Lecture 1

dinsdag 5 september 2023 12:05

a distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable

a distributed system is one in which components located at networked computers communicate and coordinate their actions only by passing messages

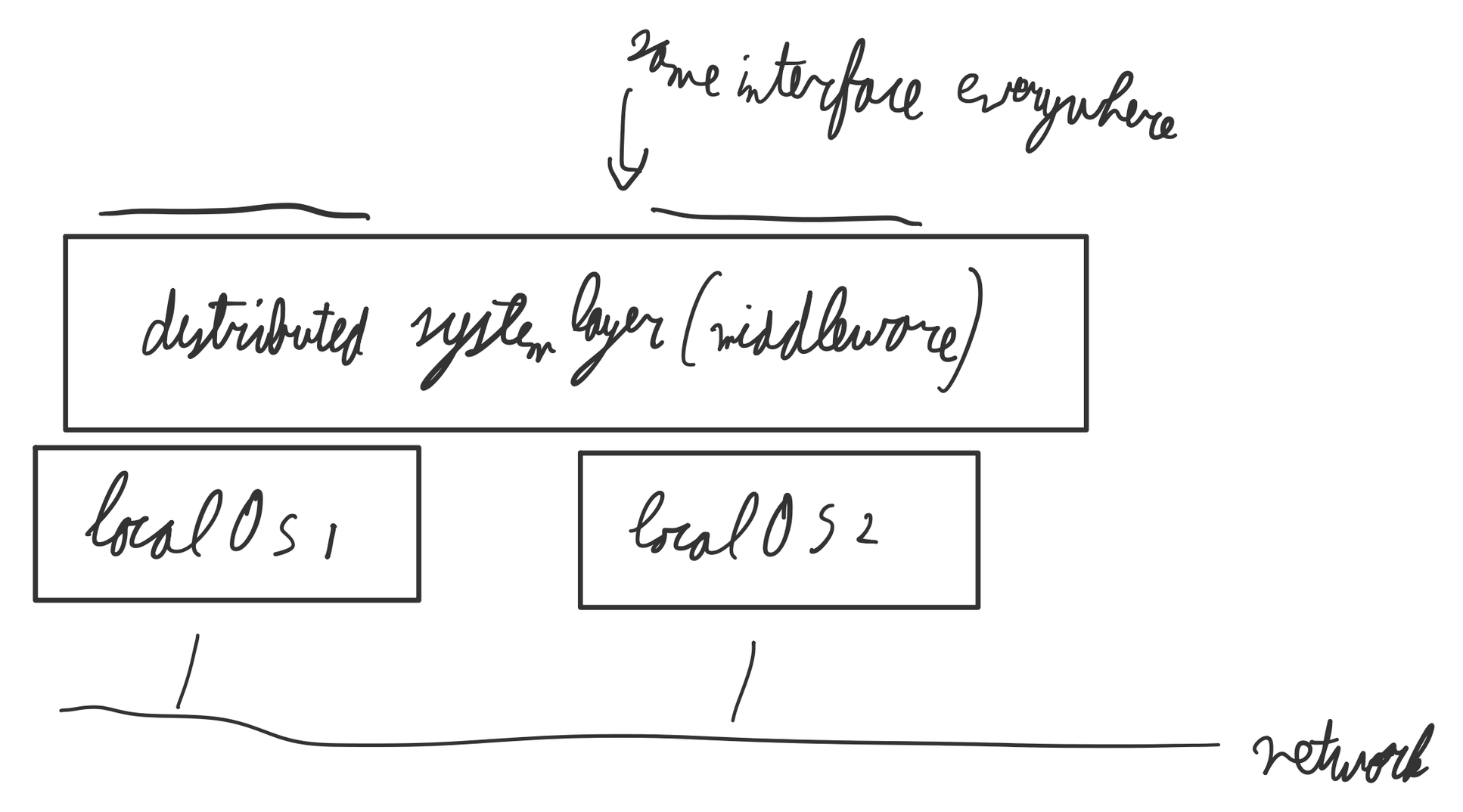
a distributed system is a collection of autonomous computing elements that appears to its users as a single coherent system

main characteristics of distributed systems

- no global notion of time
- no global notion of state
- heterogeneous resources
- network communication
- independent failures

basic ingredients

1. processing elements (nodes)
   - soft-/hardware
   - autonomous (often geographically separated)
   - heterogeneous
2. communication subsystem or network
3. software
   - operating systems, runtime systems
   - generic system services (middleware)
   - application/domain specific services (purpose of the system)

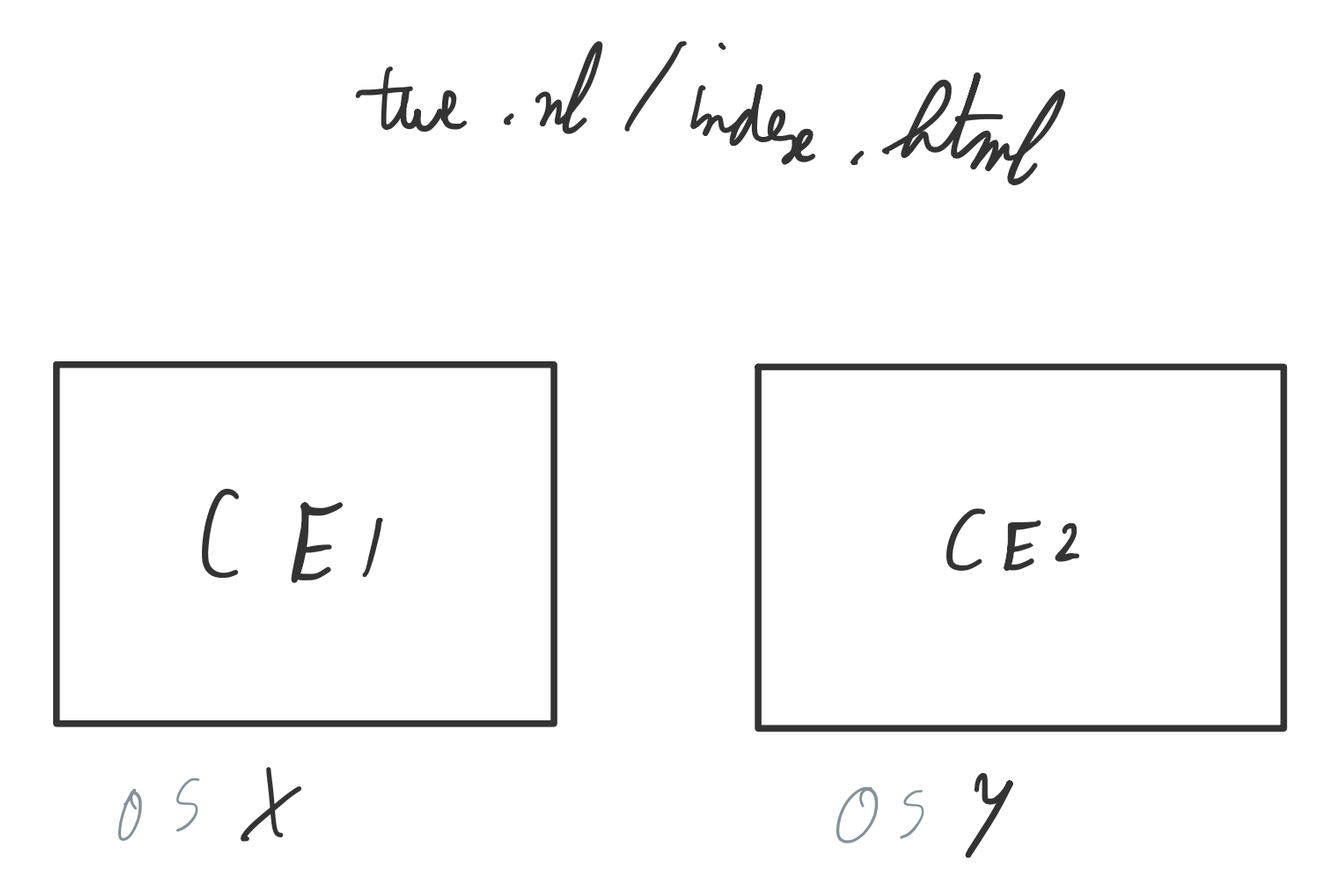


why do we need distributed systems?

unavoidable
- inherit distributed environment/application
- multiple data/resources/users at separated physical location

associated benefits
- reduced development/maintenance cost through modularity
- reduced operational cost through resource sharing
- improved performance & scalability through replication
- improved dependability through redundancy

design goals
- support of sharing resource
- distribution transparency
- scalability
- openness



access	hide differences in data representation and how objects are accessed
location	hide where an object is located
relocation	hide that an object may be moved to another location while in use
migration	hide that an object may move to another location
replication	hide that an object is replicated
concurrency	hide that an object may be shared by several independent users
failure	hide the failure and recovery of an object

ability to cope with growth

multiple dimensions

size, geographical spread, administrative domain

horizontal & vertical spreading

example: DNS ⟶ partition & distribute scales pretty well

systems should conform to well-defined interfaces

systems should easily interoperate

systems should support portability of applications

systems should be easily extensible

implementing openness requires policies & mechanisms
- allow setting caching policy, support different levels of trust for mobile code
- what level of consistency do we require for cached data, which operations may downloaded code perform?

types of distributed systems

- high-performance computing
  - cluster computing — nodes run same OS, high speed network, master-worker nodes, fully symmetric
  - grid computing — heterogeneous, dispersed across several organizations, can span a wide-area network
  - cloud computing
- distributed information systems
  - distributed transaction processing — ACID, nested transactions make this more difficult (e.g. all sub-transactions must work for the entire thing to be processed)
  - enterprise application integration — resource/service sharing, message-oriented middleware
- pervasive systems
  - ubiquitous computing systems — networked, autonomous, context-aware, intelligent
  - mobile computing systems — wireless
  - sensor networks — resource constrained (energy)

more ↑ less: reliable connectivity, computing power, data longevity, data storage, reliability, latency

cloud computing

edge computing

internet-of-things

less ↓ more: location awareness, mobility support, geographical distribution, responsiveness, interactiveness, context awareness, devices

pitfalls

- the network is reliable
- the network is secure
- the network is homogeneous
- topology doesn't change
- latency is zero
- bandwidth is infinite
- transport cost is zero
- there is one administrator