Distributed Systems
ECEG-6504

System Models

Surafel Lemma Abebe (Ph. D.)

Topics

- Types of system models
- Physical models
- Architectural models
- Fundamental models
  - Interaction, Failures, Security

Types of system models

- Difficulties and threats for DS
  - Widely varying mode of use
  - Wide range of system environments
  - Internal problems
  - External threats

- Properties and design issues of DS can be captured using descriptive models
  - Intended to provide an abstract, simplified but consistent description of a relevant aspect of DS design
    - Physical model
    - Architectural model
    - Fundamental model

Physical models

- Representation of the underlying hardware elements
  - *"A system in which (hardware or software) components located at networked computers communicate and coordinate their actions only by message passing." [Coulouris]*

  ⇒ Minimal physical model

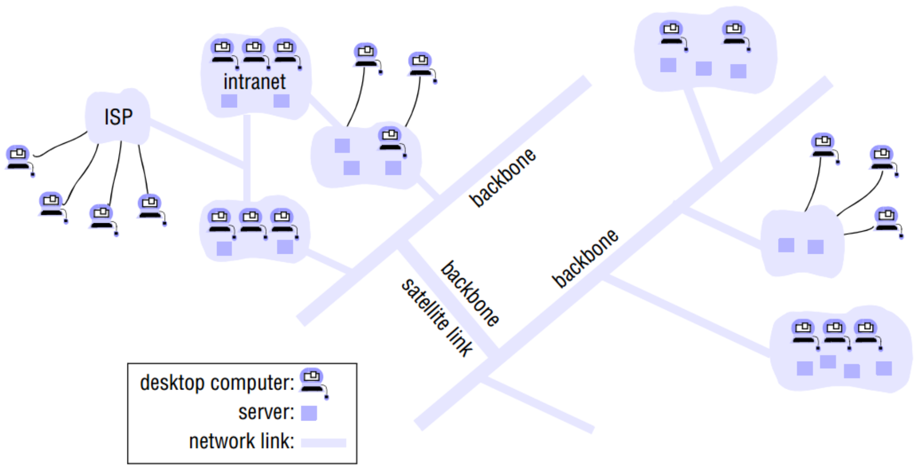
    Extensible set of computer nodes interconnected by a computer network for the required passing of messages

- Early DS (1970s and early 1980s)
  - 10 to 100 nodes interconnected by a LAN
  - Small range of services
  - Largely homogeneous systems

    => openness was not a primary concern
  - Effort was to improve quality of service

Physical models…

- Internet-scale DS (1990s)
  - Drive was growth of the Internet
  - Physical infrastructure: network of networks
  - High heterogeneity
    - Emphasis on
      => Open standards
      => Middleware technologies
      => Web services
  - Characteristics of nodes
    - Relatively static
    - Discrete
    - Autonomous



Physical models…

- Contemporary DS
  - Emergence of mobile computing
    - Need for added capability such as service discovery and support for spontaneous interoperation
  - Emergence of ubiquitous computing
    - Move from discrete nodes to architectures where computers are embedded
    - E.g., smart homes
  - Emergence of cloud computing
    - Move from autonomous nodes performing a given role to pools of nodes that together provide a given service
    - E.g., Google search

=> increase in heterogeneity

Physical models…

- Challenges

Distributed systems:	*Early*	*Internet-scale*	*Contemporary*
Scale	Small	Large	Ultra-large
Heterogeneity	Limited (typically relatively homogenous configurations)	Significant in terms of platforms, languages and middleware	Added dimensions introduced including radically different styles of architecture
Openness	Not a priority	Significant priority with range of standards introduced	Major research challenge with existing standards not yet able to embrace complex systems
Quality of service	In its infancy	Significant priority with range of services introduced	Major research challenge with existing services not yet able to embrace complex systems

Architectural models

- Architecture
  - Structure specified in terms of separately specified components and their interrelationships
  - Goal
    - Meet the present and likely future demands

  - Concerns
    - Reliability
    - Manageability
    - Adaptability
    - Cost-effectiveness
  - Architectural design provide a consistent frame of reference for the design

Architectural models…

- Architectural elements
  - Questions to understand fundamental building blocks
    - What are the entities that are communicating in the DS?
    - How do they communicate, or, more specifically, what communication paradigm is used?
    - What (potentially changing) roles and responsibilities do they have in the overall architecture?
    - How are they mapped on to the physical distributed infrastructure (what is their placement)?

Architectural models…

- Communicating entities
  - ***What are the entities that are communicating in the DS?***
    - System perspective
      - Processes
      - Caveats
        - » In systems that do not support process abstraction, entities are nodes (e.g., sensor networks)
        - » Threads (when supplementing processes)
    - Problem perspective
      - Objects
      - Components
        - » Specify interfaces + assumptions
        - » Contractual approach

Architectural models…

- Communication entities…
  - Problem perspective…
    - Web services
      - » Encapsulate behavior and access through interfaces
      - » Integrated into the WWW
      - » Partially defined by web-based technologies
      - » Decoupled complete services that can be combined to achieve value-added services
      - » Cross organizational boundaries

Architectural nodes…

- Communication paradigm
  - ***How do they communicate, or, more specifically, what communication paradigm is used?***
  - Inter-process communication
    - Low level support for communication between processes
      - Message-passing primitives
      - Socket programming (direct access to the APIs offered by Internet Protocols)
      - Multicast communication

Architectural models…

- Communication paradigm…
  - Remote invocation
    - Covers a range of techniques based on a two-way exchange
    - Request-reply protocol
      - Involve pair wise exchange of messages from client to server and back
      - Primitive
      - Used in embedded systems when performance is paramount
      - Used in HTTP protocol

Architectural models…

- Communication paradigm…
  - Remote invocation…
    - Remote procedure calls
      - Procedures in processes on remote computers can be called as if they are procedures in the local address space
      - Hides distribution, encoding, and decoding of parameters and results, passing of messages, …
      - Supports client server applications
      - Offer **access** and **location** transparency
    - Remote method invocation
      - Used with distributed objects
      - Calling object invokes a method in a remote object
      - Hides underlying details

Architectural models…

- Communication paradigm…
  - Remote invocation…
    - Common characteristics
      - Communication represents a two-way relationship
        - » Senders explicitly directing messages/invocations to the associated receiver
        - » Receivers are generally aware of the identity of senders
      - Mostly, both parties must exist at the same time

  - Indirect communication
    - Done through a third entity
    - Allow strong degree of decoupling between senders and receivers
      - Space decoupled
      - Time decoupled

Architectural models…

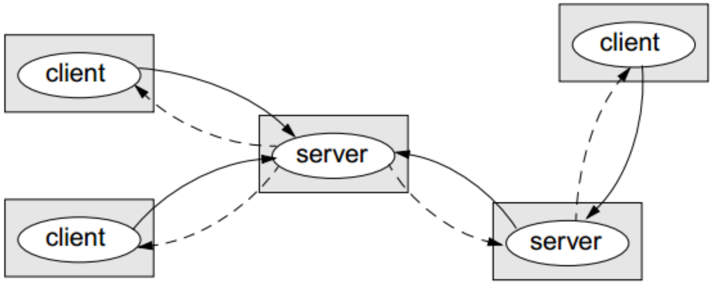
- Communication paradigm…
  - Indirect communication…
    - Group communication
      - One-to-many communication
      - Relies on the abstraction of a group => group identifier
      - Recipients elect to receive messages sent to a group by joining the group
      - Senders send messages to the group via the group identifier
        - » Senders do not need to know the recipients of the message
    - Publish-subscribe systems (Distributed event-based systems)
    - Message queues
      - Offer a point to point service
      - Consumer processes could be notified of the arrival of a new message in the queue
    - Tuple spaces
      - Write, read, or remove structured data (tuples) to/from persistent tuple space
      - Also called generative communication
    - Distributed shared memory
      - Provide an abstraction for sharing data between processes that do not share physical memory

Architectural models…

- Roles and responsibilities
  - ***What (potentially changing) roles and responsibilities do they have in the overall architecture?***
  - Architectural styles based on roles
    - Client-server
    - Peer to peer

Architectural models...

- Roles and responsibilities
  - Client-server
    - Direct and simple
    - Limitation
      - Scalability
        - » Placement of services
    - Suggested solutions
      - Placement strategies

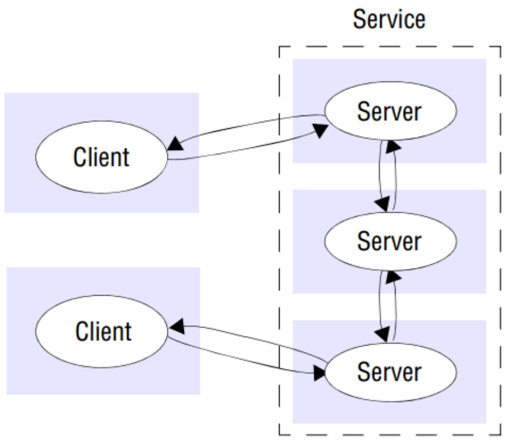


  - Peer-to-peer
    - All processes involved in a task play similar roles
    - No distinction between client and server processes/computers

Architectural models…

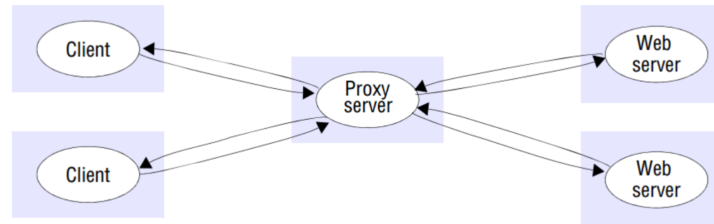
- Placement strategies
  - ***How are they mapped on to the physical distributed infrastructure (what is their placement)?***

  - Placement strategies
    - Mapping of services to multiple servers
      - E.g., Web, DNS

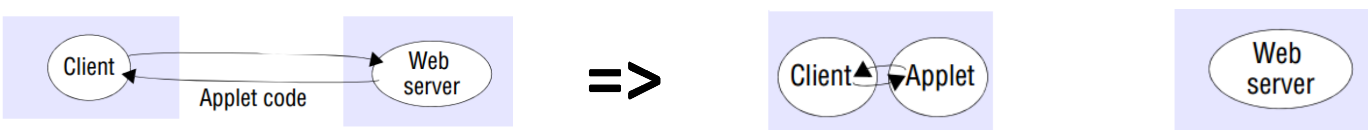


Architectural models…

- Placement strategies….
  - Caching
    - At client or proxy



  - Mobile code
    - E.g., Applet
    - Security



  - Mobile agent
    - Program travels from one computer to another in a network carrying out a task on someone’s behalf
    - Security

Fundamental models

- Models based on fundamental properties
- Allows to be specific about their characteristics and failures and security risks they might exhibit
- Contains only essential ingredients that helps to understand some aspects of a system's behavior
- Purpose
  - To make explicit all the relevant assumptions about the system we are modeling
  - To make generalization concerning what is possible and impossible

Fundamental models…

- Aspects of DS captured in the fundamental models
  - Interaction
    - Processes interact by passing messages which results in communication (information flow) and coordination (synchronization)
    - Limited level of accuracy with which independent processes can be coordinated
      - Delays
      - Difficulty to maintain the same notion of time across all machines
  - Failure
    - Define and classify faults
    - Analyze potential effects and design systems that are able to tolerate faults
  - Security
    - Define and classify attacks
    - Provide basis for the analysis of threats to a system and design of systems that resist them

Fundamental models...

- Interaction model
  - Factors affecting interaction of processes in a DS
    - Communication performance
    - Difficulty to maintain a single global notion of time
  - Performance of communication channels
    - Performance characteristics
      - Latency
        - » Time between a start of a message's transmission from one process and beginning of its receipt by another
          - Time for the first of transmitted string bits to reach its destination
          - Delay in accessing a network
          - Time taken by OS communication services at both the sending and receiving processes
      - Bandwidth of a network
        - » Total amount of information that can be transmitted over a network in a given time
      - Jitter
        - » Variation in the time taken to deliver a series of messages

Fundamental models…

- Interaction models…
  - Computer clocks and timing events
    - Two processes reading their clocks at the same time could read different time values
      - Computer clocks drift from perfect time
      - Drift rates could be different

    - Clock drift rate
      - Rate at which a computer clock deviates from a perfect reference clock

    - Correcting time on computer clocks
      - Use radio receivers to get time readings from GPS
        - » 1 microsecond accuracy
        - » GPS receivers do not work inside buildings + expensive

Fundamental models…

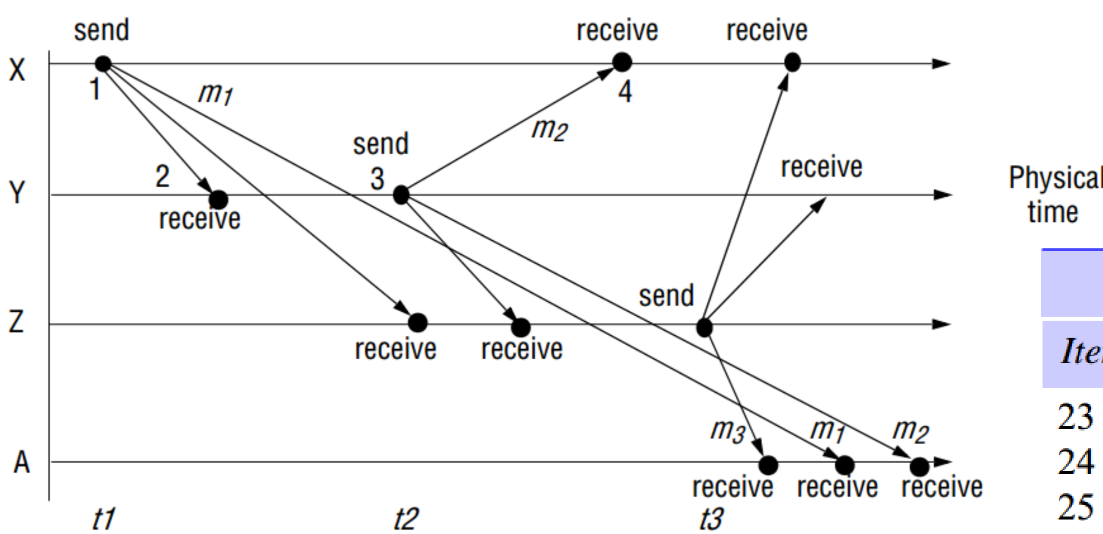
- Interaction model…
  - Two variants of interaction model
    - With and without strong assumption of time
  - Synchronous DS
    - The following bounds are defined
      - Time to execute each step of a process has known lower and upper bound
      - Each message transmitted over a channel is received within a known bounded time
      - Each process has a local clock whose drift rate from real time has a known bound
    - Difficult to arrive at realistic values and provide guarantee
      - If no guarantee, any design will be unreliable
    - Example
      - Timeout to detect the failure of a process
  - Gives an idea of how it behave in a real distributed system

Fundamental models…

- Interaction models…
  - Asynchronous DS
    - No bounds on
      - Process execution time
      - Message transmission delays
      - Clock drift rates
    - Some design problems can be solved with these assumptions
      - E.g., web cannot always provide a particular response time, a browser is designed to allow users to do other things while waiting

Fundamental models...

- Interaction models...
  - Event ordering
    - Event at one process occurred before, after, or concurrently with another event at another process
    - Execution of a system can be described in terms of events and their ordering
    - Example: email message
      - independent delays in delivery



Inbox:		
Item	*From*	*Subject*
23	Z	Re: Meeting
24	X	Meeting
25	Y	Re: Meeting

Fundamental models…

- Interaction models…
  - Event ordering…
    - Logical time
      - Provides an ordering among the events at processes running in different computers in DS
      - Example: Email ordering for X and Y
        - » X sends m_1 before Y receives m_1; Y sends m_2 before X receives m_2
        - » Y receives m_1 before sending m_2
      - Assign a number to each event corresponding to its logical ordering

Fundamental models…

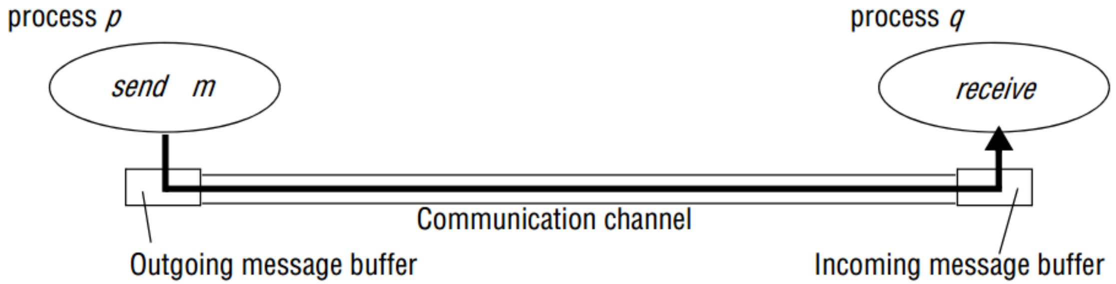
- Failure model
  - Failure
    - Processes and communication channels may depart from what is considered to be correct or desirable behavior
  - Defines the ways in which failure may occur in order to provide an understanding of the effects of failures
  - Taxonomy
    - Omission failures
    - Arbitrary failures
    - Timing failures

Fundamental models…

- Failure model…
  - Omission failures
    - Refer to cases when a process or communication channel **fails to perform actions that it is supposed to do**
  - Process omission failures
    - Mainly caused by a crash
    - Design of service that can survive the presence of faults can be simplified if it can be assumed that the services on which they depend crash cleanly
      - Detection: failure to respond
        - » Timeout
    - Fail-stop process crash
      - Other processes can certainly detect that the process has crashed
      - Can be produced in a synchronous system (if processes use timeouts)

Fundamental models…

- Failure model…
  - Communication omission failure



  - A message inserted in an outgoing message buffer never arrives at the other end's incoming message buffer
    - » Dropping messages
  - Caused by
    - » Lack of buffer space at the receiver or intervening gateway
    - » Network transmission error, detected by a checksum
- Loss of message between a sending and receiving process can also be classified as
  - Send omission failure
  - Receive omission failure
  - Channel omission failure

Fundamental models...

- Failure model...
  - Arbitrary (Byzantine) failures
    - Worst possible failure – any type of error may occur
      - E.g., a process may set or return wrong values
    - Cant be detected by seeing whether the process responds to invocations
      - Processes may arbitrarily omit intended processing steps
    - Communication channels can suffer from arbitrary failures
      - Examples
        - » Message contents could get corrupted
        - » Non-existent messages may be delivered
        - » Duplicated real messages
      - Rare b/c they are recognized by the communication software

Fundamental models...

- Failure model...
  - Timing failure
    - Applicable in synchronous DS
    - Limits are set on execution time, message delivery time and clock drift rate
    - Timing is relevant to multimedia computers with audio and video channels

Class of failure	*Affects*	*Description*
Clock	Process	Process's local clock exceeds the bounds on its rate of drift from real time.
Performance	Process	Process exceeds the bounds on the interval between two steps.
Performance	Channel	A message's transmission takes longer than the stated bound.

Fundamental models…

- Failure model…
  - Masking failures
    - Construct reliable services from components that exhibit failures
      - Knowledge of the failure characteristics of a component
    - Can be achieved by
      - Hide the failure
      - Convert the failure to a more acceptable type of failure
    - Example
      - Checksums
        - » Converts an arbitrary failure into an omission failure
        - » Request for re-transmission of omission failure
      - Replication

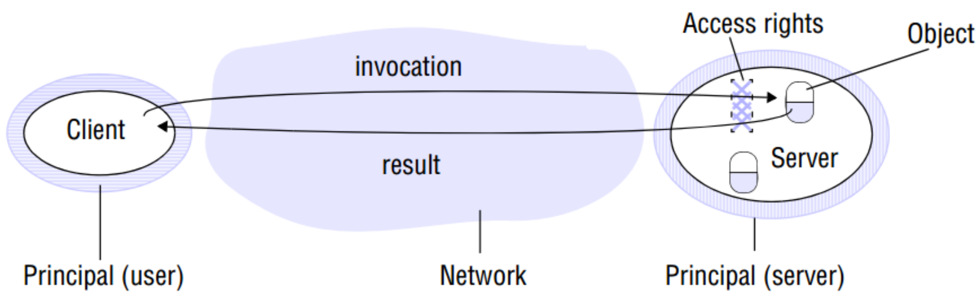
Fundamental models…

- Failure models…
  - Reliable communication
    - Defined in terms of
      - Validity
        - » Any message in the outgoing message buffer is eventually delivered to the incoming message buffer
      - Integrity
        - » The message received is identical to one sent, and no messages are delivered twice
    - Threats to integrity
      - Protocol that retransmits messages but doesn’t reject a message that arrives twice
      - Malicious users that may inject spurious messages, replay old messages, or tamper with messages

Fundamental models…

- Security model

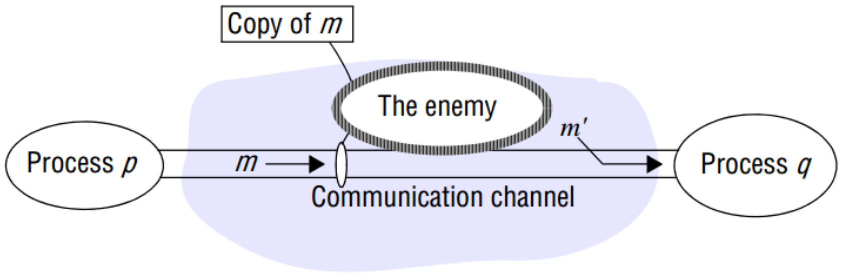
  the security of a DS can be achieved by securing the processes and the channels used for their interactions and by protecting the objects that they encapsulate against unauthorized access



  – Protecting objects
    - Objects are intended to be used in different ways by different users
      – Access rights

Fundamental models…

- Security model…
  - Securing processes and their interaction
    - Processes interact by sending messages
    - Why are processes and their interactions exposed to attacks?
      - Network and communication services used are open
      - Process interfaces are exposed
    - Enemy (adversary)
      - Capable of sending any message to any process and reading or copying any message sent between a pair of processes



Fundamental models…

- Security model…
  - Securing processes and their interaction…
    - Threats to processes

      A process that is designed to handle incoming requests may receive a message from any other process in the DS, and it cannot necessarily determine the identity of the sender

    - Threats to communication channels
      - Copy, alter, or inject messages as they travel across the network and its intervening gateways

Fundamental models…

- Security model…
  - Securing processes and their interaction…
    - Defeating security threats
      - Shared secrets
        - » Message exchanged includes information that proves the sender’s knowledge of the shared secret
      - Cryptography
        - » Keeps messages secure using encryption techniques
      - Authentication
        - » Uses shared secrets and encryption to authenticate messages
      - Secure channel
        - » Uses encryption and authentication to connect a pair of processes