Distributed Systems Roadmap

---

1. Prerequisites & Foundations

1. Understand Computer Networks
   a. TCP vs UDP, IP addressing, DNS, NAT
   b. Sockets, ports, firewalls
2. Operating Systems Fundamentals
   a. Threads vs Processes
   b. Scheduling, memory management
   c. File systems, I/O
3. Algorithms & Data Structures
   a. Graphs, Trees, Queues
   b. Sorting/searching, hashing, bloom filters

2. Core Concepts of Distributed Systems

1. What is a Distributed System?
   a. Definition: Multiple independent computers working as one system
   b. Types: Client-server, peer-to-peer, master-slave, decentralized
2. Characteristics & Goals
   a. Scalability, Fault Tolerance, Consistency, Availability, Partition Tolerance

3. Communication in Distributed Systems

1. RPC (Remote Procedure Call)
2. REST vs gRPC vs Thrift
3. Message Queues: Kafka, RabbitMQ, ZeroMQ
4. Protocol Buffers, Avro (data serialization)

4. Time & Order

1. Logical Clocks (Lamport Clocks)
2. Vector Clocks
3. NTP, Clock drift, Eventual consistency

5. Data & Storage

1. Consistency Models
    a. Strong, Weak, Eventual
    b. Causal, Read-your-writes
2. Distributed File Systems
    a. HDFS, GFS
3. CAP Theorem Deep Dive
    a. Trade-offs between Consistency, Availability, Partition tolerance

6. Consensus Algorithms

1. Why consensus is hard
2. Paxos (basic idea + challenges)
3. Raft (leader election, log replication)
4. ZAB (used by ZooKeeper)

7. Fault Tolerance & Reliability

1. Replication: Synchronous vs Asynchronous
2. Leader election
3. Quorum-based systems
4. Retry, Backoff, Circuit Breakers

8. Scalability Techniques

1. Load Balancing: DNS-based, L4 vs L7, HAProxy, Envoy
2. Caching: CDN, Memcached, Redis, cache invalidation
3. Sharding and Partitioning
4. Database scaling: read-replicas, CQRS, eventual consistency

9. Real-World Distributed Systems

1. Distributed Databases: Cassandra, MongoDB, CockroachDB
2. Messaging Systems: Kafka, Pulsar, NATS
3. Coordination Systems: ZooKeeper, etcd, Consul
4. Configuration & Service Discovery: Eureka, Consul

10. Observability & Monitoring

1. Logging: Structured vs unstructured
2. Tracing: OpenTelemetry, Jaeger
3. Metrics: Prometheus, Grafana

4. Distributed tracing and correlation IDs

11. Design Patterns & Case Studies

1. Idempotency, Retry, Sagas
2. Leader election pattern
3. Bulkhead, circuit breaker, rate limiting
4. Case studies: Google Spanner, Amazon Dynamo, Kafka, Uber’s Ringpop

12. Hands-On Practice

1. Build a simple chat system or Pub/Sub service
2. Implement your own Raft (educational)
3. Create a toy key-value store (with sharding + replication)
4. Design a fault-tolerant job queue

Recommended Resources

- "*Designing Data-Intensive Applications*" – Martin Kleppmann
- "*Distributed Systems*" – Maarten van Steen & Andrew Tanenbaum
- 🛠️ MIT 6.824 (Free course)