TCS332 Fundamental of Information Security and Blockchain

B. Tech CSE III Semester

Instructor:

Dr Mohammad Wazid

Professor, Department of CSE

Graphic Era (Deemed to be University), Dehradun, India

Email: wazidkec2005@gmail.com

Homepage: https://sites.google.com/site/mwazidiiith/home

Distributed System

- Also known as distributed computing.
- A distributed system is a collection of independent components located on different machines that share messages with each other in order to achieve common goals.
- It is a system with multiple components located on different machines that communicate and coordinate actions in order to appear as a single coherent system to the end-user.

Distributed System

- The machines that are a part of a distributed system may be computers, physical servers, virtual machines, containers, or any other node that can connect to the network, have local memory, and communicate by passing messages.

In DS nodes (sites) execute a common job and communicate among each other through some messages.

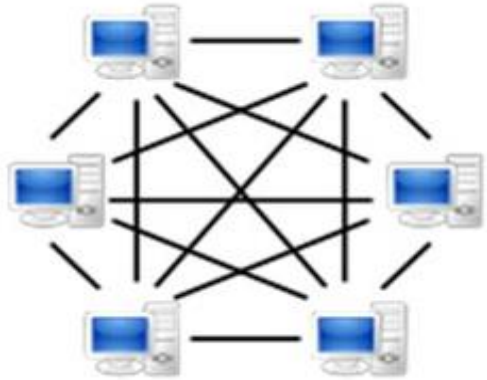

Fig. Scenario of distributed computing

Working of distributed systems

- Each machine works toward a common goal and the end-user views results as one cohesive unit.
- Each machine has its own end-user and the distributed system facilitates sharing resources or communication services.
- Distributed systems have three primary characteristics:

-- all components run concurrently,

-- there is no global clock,

-- all components fail independently of each other.

Functions of distributed systems

- **Resource sharing** - whether it's the hardware, software or data that can be shared
- **Concurrency** - multiple machines can process the same function at the same time
- **Scalability** - how do the computing and processing capabilities multiply when extended to many machines
- **Fault tolerance** - how easy and quickly the failures of the system be detected and recovered
- **Transparency** - how much access does one node have to locate and communicate with other nodes in the system.
- **Openness** - how open is the software designed to be developed and shared with each other

Architecture models of distributed systems

- **Client-server:**
- Clients contact the server for data, then server formats it and display it to the end-user.
- The end-user can also make a change from the client-side and commit it back to the server to make it permanent.

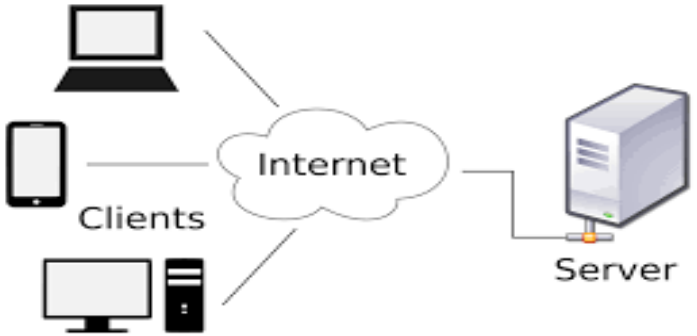

Fig. Client-server architecture of distributed systems

Architecture models of distributed systems

- **Three-tier:**
- Information about the client is stored in a middle tier rather than on the client to simplify application deployment.
- This architecture model is most common for web applications.

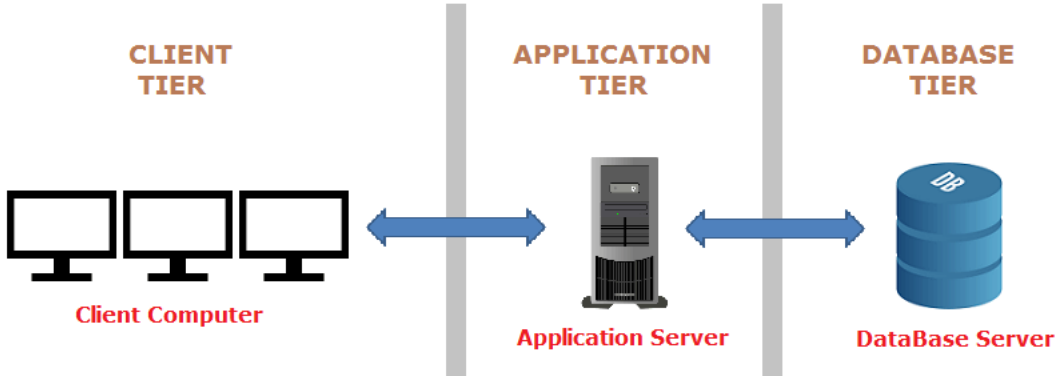

Fig. Three tier architecture of distributed systems (image source: https://www.softwaretestingmaterial.com/software-architecture/

Architecture models of distributed systems

- **n-tier:**
- Generally used when an application or server needs to forward requests to additional enterprise services on the network.

- **Adding the nodes up to n-tier.**
- **Value of n varies as per the requirements of the computing environment**

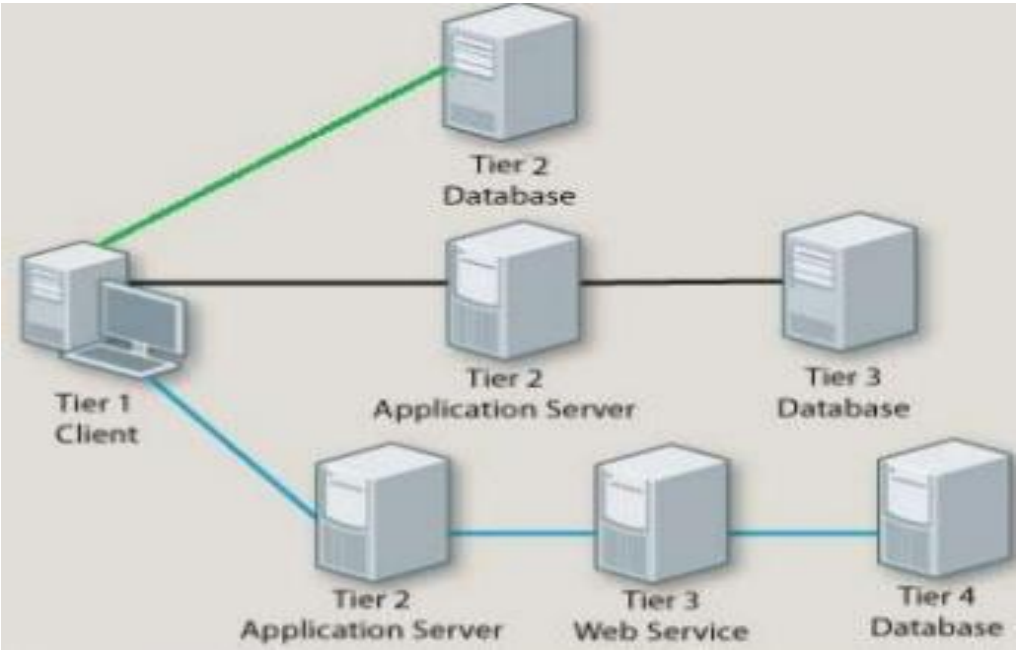

Fig. n-tier architecture of distributed systems (image source: https://www.slideshare.net/Manojksh/ntier-application-architecture

Architecture models of distributed systems

- **Peer-to-peer:**
- There are no additional machines used to provide services or manage resources.
- Responsibilities are uniformly distributed among machines in the system, known as peers, which can serve as either client or server.

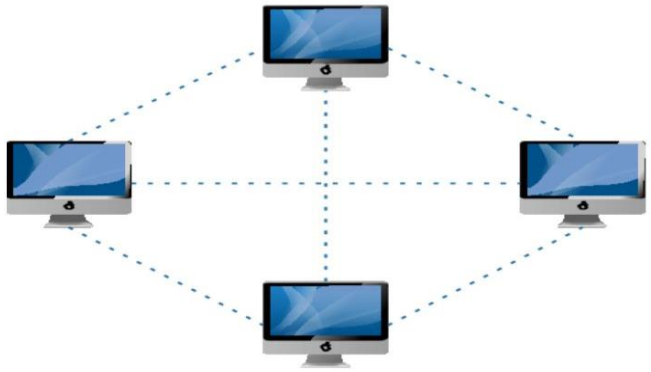

Fig. p2p architecture of distributed systems

Lesson learned

- Distributed systems can be made up of any machine that is capable of connecting to a network, having local memory, and communicating by passing messages mechanism.
- By spreading out requests and workloads, distributed systems can support far more requests and compute more jobs than a standard single system.
- You can create a fast distributed system through certain computing devices, containers, and virtual machines.

Examples

- Distributed systems have endless use cases, a few being electronic banking systems, massive multiplayer online games, and sensor networks.

Paxos

- Paxos are used for the agreement.
- Paxos is an algorithm that is used to achieve consensus among a distributed set of computers.
- Paxos is a set of protocols for solving consensus in a network of unreliable processors.

The Consensus Problem

- Suppose you have a collection of computers and want them all to agree on something.
- This is what consensus is about;
- Consensus means agreement.
- Consensus comes up frequently in distributed systems design.
- There are various reasons why we want it: **to agree on who gets access to a resource (mutual exclusion), agree on who is in charge (elections)**, or **to agree on a common ordering of events among a collection of computers** (i.e., what action should be taken next).

The Consensus Problem

- The consensus problem can be stated in a basic, generic manner: ***One or more systems may propose some value. How do we get a collection of computers to agree on exactly one of those proposed values?***
- The formal properties for asynchronous consensus are:
- **Validity:** only the proposed values can be decided. If a process decides on some value *v*, then some process must have proposed *v*.
- **Uniform Agreement:** no two correct processes can decide on different values.
- **Integrity:** each process can decide a value at most once.
- **Termination:** all processes will eventually decide on a result.

Paxos

- One or more clients propose a value to Paxos, and we have a consensus when most systems running Paxos agree on one of the proposed values.
- Paxos is widely used and is legendary in computer science since it is the first consensus algorithm that has been rigorously proved to be correct.

Paxos

- Paxos simply selects a single value from one or more values that are proposed to it and lets everyone know what that value is.
- A run of the Paxos protocol results in the selection of single proposed value.
- If you need to use Paxos to create a replicated log (for a replicated state machine), then you need to run Paxos repeatedly.
- This is called multi-Paxos. There are some optimizations that could be implemented for multi-Paxos.

Roles in paxos

- Paxos describes the actions of the processors by their roles in the protocol: client, acceptor, proposer, learner, and leader.
- **Client**
- The Client issues a request to the distributed system, and waits for a response. For instance, a write request on a file in a distributed file server.
- **Acceptor (Voters)**
- Acceptors are collected into groups called Quorums. Any message sent to an Acceptor must be sent to a Quorum of Acceptors. Any message received from an Acceptor is ignored unless a copy is received from each Acceptor in a Quorum.
- **Proposer**
- A Proposer advocates a client request, attempting to convince the Acceptors to agree on it, and acting as a coordinator to move the protocol forward.

Roles in paxos

- **Learner**
- Learners act as the replication factor for the protocol. Once a client's request has been agreed upon by the Acceptors, the Learner may take action (i.e., execute the request and send a response to the client).
- **Leader**
- Paxos requires a distinguished Proposer (called the leader) to make progress. Many processes may believe they are leaders, but the protocol only guarantees progress if one of them is eventually chosen. If two processes believe they are leaders, then there may be some mechanism to resolve this issue. **(There will be only one leader at a time.)**

Roles in paxos

- **Quorums**
- Quorums express the consistency properties of Paxos by ensuring at least some surviving processor retains knowledge of the results.
- **Quorums are defined as subsets of the set of Acceptors such that any two subsets (that is, any two Quorums) share at least one member.**
- Typically, a Quorum is any majority of participating Acceptors.
- For example, given the set of **Acceptors {A,B,C,D},** a majority Quorum would be any three Acceptors: **{A,B,C}, {A,C,D}, {A,B,D}, {B,C,D}.**

Safety/consistency due to Paxos

- To guarantee safety/consistency Paxos defines three properties and ensures the first two are always held, regardless of the pattern of failures:
- **Validity:** Only proposed values can be chosen and learned.
- **Agreement or consistency:** No two distinct learners can learn different values (or there can’t be more than one decided value). For example, value of a is 2 then it should be a=2 for all learners.
- **Termination:** If value C has been proposed, then in the end learner L will learn some value (however, sufficient processors remain non-faulty).

Working of paxos

- **Proposers:**
- Receive requests (values) from clients and try to convince acceptors to accept their proposed values.
- **Acceptors:**
- Accept certain proposed values from proposers and let proposers know if something else was accepted. A response from an acceptor represents a vote for a particular proposal.
- **Learners:**
- Announce the outcome.
- In practice, it is common for Paxos to coexist with **a single node may run proposer, acceptor, and learner role.** The service that requires consensus (e.g., distributed storage) on a set of replicated servers, **with each server taking on all three roles rather than using separate servers.**

References:

- William Stallings, “Cryptography and Network Security: Principles and Practice”, Pearson publication, 2020.
- George Icahn, “Blockchain: the complete guide to understanding blockchain technology”, 2020.
- Antony lewis, “The basics of bitcoins and blockchains: an introduction to cryptocurrencies and the technology that powers them” 2020.
- Blockchain information available at: https://mlsdev.com/blog/156-how-to-build-your-own-blockchain-architecture
- Blockchain information available at: https://www.euromoney.com/ learning/blockchain-explained/what-is-blockchain
- N. Garg, M. Wazid, A. K. Das, D. P. Singh, J. J. P. C. Rodrigues, and Y. Park. "BAKMP-IoMT: Design of Blockchain Enabled Authenticated Key Management Protocol for Internet of Medical Things Deployment," in IEEE Access, DOI: 10.1109/ACCESS.2020.2995917.2020, Online. (2018 SCI Impact Factor: 4.098)
- M. Wazid, A. K. Das, S. Shetty, and M. Jo. "A Tutorial and Future Research for Building a Blockchain-Based Secure Communication Scheme for Internet of Intelligent Things," in IEEE Access, Vol. 8, No. 1, pp. 88700-88716, 2020, DOI: 10.1109/ACCESS.2020.2992467. (2018 SCI Impact Factor: 4.098)
- pBFT information available at: https://crushcrypto.com/what-is-practical-byzantine-fault-tolerance/
- Paxos information available at: https://www.cs.rutgers.edu/~pxk/417/notes/paxos.html