Distributed Systems. Applications and Services

SAD

Goals of this session

- Get an understanding of
  - What is a distributed system
  - What are the main difficulties
- Additionally
  - Explore some examples
  - Focus on Software as a means to obtain a service.

Index

1. Concept of Distributed System
2. Goals
3. Applications vs Services
4. Application Areas

1. What is a Distributed System

- Set of autonomous agents
  - Each agent is a sequential process, proceeding at its own pace.
    - Think of it as a State Machine/Automaton
      - Its specification is the "program"
      - Input Alphabet – Output Alphabet (e.g. look at I/O Automata)
- Agents interact
  - Options:
    - Message passing
    - Shared memory (Concurrent Systems)
      - Potentially different from arbitrary distributed systems
        - Special hardware coordination primitives (e.g. test-and-set)
    - Event-driven interaction
      - ***Reactive*** systems

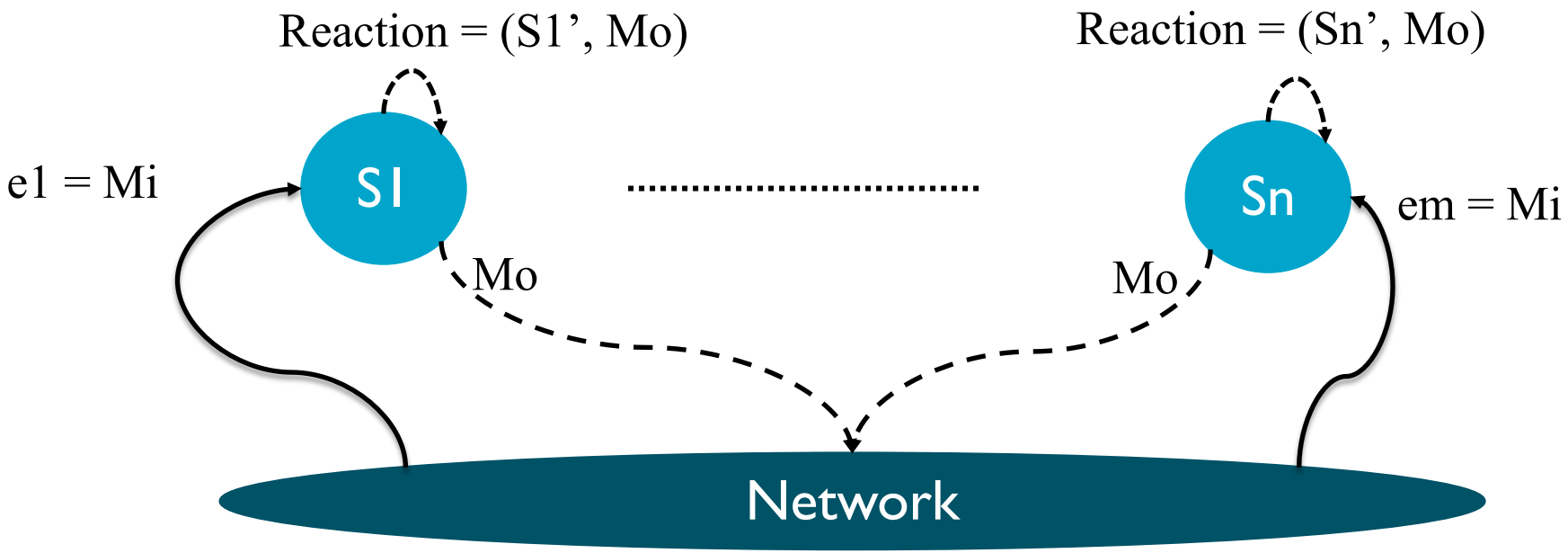
1. What is a Distributed System

- Agents have their own state
  - Independent of each other
  - Built as a result of the "reactions" to events processed.
  - The Communication Medium (e.g. Network) Also has its own state
    - Its an agent of sorts…
- There is some collective goal to this cooperation
  - By which the behavior of the whole "system" can be assessed.
    - Usually as a collection of "events"
      - Happening through each one of the elements of the system
      - Potentially enriched with state transitions at each agent

1. What is a Distributed System

A Behavior:

e1 → e2 → e3 … → em →



1. What is a Distributed System

- Correctness of a distributed system is declared by
  - Properties of behaviors that may appear in the system
  - E.g. Only some sets of behaviors are declared "Legal"
- Behaviors from different points of view:
  - Each process
  - A subset of the processes
  - A subset of the events
    - By kind of event
      - E.g. data access messages

Index

1. Concept of Distributed System
2. Goals
3. Applications vs Services
4. Application Areas

2. Goals

- Evolving field since its beginnings
  - Offshoot of concurrent systems
    - Heavily studied for their usefulness in the design of time sharing systems
    - You should be familiar with many aspects of concurrent systems (CSD, SO, TSR)
- Pushed by evolution of computer networks
  - How to make all those computers do something globally useful?

2. Goals

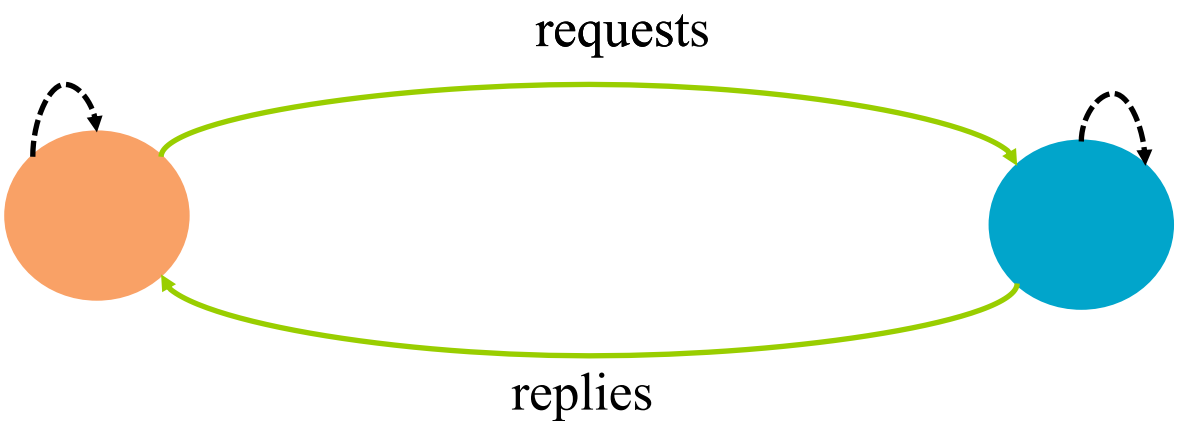
- Main original "reasons"/ "goals"
  - Speed up
    - Take a complex problem, split it in pieces, have each piece taken care of by a different computer.
  - Scale-up
    - As needs increase, add more elements to the systems and cope with the size
  - Fault tolerance
    - Basic idea:
      - If one computer breaks down, we still have other computers capable of carrying the tasks of the broken down one.
  - Resource Sharing
    - One computer may have resources (e.g., printers, disks, …) other computers do not have (and do not need to have)
    - It should be possible to access resources from everywhere

Index

1. Concept of Distributed System
2. Goals
3. Applications vs Services
4. Application Areas

3. Concept of Service

- Specification of how an interaction should proceed
  - What "behaviors" are acceptable



- Familiar concept… from what a distributed system model says
  e1 → e2 → … → en → …
  R → R' → r → … → r' → …
- Restrictions are placed on the events emitted by the clients

3. Concept of Application

- Executable specification of how a system should react to events
  - In an unambiguous language
    - Programming languages
  - An "interpreter" can execute it
    - Sets up an environment
      - To receive events and …
      - … process those events according to the spec …
      - … causing reactions
        - state changes
        - Effects for external entities (e.g., messages)
    - STATE is maintained for a running application
- A behavior can be established for a running application
  - According to the pattern of arriving events
    - *If that pattern satisfies X then the resulting behavior satisfies Y*

3. Applications and Services

- Services are obtained from the execution of an application
- Services must satisfy restrictions on their behaviors
  - That is their Service Level Agreement – SLA
  - At least two aspects
    - Functional
      - Safety properties
    - Performance
      - Liveness properties

Index

1. Concept of Distributed System
2. Goals
3. Applications vs Services
4. Application Areas

4. Application Areas: The WWW

- Servers wait for simple requests for documents
  - Requests may involve reading or writing of a document
    - But that logic is internal to the server
- The Clients are Browsers, sending/receiving documents
  - Browsers parse documents searching for metadata
  - Browsers do not need to be "Browsers" for all applications
  - Links are particular metadata pointing to other documents
    - Documents may be in different servers
- Simple and powerful paradigm
  - Initially conceived for document sharing
  - Extended to allow document requests to stand for general service requests
    - Returned "documents" encode the result of the actual request

4. Application Areas: Sensor Networks

- Driven by declining costs of hardware
- Special purpose mini-computers
  - Motes
- Embedded in common devices
  - Dishwashers, etc…
- Contain physical world sensors
  - Humidity, temperature, power consumption, …
- Wide range of potential applications
  - Surveillance
  - Biological and chemical disaster detection
  - Power monitoring
  - …

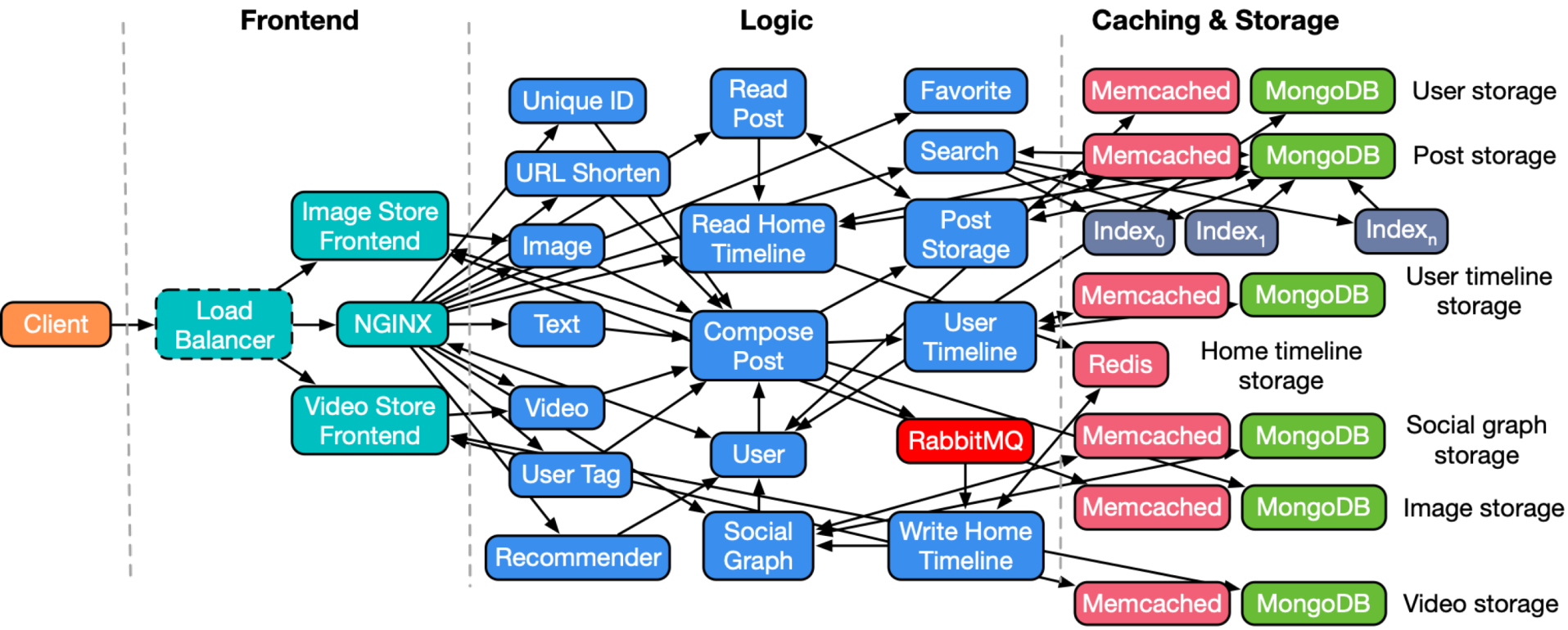
4. Application Areas: Cloud Computing

- SaaS providers
  - High volume
    - Individually
      - Google: 40,000 queries per second (1.2 Trillion per year)
      - YouTube: 1.9B active users per month,
        - viewing 5B videos per day
      - Facebook: 2.23B active users,
        - 8B video views
        - 15M photos uploaed per day
    - Combined
- At these scales, no computer can keep up.
  - By the nature of the cloud, it *has* to be massively parallel!
- IaaS providers
  - Pay-per-use

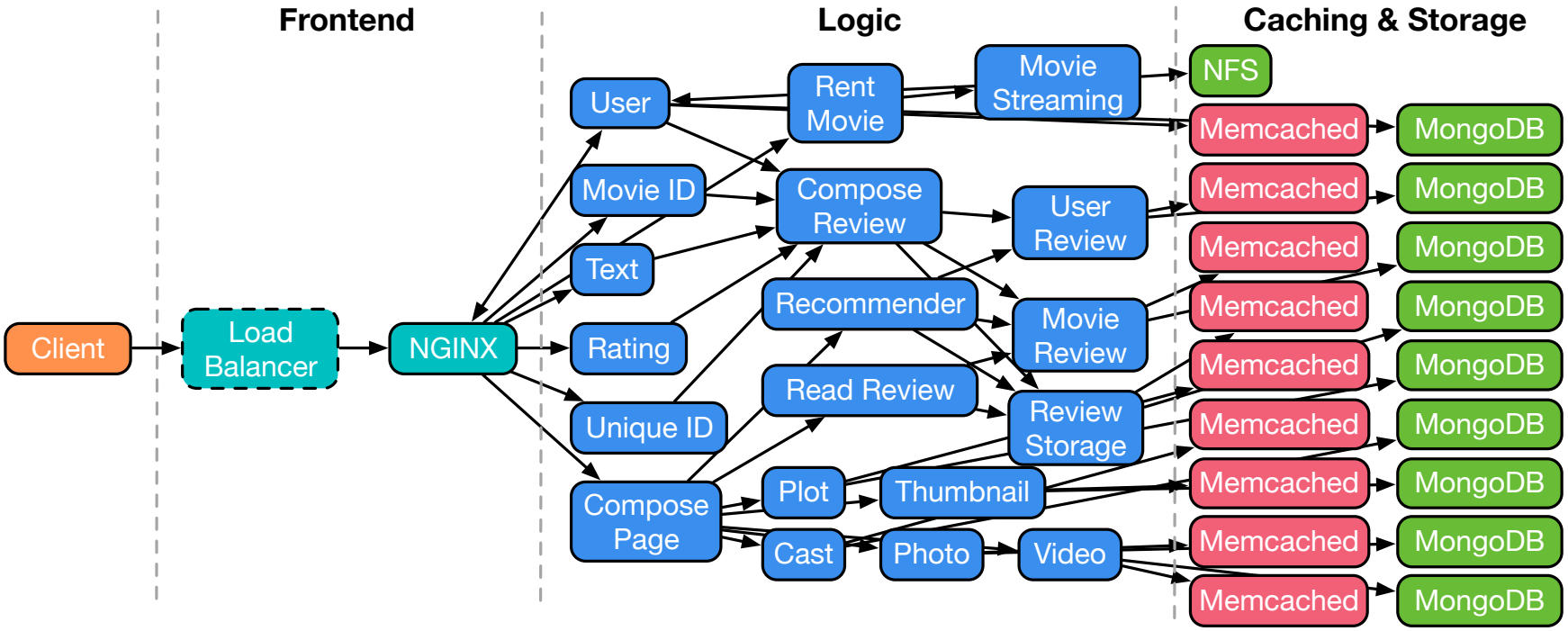
4. Application Areas: Cloud Computing

- Multi-tier
- Tier-one
  - First-line of interaction
  - Generates responses
    - Use small amounts of memory
    - Low computer power
    - Low needs for storage/network I/O
- Tier-two
  - Support services
    - Micro-services
      - Specialize on various tasks
      - E.g. Storage
    - Larger computers

4. Application Areas: Cloud/Microservices



4. Application Areas: Cloud/Microservices



4. Application Areas: Cloud/Microservices

- Every one of those little nodes is itself a small elastic pool of processes

- Microservice:
  - Out of an application
  - The data center can run one instance
    - … or many instances, "elastically",
    - … to deal with dynamically varying demand.
    - Horizontal scaling

- Any instance can handle any request equally well,
  - So, no need for very careful "routing" of specific requests to specific instances.
  - This lets the data center adapt to changing loads easily!

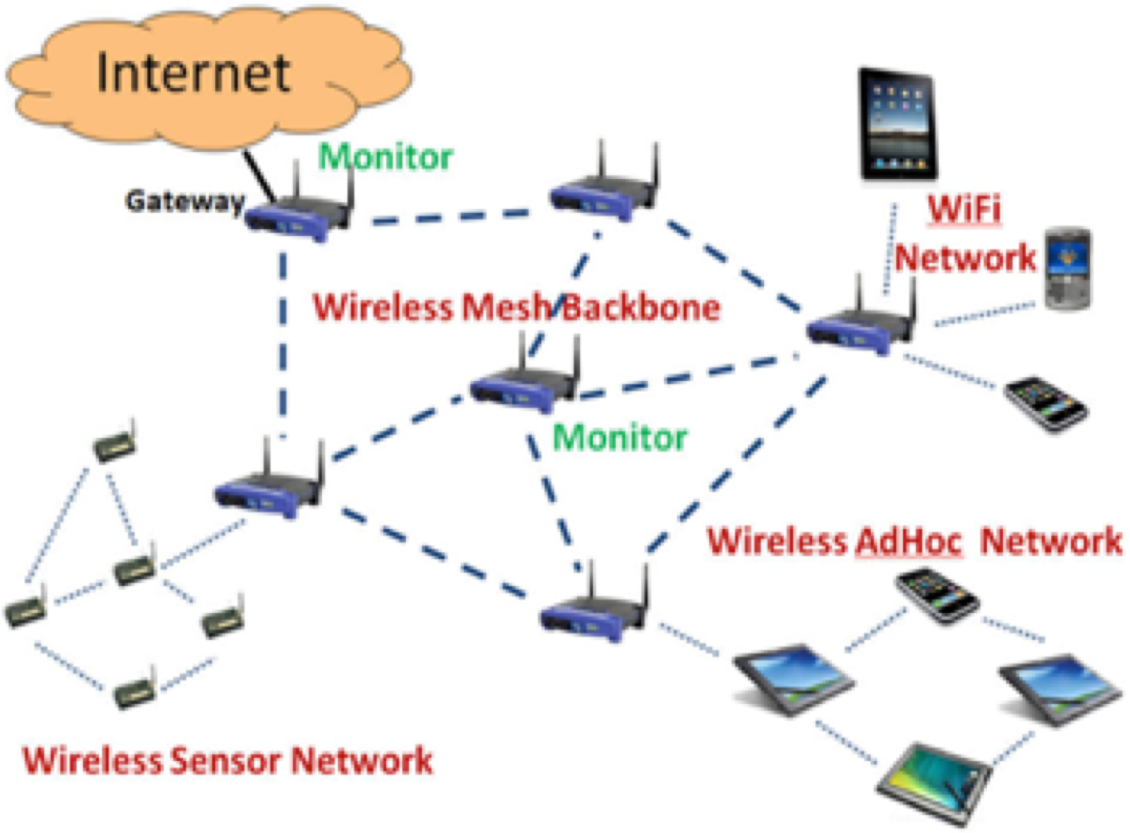
4. Application Areas: Cloud/Microservices

- Advantages of Micro-services
  - Modular → easier to understand
  - Speed of development & deployment
  - On-demand provisioning, elasticity
  - Language/framework heterogeneity

4. Application Areas: The "Internet of Things"

- Motivation: leverage ubiquitous connectivity of all devices
  - Generalization of sensor networks
    - All devices can, and will interact among them
    - Devices can also alter their physical environment
  - New scenarios open up
    - Smart cities
    - Building/Factory automation
    - Healthcare
    - …

The "Internet of Things"



4. Application Areas: Cooperative Computing

- Most computational power is underused
  - The desktops spend many hours doing nothing
- Many engineering and scientific problems can be split into pieces (tasks)
  - Each task can be resolved in a small amount of time
  - Results from each piece can be composed to complete the resolution of the whole problem
- Servers can be set up with an instance of such a problem
  - The server creates a pool of smaller tasks
- Computers across the internet can subscribe to receive tasks to solve
  - They install a special client software: the task runtime enviornment
  - The client registers with the server
- The server spreads tasks among the registered clients, and collects their results

4. Application Areas: High Availability Clusters

- So far we have seen application areas addressing resource sharing and cooperation.
- Fact:
  - Devices fail. Computers are devices. They fail at some point with a 100% probability.
- Fact:
  - Not all devices fail at the same time, always.
  - Q: when can it happen?
- Some environments need a high degree of availability
  - Banking
  - Finances
  - Mission-critical systems
  - …
- Leverage having more than one device to stand failures

4. Application Areas: High Availability Clusters

- HA Cluster:
  - Set of computers, with server programs on which clients depend constantly
  - Typically holding sensitive data
  - Designed with specific protocols to stand failures of one or more of them
  - Two main concerns:
    - Preserve data integrity/consistency
    - Preserve server operation availability

4. Application Areas: Distributed Ledgers

- Avoidance of centralized authorities to keep track of transactions
- Avoidance of centralized Notarial authorities
- Technology
  - Blockchain algorithms