Cloud computing and distributed systems

Chapter #1
Characterization of distributed systems

Zeynep Yücel

Ca' Foscari University of Venice
zeynep.yucel@unive.it
yucelzeynep.github.io

Distributed systems

- Networked computers communicate by messages.
- Interconnected, no distance limit.
- Significant role in everyday applications.
- Properties: concurrency, independence, failures.

Concurrency

- Individual tasks performed, resource sharing.
- Concurrent execution.
- More resources enhance capacity.

No global clock

- Coordination through message exchange.
- Shared concept of time needed.
- Struggles with clock synchronization.
- No universal correct time.
- Limitation due to message reliance.

Independent failures

- Failures may isolate computers, but they continue operation.
- Programs may not detect failures.
- Unawareness of crashes by components.

Resources in Distributed Systems

- Hardware Resources
  - Storage: HD, SSD, Cloud, etc.
  - I/O Devices: Reduce redundancy, costs.
- Software Resources:
  - Files: Access regardless of location.
  - Databases: Real-time sharing, collaboration.
  - Data Objects: Object-oriented programming.
- Media and Communication:
  - Streaming services: Video, audio.

Examples of distributed systems

- Networks enable everyday services.
  - Distributed technology crucial for modern computing.
- Diverse applications:
  - Localized to global systems.
  - Data-centric to processor-intensive.
  - Simple sensors to powerful elements.

Some examples of distributed systems

Web search

- Billions of web pages.
- Web search engine indexes content.
- Complex indexing and processing.
- Google's sophisticated distributed system:
  - Global data center network.
  - Optimized distributed file system.
  - Structured storage for quick access.
  - Lock service for synchronization.
  - Programming model for parallel computations.

Some examples of distributed systems

Massively multiplayer online games (MMOGs)

- Immersive experiences in virtual worlds.
- Supports thousands of players.
- Challenges: response times, consistent views.
- Commercial applications vary:
  - Centralized client-server architecture.
  - Distributed architectures based on usage.
- Research explores peer-to-peer architectures.

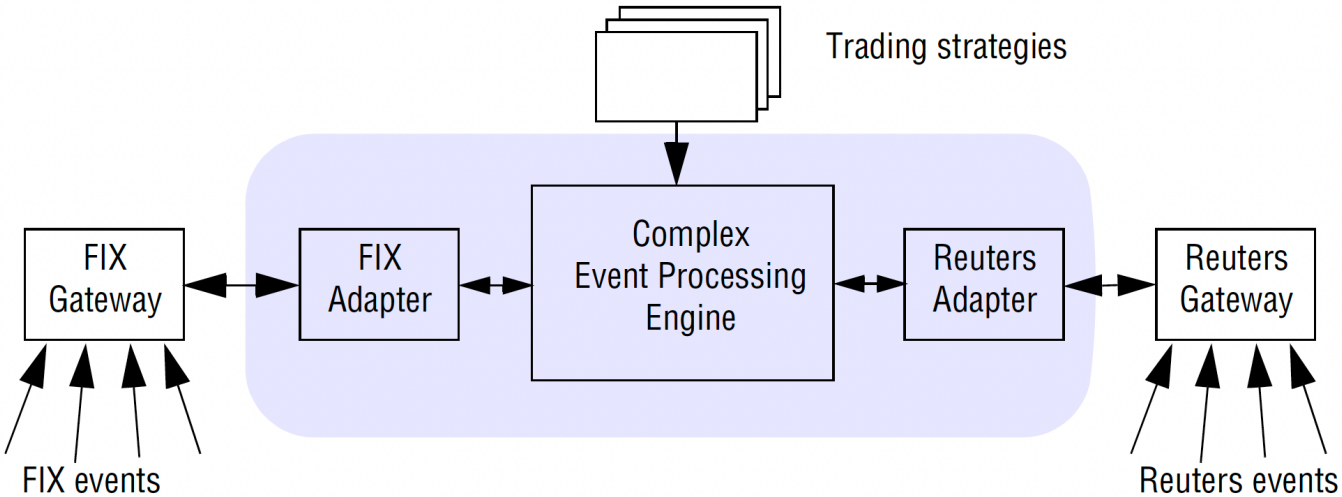
Some examples of distributed systems

Financial trading

- Real-time information access critical.
- Focus on event communication and processing.
  - Requires distributed event-based architecture.
  - Heterogeneous event feeds from various sources.
- Adapters unify formats for processing.
- Complex Event Processing automates opportunities.

Some examples of distributed systems

Financial trading



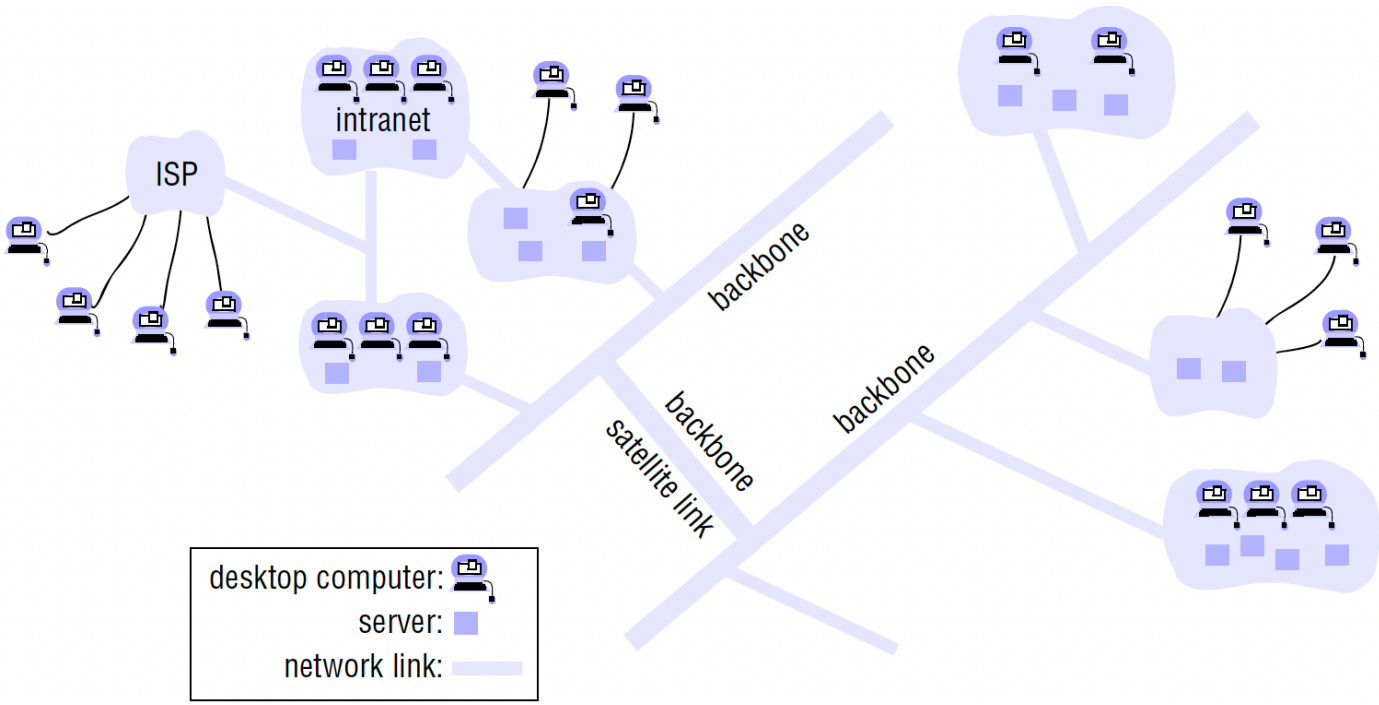
- Event feeds from various sources.
  - Heterogeneity issues in formats.
  - Adapters convert formats to common standard.
- Handles multiple event streams rapidly.
- Real-time processing to identify opportunities.
- Automation due to competitive pressures.

Other examples of distributed systems

- Media: User-generated content platforms.
- Healthcare: Telemedicine and electronic records.
- Education: E-learning and collaborative environments.
- Transport: GPS and mapping services.
- Science: Grid technology for data processing.
- Environmental Management: Networked sensors for monitoring.

Trends in distributed systems

Pervasive networking and the modern Internet



- Modern Internet: interconnected computer networks.
- Communication mechanisms across technologies.

Trends in distributed systems

Pervasive networking and the modern Internet

- Large distributed system services.
- Intranets secured by firewalls.
- ISPs connect intranets via backbones.
- Some networks isolated from Internet.

Trends in distributed systems

Mobile computing

- Mobile computing allows tasks on-the-go.
- Devices: laptops, handhelds, wearables.
- Continued Internet access while moving.

Trends in distributed systems

Ubiquitous computing

- Integrating small computing devices.
- Devices are common, unnoticed functions.
- Communication enables control and notifications.

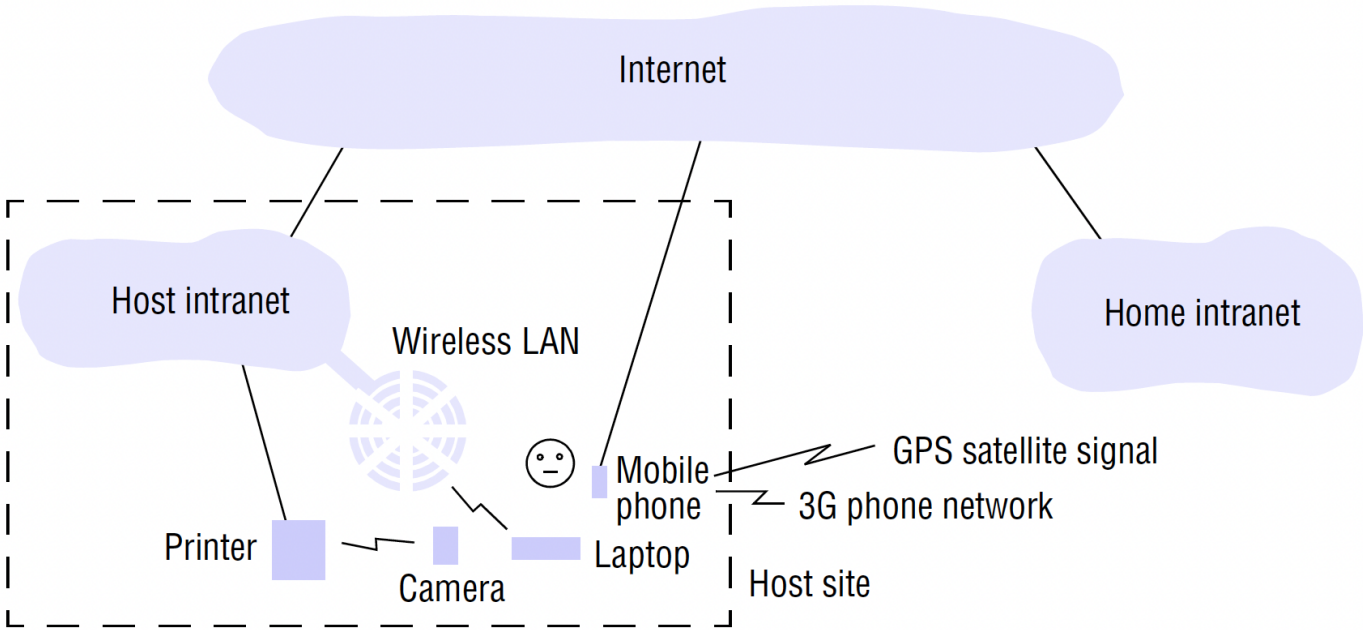
Trends in distributed systems

Mobile and ubiquitous computing

- Distinct definitions, yet work together.
  - Mobile computing with conventional devices.
  - Ubiquitous within single environments.
- Spontaneous interoperation is possible.

Trends in distributed systems

Mobile and ubiquitous computing



- Spontaneous interoperation.
- Easily creating and removing associations.
- Fast and convenient, even in unfamiliar environments.
- Service discovery.

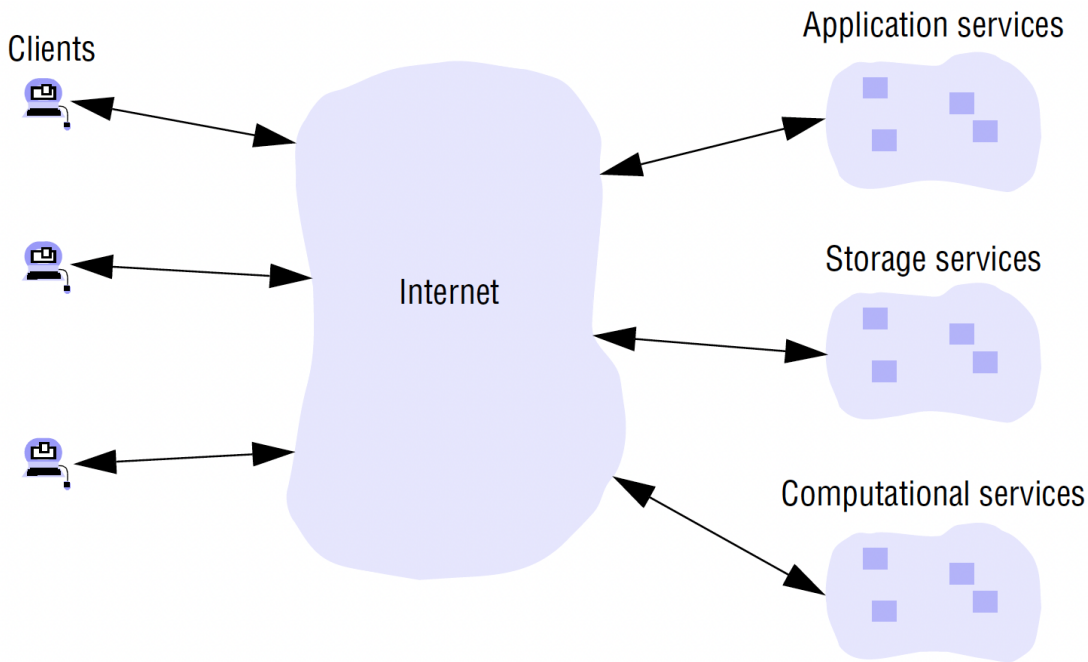
Distributed multimedia systems

- Integration of discrete and continuous media.
- Distributed multimedia system manages storage, transmission, presentation.
- Continuous types also maintain real-time relationship.
  - Applications: IP telephony, webcasting.
- Demand for robust infrastructure support.

Distributed computing as a utility

- Resources offered as utility.
- Users can rent resources.
  - Physical resources: remote storage, computational power.
  - Software services: enhance rental experience.
- Cloud computing concept introduced.

Cloud computing, cluster and grid



- Cloud computing views computing as utility.
  - Internet-based applications and services.
  - Offered as service, pay-per-usage model.
  - Decreased reliance on local devices.

Cloud computing, cluster and grid

- Clouds hosted on cluster computers.
  - Cluster computers for high-performance computing.
  - Standard Linux on commodity PCs.
  - Mass storage solutions included.
- Grid computing as cloud computing precursor.
  - Focused on scientific applications.

Resource sharing

- Resource sharing often taken for granted.
- Higher-level resource sharing significant.
  - Search engines depend on servers.
- "Service" as resource management.

Resource sharing

- "Server" as program accepting requests.
  - Clients send requests, servers reply.
  - Remote invocation describes interaction.
  - Processes may act as both client/server.
- Client-server model common in many real-world systems.

Challenges of distributed systems

Heterogeneity of distributed systems

- Heterogeneity in networks, hardware, OS.
  - Internet protocols mask network differences.
  - Data types vary across hardware.
  - Different OS may use varied calls.
  - Programming languages have different representations.
  - Implementations require common standards.

Challenges of distributed systems

Middleware

- Middleware provides programming abstraction.
- Conceals heterogeneity, unified computational model.
- Examples include remote object invocation.

Challenges of distributed systems

Heterogeneity and mobile code

- Mobile code transfers and executes remotely.
- Virtual machine approach enables execution.
- Java compiler generates platform-independent bytecode.

Challenges of distributed systems

Openness

- Openness allows integration of new services.
- Specification and documentation availability essential.
  - E.g. RFCs
- Standards ensure system component compatibility.
  - Hardware and software extensions
  - Test and verify for conformance

Challenges of distributed systems

Security

- High value of information resources.
- Key security components: confidentiality, integrity, availability.
- Firewalls control traffic, but not usage.
- Sensitive information needs concealment and authentication.
- Encryption addresses both challenges.

Challenges of distributed systems

Security

- Unmet security challenges:
- Denial of Service Attacks:
  - Excessive requests disrupt services.
  - Legitimate access denied.
- Security of Mobile Code:
  - Executable attachments may pose risks.
  - Unpredictable consequences from execution.

Challenges of distributed systems

Scalability

Date	*Computers*	*Web servers*	*Percentage*
1993, July	1,776,000	130	0.008
1995, July	6,642,000	23,500	0.4
1997, July	19,540,000	1,203,096	6
1999, July	56,218,000	6,598,697	12
2001, July	125,888,197	31,299,592	25
2003, July	~200,000,000	42,298,371	21
2005, July	353,284,187	67,571,581	19

- Scalable systems effective with increased resources.

Challenges of distributed systems

Scalability

- Challenges in scalable systems design:
  - Controlling resource costs: proportional to user growth.
  - Performance loss: larger data sizes degrade performance.

Challenges of distributed systems

Scalability

- Preventing software resource depletion.
  - Limited IP address space issues.
  - Predicting demand is difficult.
- Avoiding performance bottlenecks.

Challenges of distributed systems

Failure handling

- Managing failures due to partial failures.
  - Some components fail, others continue.
- Failure management methods:
  - Failure detection: checksums, remote server crashes.
  - Failing masking: retransmitting messages.
  - Failure tolerance: clients designed to handle.
  - Recovery: rollback to consistent state.
  - Redundancy: services withstand failures.

Challenges of distributed systems

Concurrency

- Multiple clients access shared resources.
- Processing a single request limits throughput.
- Concurrent processing improves efficiency.
- Conflicts may lead to inconsistencies.
- Adaptation required for distributed objects.

Transparency

- Transparency hides component separation.
- System perceived as cohesive whole.
- Forms of transparency:
  - Access
  - Location
  - Concurrency
  - Failure
  - Replication
  - Mobility
  - Performance
  - Scaling

Transparency

- Access transparency abstracts access mechanisms.
  - Seamless file interaction is ideal.
- Location transparency enables resource access without location knowledge.
  - Specific data center access limits transparency.
- Access and location transparency combined as network transparency.

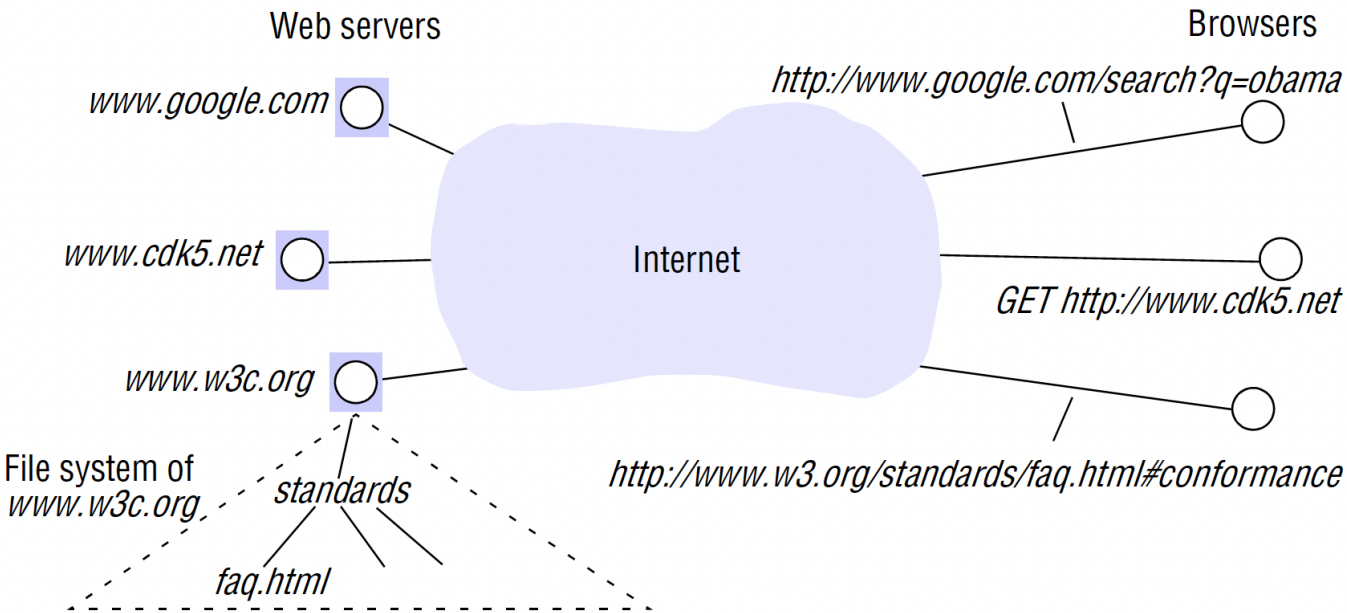
Quality of service

- QoS influenced by nonfunctional properties:
  - Reliability
  - Security
  - Performance
  - Adaptability
- Performance focuses on timeliness.
  - Multimedia requires timely processing.
  - QoS ensures task deadline meeting.
- QoS depends on resource availability.
- Performance degrades under heavy loads.
- QoS applies to systems and networks.
- Unmet requests rejected.

Case study: WWW

- WWW originated in 1989 at CERN.
- Openness is a defining feature.
  - Communication and content standards publicly accessible.
  - Diversity of resources welcomed.

Case study: WWW



- Core of Web: three fundamental components.
  - HTML, URLs, HTTP.

Case study: WWW

HTML

- HTML lays out document content and structure.
  - Hypertext allows document linking.
  - Tags annotate documents.
  - Browser retrieves and interprets HTML files.

Case study: WWW

URL

- URL identifies a resource.
- General URL format includes distinct parts.
- `<scheme>://<user>:<password>@<host>:<port>/<path>;<params>?<query>#<frag>`
- Server name mapped to IP address via DNS.

Case study: WWW

URL

- HTTP defines browser-server interactions.
  - Request-reply protocol for resource access.
  - Key operations: GET, POST, etc.
  - Single resource targeted per request.
  - Concurrent requests speed up loading.
  - Simple access control by default.

Discussion topics

Synchronization

How might the clocks in two computers that are linked by a local network be synchronized without reference to an external time source? What factors limit the accuracy of the procedure you have described? How could the clocks in a large number of computers connected by the Internet be synchronized? Discuss the accuracy of that procedure.

Discussion topics

Cloud computing

Compare and contrast cloud computing with more traditional client-server computing? What is novel about cloud computing as a concept?

Discussion topics

Heterogeneity

A server program written in one language (for example C++) provides the implementation of a BLOB object that is intended to be accessed by clients that may be written in a different language (for example Java). The client and server computers may have different hardware, but all of them are attached to an internet. Describe the problems due to each of the five aspects of heterogeneity that need to be solved to make it possible for a client object to invoke a method on the server object.