**CS3551-DISTRIBUTED COMPUTING**

**PART B (13 Marks)**

**UNIT-1 INTRODUCTION**

**Q1: Define Distributed Computing and explain how it relates to Computer System Components.**

**Definition of Distributed Computing**

Distributed Computing is a model where multiple independent computers work together as a single system to execute tasks. These systems **communicate via a network** to achieve a common goal.

**Relation to Computer System Components**

Distributed computing involves:

1. **Processors** – Each node has a processor.
2. **Memory** – Distributed across different nodes.
3. **Network** – Facilitates communication.
4. **Software** – Manages communication and coordination.
5. **Storage** – Data is distributed across multiple machines.

**Q2: Compare Message-Passing Systems and Shared Memory Systems in Distributed Computing.**

| **Feature** | **Message-Passing System** | **Shared Memory System** |
| --- | --- | --- |
| **Communication** | Uses messages for interaction. | Uses shared memory to communicate. |
| **Synchronization** | Explicit message exchanges. | Implicit via memory access. |
| **Latency** | Higher due to network overhead. | Lower as memory access is faster. |
| **Scalability** | More scalable. | Less scalable due to memory contention. |
| **Example** | Computer clusters. | Multi-core processors. |

**Q3: Discuss the key challenges in designing Distributed Systems.**

1. **Concurrency** – Multiple processes executing simultaneously.
2. **Fault Tolerance** – The system must continue functioning despite failures.
3. **Scalability** – Must handle increased load efficiently.
4. **Security** – Ensuring safe communication over networks.
5. **Heterogeneity** – Different hardware/software combinations.
6. **Synchronization** – Maintaining a consistent state across nodes.
7. **Transparency** – Users should not notice distributed execution.

**Q4: Differentiate between Synchronous and Asynchronous Distributed Systems.**

| **Feature** | **Synchronous System** | **Asynchronous System** |
| --- | --- | --- |
| **Timing Assumptions** | Has known time limits for execution. | No assumption on execution time. |
| **Communication** | Messages arrive within a bounded time. | Messages can be delayed indefinitely. |
| **Clock Synchronization** | Easier to achieve. | Harder to synchronize clocks. |
| **Failure Handling** | Easier due to known time constraints. | Harder due to unpredictable delays. |
| **Example** | Real-time systems, Banking transactions. | Internet-based applications, Cloud computing. |

**Q5: Explain the concept of the Global State in a Distributed System.**

* **Global State** refers to the **combined state of all processes** and **message channels** at a given point in time.
* Difficult to determine in a distributed system due to **asynchrony** and **lack of a global clock**.
* Used for **checkpointing, debugging, and recovery mechanisms**.
* **Chandy-Lamport Algorithm** helps in capturing consistent global states.
* **Example:** A banking system needs to ensure all accounts are synchronized across different branches.

**UNIT-II LOGICAL TIME AND GLOBAL CLOCK**

**Q1: Explain the importance of Physical Clock Synchronization in Distributed Systems. How does NTP work?**

**Importance of Clock Synchronization**

* In distributed systems, each machine has its own **clock**, leading to **clock drift**.
* Clock synchronization ensures **consistent timestamps** for events like transactions and logging.

**Network Time Protocol (NTP)**

* **NTP synchronizes clocks** across a distributed network.
* Works in a **hierarchical structure** with **Stratum levels** (Stratum 1 is the most accurate).

**NTP Synchronization Process**

1. A client requests time from an **NTP server**.
2. The server responds with a **timestamp**.
3. The client calculates **clock offset** and **adjusts** its time accordingly.

**Example:**

* **Stock Trading Systems** use NTP to ensure accurate timestamps for transactions.

**Q2: Define Logical Clocks. Differentiate between Scalar Time and Vector Time.**

**Logical Clocks**

* Used when physical clocks are unreliable in distributed systems.
* Ensure **event ordering** instead of exact time.

**Scalar Time (Lamport Timestamps)**

* Uses a **single counter** that increments with each event.
* **Rules:**
  1. Each process maintains a **logical clock (LC)**.
  2. When sending a message, attach **LC value**.
  3. Upon receiving, update **LC = max(LC, received LC) + 1**.
* **Limitation:** Cannot detect **concurrent events**.

**Vector Time**

* Uses **a vector of timestamps** (one per process).
* Helps in detecting **causality** and **concurrent events**.

| **Feature** | **Scalar Time** | **Vector Time** |
| --- | --- | --- |
| **Number of Values** | Single counter | Vector of counters |
| **Detects Concurrency?** | No | Yes |
| **Used in** | Event ordering | Causal consistency |

**Q3: Explain Message Ordering Paradigms in Distributed Systems.**

**Types of Message Ordering**

1. **FIFO Ordering** → Messages from the same sender arrive **in the order sent**.
2. **Causal Ordering** → If **event A happened before B**, all processes see A **before** B.
3. **Total Ordering** → All processes see messages in the **same order**.

**Example:**

* **Bank Transactions** require **total ordering** to avoid discrepancies in account balances.

**Q4: What is Group Communication? Explain its importance.**

**Group Communication in Distributed Systems**

* A method for processes to communicate **as a group** instead of one-to-one.
* Ensures **reliable, ordered, and atomic message delivery**.

**Types of Group Communication**

1. **Unicast** → One sender to one receiver.
2. **Multicast** → One sender to multiple receivers.
3. **Broadcast** → Message sent to **all nodes**.

**Importance:**

* Used in **replication, fault tolerance, and distributed databases**.
* **Example:** In **Google Drive**, group communication keeps multiple copies of files synchronized.

**Q5: Explain the concept of Snapshot Algorithms for FIFO Channels.**

**Snapshot Algorithm**

* Captures a **global state** of a distributed system.
* Used for **failure recovery, debugging, and rollback**.

**Chandy-Lamport Algorithm (for FIFO Channels)**

1. A process initiates a **snapshot** by sending a marker.
2. Upon receiving a marker, a process:
   * Saves its **current state**.
   * Forwards the marker to others.
3. Records incoming messages between marker arrivals.

**Example:**

* **Distributed Banking Systems** use snapshot algorithms to **recover from failures**.

**UNIT-III DISTRIBUTED MUTEX AND DEADLOCKS**

**Q1: Explain the need for Distributed Mutual Exclusion in distributed systems.**

**Need for Mutual Exclusion in Distributed Systems**

* **Mutual exclusion** ensures that **only one process** can access a shared resource at a time.
* In distributed systems, there is **no global clock** or shared memory, making coordination difficult.

**Challenges:**

1. **Synchronization Delay** → Messages take time to propagate.
2. **Failure Handling** → If a node fails, reassigning control is complex.
3. **Fairness** → Ensuring every process gets fair access.

**Example:**

* **Distributed File Systems** (e.g., Google Drive) use mutual exclusion to **prevent data corruption** when multiple users edit a file.

**Q2: Describe Lamport’s Distributed Mutual Exclusion Algorithm.**

**Lamport’s Algorithm Steps:**

1. Each process maintains a **logical clock**.
2. A process sends a **request message (REQ)** with its timestamp.
3. Other processes respond with an **acknowledgment (ACK)** if they:
   * Are not using the resource.
   * Have a lower priority request.
4. The requesting process enters the **critical section** when:
   * It receives ACK from all other processes.
5. After exiting, it sends a **release message (REL)**.

**Example:**

* Used in **banking transactions** to ensure **sequential processing of operations**.

**Q3: What are Token-Based Algorithms in Distributed Mutual Exclusion?**

**Token-Based Algorithms**

* A unique **token** circulates in the system.
* The process holding the token **can enter the critical section (CS)**.

**Advantages:**

✔ No need for **continuous message exchanges**.  
✔ Guarantees **fair access** to processes.

**Disadvantages:**

❌ **Token Loss** → System failure if the token is lost.  
❌ **Single Point of Failure** → If the token holder crashes, a new token must be generated.

**Example:**

* Used in **Ethernet Token Ring networks**.

**Q4: Explain Suzuki-Kasami’s Broadcast Algorithm.**

**Suzuki-Kasami’s Algorithm Steps:**

1. A process that wants access **broadcasts a request message**.
2. The process holding the **token** replies.
3. The token contains a **queue** of waiting processes.
4. When the process finishes, it **passes the token** to the next requester.

**Advantages:**

✔ **Less overhead** compared to permission-based approaches.  
✔ **Efficient for large systems**.

**Example:**

* Used in **cloud storage services** to manage concurrent file accesses.

**Q5: What are the different Models of Deadlocks in Distributed Systems?**

**Deadlock Models:**

1. **AND Model** → A process waits for **all** requested resources before proceeding.
2. **OR Model** → A process proceeds if **at least one** of the requested resources is available.

| **Feature** | **AND Model** | **OR Model** |
| --- | --- | --- |
| **Waiting Condition** | Process waits for all resources. | Process waits for any available resource. |
| **Detection Complexity** | Higher | Lower |
| **Example** | Transaction Processing | Cloud Computing |

**UNIT-IV CONSENSUS AND RECOVERY**

**Q1: What is Consensus in Distributed Systems? Explain its importance.**

**Definition:**

* Consensus is the process where **all nodes agree** on a single value despite failures.
* It is **critical for fault tolerance** in distributed computing.

**Importance of Consensus:**

1. **Coordination** → Ensures nodes operate consistently.
2. **Fault Tolerance** → Handles node failures gracefully.
3. **Security** → Prevents malicious attacks (e.g., blockchain consensus).
4. **Data Consistency** → Avoids conflicts in replicated databases.

**Example:**

* **Blockchain Networks** (Bitcoin) use **Proof of Work (PoW)** for consensus.

**Q2: Explain Agreement in Failure-Free Synchronous and Asynchronous Systems.**

**Failure-Free Synchronous Agreement:**

* All nodes communicate in **fixed time intervals**.
* Messages arrive **in a known order**.
* Agreement is **easier** since time is predictable.

**Failure-Free Asynchronous Agreement:**

* No fixed time constraints.
* Nodes may receive messages at **different times**.
* **Fischer, Lynch, and Paterson (FLP) Theorem** proves consensus is **impossible** in pure asynchronous systems.

**Example:**

* **Synchronous** → Military control systems.
* **Asynchronous** → Cloud storage synchronization.

**Q3: What are the Issues in Failure Recovery in Distributed Systems?**

**Key Issues in Failure Recovery:**

1. **Checkpointing Overhead** → Saving system state increases resource usage.
2. **Rollback Propagation** → Recovery in one process may cause others to rollback.
3. **Message Loss** → Unreliable network communication can lead to missing data.
4. **Consistency Issues** → Ensuring consistency after recovery is challenging.

**Example:**

* **Online banking** uses **checkpointing** to recover transaction history after failure.

**Q4: Describe the Coordinated Checkpointing Algorithm.**

**Steps in Coordinated Checkpointing:**

1. A **coordinator** process initiates checkpointing.
2. All processes **pause and save** their state.
3. The checkpoint is stored **atomically** across all nodes.
4. If a failure occurs, the system **rolls back to the latest checkpoint**.

**Advantages:**

✔ Prevents **orphan messages**.  
✔ Ensures **global consistency**.

**Disadvantages:**

❌ Causes **synchronization delays**.  
❌ Requires **more storage**.

**Example:**

* Used in **distributed databases** (e.g., MySQL replication).

**Q5: Explain Asynchronous Checkpointing and Recovery Algorithm.**

**Asynchronous Checkpointing Steps:**

1. Each process **independently** saves its state.
2. No global coordination is required.
3. If failure occurs, a process **rolls back to its own checkpoint**.

**Advantages:**

✔ **Lower overhead** compared to coordinated checkpointing.  
✔ **Better performance** in large-scale systems.

**Disadvantages:**

❌ Can lead to **rollback propagation**.  
❌ Requires **dependency tracking** between checkpoints.

**Example:**

* Used in **cloud-based applications** for auto-recovery.

**UNIT-V CLOUD COMPUTING**

**Q1: Define Cloud Computing and Explain its Characteristics.**

**Definition:**

Cloud computing is a technology that enables **on-demand access** to computing resources over the internet, without direct user management of infrastructure.

**Characteristics of Cloud Computing:**

1. **On-Demand Self-Service** → Users can provision resources automatically.
2. **Broad Network Access** → Access from anywhere via the internet.
3. **Resource Pooling** → Shared resources serve multiple customers.
4. **Rapid Elasticity** → Resources scale up/down as needed.
5. **Measured Service** → Usage is monitored and billed accordingly.

**Example:**

* Google Drive allows users to store and access data anytime.

**Q2: What are the Different Cloud Deployment Models?**

Cloud computing deployment models define how cloud services are hosted and accessed.

**Types of Cloud Deployment Models:**

1. **Public Cloud**
   * Hosted by third-party providers (e.g., AWS, Google Cloud).
   * Available to multiple users over the internet.
   * **Example:** Gmail, Dropbox.
2. **Private Cloud**
   * Used exclusively by a single organization.
   * Higher security and customization.
   * **Example:** Banking institutions using private data centers.
3. **Hybrid Cloud**
   * Combination of public and private clouds.
   * Provides flexibility and optimized cost.
   * **Example:** E-commerce platforms (private for payments, public for product display).
4. **Community Cloud**
   * Shared among organizations with similar needs.
   * **Example:** Healthcare sector data sharing.

**Q3: Explain Cloud Service Models with Examples.**

Cloud services are categorized into three major models:

1. **Infrastructure as a Service (IaaS)**
   * Provides virtualized computing resources.
   * Users manage OS, applications, and networking.
   * **Example:** Amazon EC2, Microsoft Azure Virtual Machines.
2. **Platform as a Service (PaaS)**
   * Provides a platform to develop, test, and deploy applications.
   * Developers focus on coding without managing infrastructure.
   * **Example:** Google App Engine, AWS Lambda.
3. **Software as a Service (SaaS)**
   * Ready-to-use applications hosted on the cloud.
   * No need for installation or maintenance.
   * **Example:** Google Docs, Microsoft 365.

**Q4: What are the Challenges of Cloud Computing?**

1. **Security and Privacy** → Sensitive data is stored on external servers.
2. **Downtime and Reliability** → Cloud providers may experience failures.
3. **Latency Issues** → Delay in accessing cloud-hosted applications.
4. **Compliance Requirements** → Industries need to follow data regulations.
5. **Vendor Lock-in** → Difficult to migrate data between cloud providers.

**Example:**

* **AWS outage (2021)** affected businesses using Amazon’s cloud services.

**Q5: Explain Virtualization and Its Role in Cloud Computing.**

**Definition:**

Virtualization is the process of creating **virtual instances** of hardware, storage, or network resources.

**Types of Virtualization:**

1. **Server Virtualization** → Multiple virtual servers run on a single physical server.
2. **Storage Virtualization** → Multiple storage devices appear as a single unit.
3. **Network Virtualization** → Software-defined networks (SDN) manage networking.

**Role in Cloud Computing:**

✔ **Optimizes resource utilization**.  
✔ **Enables scalability and cost savings**.  
✔ **Supports multi-tenancy** (multiple users on the same hardware).

**Example:**

* **VMware ESXi** for virtualizing servers in data centers.