Distributed Computing*

Jie Wu

Department of Computer and Information Sciences

Temple University

*Part of the materials come from Distributed System Design, CRC Press, 1999. (Chinese Edition, China Machine Press, 2001.)

The Structure of Classnotes

- Focus
- Example
- Exercise
- Project

Table of Contents

- Introduction and Motivation
- Theoretical Foundations
- Distributed Programming Languages
- Distributed Operating Systems
- Distributed Communication
- Distributed Data Management
- Reliability
- Applications

Development of Computer Technology

- 1950s: serial processors
- 1960s: batch processing
- 1970s: time-sharing
- 1980s: personal computing
- 1990s: parallel, network, and distributed processing
- 2000s: wireless networks
- 2010s: mobile and cloud (edge, fog) computing
- 2020s: IoT, big data (AI), and blockchain (security)

Application 1: Cloud

Cloud computing

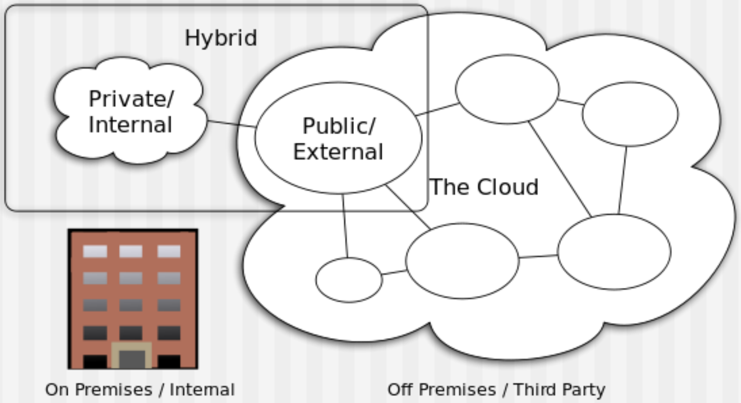
Ubiquitous access to shared pools of configurable system resources that can be rapidly provisioned with minimal management effort, often over the Internet

Characteristics (by NIST)

On-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service

Types

- Public cloud and private cloud



Cloud Computing Types

CC-BY-SA 3.0 by Sam Johnston

Products

- Amazon AWS, Microsoft Azure
- Others: Google Cloud , Alibaba, and IBM

Fog: distributed cloud

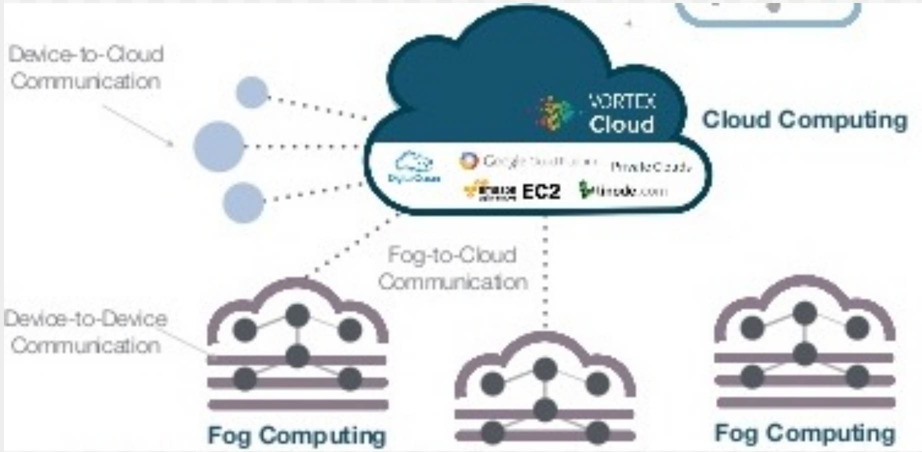
Edge: devices at the edge network (e.g., IoT)

Fog: distributed cloud (e.g. cloud + IoT)

- Reduce data communication and process demands
- Data storage and processing outside the cloud

Products

- Azure intelligent edge
- Cloudlets (CMU)


Device-to-Cloud Communication
VORTEX Cloud
Cloud Computing
EC2
Fog-to-Cloud Communication
Device-to-Device Communication
Fog Computing
Fog Computing

Application 2: Hadoop

Apache Hadoop is built for big data (ML) processing

- MapReduce: map, shuffle, and reduce
  - Pipeline
  - Data parallelism
- HDFS (Hadoop distributed file systems)

Apache HIVE

- Data warehouse on top of Hadoop
- SQL-like interface (distributed database)

* gray color: concepts to be covered in this class

SPARK: beyond Hadoop

Apache Spark is built for speed, mainly for ML

- Speed (10x to 100x compared to Hadoop)
- Data in memory (Hadoop in hard disk)
- RDD: resilient distributed dataset (extension from distributed shared memory, DSM and fault tolerance)
- Streaming
- Better API

New paradigm for reinforcement learning (RL)

- Stanford DAWN
- Berkeley Ray

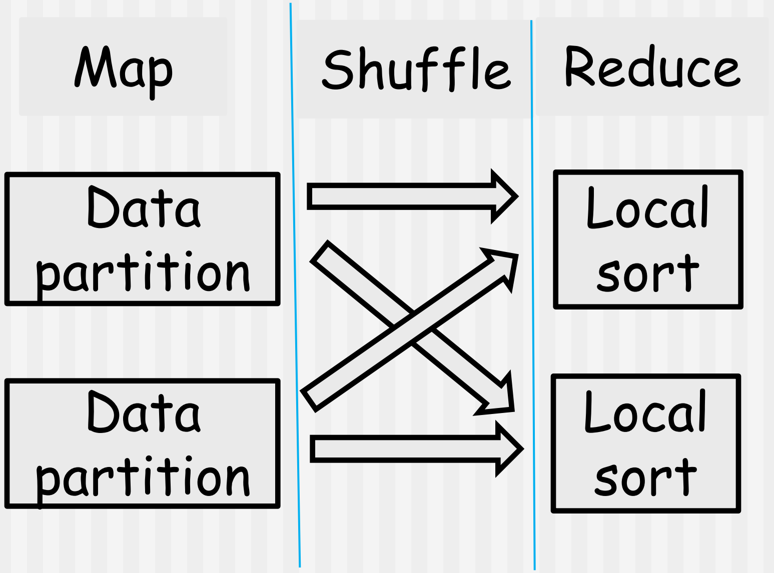
TeraSort: map-shuffle-reduce

Map-Shuffle-Reduce

- Map and Reduce: CPU-intensive
- Shuffle: I/O-intensive

TeraSort (and word count)

- Map: sample & partition data
- Shuffle: partitioned data
- Reduce: locally sort data



Parallelism and Data Size

- At least 64 Gigabytes

Minimal MapReduce Algorithms, SIGMOD'13

Application 3: Bitcoin

Bitcoin: crytocurrency and worldwide payment system

- First decentralized digital currency without a central bank or single administrator
- Transactions: use of cryptograph and is recorded in a distributed ledger called blockchain

Most crowded trade

Blockchain: building block

Blockchain: distributed database on a set of communicating nodes

A list of records (transactions), called blocks.

- Transactions: input node(s) to output node(s)
- Mining: distributed book-keeping to ensure consistency, complete, and unalterable (using cryptograph hash chain)

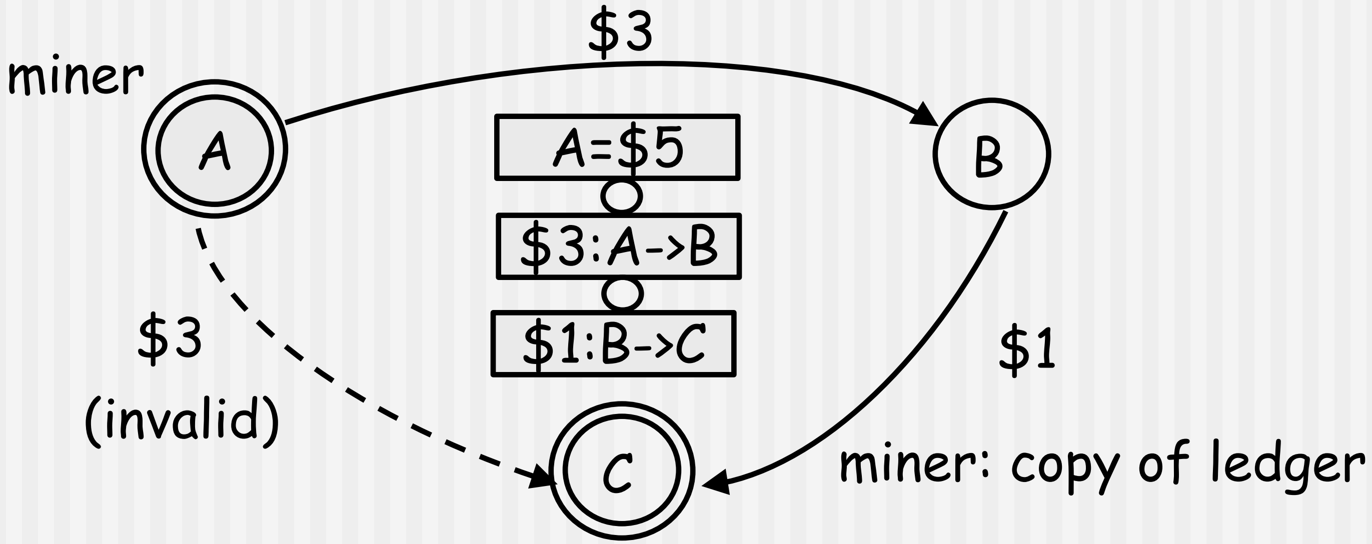
Access through smart contracts: instance of programs, with state and methods

Properties: decentralized (trust-less) service

fault tolerance and decentralized consensus

Money transfer: ledger and minor

Distributed ledger in blockchain: Users broadcast its local transactions



Miner: complete through a random process to get bitcoin

(1) validate, (2) find a key (puzzle solving based on Proof of Work, POW), and (3) broadcast result

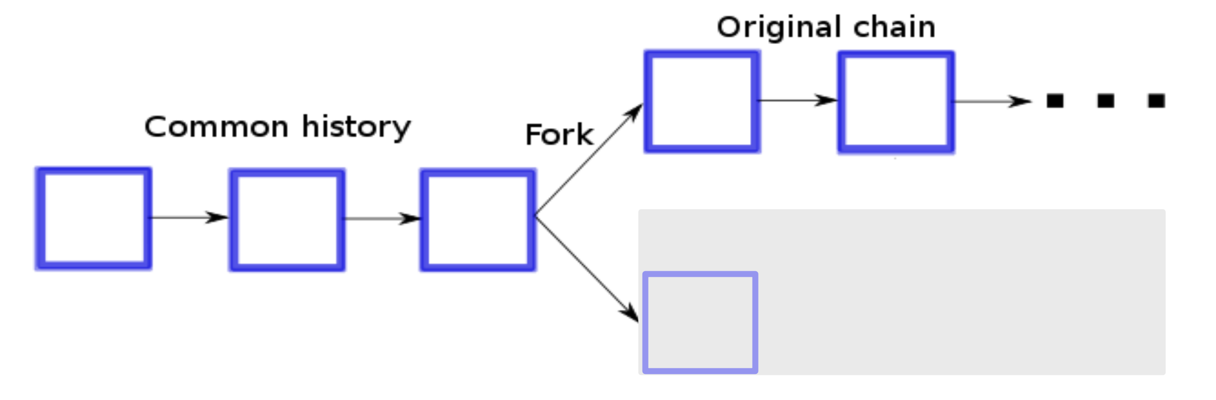
Security: digital signature, hash of previous data

Ledger consistency: removing fork

No strict consistency: temporary fork may occur

The fork consistency: following the longest chain

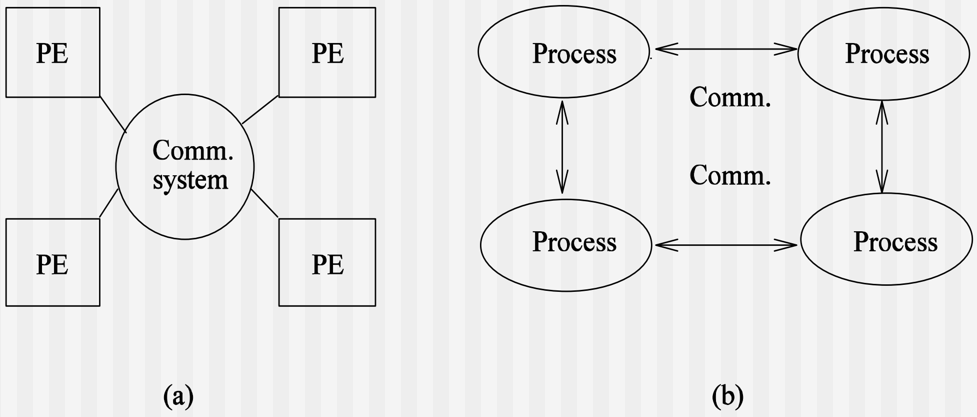
- The branch with the maximum total POW will dominate



- Scalability: layer-two solutions and sharding (store a database across multiple machines)

A Simple Definition

- A **distributed system** is a collection of independent computers that appear to the users of the system as a single computer.
- Distributed systems are "**seamless**": the interfaces among functional units on the network are for the most part invisible to the user.



System structure from the physical (a) or logical point of view (b).

Motivation

- People are distributed, information is distributed (Internet and Intranet)
- Performance/cost
- Information exchange and resource sharing (WWW and CSCW)
- Flexibility and extensibility
- Dependability

Two Main Stimuli

- Technological change
- User needs

Goals

- **Transparency**: hide the fact that its processes and resources are physically distributed across multiple computers.
  - Access
  - Location
  - Migration
  - Replication
  - Concurrency
  - Failure
  - Persistence
- **Scalability**: in three dimensions
  - Size
  - Geographical distance
  - Administrative structure

Goals (Cont'd.)

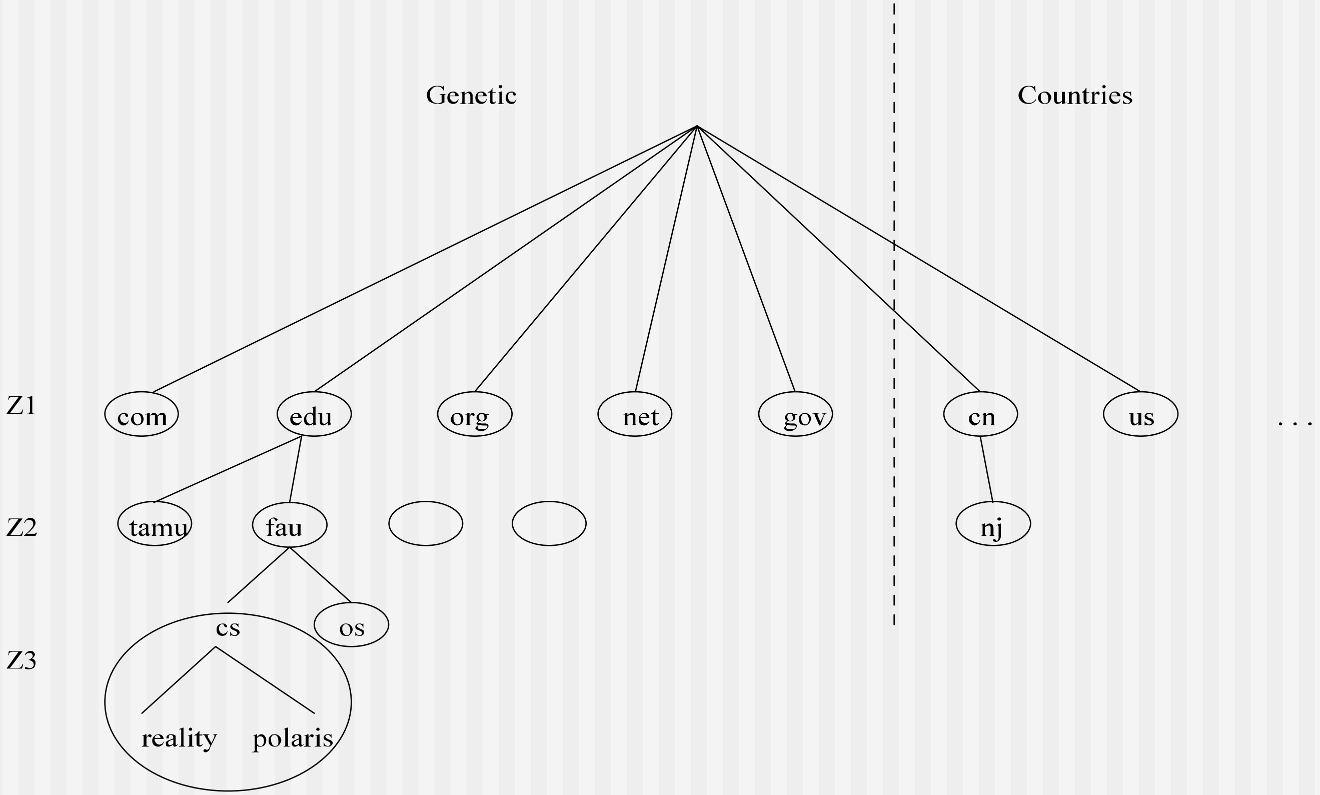
- **Heterogeneity** (mobile code and mobile agent)
  - Networks
  - Hardware
  - Operating systems and middleware
  - Program languages
- **Openness**
- **Security**
- **Fault Tolerance**
- **Concurrency**

Scaling Techniques

- Latency hiding (pipelining and interleaving execution)
- Distribution (spreading parts across the system: sharding)
  - Horizontal partitioning
    - Splits large databases into smaller components (similar to map in Terasort)
- Replication (replicate parts: caching)
  - Web caching (i.e., cookies)
  - DNS (domain name space) caching
    - Translates human readable domain names (e.g., www.cis.temple.edu) to machine readable IP addresses (e.g., 155.247.19.2)

Example 1: (Scaling Through Distribution)

URL searching based on hierarchical DNS name space (partitioned into zones).



DNS name space.

Design Requirements

- Performance Issues
    - Responsiveness
    - Throughput
    - Load Balancing
- Quality of Service
    - Reliability
    - Security
    - Performance
- Dependability
    - Correctness
    - Security
    - Fault tolerance

Similar and Related Concepts

- Distributed
- Network
- Parallel
- Concurrent
- Decentralized

Schroeder's Definition

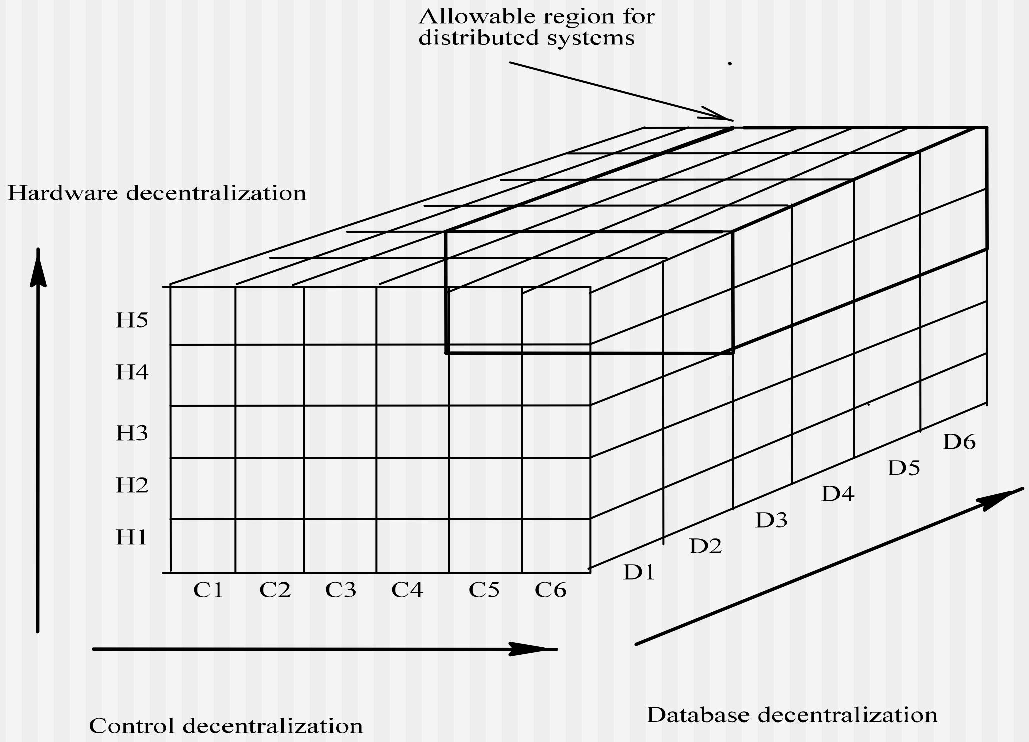
- A list of **symptoms** of a distributed system
  - Multiple processing elements (PEs)
  - Interconnection hardware
  - PEs fail independently
  - Shared states

Focus 1: Enslow's Definition

Distributed system = distributed hardware + distributed control + distributed data

A system could be classified as a distributed system if all three categories (hardware, control, data) reach a certain degree of decentralization.

Focus 1 (Cont'd.)


Allowable region for distributed systems
Hardware decentralization
H5
H4
H3
H2
H1
C1
C2
C3
C4
C5
C6
D1
D2
D3
D4
D5
D6
Control decentralization
Database decentralization

Enslow's model of distributed systems.

Hardware

- A single CPU with one control unit.
- A single CPU with multiple ALUs (arithmetic and logic units).There is only one control unit.
- Separate specialized functional units, such as one CPU with one floating-point co-processor.
- Multiprocessors with multiple CPUs but only one single I/O system and one global memory.
- Multicomputers with multiple CPUs, multiple I/O systems and local memories.

Control

- Single fixed control point. Note that physically the system may or may not have multiple CPUs.
- Single dynamic control point. In multiple CPU cases the controller changes from time to time among CPUs.
- A fixed master/slave structure. For example, in a system with one CPU and one co-processor, the CPU is a fixed master and the co-processor is a fixed slave.
- A dynamic master/slave structure. The role of master/slave is modifiable by software.
- Multiple homogeneous control points where copies of the same controller are used.
- Multiple heterogeneous control points where different controllers are used.

Data

- Centralized databases with a single copy of both files and directory.
- Distributed files with a single centralized directory and no local directory.
- Replicated database with a copy of files and a directory at each site.
- Partitioned database with a master that keeps a complete duplicate copy of all files.
- Partitioned database with a master that keeps only a complete directory.
- Partitioned database with no master file or directory.

Network Systems

- Performance scales on **throughput** (transaction response time or number of transactions per second) versus **load**.
- Work on burst mode.
- Suitable for small transaction-oriented programs (collections of small, quick, distributed **applets**).
- Handle uncoordinated processes.

Parallel Systems

- Performance scales on **elapsed execution times** versus number of processors (subject to either Amdahl or Gustafson law).
- Works on bulk mode.
- Suitable for numerical applications (such as SIMD or SPMD vector and matrix problems).
- Deal with one single application divided into a set of coordinated processes.

Distributed Systems

A compromise of network and parallel systems.

Comparison

Item	Network sys.	Distributed sys.	Multiprocessors
Like a virtual uniprocessor	No	Yes	Yes
Run the same operating system	No	Yes	Yes
Copies of the operating system	N copies	N copies	1 copy
Means of communication	Shared files	Messages	Shared files
Agreed up network protocols?	Yes	Yes	No
A single run queue	No	Yes	Yes
Well defined file sharing	Usually no	Yes	Yes

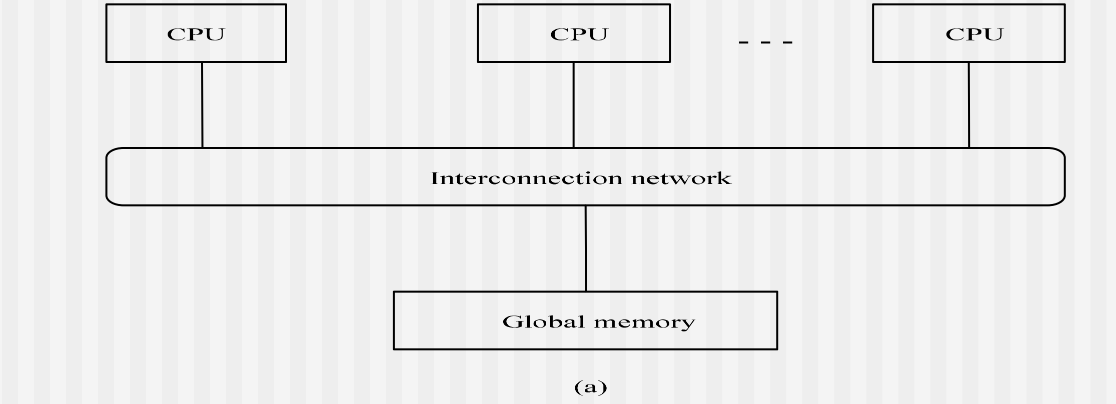
Comparison of three different systems.

Focus 2: Different Viewpoints

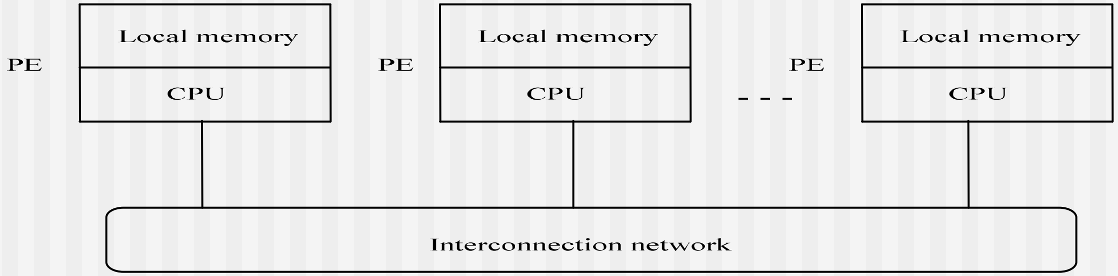
- Architecture viewpoint
- Interconnection network viewpoint
- Memory viewpoint
- Software viewpoint
- System viewpoint

Architecture Viewpoint

- **Multiprocessor**: physically shared memory structure
- **Multicomputer**: physically distributed memory structure.



(a)

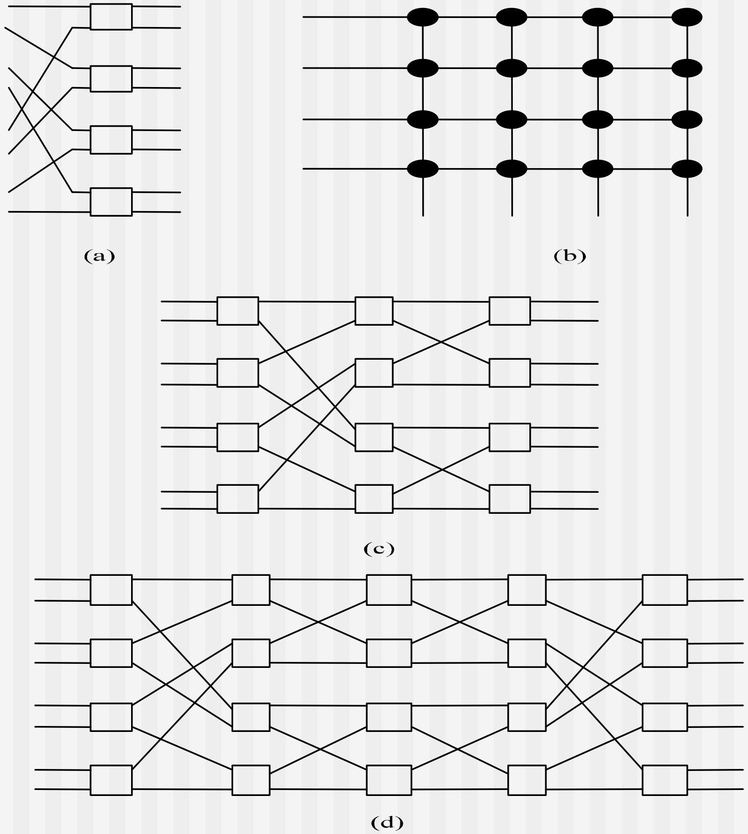


(b)

Interconnection Network Viewpoint

- static (point-to-point) vs. dynamics (ones with switches).
- bus-based (Fast Ethernet) vs. switch-based (routed instead of broadcast).

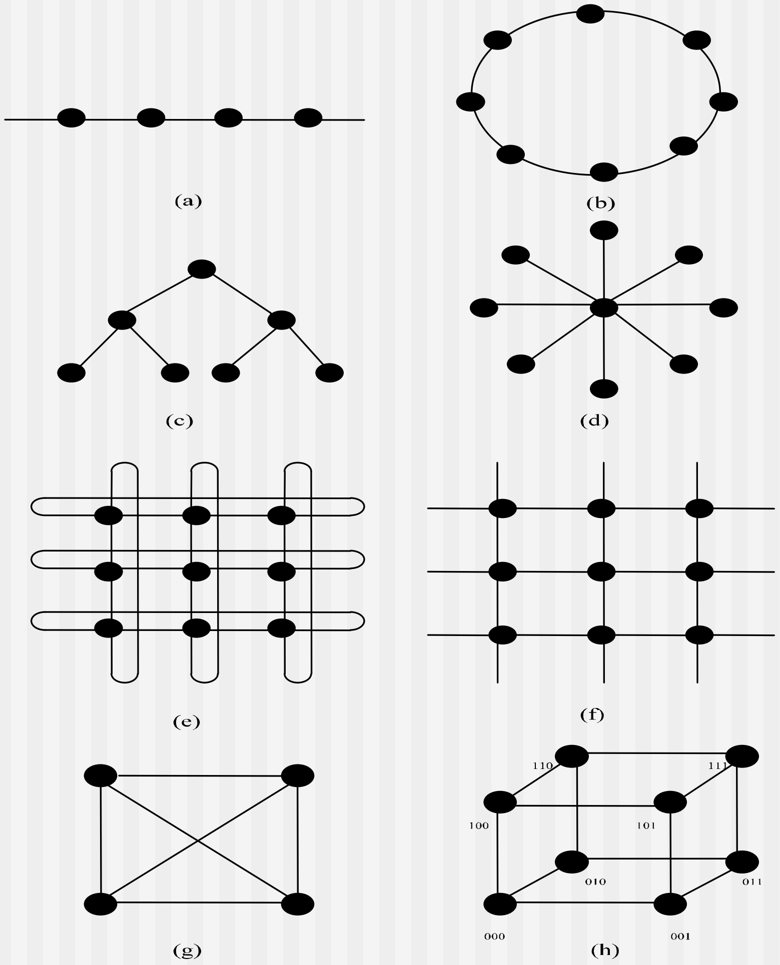
Interconnection Network Viewpoint (Cont’d.)



Examples of dynamic interconnection networks: (a) shuffle-exchange, (b) crossbar (e.g., Cisco 12000 series switches), (c) baseline, and (d) Benes.

(a) and (c) are blocking
(d) is non-blocking (i.e., any unused input and output can be connected)

Interconnection Network Viewpoint (Cont’d.)



Examples of static interconnection networks: (a) linear array, (b) ring, (c) binary tree, (d) star, (e) 2-d torus, (f) 2-d mesh, (g) completely connected, and (h) 3-cube.

Measurements for Interconnection Networks

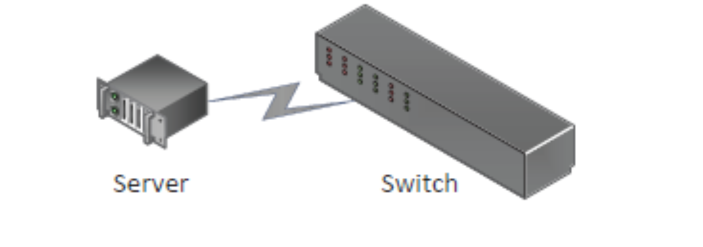
- *Node degree*. The number of edges incident on a node.
- *Diameter*. The maximum shortest path between any two nodes.
- *Bisection width*. The minimum number of edges along a cut which divides a given network into equal halves.

Application 4: Data Center Network (DCN)

- DCN connects many servers to handle demands of cloud

Three types of connections:

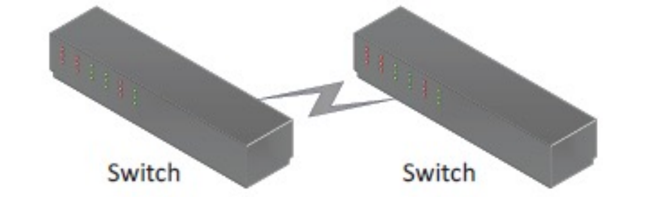
- Server-switch connection (a)
- Switch-switch connection (b)
- Server-server connection (c)



(a) Server-switch connection

Two classes of DCNs:

- Switch-centric
  - Only server-switch and switch-switch connections (a and b), no server-server
  - Eg, Flattened Clos, Fat-Tree
- Server-centric
  - Mostly, only server-switch and server-server connections (a and c), no switch-switch
  - Eg: Dcell, BCube, FiConn



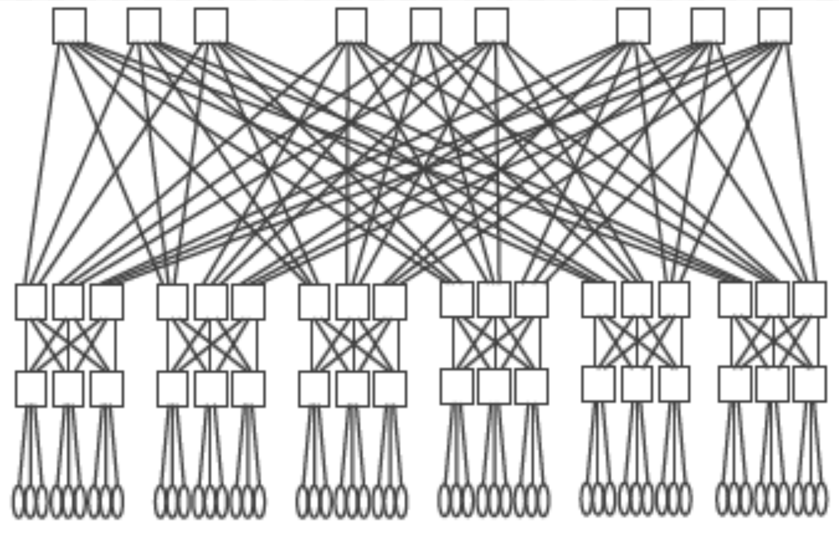
(b) Switch-switch connection

Server
Server

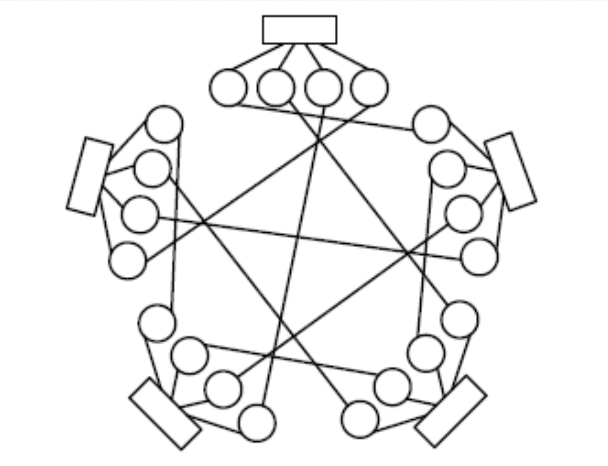
(c) Server-server connection

Switch- and Server-centric DCNs

Folded Clos (switch-centric)



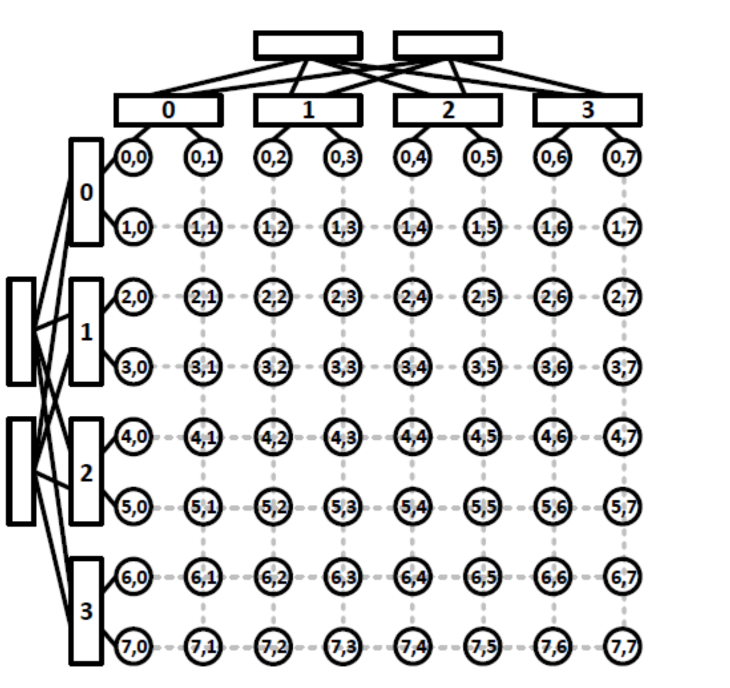
Dcell (server-centric)



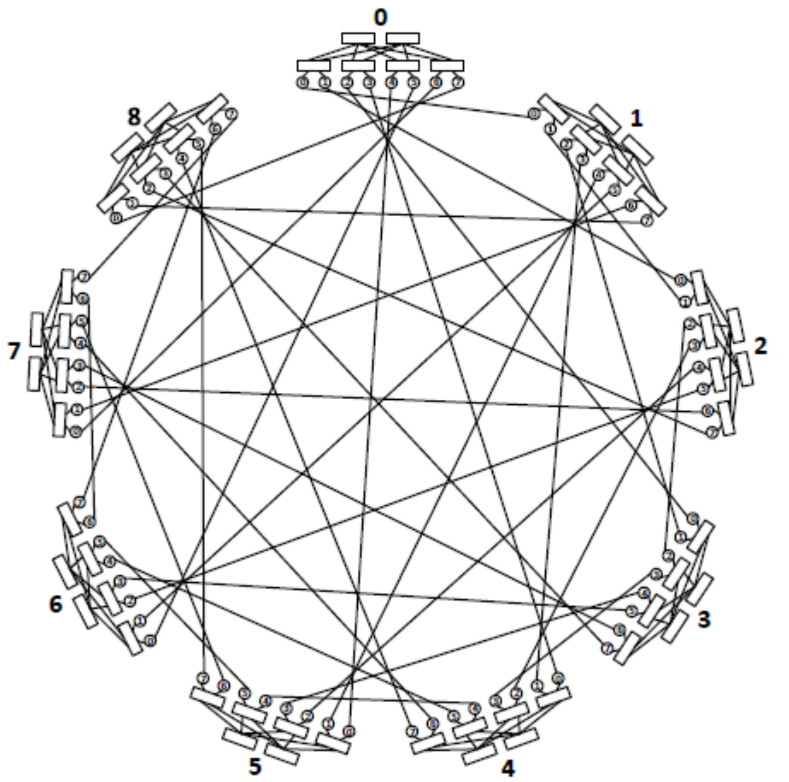
D. Li, J. Wu, Z. Liu, and F. Zhang, "Dual-Centric Data Center Network Architectures," ICPP, 2015.

Dual-Centric DCNs

Three configurations (a), (b), and (c)



(a) Final interconnection of FSquare(4)



(b) Final interconnections of FCell(4).

What's the Best Choice? (Siegel 1994)

- A **compiler-writer** prefers a network where the transfer time from any source to any destination is the same to simplify the data distribution.
- A **fault-tolerant researcher** does not care about the type of network as long as there are three copies for redundancy.
- A **European researcher** prefers a network with a node degree no more than four to connect Transputers.

What's the Best Choice? (Cont’d.)

- A **college professor** prefers hypercubes and multistage networks because they are theoretically wonderful.
- A **university computing center official** prefers whatever network is least expensive.
- A **NSF director** wants a network which can best help deliver health care in an environmentally safe way.
- A **Farmer** prefers a wormhole-routed network because the worms can break up the soil and help the crops!

Memory Viewpoint

	Logically shared	Logically distributed
Physically shared	Shared memory	Simulated message passing
Physically distributed	Distributed shared memory	Message passing

Physically versus logically shared/distributed memory.

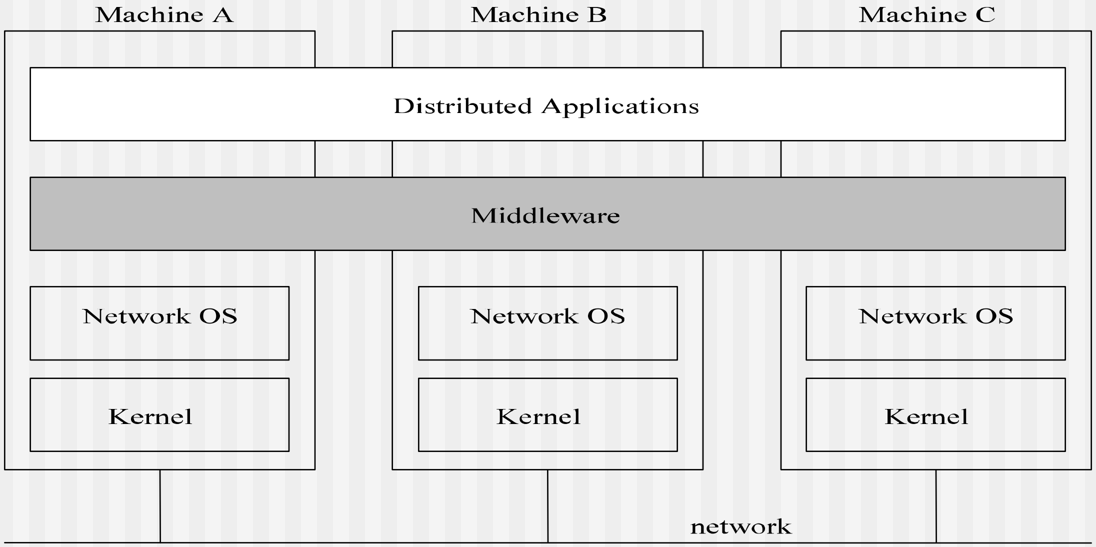
Software Viewpoint

- Distributed systems as resource managers like traditional operating systems.
  - Multiprocessor/Multicomputer OS
  - Network OS
  - Middleware (on top of network OS)

- Modern OS
  - Hypervisor creates and runs virtual machines (VMs) by virtually sharing its resources, such as memory and processing units.
  - Ability to run different operating systems and configurations.
  - Example: VMware, Zen, and Kurbernetes (containers on Docker)

Service Common to Many Middleware Systems

- High level communication facilities (access transparency)
- Naming
- Special facilities for storage (integrated database)



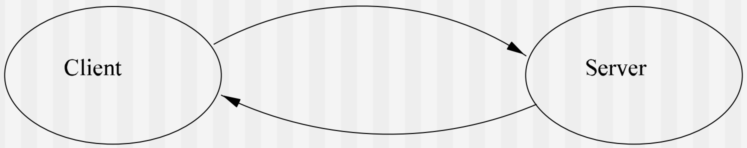
Middleware

System Viewpoint

- The division of responsibilities between system components and placement of the components.

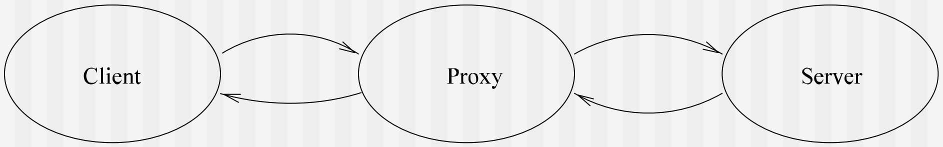
Client-Server Model

- Client-server



(a)

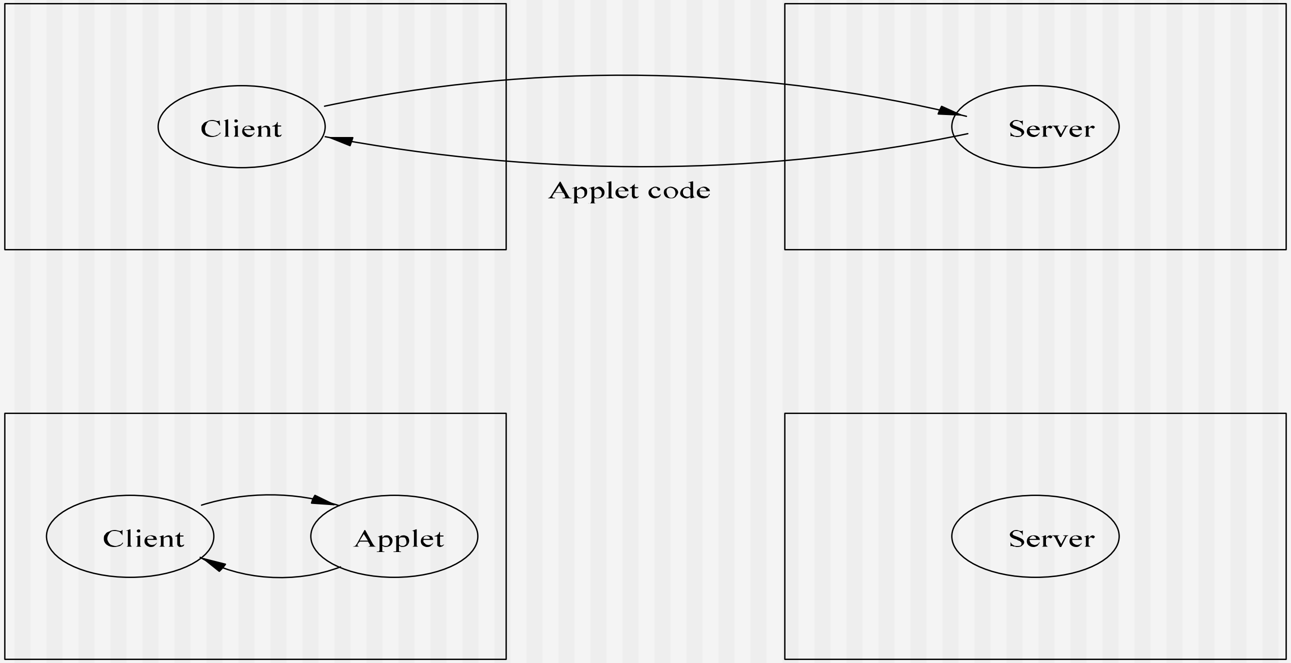
- Proxy server (multiple instance cases)



(b)

(a) Client and server and (b) proxy server.

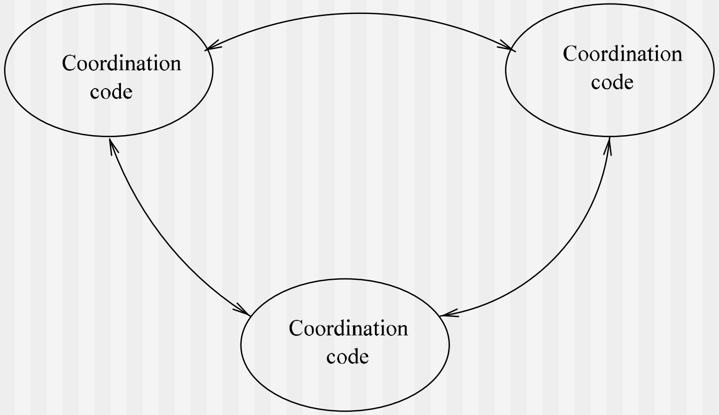
Mobile Code and Mobile Agents



Mobile code (web applets).

Peer Processes

Peer-to-Peer: P2P



Future trend

Microservice: P2P

(instead of monolithic based on client-server)

Hardware vs. Software

- Hardware is physical and tangible components of a computer.

- Software is a set of well-written instructions written in programming language.

- Certain functions can be implemented by either hardware or software
  - E.g., software-defined network (SDN)

Future Trend on Distributed Systems

- Cloud as a service
  - Infrastructure as a Service (IaaS)
  - Planform as a Service (PaaS)
  - Software as a Service (SaaS)
- Elastic computing (adapted to workload changes)
  - Virtualization
    - Virtual machines (containers)
    - Virtual networks
    - Virtual storage (NAS and SAN technology)
  - Automation and orchestration
    - Automation: AIops using ML/AI
    - Orchestration: automated replication and parallelism (Kubernetes)

D. Comer: The Cloud Computing Book: The Future of Computing Explained, 2021

Key Issues (Stankovic's list)

- Theoretical foundations
- Reliability
- Privacy and security
- Design tools and methodology
- Distribution and sharing
- Accessing resources and services
- User environment
- Distributed databases
- Network research

Wu's Book

- Distributed Programming Languages
  - Basic structures
- Theoretical Foundations
  - Global state and event ordering
  - Clock synchronization
- Distributed Operating Systems
  - Mutual exclusion and election
  - Detection and resolution of deadlock
  - self-stabilization
  - Task scheduling and load balancing
- Distributed Communication
  - One-to-one communication
  - Collective communication

Wu's Book (Cont’d.)

- Reliability
  - Agreement
  - Error recovery
  - Reliable communication
- Distributed Data Management
  - Consistency of duplicated data
  - Distributed concurrency control
- Applications
  - Distributed operating systems
  - Distributed file systems
  - Distributed database systems
  - Distributed shared memory
  - Distributed heterogeneous systems

Wu's Book (Cont’d.)

- Part 1: Foundations and Distributed Algorithms
- Part 2: System infrastructure
- Part 3: Applications

What is Distributed Algorithms

- Parallel Computing: efficiency
- Real-Time: On-time computing
- Distributed (Message-Passing) Algorithms
  - Dealing with delay and uncertainty
  - Simplicity, elegance, and beauty are first-class citizens

  (Michel Raynal, 2013)

Distributed Algorithms

- Termination
  - In a social network, each person exchanges his/her friend list with friends. What is the stoppage condition?
- Global State
  - How to design an observation algorithm by observing an execution without modifying its behavior?
- Distributed Consensus
  - How to reach distributed consensus (e.g., binary decisions) in the presence of traitors?

Distributed Algorithms (Cont’d)

- Logical Clock
  - How to order events in different systems with asynchronous clocks? How to discard obsolete data?
- Data
  - How to replicate data and keep them consistent?
- Load
  - How to distribute load in a load balanced way?
- Routing
  - How to perform efficient routing that is deadlock-free and fault-tolerant?

References

- IEEE Transactions on Parallel and Distributed Systems (TPDS)
- IEEE International Conference on Distributed Computing Systems (ICDCS) (theory/system)
- IEEE International Conference on Reliable Distributed Systems (SRDS) (theory/system)
- ACM Symposium on Principles of Distributed Computing (PODC) (theory)
- IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGrid) (system)
- ACM Special Interest Group on Data Communication (SIGCOMM) (system)

Exercise 1

1. In your opinion, what is the future of the computing and the field of distributed systems?

2. Use your own words to explain the differences between distributed systems, multiprocessors, and network systems.

3. Calculate (a) node degree, (b) diameter, (c) bisection width, and (d) the number of links for an *n* x *n* 2-d mesh, an n x *n* x *n* 3-d torus, and an *n*-dimensional hypercube.