Operating System – Week 10

1. Protocol Comparison Table

Feature	Centralized Algorithm	Distributed (Ricart-Agrawala)	Token-Ring Algorithm
Messages Required	3 messages per critical section entry (request, grant, release)	2 × (N − 1) messages per entry	1 message (token passing)
Drawbacks	Single point of failure; can cause delays if coordinator crashes	High message overhead; clock synchronization needed	Token loss or duplication can cause issues
Failure Scenarios	Coordinator failure stops all requests	Node failure requires reconfiguration	Lost token must be regenerated
Real-World Use Case	Centralized database lock manager	Distributed file system coordination	Token-based LANs (e.g., Token Ring networks)

Each algorithm manages mutual exclusion differently — centralized is simple but risky, distributed is fair but chatty, and token-ring is efficient but needs token control.

2. Raft Algorithm

The Raft algorithm is a modern consensus algorithm used to manage a replicated log across a distributed system. It ensures all nodes agree on shared data, even when some nodes fail. Raft is designed to be easier to understand than Paxos while achieving the same reliability.

Raft works through a leader-based approach. In a cluster, one node becomes the leader through an election process. The leader handles all client requests that change data and replicates the updates to the follower nodes. This keeps the system consistent and organized because only one node makes decisions at a time.

When a new log entry is created, the leader appends it to its log and sends AppendEntries messages to followers. Once the majority of nodes confirm the entry, it is considered committed and applied to all servers' states. If the leader fails, a new election starts automatically, and a follower with the most recent log becomes the new leader.

Raft divides the process into three clear parts: leader election, log replication, and safety. These make the algorithm easier to read and explain compared to Paxos, which has a complex structure and overlapping roles.

Because Raft organizes the system around a single leader and clear phases, it is widely adopted in systems like etcd, Consul, and RethinkDB. It offers strong consistency, automatic recovery from node failure, and is much simpler to reason about, which is why developers prefer it over Paxos.

3. etcd and the Raft Algorithm

etcd is a distributed key-value store used for configuration management and service coordination. It is a core component of Kubernetes, providing reliable storage for cluster state. etcd achieves high reliability using the Raft consensus algorithm, which ensures all cluster nodes agree on the same data even in case of failures.

In etcd, one node is always the leader, while the others act as followers. The leader handles all write operations, ensuring consistency. When a client sends a change (like updating a configuration), the leader adds the request to its log and sends AppendEntries messages to the followers. When the majority of followers acknowledge the entry, it becomes committed. This means all nodes apply it to their local data, keeping the system synchronized.

If the leader fails or becomes unreachable, the followers automatically start a new election. A follower with the most up-to-date log becomes the next leader, ensuring minimal downtime. This election process allows etcd to continue functioning smoothly even during node failures, maintaining high availability.

etcd also uses Raft to support distributed locks, leader election, and watchers. Distributed locks ensure that only one client can access a resource at a time across the entire cluster. Leader election is used by higher-level systems (like Kubernetes controllers) to select a single master node for specific tasks. Watchers let clients receive updates automatically whenever data changes, avoiding the need for constant polling.

The use of Raft gives etcd strong consistency and fault tolerance. Every operation is safely replicated across nodes, ensuring data is not lost even if some servers go offline. Its simplicity also allows developers to maintain and debug it easily.

In summary, etcd’s use of the Raft algorithm allows it to provide reliable coordination services, making it a trusted backbone for distributed systems like Kubernetes, Docker Swarm, and Cloud Foundry.