Lesson 1: Foundations and Advantages

To understand distributed systems, you first need to understand what they look like and why they are often better than having just one giant computer.

1. What is a Distributed System? At its most basic level, a distributed system is a collection of processors that are "loosely coupled" and interconnected by a communications network. Here are the key terms you need to know for your exam:

- **Processors:** These are the individual "brains" of the system. Depending on who you talk to, they might be called **nodes, computers, machines,** or **hosts**.
- **Site:** This refers to the physical location where a processor is sitting.
- **The Client-Server Relationship:** In these systems, we generally see a **server** that holds a specific resource (like a file or a printer) and a **client** node at a different site that wants to use that resource.

2. Why do we use Distributed Systems? Your lecture notes highlight several critical reasons why these systems are used instead of standalone computers:

- **Resource Sharing:** This is the most common reason. It allows users at different sites to share files, print documents on remote printers, process information in a shared distributed database, or even use specialized hardware devices that only exist at one location.
- **Computation Speedup:** If a specific task is too big for one computer, a distributed system allows for **load sharing**. This means you can migrate parts of a job to other processors in the network so they can work on them simultaneously, getting the result much faster.
- **Reliability:** This is a major advantage. If one site in a distributed system fails, the remaining sites can detect the failure and take over the necessary functions. The system can recover and eventually reintegrate the failed site once it is back online.
- **Communication:** Distributed systems facilitate "message passing." This means that all the high-level functions you would expect from a single computer can be expanded so that people and processes can talk to each other across vast distances.
- **Flexibility and Maintenance:** By moving away from one massive, expensive system to multiple smaller systems, companies gain more flexibility. These smaller systems are often easier to maintain, have better user interfaces, and allow the computing power to be "downsized" into more efficient units.

Visualization of the Concept: Imagine three different locations: Site A, Site B, and Site C.

- **Site A** has a server.
- **Site B** has a client.
- **Site C** has a cluster of resources. All of these are connected by a "cloud" (the network). When the client at Site B needs something, it sends a message through the network to the server at Site A or the resources at Site C to get the job done.

Lesson 2: Operating System Types and Network Architectures

To build on our foundations, we must now examine how these systems are managed by software and the physical ways they are wired together. For your exam, you need to understand the fundamental split between two types of operating systems and the specific technical standards of modern networks.

1. Two Main Distributed Operating System Types

The user experience of a distributed system is defined by the operating system's design.

Here is a direct comparison between **Network Operating Systems (NOS)** and **Distributed Operating Systems (DOS)**.

Feature	Network Operating System (NOS)	Distributed Operating System (DOS)
User Awareness	Users are **fully aware** of the multiple different machines.	Users are **not aware** of the multiplicity of machines.
Resource Access	Done **explicitly** (you must tell the system exactly what to do).	**Seamless**; remote access feels exactly like local access.
User Experience	**More difficult**; requires a paradigm shift to manage network sessions.	**Transparent**; the system looks like one centralized unit.
Core Method	Users log in or move files **manually**.	The system **migrates** data, math, or processes automatically.

1- Network Operating Systems (The Manual Approach)

- **Explicit Tools:** You must use specific commands to reach other sites:
  - **Remote Logging:** Using `telnet` or `ssh` to sign into another machine.
  - **Remote Desktop:** Specifically used in Microsoft Windows.
  - **File Transfer:** Using the **File Transfer Protocol (FTP)** to move data.
- **The Paradigm Shift:** Users must establish a "session" and give network-based commands, which makes the system harder to use than a standard PC.

2- Distributed Operating Systems (The Transparent Approach)

Instead of the user doing the work, the OS uses **Migration Strategies** to move resources behind the scenes:

- **Data Migration:** The system transfers either the **entire file** or just the **portions** needed for the immediate task.
- **Computation Migration:** Instead of moving data, the system transfers the "math" to another site via **Remote Procedure Calls (RPCs)** or a **messaging system**.
- **Process Migration:** The system moves an entire running process to another site for:
  - **Load balancing:** Evening out the workload across the network.
  - **Computation speedup:** Running subprocesses concurrently on different sites.
  - **Preferences:** Moving a job to a site with specialized hardware or specific software licenses.
  - **Data access:** Running the process right where the data is stored instead of moving the data.
- **Key Example:** You should consider the **World Wide Web** as a major real-world example of these concepts.

2. Migration Strategies (How Work Moves)

In a Distributed Operating System, the system must decide how to move work to be most efficient.

- **Data Migration:** The system can move data to the user, either by transferring the **entire file** or just the specific **portions** needed for the current task.
- **Computation Migration:** Instead of moving the data, the system moves the "math" to the data. This is handled through **Remote Procedure Calls (RPCs)** or specialized messaging systems.
- **Process Migration:** This is the most complex form, where an entire running process is moved to a different site for several reasons:
  - **Load Balancing:** Spreading work out so no single machine is overwhelmed.
  - **Computation Speedup:** Breaking a job into subprocesses that run concurrently on different sites.

- **Hardware/Software Preferences:** Moving a task to a site that has a specialized processor (like a GPU) or a specific software license available only there.
- **Data Access:** Running the process at the remote site where the data lives to avoid moving massive files across the network.

3. Network Structure: LAN vs. WAN

The geographical distance between sites determines the network's technical design.

- **Local-Area Network (LAN):**
  - **Scope:** These cover small areas like a single office, building, or campus.
  - **Components:** A LAN isn't just computers; it includes everything from mainframes to mobile devices, as well as **peripherals** like printers and **storage arrays**, all connected by routers.
  - **Ethernet Standards:** This is the most common LAN structure. It uses a **multiaccess bus-based** system (defined by **IEEE 802.3**) with speeds ranging from 1 Mbps up to **40 Gbps** over twisted pair copper or optical fibre.
  - **Wireless (WiFi):** Modern LANs frequently use the **IEEE 802.11g** standard, which operates at 54 Mbps.
- **Wide-Area Network (WAN):**
  - **Scope:** These link sites that are geographically separated, even across the globe.
  - **Diversity:** The Internet is a WAN that allows hosts that **differ in all dimensions** (size, OS, hardware) to communicate seamlessly.
  - **Connections:** These use **point-to-point connections** over long-haul lines that are often leased from telephone companies.
  - **The Communication Subsystem:** The collection of routers (connection processors) that handle the actual movement of data packets in a WAN is formally known as the communication subsystem.
  - **Standard Link Speeds:** Memorize these three specific tiers for your exam:
    - **T1:** 1.544 Mbps.
    - **T3:** 45 Mbps (which equals 28 T1 links).
    - **OC-12:** 622 Mbps.
  - **Interconnection Analogy:** Think of it like a cell phone network where your phone uses radio waves to talk to a local tower (LAN), which then connects to a massive global grid of hubs (WAN).

Visualization Summary: Imagine a spiderweb (the network structure). If the web is small and tight, it's a **LAN** (Ethernet/WiFi). If the web stretches across the world, it's a **WAN** (T1/T3/OC-12). The "Spider" is the OS; if you have to tell the spider exactly where to go, you are using a **NOS**. If the spider moves automatically to find what you need without you asking, you are using a **DOS**.

Lesson 3: Communication Structures and the TCP/IP Model

In this lesson, we shift our focus from the "what" and "where" of distributed systems to the **"how"**—specifically, how these separate machines actually manage to talk to one another across a network. As a professor would explain, this is essentially the "postal system" of the computing world.

1. The Four Fundamental Design Issues

When engineers design a communication network for a distributed system, they must solve four basic problems:

- **Naming and Name Resolution:** How do two processes find each other? Just like you need a person's name or address to send a letter, a computer needs a way to locate another process to begin communication.
- **Routing Strategies:** Once you know where you are going, what path should the message take? In a complex network, there might be many ways to get from Site A to Site B; the system must choose the best route.
- **Connection Strategies:** How do two processes send a long sequence of messages? They need a plan for keeping the conversation going without losing parts of the data.
- **Contention:** Since the network is a shared resource, what happens when multiple machines want to talk at the same time? The system needs a way to resolve these conflicting demands for bandwidth.

2. The ISO Network Model (OSI)

To make communication consistent, the industry uses the **ISO (International Standards Organisation) Model**. It breaks communication down into seven distinct layers, each with a specific job:

1. **Physical Layer (L1):** The actual wires, radio waves, or fiber optics.
2. **Link Layer (L2):** Handles data transfer between two directly connected nodes.
3. **Network Layer (L3):** Manages the routing of data across different networks.
4. **Transport Layer (L4):** Ensures data arrives correctly and in order.
5. **Session Layer (L5):** Manages the "conversation" between applications.
6. **Presentation Layer (L6):** Translates data into a format the application understands.
7. **Application Layer (L7):** Where the user interacts with the network (e.g., a web browser).

3. The TCP/IP Protocol Suite

While the ISO model is the theoretical standard, the real world uses the **TCP/IP model**. It simplifies those seven layers into four main categories:

- **Application Layer:** This combines the top three ISO layers. It includes protocols you use every day, such as **HTTP** (web), **DNS** (naming), **Telnet/SSH** (remote login), **SMTP** (email), and **FTP** (file transfer).
- **Transport Layer:** This uses **TCP (Transmission Control Protocol)** for reliable connections or **UDP** for faster, "best-effort" communication.
- **Network/IP Layer:** This uses the **Internet Protocol (IP)** to handle the actual addressing and routing of packets.
- **The Bottom Layers:** While not explicitly defined in the TCP/IP stack shown in your notes, these correspond to the physical hardware like **Ethernet**.

4. A Practical Example: How a Packet Moves

For your exam, you should understand the "handshake" that happens when one computer sends a packet to another on an Ethernet network:

- **Dual Addressing:** Every host has two unique identifiers: a logical **IP Address** and a physical hardware address called a **MAC (Media Access Control) Address**. You need **both** to communicate.
- **DNS:** If you only have a name (like "google.com"), you use the **Domain Name Service (DNS)** to find the corresponding IP address.
- **ARP (The Map-Maker):** To find the MAC address that goes with an IP, the system uses the **Address Resolution Protocol (ARP)**. It sends a **broadcast** message to every machine on the local network asking, "Who has this IP?".
- **Same vs. Different Networks:** If the hosts are on the **same network**, they use ARP to talk directly. If they are on **different networks**, the sender sends the packet to a **router**, which then forwards it toward the destination network.

5. Anatomy of an Ethernet Packet

When data travels over the wire, it is wrapped in an "Ethernet Packet". Here is exactly what is inside, from start to finish:

- **Preamble (7 bytes):** A repeating pattern (10101010) that tells the receiver a packet is coming.
- **Start of Frame Delimiter (1 byte):** A specific pattern (10101011) that marks the start.
- **Destination Address (2 or 6 bytes):** The MAC address of the receiver.
- **Source Address (2 or 6 bytes):** The MAC address of the sender.
- **Length of Data Section (2 bytes):** Tells the receiver how much data to expect.
- **Data (0–1500 bytes):** The actual message being sent.
- **Pad (optional, 0–46 bytes):** Extra bits to ensure the packet is at least 63 bytes long.

- **Frame Checksum (4 bytes):** A math check used for **error detection** to ensure the data wasn't corrupted during transit.

Summary Tip: Think of the **IP address** as the house address on an envelope and the **MAC address** as the specific person living there. You use **DNS** to look up the address in a phone book and **ARP** to shout into the house to see who is home.

Lesson 4: Robustness and Design Challenges

In a distributed system, things will inevitably go wrong. **Robustness** is the system's ability to detect these problems and keep running, while **Design Issues** are the high-level goals engineers strive for to make the system usable and powerful.

1. Robustness: Failure Detection

Detecting a hardware failure is notoriously difficult. To manage this, systems use a **Heartbeat Protocol** to monitor the health of connections:

- **The "I-am-up" Message:** If Site A and Site B are linked, they exchange "I-am-up" messages at fixed intervals.
- **Missing Heartbeats:** If Site A doesn't get a message, it assumes either Site B is down or the message was simply lost.
- **The "Are-you-up?" Check:** Site A then sends an explicit "Are-you-up?" query. If there's still no reply after retries or trying alternate routes, it concludes a failure has occurred.

Crucial Exam Note: Even if a failure is confirmed, Site A **cannot determine exactly why** it happened. It could be that Site B is down, the direct link is broken, the alternate link is broken, or the message itself was just lost.

2. Robustness: Reconfiguration

Once a failure is detected, the system must immediately **reconfigure** itself so other nodes don't try to use broken paths or dead services:

- **Broadcasting Failures:** If a specific link between A and B fails, or if an entire site fails, this must be broadcast to **every other site** in the system.
- **Service Updates:** Sites are notified that any services previously offered by the failed site are now unavailable.
- **Recovery:** When the link or site comes back online, another broadcast is sent to let everyone know it is safe to use again.

3. Core Design Issues

When building these systems, professors look for four key design "pillars":

- **Transparency:** The entire distributed system should feel like one single, conventional, centralized computer to the user.
- **Fault Tolerance:** The system must be designed to continue functioning even when parts of it fail.
- **Scalability:** As demand grows, the system should easily accept new resources (like more servers) to handle the load.
    - **Hadoop:** An open-source framework used for processing massive datasets in distributed environments (originally based on Google's search indexing) is a prime example of scalability.
- **Clusters:** This refers to a collection of semi-autonomous machines that act together as a single, unified system.

Visualization of Failure Detection:

- **Site A:** "Ping! (I-am-up)" ➔ **Site B:** "Ping! (I-am-up)" (Everything is fine).
- **Site A:** "Ping?" ... (Silence) ... "Are you there?" ... (Silence).
- **Site A to All Sites:** "Attention! Site B is currently unreachable. Do not send data there.".

Lesson 5: Distributed File Systems (DFS)

In our final lesson, we will explore the **Distributed File System (DFS)**. This is a distributed implementation of the classical time-sharing model of a file system, allowing multiple users to share files and storage resources across a network. As a professor would describe it, a DFS acts as a manager for a set of dispersed storage devices, making a collection of remotely located storage spaces appear as one unified system.

1. The Structure of a DFS

To understand how a DFS operates, you must be familiar with its three primary components and how they interact:

- **Service:** This is a software entity running on one or more machines that provides specific functions to "a priori unknown" clients.
- **Server:** This refers specifically to the service software when it is running on a single machine.
- **Client:** This is a process that invokes a service using a set of operations called a **client interface**.
- **The Client Interface:** For a file service, this interface consists of primitive operations you use every day, such as **create, delete, read, and write**. Crucially, this interface should be **transparent**, meaning it does not distinguish between local and remote files. In some cases, a lower-level intermachine interface is needed to handle actual cross-machine interactions.

2. Naming and Transparency

Naming is the process of mapping logical objects (the file name you see) to physical objects (where it actually sits on a disk). A high-quality DFS uses **multilevel mapping** to hide the complex details of how and where a file is stored. There are two specific naming concepts you must memorize:

- **Location Transparency:** The name of the file does not reveal anything about its physical storage location.
- **Location Independence:** This is even more advanced; it means the file name does not have to change even if the physical storage location of that file moves to a different disk or site.
- **Replication Transparency:** If a file is replicated (copied) across several sites for safety, a transparent DFS hides both the existence of these multiple copies and their specific locations from the user.

3. Remote File Access and Caching

Moving files across a network constantly would be too slow. To solve this, use **caching**.

- **The Caching Process:** The system retains recently accessed disk blocks in a local cache. If you need data that isn't cached yet, a copy is brought from the server to the user, and all subsequent accesses are performed on that local copy.
- **The Master Copy:** While parts of a file may be scattered in different caches across the network, there is always one **master copy** residing on the server machine.
- **Network Virtual Memory:** Because the system manages these cached copies across the network, this process is sometimes referred to as network virtual memory.

4. The Cache-Consistency Problem

The biggest challenge in a DFS is ensuring that a locally cached copy stays consistent with the master copy on the server. There are two main ways to handle this:

- **Client-Initiated Approach:** The client starts a "validity check" where it asks the server to verify if the local data is still consistent with the master copy.
- **Server-Initiated Approach:** The server keeps a record of which clients are caching which files. If the server detects a potential inconsistency (like another user modifying the master copy), it must take action to notify the clients.

Visualization of DFS Naming: Imagine you have a file named "Exam_Notes.pdf".

- **With Location Transparency:** You see the name, but you have no idea if it is on a server in New York or London.
- **With Location Independence:** If the IT department moves that file from the New York server to the London server, you still just click "Exam_Notes.pdf" and it works perfectly without you having to update any links.