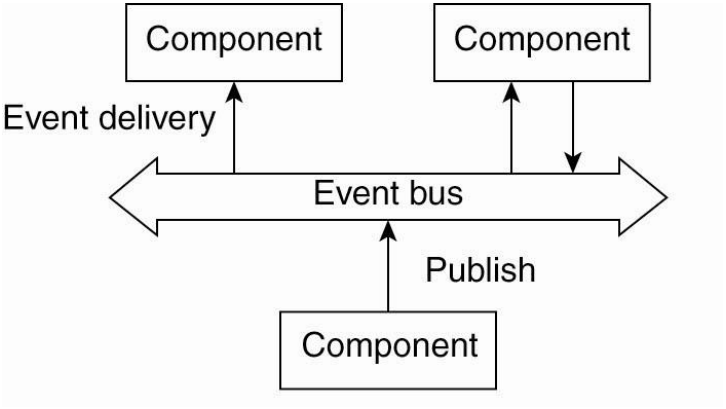
Object-based



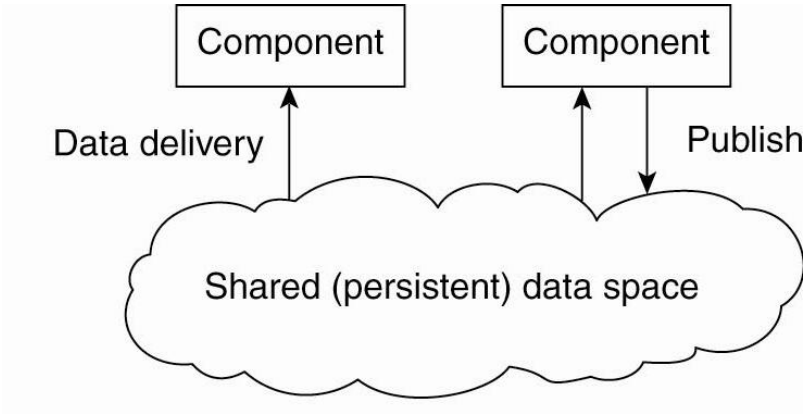
Architecture styles

- Decoupling processes in
  - Space (“anonymous” or referential decoupling) and
  - Time (“asynchronous” or temporal decoupling)
- Alternative styles

Event-based



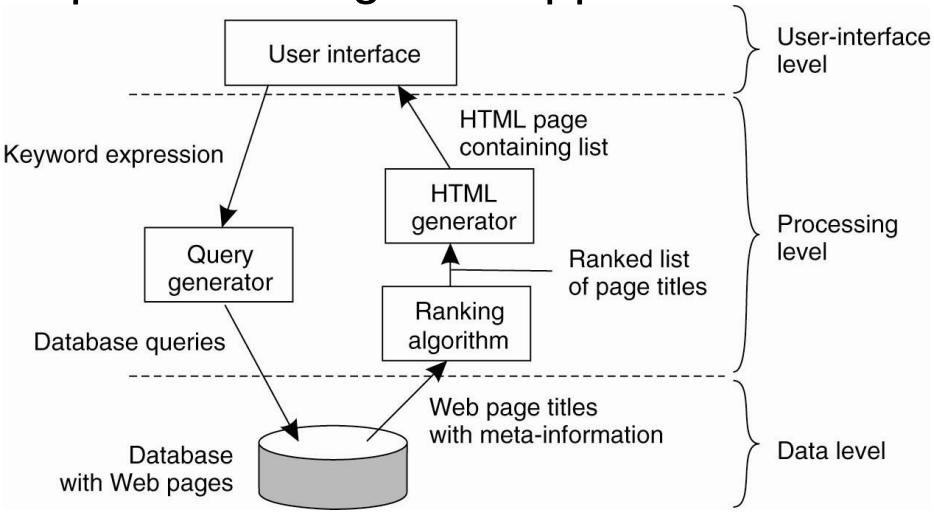
Shared data-space



System arch – vertical distribution

- Basic client/server model
  - Server processes offer services use by clients processes
  - Clients follow request/reply model in using services
  - Clients/servers can be distributed across different machines
- Traditional three-layered view
  - User-interface layer – an application's user interface
  - Processing layer – application, i.e. without specific data
  - Data layer – data to manipulate through the application

Internet search engine



System arch – horizontal distribution

- Multi-tiered client/server architecture – result from dividing the application into a user-interface, processing components and data level - *vertical*
- What's horizontal distribution?
  - Clients and servers split into logically equivalent parts, each operating on its own share of the data set
  - Each process logically equal to each other, acting both as client *and* server – peer-to-peer
- In all cases, we are dealing with overlay networks: data is routed over connections setup between the processes

Peer-to-peer architectures

- Structured, DHT-based, P2P: nodes are organized following a specific distributed data structure
  - A structured network such as a logical ring, and make specific nodes responsible for services based only on their ID
- Unstructured P2P: nodes have randomly selected neighbors
- Hybrid P2P:
  - Potentially combining each model's strengths
  - Some nodes are appointed special functions in a well-organized fashion

Architecture and middleware

- A key goal for middleware is to provide distribution transparency
- Typically, however, middleware adopts particular architecture styles
  - Makes it simpler to develop applications for that style
  - *Makes it hard/inefficient to do it with any other!*
- To alternatives – build different versions or make them easy to adapt dynamically
  - Interceptors: Intercept the usual flow of control when invoking a remote object
    - Make replication transparent
    - Make handling MTU transparent
    - …

Adaptation and self-management

- To deal with changing environments/demands – adaptive middleware
- To facilitate software adaptation
  - Separation of concerns: Separate specific from general functionalities (e.g. reliability, security, …) and later weave them together into an implementation
  - Computational reflection: Let program inspect itself at runtime and adapt/change its settings dynamically if necessary
  - Component-based design: Organize a distributed application through components that can be dynamically replaced when needed

Self-management

- Too many features, too many knobs, no sufficient foresight, …
- Self-*/Autonomics systems – self-configurable, self-manageable, self-healing, self-optimizing
  - Commonly, organized as a feedback control system
- Structure of an autonomic element
  - Managed element +
  - Autonomic manager
    - Monitor – both the managed element and its environment
    - Analyze – the monitoring information to
    - Plan – construct plan and
    - Execute – execute those plans
- One of today's grand challenges

Question 2

- *How the architectural style adopted by a distributed application impact its adaptability?*