Unit 5
Distributed Systems

Structure of the Unit

1. Unit Outcomes
2. Distributed Systems
3. Advantages of Distributed Systems
4. Types of Distributed Operating Systems
5. Robustness and Design Issues of Distributed OS
6. Self-Assessment Questions
7. Self-Assessment Activities
8. Multiple Choice Questions
9. Keys to Multiple Choice Questions
10. Summary of the Unit
11. Keywords
12. Recommended Resources for Further Reading

5.1 Unit Outcomes

After the successful completion of this unit, the student will be able to:

1. Gain a comprehensive understanding of distributed systems, including their advantages
2. Explore various types of distributed operating systems.
3. Analyze the robustness and design issues inherent to distributed operating systems.

5.2 Distributed Systems

A distributed system is a network of processors that operate independently without sharing memory or a clock. Each processor has its local memory, and they communicate with each other through networks like local-area or wide-area networks. These systems can encompass various types of processors, from small handheld devices to personal computers, workstations, and large mainframe computers.

In the context of distributed systems, a distributed file system is a service that spreads its users, servers, and storage devices across different locations within the distributed network. Unlike a centralized system with a single data repository, a distributed file system relies on multiple independent storage devices. This approach requires service activities to be conducted across the network.

The advantages of a distributed system include providing users with access to the system's resources, which enhances computation speed and improves data availability and reliability. However, due to their distributed nature, these systems must address challenges related to process synchronization, communication, deadlock prevention, and handling failures, which are not encountered in centralized systems. This unit delves into the fundamental structure of distributed systems and the networks that

link them. Furthermore, it highlights the key distinctions in operating system design when comparing these systems to centralized ones.

A distributed system consists of loosely connected processors, linked through a communication network. From the perspective of a specific processor within this system, all other processors and their resources are considered remote, while its resources are considered local.

These processors in a distributed system come in various sizes and serve different functions. They can range from small microprocessors to workstations, minicomputers, and large general-purpose computer systems. Depending on the context, they may be referred to as sites, nodes, computers, machines, or hosts. In this context, "site" denotes the location of a machine, while "host" refers to a specific system at a site. Typically, one host at one site, known as the server, possesses a resource that another host at a different site, referred to as the client or user, wishes to access. A generalized structure of a distributed system is depicted in Figure 5.1.

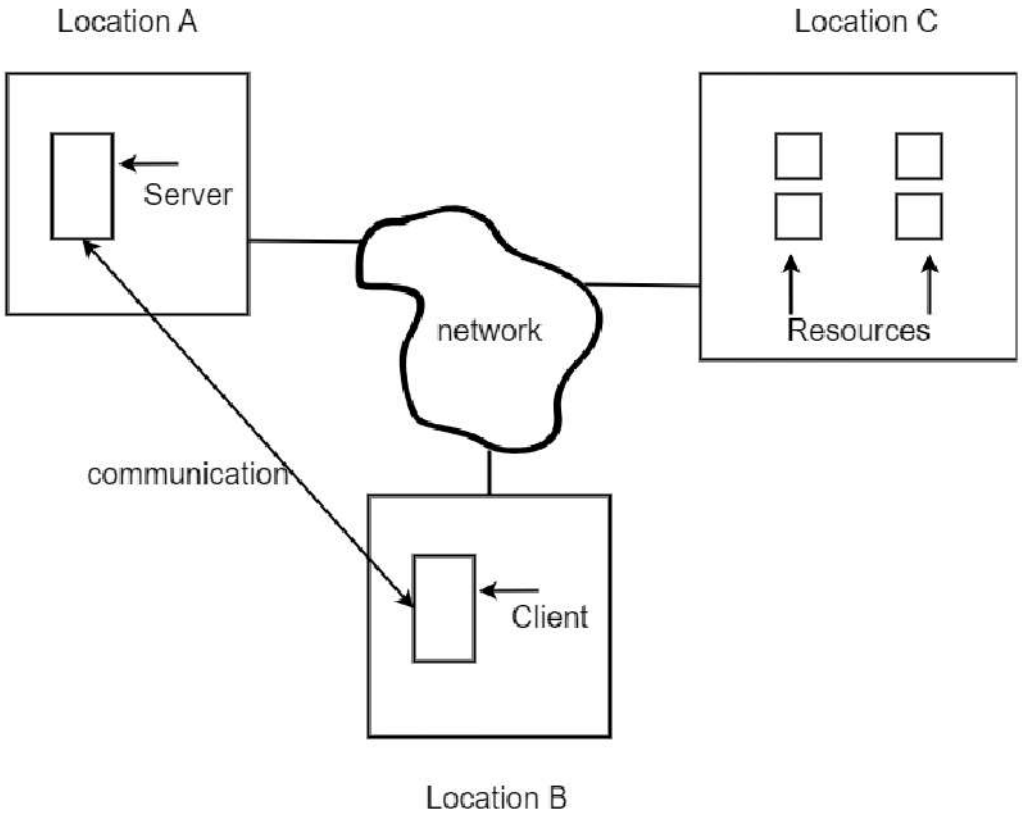

Figure 5.1: Distributed System

In summary, a Distributed System comprises a network of autonomous computers, which are often physically and geographically distant from each other, and do not have shared physical memory or synchronized clocks. Instead, each processor within this system possesses its dedicated local memory, and these processors interact through both local and wide area networks. It's important to note that the nodes or computers in a distributed system can have diverse hardware architectures, meaning they may differ significantly in terms of their underlying hardware components and capabilities.

5.3 Advantages of Distributed Systems

There are four primary motives for constructing distributed systems: resource sharing, computation acceleration, reliability, and communication. Each of these rationales is briefly discussed below.

1. Resource Sharing: In a distributed system, different sites with varying capabilities are interconnected. This allows users at one site to access resources available at other sites. For example, a user at Site A can utilize a laser printer at Site B, while a user at Site B can access a file residing at Site A. In general, distributed systems provide mechanisms for remote file sharing, distributed database processing, remote file printing, utilization of specialized remote hardware devices (such as high-speed array processors), and various other operations.

2. Computation Speedup: Distributed systems facilitate the division of a complex computation into sub computations that can run concurrently. These sub computations can be distributed across multiple sites, enabling concurrent execution and improved performance. Moreover, if one site is overwhelmed with tasks, some jobs can be shifted to less loaded sites, a process known as automated load sharing, although this is not yet common in commercial systems.

3. Reliability: Distributed systems offer increased reliability because if one site fails, the remaining sites can continue to function. In scenarios where the system comprises multiple large autonomous installations, the failure of one should not impact the rest. However, in systems with small machines, each responsible for a critical function, a single failure can disrupt the entire system. Redundancy in both hardware and data can help the system continue operations even if some sites fail. Detecting site failures, taking appropriate action, and integrating failed sites back into the system poses complex challenges.

4. Communication: When various sites are linked by a communication network, users at different locations can exchange information. Messages are passed between systems similar to how processes communicate within a single-computer message system. This communication network allows all the higher-level functionalities present in standalone systems, including file transfer, login, mail, and remote procedure calls (RPCs), to extend to distributed systems. The advantage of distributed systems is that these functions can be carried out across significant distances, enabling remote collaboration on projects, file sharing, program execution, and coordination, even for geographically distant users.

The benefits of distributed systems have led to a widespread trend in decentralization, with many companies replacing mainframes with networks of workstations or personal computers. This shift results in increased functionality for the cost, enhanced flexibility in resource location and facility expansion, improved user interfaces, and easier maintenance.

5.4 Distributed Operating System

5.4.1 Distributed OS Basics

A distributed operating system is a complex network of interconnected computers that operate within a virtual environment. Users interact with this virtual shell to perform various tasks, while the system's architecture delegates task execution to one or more computers within this virtual environment. Here, we delve into the key aspects:

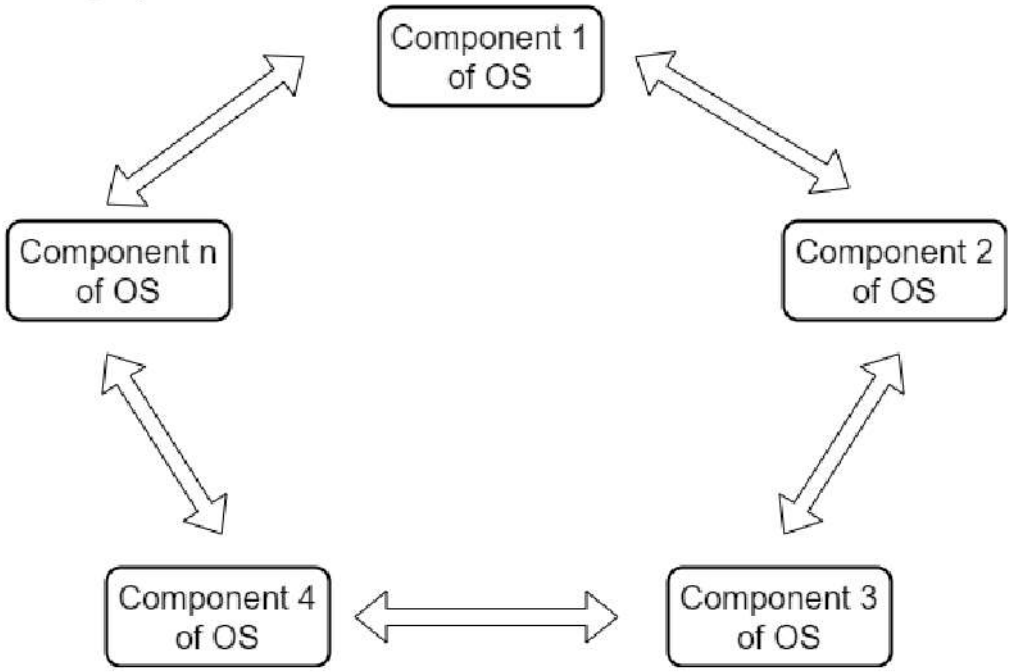

Figure 5.2:View of Distributed OS

- **User and Developer Perspectives**: The distributed operating system encompasses a boundary represented by a circle with arrows (as depicted in Figure 5.2). This boundary separates the user's viewpoint from the internal system details, ensuring that users remain unaware of the intricacies. Developers, on the other hand, work with components within this boundary. The number of available components, denoted from 1 to N, can vary depending on the specific scenario.

- **Component Diversity**: Within a distributed operating system, the components can comprise computers with varying operating systems located at different sites. These components are seamlessly linked via a robust local area network, playing a pivotal role in executing tasks within this distributed architecture.

- **Multifaceted Operating System Functions**: A distributed operating system handles a range of functions commonly associated with operating systems, such as managing processes, input/output operations, memory allocation, file organization, security and protection, and network management. Each of these functions is managed by distinct components interconnected through the network.

- **Unified User Experience**: When a user submits a command to the distributed operating system, the instruction may necessitate the involvement of one or more components within the operating system. Despite the user's perception of a unified system, the internal architecture comprises subsystems working collaboratively to achieve the overarching goals of the distributed operating system.

- **Distributed Processing**: Distributed systems offer the advantage of distributing workloads among computers that can communicate with one another. True distributed processing, on the other hand, involves separate computers working in harmony to accomplish a common objective. This approach necessitates a highly structured environment where hardware and software interact seamlessly, share resources, and freely exchange information.

5.4.2 Resources in Distributed OS

In a distributed operating system, users can access remote resources much like they would access local ones. The system manages data and processes migration between different sites, ensuring efficient resource utilization and flexibility. This section explores these aspects in detail:

- **Data Migration**: If a user on one site (e.g., Site A) needs access to data residing on another site (e.g., Site B), there are two basic methods for transferring the data. The first approach is to transfer the entire file to Site A, making all further access local. However, this can be inefficient, especially when only a portion of the file is needed. The second approach is to transfer only a portion of the file that is needed. The second approach is more efficient, as it transfers only the necessary parts of the file, akin to demand paging. This method, used by protocols like Sun Microsystems' NFS and Microsoft's SMB, minimizes data transfer over the network.

- **Computation Migration**: In some cases, it's more efficient to transfer the computation itself rather than the data. For instance, if a task requires processing data from various sites, it's better to access the data at their locations and return the results. This can be done through remote procedure calls (RPCs) or by creating a new process on the target site. Processes may run concurrently, and this method increases efficiency when the data transfer time is longer than executing the remote command.

- **Process Migration**: An extension of computation migration, process migration allows the execution of a process or its parts at different sites. This can be motivated by load balancing, computation speedup, hardware or software preferences, and efficient data access. Two techniques are used: one that hides the migration from the user, often for load balancing and computation speedup, and another that involves user input, typically for situations requiring hardware or software preferences.

The World Wide Web, for example, exhibits several aspects of a distributed computing environment. It

enables data migration, computation migration (e.g., database operations), and even process migration in the form of Java applets sent from servers to clients for execution. While network operating systems offer some of these features, a distributed operating system streamlines and simplifies them, resulting in a powerful and user-friendly system, contributing to the immense growth of the World Wide Web.

5.4.3 Evolution of Distributed Operating Systems

In the early days of computing, computers had limited computing power. However, advancements in technology led to exponential increases in processing speed and power. As technology progressed, minicomputers emerged to handle varying data volumes, while personal computers managed more complex tasks. Nevertheless, tasks involving massive data analysis were still reliant on mainframe computers. Researchers explored alternatives and introduced the concept of distributed operating system architecture. This architecture involved the delegation of responsibilities among multiple interconnected computers within a network, resulting in enhanced processing speed and power. This architectural approach is not solely about hardware but also involves a synergy of hardware and software to enhance system efficiency and effectiveness.

- **Role of RPCs**: Remote Procedure Calls (RPCs) significantly contributed to the development of distributed architecture. Their importance grew with the advent of the TCP/IP protocol (Transmission Control Protocol/Internet Protocol). RPCs enable individuals to execute various commands and instructions on remote computers within a network, provided they have the necessary permissions. This feature has elevated the prominence and relevance of distributed operating system architectures, ultimately delivering higher processing power to end-users.

- **Architectural Advancements**: The initial architectural framework was designed for batch processing and was further refined with the introduction of minicomputers. Minicomputers allowed researchers to incorporate the master/slave concept, where one computer served as the master, overseeing the execution of tasks by connected slave nodes. As computer sizes significantly decreased and the processing power of microcomputers increased, processing within a distributed operating system became more efficient.

- **Modern Network Capabilities**: Present-day networks offer exceptional performance, scalability, and security, enabling high-end servers to leverage network advancements for distributed computing. This has been made possible through the integration of strong, reliable, and secure networks.

In summary, a distributed operating system creates a network of interconnected computers operating within a virtual environment. Users interact with this virtual shell, while the architecture optimizes task execution and enhances processing power. The evolution of distributed operating systems has been driven by advancements in technology, making it a versatile and powerful system.

5.5 Types of Distributed Operating System

5.5.1 Peer-to-Peer Systems:

In a peer-to-peer (P2P) distributed operating system, every computer or node is considered equal in terms of functionality. Each node can act both as a client and a server, meaning it can request resources and provide resources to other nodes. Nodes can share resources like processing power, storage, and bandwidth with each other, promoting a decentralized approach. P2P systems are commonly used in applications like file sharing, instant messaging, and gaming. These systems are also known as "Loosely Coupled Systems" because there is no central authority; instead, peers interact directly with one another.
Example: BitTorrent is a classic example of a P2P system. In BitTorrent, users can both download and upload files. There is no central server; instead, users connect to each other to exchange parts of files, making it a distributed file-sharing system.

5.5.2 Client-Server Systems:

In a client-server distributed operating system, the architecture is based on a client-server model.
Servers offer specific services or resources, and clients initiate requests to the server to access these services. The server responds by providing the requested service or resource. Client-server systems are frequently used in enterprise applications, where centralization and control over resources are essential.
Example: The World Wide Web operates on a client-server model. When you access a website, your web browser (the client) sends a request to a web server, which then provides the web page's content, such as HTML, images, and videos. The web server is the central authority that serves these resources to clients.

5.5.3 Middleware:

Middleware is a distinct type of distributed operating system, and it's more of a software layer that resides between the operating system and application software. It provides a suite of services that facilitate communication between different applications operating on separate machines. Middleware is instrumental in building distributed systems capable of running across multiple platforms and enabling interoperability.
Example: Java RMI (Remote Method Invocation) is middleware used in Java applications. It allows Java objects to invoke methods on objects running on remote machines, enabling distributed applications to communicate seamlessly. Middleware like RMI abstracts the complexities of distributed communication, making it easier for developers to build distributed systems.

5.5.4 N-tier Systems:

N-tier distributed operating systems are designed based on the concept of dividing an application into multiple tiers or layers, each with specific responsibilities. For example, a three-tier system typically consists of a presentation tier, a business logic tier, and a data storage tier. These tiers can be deployed on separate machines, which offers benefits such as scalability, fault tolerance, and improved performance. N-tier architectures are commonly used in enterprise applications and web development, where a clear separation of concerns and functionalities is crucial.

Example: An e-commerce website is often built using an N-tier architecture. The presentation tier handles the user interface and displays products to users. The business logic tier manages processes like order processing and inventory management. The data storage tier stores product data, user profiles, and orders. By separating these functions into different tiers, the system becomes more scalable and fault-tolerant.

5.5.5 Three-tier Systems:

A three-tier distributed operating system is a specific form of N-tier architecture with three distinct layers. The presentation tier serves as the user interface, responsible for displaying information to users. The application tier manages the business logic, including processing and managing data. The data storage tier handles data storage and retrieval operations. These systems are often used in applications that require a clear separation of user interaction, data processing, and data storage, such as web applications.
Example: A web-based email system follows a three-tier architecture. The presentation tier is the webmail interface that users interact with. The application tier handles email processing, such as sending, receiving, and filtering emails. The data storage tier stores email data and attachments. This separation of concerns ensures efficient email management and provides a clear boundary between user interaction and data storage.
In summary, different distributed operating systems have distinct architectural approaches tailored to specific use cases. These approaches vary in terms of centralization, resource sharing, and separation of concerns, depending on the nature of the applications and their requirements.

5.6 Robustness and Design Issues of Distributed OS

5.6.1 Robustness

In a distributed system, hardware failures are a common concern, including link failures, site failures, and message losses. Detecting and responding to these failures is essential for maintaining system robustness.

Failure Detection: In an environment with no shared memory, distinguishing between link failure, site failure, and message loss can be challenging. Typically, we can only detect that some form of failure has occurred. Heartbeat procedures are often used to detect link and site failures. Sites periodically exchange "I-am-up" messages, and if a message is not received within a specified time, the system assumes a failure has occurred. Depending on the situation, actions like sending "Are-you-up?" messages and timeouts help determine the type of failure.

Reconfiguration: When a failure is detected, the system must reconfigure to continue normal operation. For example, if a direct link between two sites fails, this information is broadcast to update routing tables. If a site is believed to have failed, all sites must be notified so they no longer use the services of the failed site. This may involve electing new coordinators or reconstructing logical rings.

Recovery from Failure: When a failed link or site is repaired, it should be smoothly integrated back into the system. Continuous heartbeat procedures help notify all relevant parties. If a site recovers, it must

inform other sites and update its local tables with necessary information.

Fault Tolerance: A distributed system should tolerate various types of failures and continue functioning, albeit possibly in a degraded form. This can include communication faults, machine failures, storage device crashes, and storage media degradation. Achieving fault tolerance may require redundant communication paths, storage components, and monitoring software. Specific applications can improve fault tolerance by implementing lock management, allowing data access on shared disks concurrently and seamless recovery from failures.

In summary, maintaining a distributed system's robustness involves detecting, responding to, and recovering from hardware failures while ensuring fault tolerance through redundancy and monitoring mechanisms.

5.6.2 Design

Designing distributed systems comes with several key issues and challenges. These issues include achieving transparency, enabling user mobility, addressing scalability, and ensuring fault tolerance.

Transparency: A transparent distributed system aims to make the presence of multiple processors and storage devices invisible to users. Users should access resources, whether local or remote, in a consistent manner. The system is responsible for locating and facilitating the interaction with resources. User mobility is also a part of transparency, allowing users to log into any machine in the system without restrictions.

Scalability: Scalability is the system's ability to handle increased service loads. A scalable system responds gracefully to higher demand by degrading performance moderately and reaching a saturated state later. Designing for scalability is crucial, especially in distributed systems, where growth and integration of new resources are common. A scalable design should accommodate growth without causing issues like network congestion or expensive design modifications.

Fault Tolerance: Fault tolerance is closely related to scalability. Heavily loaded components can behave like faulty ones, and shifting the load to backups can also saturate the system. Spare resources are essential for ensuring reliability and handling peak loads gracefully. A distributed system inherently offers advantages in fault tolerance and scalability, but these benefits must be realized through proper design, emphasizing the distribution of control and data.

Example: Hadoop, an open-source framework, was designed to address the challenges of scalability and fault tolerance in big data projects. Hadoop, based on Google's MapReduce and Google File System projects, enables the processing of large data sets in distributed computing environments. It allows thousands of systems to work together, managing petabytes of data efficiently, making it ideal for big data tasks like mining social media for company-related information and analyzing financial data for stock pricing trends.

5.7 Self-Assessment Questions

Q1. What types of processors can be part of a distributed system, and what are the advantages of having such diversity? [4 marks, L2]

Q2. Describe the key differences between a distributed file system and a centralized file system. [4 marks[L2]

Q3. In a distributed system, what is the perspective of a specific processor regarding resources located in different parts of the network? [4 marks, L2]

Q4. Summarize the key characteristics of a distributed system. [4 marks, L2]

Q5. Explain the concept of reliability in distributed systems and how redundancy can help maintain system operations. [4 marks, L2]

Q6. How does automated load sharing contribute to improved performance in a distributed system? [4 marks, L2]

Q7. In a distributed operating system, how can users access remote resources compared to local ones? [4 marks, L2]

Q8. What is process migration in a distributed operating system, and what are the motivations for using it? [4 marks, L2]

Q9. How does the World Wide Web exhibit aspects of a distributed computing environment, and what types of migrations can be observed in its operations? [6 marks, L3]

Q10. Explain the different types of distributed OS. [10 marks, L2]

Q11. What is the primary role of the presentation tier in a three-tier system? [2 marks, L2]

Q12. Explain the design issues in distributed OS. [8 marks, L2]

5.8 Self-Assessment Activities

A1. You are tasked with setting up a distributed system for a research project. The system involves multiple computers with different operating systems located in different parts of the world. Each computer needs to share data and perform computations collaboratively. How would you design and implement this distributed system to ensure effective communication and coordination among the remote computers?

A2. You are developing a high-traffic e-commerce website that needs to handle a large number of concurrent user requests. How would you implement load balancing in a distributed system to ensure that the web servers efficiently distribute the incoming traffic while maintaining high availability and reliability?

A3. You are developing a mobile application that allows users to access and interact with a distributed system from their smartphones. What challenges would you encounter in ensuring a seamless user experience, and how would you address issues related to network connectivity, latency, and limited resources on mobile devices?

A4. How would you design and implement a distributed file system that ensures efficient resource sharing, fault tolerance, and robust data access? What key considerations and technologies would you incorporate into your design?

A5. Your organization runs a global chain of retail stores, and you need to manage a distributed database system to track inventory, sales, and customer information. How would you design the

distributed database to ensure data consistency, security, and efficient access while also handling potential network failures and data synchronization issues?

5.9 Multiple-Choice Questions

Q1. What is a fundamental characteristic of distributed systems? [1 mark, L1]

A. Shared memory and synchronized clocks
B. Independent processors without shared memory or a clock
C. Centralized processors with synchronized clocks
D. Independent processors with synchronized clocks

Q2. What are the advantages of a distributed system? [1 mark, L1]

A. Reduced computation speed and data availability
B. Centralized data storage
C. Enhanced computation speed and improved data availability
D. Simplified communication and synchronization

Q3. What is the key distinction between a distributed file system and a centralized file system? [1 mark, L1]

A. A distributed file system relies on multiple independent storage devices, while a centralized system has a single data repository.
B. A centralized file system relies on multiple independent storage devices, while a distributed system has a single data repository.
C. Both use shared memory for data storage.
D. Centralized file systems are faster than distributed file systems.

Q4. What are the key features of a distributed system, as described in the content? [1 mark, L1]

A. Shared physical memory and synchronized clocks
B. Geographical proximity of processors
C. Independent processors, dedicated local memory, and diverse hardware architectures
D. Centralized resource management and clock synchronization

Q5. What types of processors can be part of a distributed system? [1 mark, L1]

A. Only personal computers and large mainframe computers
B. Only small handheld devices.
C. A wide range, including small handheld devices, personal computers, workstations, and large mainframe computers.
D. Only workstations

Q6. What is automated load sharing in a distributed system? [1 mark, L1]

A. The process of automating resource allocation for increased security.
B. The automatic backup of data and hardware for redundancy.
C. Shifting tasks from overloaded sites to less loaded ones for improved performance.
D. Automated communication between distributed sites for reliability.

Q7. In what circumstances can redundancy in both hardware and data be particularly valuable in a distributed system? [1 mark, L1]

A. Redundancy is never valuable in a distributed system.
B. Redundancy is useful only for reducing costs.
C. Redundancy can help maintain system operations even if some sites fail.

D. Redundancy is mainly for improving performance

Q8. What does true distributed processing in a distributed system involve? [1 mark, L1]
A. Individual computers working independently on separate tasks.
B. Separate computers working together to achieve a common objective.
C. A single computer handling all processing tasks.
D. The centralization of all processing tasks

Q9. When is it more efficient to transfer the computation itself rather than the data in a distributed operating system?[1 mark, L1]
A. When the data transfer time is shorter than executing the remote command.
B. When the task requires processing data from various sites
C. When the computation is too complex for the remote site.
D. When data migration is preferred.

Q10. How does the World Wide Web exhibit aspects of a distributed computing environment? [1 mark, L1]
A. By centralizing all data and processing.
B. By isolating data at remote sites.
C. By enabling data, computation, and process migration.
D. By avoiding all forms of migration.

Q11. How is an application divided in an N-tier distributed operating system? [1 mark, L1]
A. Into separate copies for redundancy.
B. Into a single tier for simplicity
C. Into multiple tiers or layers, each with specific responsibilities.
D. It is not divided in N-tier systems.

Q12. What role does middleware play in a distributed operating system? [1 mark, L1]
A. It provides a direct user interface.
B. It serves as the central server in the system.
C. It facilitates communication between different applications operating on separate machines.
D. It handles data storage and retrieval operations.

Q13. What is fault tolerance in a distributed system?
A. The ability to achieve high performance under heavy loads
B. The system's resistance to scalability
C. The system's ability to continue functioning in the presence of failures
D. The system's ability to maintain a single point of failure

Q14. In a distributed system, what action is taken when a direct link between two sites fails?
A. The system continues normal operation without changes
B. The failed link is automatically repaired.
C. Information about the failure is broadcast to update routing tables
D. All other sites are shut down.

Q15. What method is often used to detect link and site failures in a distributed system?
A. Shared memory
B. Heartbeat procedures.
C. Message redundancy
D. Centralized monitoring.

5.10 Keys to Multiple-Choice Questions

Q1. Independent processors without shared memory or a clock (B)
Q2. Enhanced computation speed and improved data availability (C)
Q3. A distributed file system relies on multiple independent storage devices, while a centralized system has a single data repository. (A)
Q4. Independent processors, dedicated local memory, and diverse hardware architectures (C)
Q5. A wide range, including small handheld devices, personal computers, workstations, and large mainframe computers. (C)
Q6. Shifting tasks from overloaded sites to less loaded ones for improved performance. (C)
Q7. Redundancy can help maintain system operations even if some sites fail. (C)
Q8. Separate computers working together to achieve a common objective. (B)
Q9. When the task requires processing data from various sites. (B)
Q10. By enabling data, computation, and process migration. (C)
Q11. Into multiple tiers or layers, each with specific responsibilities. (C)
Q12. It facilitates communication between different applications operating on separate machines. (C)
Q13. The system's ability to continue functioning in the presence of failures (C)
Q14. Information about the failure is broadcast to update routing tables. (C)
Q15. Heartbeat procedures(B)

5.11 Summary of the Unit

A distributed system is a network of processors that operate independently without sharing memory or a clock. It consists of various types of processors, ranging from small handheld devices to large mainframe computers. These processors communicate through local and wide-area networks, enabling users to access system resources, enhance computation speed, and improve data availability and reliability. However, distributed systems introduce unique challenges, including process synchronization, communication, deadlock prevention, and handling failures. In a distributed system, resource sharing, computation acceleration, reliability, and communication are the primary motives for construction. Users can access resources across different sites, and share files, databases, and specialized hardware devices. Computation can be divided into subcomputations that run concurrently, and reliability is achieved through redundancy and spare resources. Communication allows users to exchange information over significant distances, facilitating remote collaboration.

Distributed operating systems are complex networks of interconnected computers within a virtual environment. Users interact with a unified system, but the architecture delegates task execution to multiple computers. These systems handle multifaceted operating system functions, manage distributed processing, and ensure a unified user experience. The evolution of distributed operating systems has been driven by advancements in technology, enhancing processing speed and power. Remote Procedure Calls (RPCs) play a significant role in distributed architecture, improving processing power. Architectural advancements like the master/slave concept optimize task execution, and modern network capabilities provide exceptional performance and security.

In summary, distributed systems and distributed operating systems provide powerful solutions for

resource sharing, computation acceleration, reliability, and communication. However, they require careful design to address the challenges of fault tolerance, scalability, and efficient resource management.

5.12 Keywords

Distributed system, Processors, Local memory, Communication, High-speed buses, Data migration, Computation migration, Process migration, Fault tolerance, Error recovery, Scalability

5.13 Recommended Learning Resources

[1] Silberschatz, A., Galvin, P., & Gagne, G. (2005). Operating System Concepts, 7th ed., Hoboken.

[2] William stalling. Operating Systems: Internal and design principles, 7th edition PHI

[3] D.M. Dhamdhere. Operating Systems: A concept-based Approach, 2nd Edition, TMH