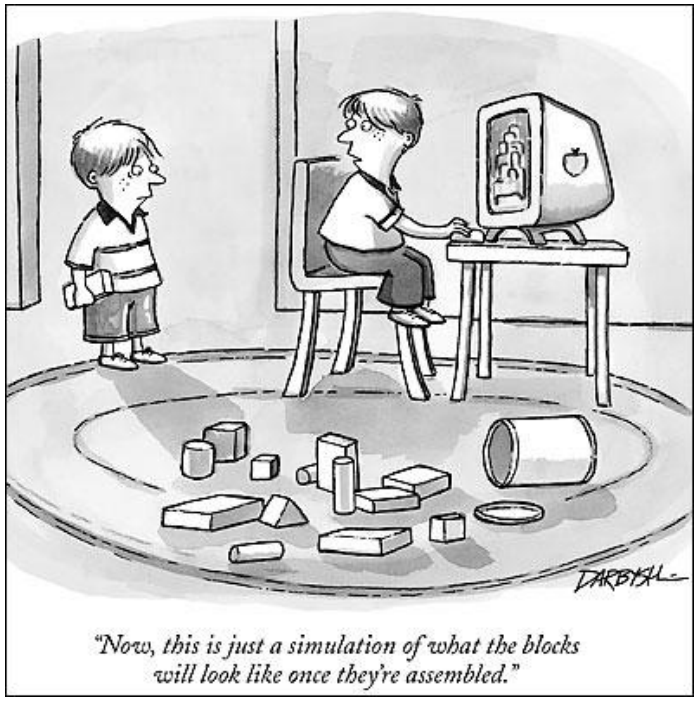
Distributed Systems Architectures



Today

- Architectural styles
- Software architectures
- LOCKS

Next time

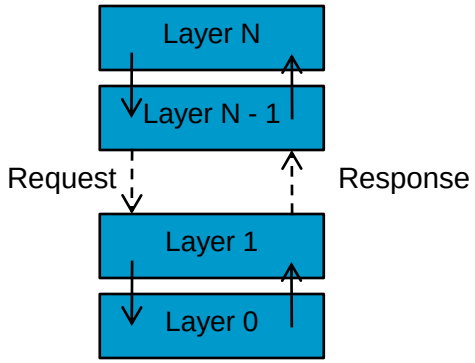
- ….

Architectural styles

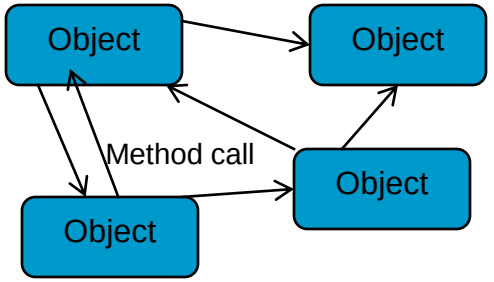
- Organization as the key to master complexity
- Two ways to look at distributed systems organization
  - Software architecture – how to organize the system's software components and how those component should interact
  - System architecture – how to instantiate the set of components on real machines
- Basic idea – Organize a system into logically different components and distributed those components over different machines

Interconnected components …

- Component
  - A modular unit with well-defined required/provided interfaces
- Connector
  - A mechanism that mediates communication, coordination or cooperation among components
- Different software architectural styles
  - Layered style as used in client-server models

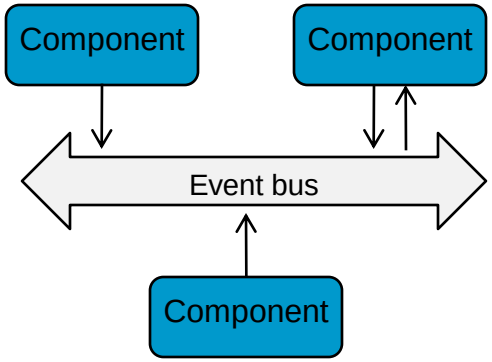


  - Object-based as in distributed object systems

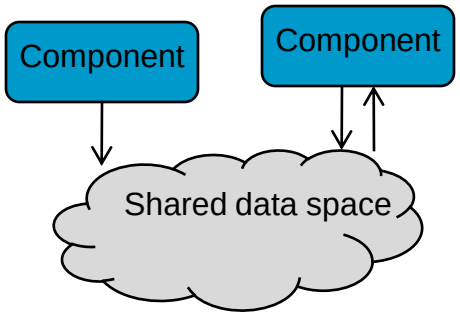


Interconnected components

- Decoupling components for greater flexibility
  - Referential decoupling (no need to know you name) – Event-based, publish-subscribe



  - Temporal decoupling (no need to be around) – Shared data spaces

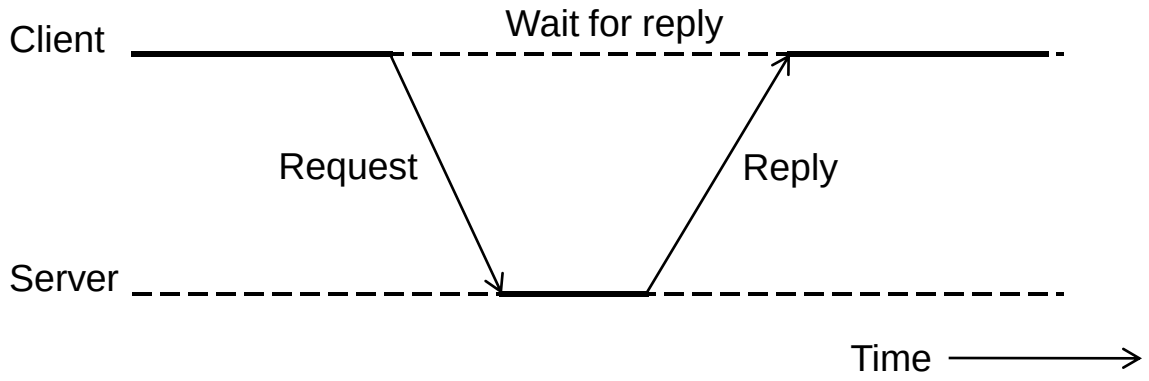


System architectures

- Deciding on this tell us a bit about performance and reliability requirement of each component
- A classification based on the symmetry of the interaction between components
  - Client-server and its variations
  - Peer-based architecture
  - Hybrid models

Client-server

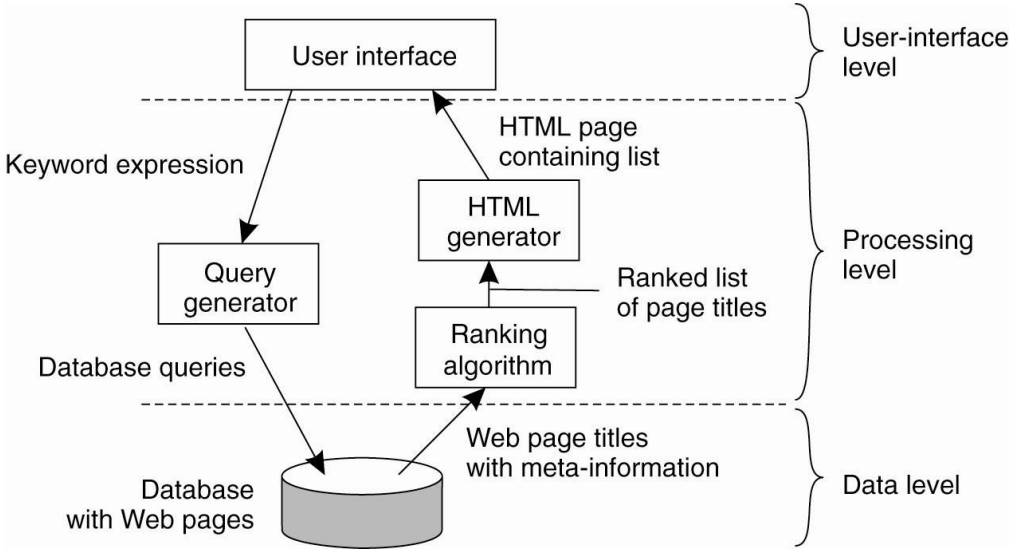
- Basic idea
  - There are processes offering services
  - There are other processes using them
  - Client and servers can be on different machines
  - Interaction follows a request/reply model
    - The underlying connection can use a connectionless (UDP) or connection-oriented protocol (TCP)



Introducing layering

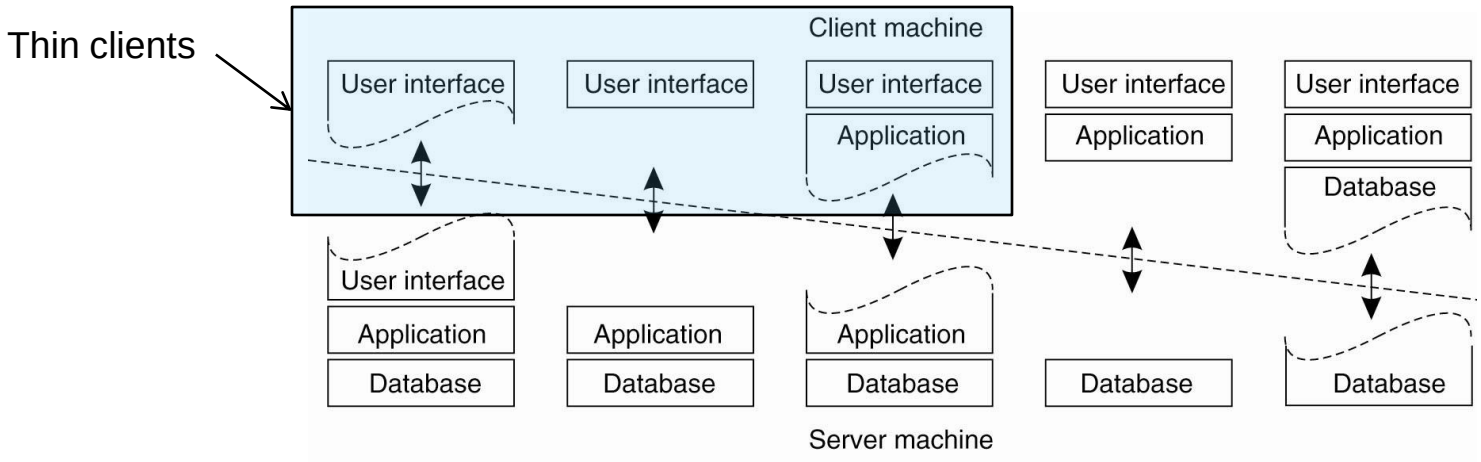
- Server for some and client of others
- A typical three-layer model for database access
  - User-interface – UI part of an application
  - Processing – functionality of an application without data
  - Data – data to be manipulated through the application
    - Responsible for ensuring data is persistent and consistent
    - Many times a relational database

A simple example – an Internet search engine



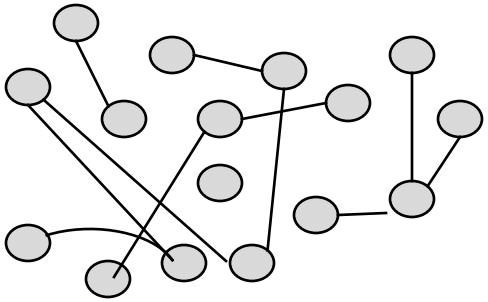
Multitiered architectures

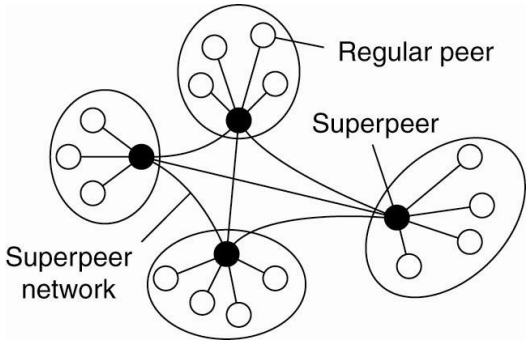
- Three logical layers with multiple instantiations
  - Over two or more tiers
- How much to place on each side?
  - Thinner or fatter clients
    - Fat's good – Short(er) response times and leverage clients' resources
    - Thin's good – Easy to manage



We are all (mostly) equal

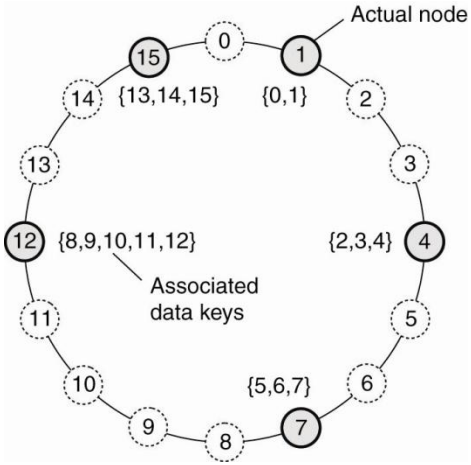
- Unstructured P2P systems
  - Peers connect with other random peers
  - Semi-structured models for scalability (superpeers)





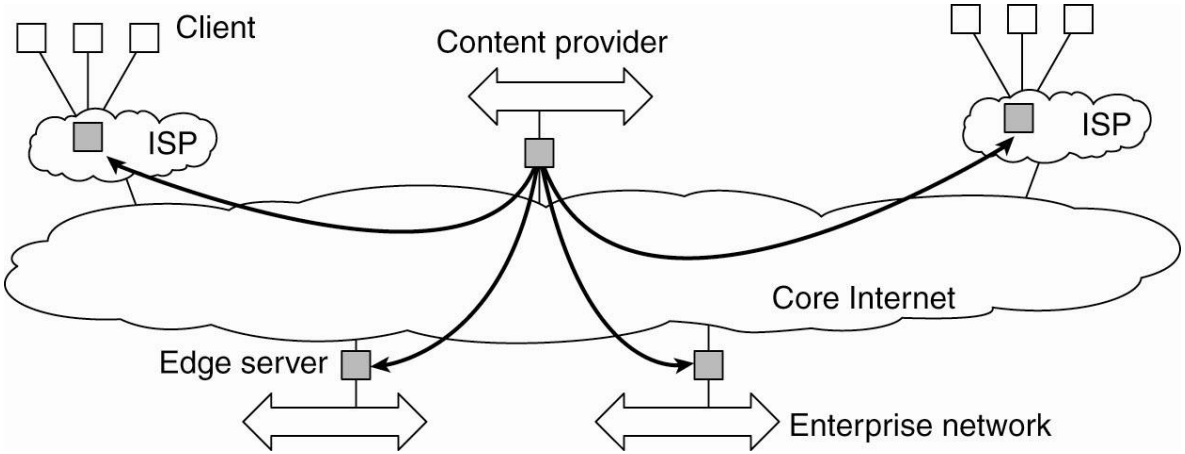
- Structured P2P systems (e.g. Distributed Hash Tables)

Peers' connection and content mapping in Chord



Hybrid models

- Edge-server systems – typical of content distribution networks (CDNs)
  - Clients get content through an edge server



- Early versions of P2P systems
  - Napster
  - Traditional BitTorrent

Next time

- Some concrete examples
  - LOCKSS – a P2P system for digital preservation
    - Paper: Maniatis et al., The LOCKSS Peer-to-Peer Digital Preservation System (Fabián)
  - Akamai CDN
    - Paper: Sherman et al., ACS: The Akamai Configuration Management System (Mario)
  - Google
    - Paper: Dean and Ghemawat, MapReduce: Simplified Data Processing on Large Clusters (Zach)