📘 Distributed Systems

Slides 36–44 — Complete Academic Summary

🔹 Slide 36 — Enterprise Application Integration (EAI)

What is EAI?

Enterprise Application Integration (EAI) is the process of enabling **different enterprise applications** (often developed independently) to **work together and share data and processes** in a coordinated way.

Transactions in EAI

A **transaction** is a sequence of operations that must be executed **as a single logical unit**.

Transaction Primitives

Primitive	Description
BEGIN_TRANSACTION	Marks the start of a transaction
END_TRANSACTION	Terminates the transaction and attempts to commit
ABORT_TRANSACTION	Cancels the transaction and restores old values
READ	Reads data from a file, table, or resource
WRITE	Writes data to a file, table, or resource

The Core Issue: *All-or-Nothing*

- Either **all operations succeed**, or **none of them take effect**
- Partial execution is **not allowed**

ACID Properties (Very Important for Exams ⭐)

1️⃣ Atomicity

- The transaction happens **indivisibly**

- Either fully executed or fully aborted

2️⃣ **Consistency**

- The transaction does **not violate system rules or constraints**

3️⃣ **Isolation**

- Concurrent transactions do **not interfere** with each other

4️⃣ **Durability**

- Once committed, changes are **permanent**, even after failures

📌 **Exam tip:**

ACID guarantees correctness and reliability in distributed enterprise systems.

🔹 Slide 37 — TPM: Transaction Processing Monitor

Observation

In many enterprise systems:

- Data involved in a transaction is **distributed across multiple servers**
- Coordinating these servers is **complex**

What is a TPM?

A **Transaction Processing Monitor (TPM)** is middleware responsible for:

- Coordinating the execution of a transaction
- Managing communication between clients and servers
- Ensuring ACID properties across distributed resources

How TPM Works (Conceptually)

1. Client sends a **transaction request**
2. TPM:
   - Breaks it into sub-requests
   - Sends them to different servers
3. Servers process requests and send replies
4. TPM:
   - Collects responses
   - Decides whether to **commit or abort**
5. Client receives the final result

📌 **Key exam idea:**

TPM hides the complexity of distributed transactions from applications.

🔹 Slide 38 — Middleware and EAI

Role of Middleware

Middleware provides **communication facilities** that allow distributed applications to interact and integrate.

It sits **between clients and servers**, abstracting networking and coordination details.

Middleware Communication Models

1️⃣ **Remote Procedure Call (RPC)**

- A client calls a remote operation **as if it were local**
- The call is:
  - Packaged as a message
  - Sent to the server
  - Processed
  - Result returned as a normal function return

📌 Limitation:

- Caller and callee must be **active at the same time**

2️⃣ **Message-Oriented Middleware (MOM)**

- Messages are:
  - Sent to a **logical contact point** (published)
  - Forwarded to **subscribed applications**
- Communication is **asynchronous**

📌 Advantage:

- Sender and receiver are **decoupled in time and space**

Exam Comparison

RPC → synchronous, tightly coupled
MOM → asynchronous, loosely coupled

🔹 Slide 39 — How to Integrate Applications

1️⃣ File Transfer

- Technically simple
- Not flexible

Problems:

- Need agreement on file format and layout
- File management complexity
- Update propagation and notifications are difficult

2️⃣ Shared Database

- More flexible than file transfer
- Applications share the same data store

Issues:

- Requires a **common data schema**
- Risk of:
  - Bottlenecks
  - Tight coupling

3️⃣ Remote Procedure Call (RPC)

- Effective when a **sequence of actions** must be executed
- Still synchronous

4️⃣ Messaging

- Allows **decoupling**
- No need for both applications to be running simultaneously
- Better suited for large distributed systems

📌 **Exam tip:**

> Messaging is preferred in scalable, loosely coupled architectures.

🔹 Slide 40 — Distributed Pervasive Systems

Observation

A new generation of distributed systems where:

- Nodes are **small and mobile**
- Often **embedded** in larger systems
- Systems **blend naturally into the user's environment**

Users may not even be aware they are interacting with a distributed system.

Three Overlapping Subtypes

1️⃣ **Ubiquitous Computing Systems**

- Continuously present
- Continuous interaction between user and system

2️⃣ **Mobile Computing Systems**

- Devices are inherently **mobile**
- Location changes are essential

3️⃣ **Sensor (and Actuator) Networks**

- Emphasis on **collaborative sensing**
- May also perform actions (actuation) on the environment

🔹 Slide 41 — Ubiquitous Systems

Core Elements of Ubiquitous Computing

1️⃣ **Distribution**

- Devices are networked, distributed, and transparently accessible

2️⃣ **Interaction**

- User interaction is **highly unobtrusive**
- Technology "fades into the background"

3️⃣ **Context Awareness**

- System understands user context (location, activity, time)
- Uses it to optimize interaction

4️⃣ **Autonomy**

- Devices operate **without human intervention**
- Self-managed systems

5️⃣ **Intelligence**

- System handles complex, dynamic interactions
- Adapts to changing conditions

📌 **Key phrase to remember:**

Ubiquitous computing is invisible, adaptive, and context-aware.

🔹 Slide 42 — Mobile Computing

Distinctive Features

1️⃣ **Diverse Devices**

- Smartphones
- Tablets
- GPS devices
- Remote controls
- Active badges

2️⃣ **Mobility**

- Device location changes over time
- Leads to:
  - Changing local services
  - Dynamic reachability

Keyword: **Discovery**

3️⃣ **Unstable Communication**

- No fixed route
- No guaranteed connectivity
- Leads to **disruption-tolerant networking**

📌 **Exam note:**

Mobile computing systems must handle intermittent connectivity.

🔹 Slide 43 — Sensor Networks

Characteristics of Sensor Nodes

Sensor nodes are:

- **Many**
  - Typically tens to thousands

- **Simple**
  - Limited memory
  - Limited processing power
  - Limited communication capability
- **Energy-constrained**
  - Often battery-powered
  - Sometimes battery-less

📌 **Key challenge:**

Energy efficiency is the dominant design concern in sensor networks.

🔹 Slide 44 — Sensor Networks as Distributed Databases

Two Extreme Approaches

1️⃣ Centralized Approach

- Sensors send **all data directly** to the operator's site
- Operator stores and processes data

✔️ Simple
❌ High communication cost
❌ Energy inefficient

2️⃣ In-Network Processing Approach

- Each sensor:
  - Processes and stores data locally
- Operator sends **queries**
- Sensors return **only answers**, not raw data

✔️ Energy efficient
✔️ Scalable
✔️ Reduces communication

📌 **Exam takeaway:**

Treating sensor networks as distributed databases enables efficient querying and aggregation.