Unit 4: Software Availability (4 Hours)

What is Software Availability?

Software Availability refers to the **degree to which a software system or service is operational and accessible when required for use.** It's a core measure of a system's **dependability** and **quality of service (QoS).**

It answers the question:

"**Is the system available when the user needs it?**"

It's typically expressed as a **percentage** over a given time period:

$$\textbf{Availability (A)} = \textbf{(Uptime / (Uptime + Downtime))} \times \textbf{100}$$

Importance of Availability in System Dependability

1. User Trust and Experience

- Systems that are unavailable frustrate users and erode trust.
- Even highly reliable systems can be **useless** if they are **not available** when required.

Example: A reliable online banking app that's down during payday is a failure in terms of user experience.

2. Business Continuity and Revenue

- Downtime directly impacts revenue, productivity, and customer satisfaction.
- Critical business operations rely on systems being accessible.

Example: Amazon can lose **millions per minute** of downtime during peak sales.

3. Foundation for Fault Tolerance

- Availability includes recovery time; hence it drives investment in:
  - **Redundancy**
  - **Failover systems**
  - **Self-healing infrastructure**

These technologies are vital for building **fault-tolerant** dependable systems.

4. Time-Critical Operations

- In real-time systems (e.g., air traffic control, healthcare), availability is essential to prevent disasters.

Example: An unavailable emergency dispatch system could cost lives.

5. Quantifiable SLA Guarantees

- Availability can be **measured** and included in **Service Level Agreements (SLAs)**.
- SLAs guarantee performance and **bind service providers** to certain uptime commitments (e.g., 99.99%).

6. Links with Maintainability

- A system might **fail often**, but if it's **restored quickly**, availability remains high.
- Availability highlights the **system's recoverability** and **repair efficiency**.

7. Dependency on Other Dependability Attributes

- High availability usually implies:
  - Good **reliability** (few failures)
  - Good **maintainability** (fast recovery)
  - Adequate **security** (protected from attacks)

Thus, it integrates and **reflects the effectiveness** of other dependability properties.

8. Availability Impacts Redundancy Design

- To meet availability targets (e.g., 99.999%), systems must use:
  - **Hot or cold standby systems**
  - **Clustering(group acting as a single system)**
  - **Load balancing**
  - **Distributed architectures**

This directly affects **system architecture decisions** and **costs**.

Availability Metrics:

Availability metrics provide **quantitative ways** to measure and analyze how often a software system or service is available for use. These metrics are essential for:

- Performance evaluation
- SLA (Service Level Agreement) compliance
- System reliability and design optimization
- Business continuity planning

These metrics help answer questions like:

- *How often does the system fail?*
- *How long does it take to recover?*

- *What is the impact of downtime on users and operations?*

1. Uptime

Definition:

Uptime is the total amount of time a system remains operational without any interruption.

Measured in:

- Hours, minutes, or seconds
- Typically over a day, week, month, or year

Example:

If a system runs for 23 hours without failure in a 24-hour window, uptime = 23 hours.

Downtime

Definition:

Downtime is the period when a system is **not available** due to failures, crashes, maintenance, or other issues.

Causes of Downtime:

- Software bugs
- Server crashes
- Network outages
- Cyberattacks
- Planned maintenance

Types:

- **Planned Downtime**: For upgrades, backups, patches
- **Unplanned Downtime**: Failures, crashes, security breaches

3. Availability Ratio

$$\textbf{Availability (A) = (Uptime / (Uptime + Downtime)) × 100}$$

Example:

If uptime = 720 hours, downtime = 2 hours:

A= 99.72%

Along with these three above metrics, the software availability also use most of the metrics like MTTF, MTTR, MTBF and Availability. Please refer the previously done numerical on those topics.

Mathematical Models of Availability

A **mathematical model of availability** is a formal way of quantifying how **often** and **for how long** a system is operational, using **probabilistic, statistical, and stochastic techniques**. These models help predict and optimize **uptime**, **recovery**, and **failure response** in software systems.

They answer:

- *What percentage of time is the system available?*
- *How often does it fail, and how quickly does it recover?*
- *What is the steady-state (long-term) behavior of system availability?*

1. Basic Availability Model

This is the **most commonly used model**, especially for systems with **constant failure and repair rates**.

Formula:
Availability (A) =MTBF/MTBF+MTTR

Where:

- **MTBF (Mean Time between Failures)**: Average operational time before a failure occurs.
- **MTTR (Mean Time to Repair)**: Average time taken to fix a failure.

2. Markov Model (Two-State Continuous-Time)

Markov models are used when systems switch **randomly** between **operational and failed states**, and the transition rates follow **exponential distributions**.

System States:

- **State 0 (Up)**: System is operational.
- **State 1 (Down)**: System is failed.

Transition Rates:

- λ: Failure rate (transitions from Up → Down)
- μ: Repair rate (transitions from Down → Up)

Availability Equation: A = μ/λ+ μ

This is **mathematically equivalent** to the basic MTBF/MTTR formula if:

- MTBF = 1/λ
- MTTR = 1/μ

3. Multi-State Markov Models

Used for **complex software systems** with:

- **Multiple degradation levels**

- **Sub-component failures**
- **Redundant systems**

Example States:

State	Description
S0	Fully operational
S1	Degraded (partial failure)
S2	Failed completely

Transitions between states have rates:

- λ_1, λ_2 for failures
- μ_1, μ_2 for repairs

To calculate availability:

1. Write all **balance equations**
2. Solve for **steady-state probabilities**
3. Sum the probabilities of all **"operational" states**

These models require solving **systems of linear equations** and are often computed using tools like **SHARPE**, **MATLAB**, or **PRISM**.

4. Instantaneous and Steady-State Availability

Type	Definition
Instantaneous A(t)	Availability at an exact moment t. Depends on time-dependent behavior.
Steady-State A(∞)	Long-run average availability. Assumes many failure-repair cycles have occurred.

Most systems use **steady-state availability** for SLAs and design decisions.

5. Redundant System Models (Parallel Systems)

In **redundant software systems**, availability is modeled using combinations of component availabilities.

Remember the formulas for Series System and Parallel System

6. Downtime Estimation from Availability

Given availability A, total time T:

$$\text{Downtime} = T \times (1 - A)$$

Example:

For A=99.9%, over a year (8760 hours):

$$\text{Downtime} = 8760 \times (1 - 0.999) = 8.76 \text{ hours/year}$$

Conclusion:

 Mathematical models of availability help **design, evaluate, and optimize** software systems.

 Choosing the right model depends on:

- Complexity of system
- Type of failure/recovery
- Need for time-specific or long-run estimates

 These models form the basis for achieving **highly available**, **dependable**, and **cost-effective** software solutions.