# Architectural Considerations

SAD

# 1. DS Organization: Architectural approaches

- Base concept of Distributed System too complex
  - Good for formal modelling
    - Establish formal properties
    - Prove correctness
    - APPLICATION:
      - Specialized middleware
  - Bad for general actual programming
    - Pure event-based anarchy
      - Unstructured inter-agent interaction
      - Non-viable starting point for imprecisely specified systems
        - All?

# 1. DS Organization: Architectural approaches

➔ Add structure

- Tension
  - Centralized systems
    - Some agents do more than their fair share of work
      - Server
    - Easy to reason about
      - Server coordinates: there is a Boss
        - We know who to blame
  - Fully distributed systems
    - All participate on equal footing
      - Need consensus
    - Difficult to reason about
      - Many things going on at the same time
        - Notion of causality a bit diffuse

# System Concerns: Configuration and Location

- Basic practical questions in a system
  - How are resources located?
    - Including agents
  - How does an agent know what resources are available?
  - How does an agent know who to ask for a particular resource?
  - How does an agent communicate with another?
  - How is all this configured?
- Answers to these questions determine degrees of centralization
  - Elements
    - Access the location service itself
      - Query the location service for a resource
      - Register available resources
      - Discover available resources
  - Above are related by the architectural structure
    - Centralized access usually means centralized query

# Centralization

- Centralized architectures imply central delivery of services/resources
  - Generic meaning of the term "resource"
    - Any piece of information
  - If an agent has a resource to share, it is placed on a server. e.g.
    - A directory server
    - A file server
  - Other agents can access those resources contacting the server

# Centralization

- Potential issues:
  - Scale
    - Can the central "server" cope with large populations of clients
    - Changes in the load of arriving requests?
    - Increases in the amount of information it needs to store?
  - Availability
    - Can the server fail?
    - How critical is the work of the server?
    - Can the server recover quickly?
  - Configuration
    - How are servers found?

# Distribution

- Any agent can hold any resource
  - Resources are spread over the population of agents forming the system
  - The same resource may be replicated and held by more than one agent
- Agents interested in other agents accessing their resurces, publish the fact they have it
  - Explicitly: Flooding the network (announcing)
  - Implicitly: Answering queries
- Agents interested on a resource search for it
  - Reach out to the network for other agents knowing something about it.
  - Contact its produced directly
  - No agent contains/serves all of one resource

# Distribution

- Potential Issues
  - Resource Location
    - Is there a way of economically search the network for a resource?
    - Are there many replicas of resource location knowledge?
  - Resource Access
    - Are there many replicas of a resource?
    - Are we wasting too much space?
  - Agent communication
    - Can agent communication proceed directly?
    - Is it necessary to follow some "breadcrumb" path to talk to the agent holding a resource?
      - E.g. when resources migrate

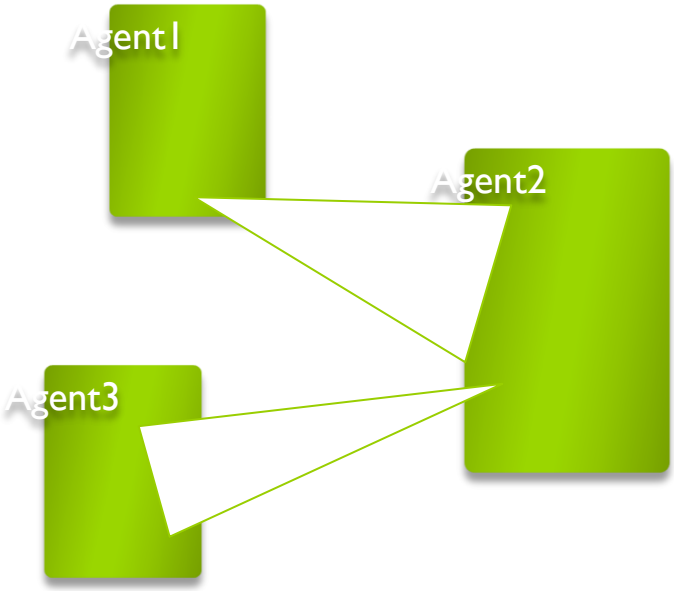
# Client Server

- Two usages
  - As an interaction pattern
  - As an architectural pattern
- Two Roles
  - The server.
    - Waits for requests from a population of Clients
  - The Client
    - Connects with a Server, sends a request message, and waits for the result.

# Client Server

- Modeled after subroutine invocations
  - The main program is the client
  - The invoked subroutine is the server, which may hold state
- Fully asymmetric role set
  - The server does more than its fair share of work
- Fits well a centralized system model
  - Architectural pattern

# Client Server: Interaction Pattern

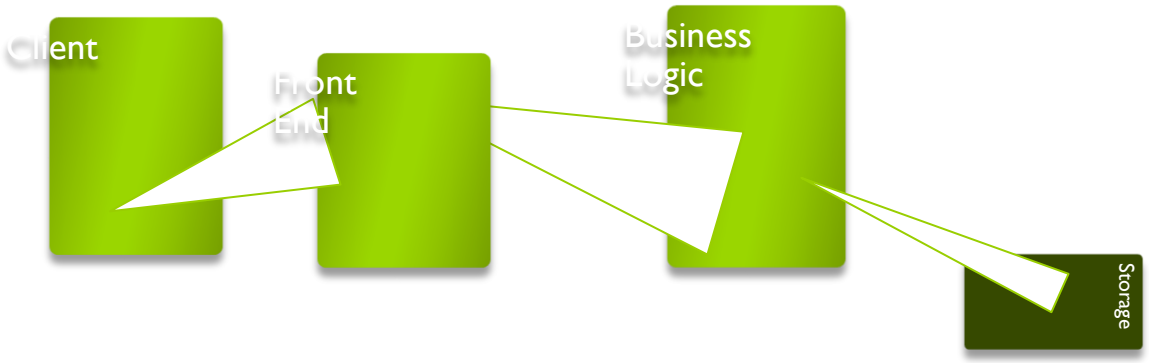


# Client Server

- As an interaction Pattern
  - Useful way of modelling agent-to-agent interaction
  - Does not impose a centralized architecture
  - Client/Server roles can be dynamic
    - Who acts as client/server varies as a result of system dynamics
  - Subset of fully asynchronous pattern
- As an Architecture:
  - Fully centralized: one server, many clients
  - Non-fault tolerant (a priori)
    - Server failure means system failure
  - Non-scalable
    - Single bottleneck point

# Client Server: Architecture Pattern

- N-tier architecture



# Client Server

- Extremely successful simplification
  - Helped by the success of a particular network stack: IP
  - Many examples on that stack
    - Network file systems (NFS)
    - Windowing Systems (X-Windows)
    - Printer Servers
    - The mail daemon (SMTP)
    - Database systems (SQL)
    - …

# Client Server

- Web protocols are based on this approach
  - Web Servers wait for requests
  - Browsers are the Clients
    - Slightly more complex situations due to the need for push communication
      - Websockets… for push communications
- What is understood as SOA is based on this approach
  - Servers are component services within a larger structure
  - Client/Server roles adopted by all agents
- Extension: Distributed Objects

# General Architectural Concerns

- Factors determining how centralized a system is
  - Resource Location
    - How are agents actually found in the system
  - Resource availability
    - How are available resources actually discovered
  - Communication
    - How does communication proceed among agents

# Search

- Centralized
  - Server with a dictionary
  - Providers register
  - Consumers query
  - Providers/consumers must be configured with server addresses
- Decentralized
  - Providers broadcast availability
  - Consumers consult their local cache

# Access

- Centralized
  - A few servers hold the resources
  - Providers send resources to servers
  - Consumers request resources from servers
  - Consumers must be configured with server addresses
- Decentralized
  - Consumers directly access the resources, once found

# Inter-agent communication

- Centralized
  - Mediated
    - Intermediary routes all traffic
      - (Network actually does this…)
  - Failure in intermediary ➔ isolated agents/network partitions
  - No need to know the actual agents
    - Or its addresses
      - Why may this be good?
- Decentralized
  - Direct
  - Need to know the agent's address
  - Better scalability
    - Aggregates network bandwidth

# Location examples

- DNS
  - Search
    - Hierarchical
  - Access
    - Centralized
- Google
  - Search
    - Decentralized
  - Access
    - Centralized