MICROLINK INFORMATION TECHNOLOGY COLLEGE
DISTRIBUTED SYSTEM ASSIGNMENT
COMPUTER SCIENCE DEPARTMENT
DEFINITIONS OF DISTRIBUTED SYSTEMS

Group Name:

- Girum Sisay —--- 14873/20
- Kasahun Alemayehu —--- 15032/20
- Mikias Endale —--- 14672/20
- Nahom Asgele —-- 14872/20
- Yoseph Zemede —-- 14875/20

Submitted to: Ins. Anteneh
Date: May, 2, 2023

Definition of Distributed Systems

A distributed system is a way to use computers in which different parts are spread out over a network of computers or other computing devices. These devices divide up the work and work together to get the job done faster and better than if just one device had been in charge of it.

The term "distributed system" refers to a group of independent computer systems that are linked via a centralized computer network using distributed system software while being geographically distant from one another. Each system's autonomous computers will interact with one another by exchanging data and resources while carrying out the duties given to them.

A distributed system contains multiple nodes that are physically separate but linked together using the network. All the nodes in this system communicate with each other and handle processes in tandem. Each of these nodes contains a small part of the distributed operating system software.

Distributed computing is a model in which software system components are shared between multiple processors or nodes. Even if the software components are distributed across multiple computers in multiple locations, they are operated as a single system. This is done to increase productivity and effectiveness. Sending messages back and forth, the systems on various networked computers communicate and coordinate to complete a predefined task.

Distributed systems are an important part of IT and computer science because more and more jobs are getting so big and complicated that a single computer wouldn't be able to handle them. But distributed computing has more benefits than standard ways of working with computers. Distributed systems lower the risks of having a single point of failure, making them more reliable and able to handle mistakes. Modern distributed systems are usually made to be scalable in close to real time. You can also add more computing resources on the fly, which boosts speed and shortens the time it takes to finish.

Why Distributed Systems?

Distributed systems are necessary for modern computing to exist. They are indispensable for the operation of wireless networks, cloud computing services, and the Internet. None of these technologies would exist absent distributed systems.

Do we still require distributed systems for enterprise-level tasks that lack the complexity of a full-scale communications network? The answer is typically affirmative. Distributed systems provide scalability and enhanced performance in ways that monolithic systems cannot, and because they can rely on the capabilities of other computing devices and processes, they can offer features that would be difficult or impossible to implement on a single system.

Distributed systems offer a number of advantages over monolithic, or single, systems, including:

- **Greater flexibility**: It is easier to add computing power as the need for services grows. In most cases today, you can add servers to a distributed system on the fly.
- **Reliability**: A well-designed distributed system can withstand failures in one or more of its nodes without severely impacting performance. In a monolithic system, the entire application goes down if the server goes down.
- **Enhanced speed**: Heavy traffic can bog down single servers when traffic gets heavy, impacting performance for everyone. The scalability of distributed databases and other distributed systems makes them easier to maintain and also sustain high-performance levels.

Problems of Distributed Systems

Distributed systems are considerably more complex than monolithic computing environments, and raise a number of challenges around design, operations and maintenance. These include:

- **Increased opportunities for failure:** The more systems added to a computing environment, the more opportunity there is for failure. If a system is not carefully designed and a single node crashes, the entire system can go down. While distributed systems are designed to be fault tolerant, that fault tolerance isn't automatic or foolproof.

- **Synchronization process challenges:** Distributed systems work without a global clock, requiring careful programming to ensure that processes are properly synchronized to avoid transmission delays that result in errors and data corruption. In a complex system — such as a multiplayer video game — synchronization can be challenging, especially on a public network that carries data traffic.

- **Imperfect scalability:** Doubling the number of nodes in a distributed system doesn't necessarily double performance. Architecting an effective distributed system that maximizes scalability is a complex undertaking that needs to take into account load balancing, bandwidth management and other issues.

- **More complex security:** Managing a large number of nodes in a heterogeneous or globally distributed environment creates numerous security challenges. A single weak link in a file system or larger distributed system network can expose the entire system to attack.

- **Increased complexity:** Distributed systems are more complex to design, manage and understand than traditional computing environments.

The challenges of distributed systems as outlined above create a number of correlating risks. These include:

- **Security:** Distributed systems are as vulnerable to attack as any other system, but their distributed nature creates a much larger attack surface that exposes organizations to threats.

- **Risk of network failure:** Distributed systems are beholden to public networks in order to transmit and receive data. If one segment of the internet becomes unavailable or overloaded, distributed system performance may decline.

- **Governance and control issues:** Distributed systems lack the governability of monolithic, single-server-based systems, creating auditing and adherence issues around global privacy laws such as GDPR. Globally distributed environments can impose barriers to providing certain levels of assurance and impair visibility into where data resides.

- **Cost control:** Unlike centralized systems, the scalability of distributed systems allows administrators to easily add additional capacity as needed, which can also increase costs. Pricing for cloud-based distributed computing systems are based on usage (such as the number of memory resources and CPU power consumed over time). If demand suddenly spikes, organizations can face a massive bill.

Characteristics of Distributed Systems

Distributed systems are commonly defined by the following key characteristics and features:

- **Scalability**: The ability to grow as the size of the workload increases is an essential feature of distributed systems, accomplished by adding additional processing units or nodes to the network as needed.
- **Concurrency**: Distributed system components run simultaneously. They're also characterized by the lack of a "global clock," when tasks occur out of sequence and at different rates.
- **Availability/fault tolerance**: If one node fails, the remaining nodes can continue to operate without disrupting the overall computation.
- **Transparency**: An external programmer or end user sees a distributed system as a single computational unit rather than as its underlying parts, allowing users to interact with a single logical device rather than being concerned with the system's architecture.
- **Heterogeneity**: In most distributed systems, the nodes and components are often asynchronous, with different hardware, middleware, software and operating systems. This allows the distributed systems to be extended with the addition of new components.

Organization and Goals of Distributed Systems

Organization

Distributed systems are frequently composed of complex software whose components are, by definition, distributed across multiple computers. In order to control their complexity, it is essential that these systems be well-organized. There are numerous methods to view the organization of a distributed system, but one of the most apparent is to distinguish between the logical organization of the collection of software components and their physical manifestation.
A distributed system is broadly divided into two essential concepts — software architecture (further divided into layered architecture, object-based architecture, data-centered architecture, and event-based architecture) and system architecture (further divided into client-server architecture and peer-to-peer architecture).

Goals

- **Making Resources Accessible:** The main goal of a distributed system is to make it easy for the users (and applications) to access remote resources, and to share them in a controlled and efficient way. Resources can be just about anything, but typical examples include things like printers, computers, storage facilities, data, files, Web pages, and networks, to name just a few. There are many reasons for wanting to share resources. One obvious reason is economics. For example, it is cheaper to let a printer be shared by several users in a small office than having to buy and maintain a separate printer for each user. Likewise, it makes economic sense to share costly resources such as supercomputers, high-performance storage systems, imagesetters, and other expensive peripherals.

- **Distribution Transparency:** An important goal of a distributed system is to hide the fact that its processes and resources are physically distributed across multiple computers. A distributed system that is able to present itself to users and applications as if it were only a single computer system is said to be transparent

- **Openness:** Another important goal of distributed systems is openness. An open distributed system is a system that offers services according to standard rules that describe the syntax and semantics of those services. For example, in computer networks, standard rules govern the format, contents, and meaning of messages sent and received. Such rules are formalized in protocols. In distributed systems,services are generally specified through interfaces, which are often described in an Interface Definition Language (IDL). Interface definitions written in an IDL nearly always capture only the syntax of services. In other words, they specify precisely the names of the functions that are available together with types of the parameters, return values, possible exceptions that can be raised, and so on. The hard part is specifying precisely what those services do, that is, the semantics of interfaces. In practice, such specifications are always given in an informal way by means of natural language.

- **Scalability:** scalability is one of the most important design goals for developers of distributed systems. Scalability of a system can be measured along at least three different dimensions (Neuman, 1994). First, a system can be scalable with respect to its size, meaning that we can easily add more users and resources to the system. Second, a geographically scalable system is one in which the users and resources may lie far apart. Third, a system can be administratively scalable, meaning that it can still be easy to manage even if it spans many independent administrative organizations. Unfortunately, a system that is scalable in one or more of these dimensions often exhibits some loss of performance as the system scales up.