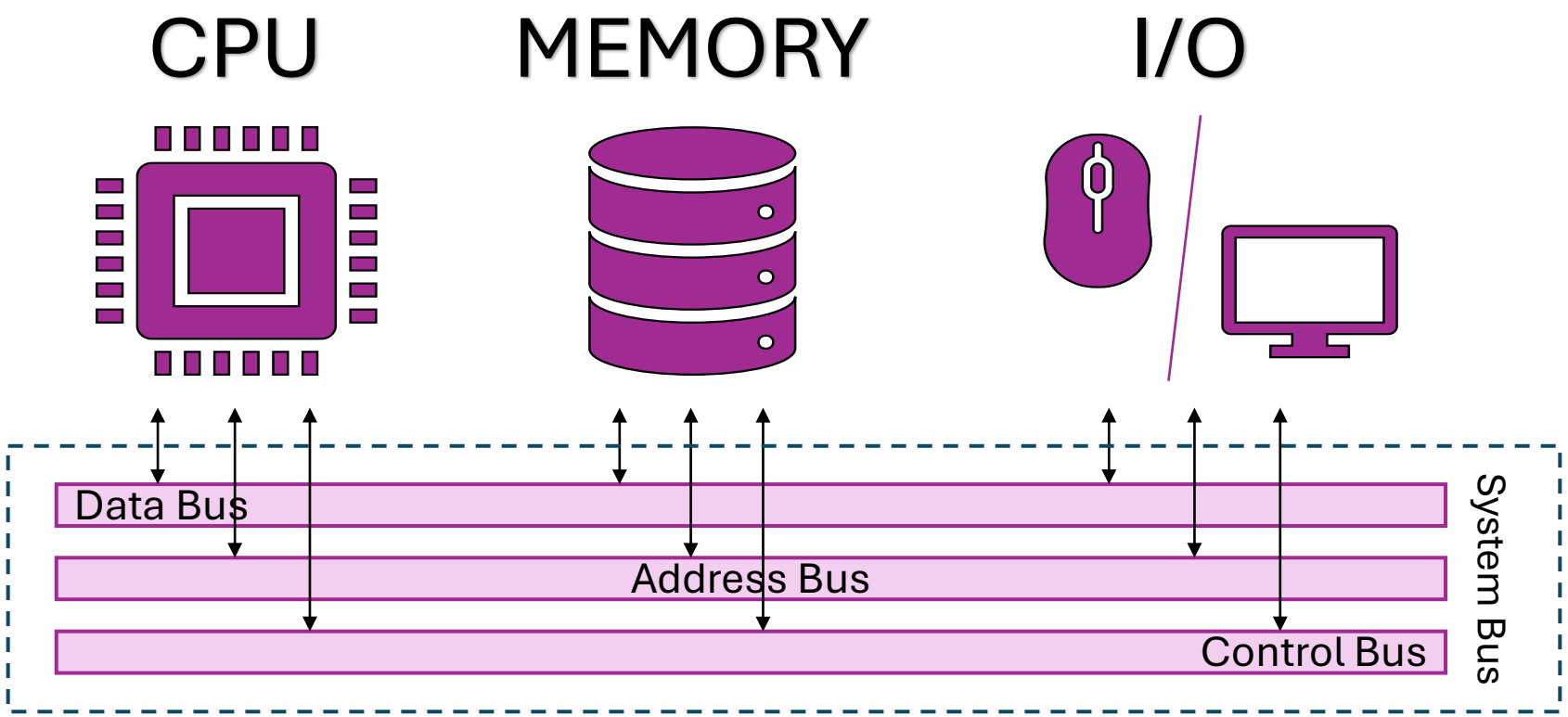
2. Computer Architectures

CPU, Memory, and I/O

Summary

- The Von Neumann Model
- System Bus
- CPU
- Instruction Cycle
- Memory
- MDR and MAR Registers
- FETCH and STORE Operations
- INPUT and OUTPUT Devices
- INPUT/OUTPUT Devices

The Von Neumann Model



System Bus

The **system bus** typically connects the CPU, RAM, I/O peripherals, and other system components. It allows for the **transfer of data**, **instructions**, and **control signals** between these components, enabling them to **collaborate** and **interact**.

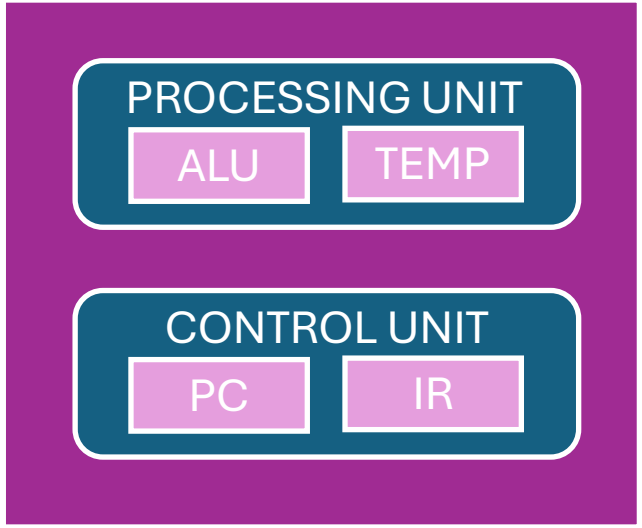
- **Data Bus:** Transmits binary data from one component to another. The data bus lines allow the transfer of information such as program instructions, data to be processed, or calculation results.

- **Address Bus:** Indicates a specific location in memory or I/O peripherals. The address bus allows system components to specify the source or destination of the data to be transferred.

- **Control Bus:** Carries control signals that coordinate and manage the operations of various system components. Control signals can include synchronization signals, interrupt signals, operation enable or disable signals, and status signals.

CPU

The **CPU (Central Processing Unit)** is the primary component of a computer that performs most of the processing inside the system. It **executes instructions** from programs, **performs calculations**, and **manages data flow** to and from other components like memory and peripherals. The CPU is often referred to as the **"brain"** of the computer, as it handles all critical tasks to ensure the system operates efficiently.



CPU – Processing Unit

The **processing unit** is a central part of the CPU responsible for **executing instructions** and managing the **data processing** tasks. It includes several key components that work together to perform **arithmetic** and **logical** operations, **control instruction flow**, and **store intermediate results**.

- **ALU (Arithmetic Logic Unit)**

The ALU is a critical component of the CPU that performs **arithmetic** operations (such as addition, subtraction, multiplication, and division) and **logical** operations (such as AND, OR, NOT, and XOR). It is essential for executing calculations and making decisions based on logical comparisons.

- **TEMP (Temporary Registers)**

Temporary registers (TEMP) are **small**, **fast storage locations** within the CPU used to hold **intermediate data** and **results** during instruction execution. They provide quick access to frequently used values and help optimize the processing speed by reducing the need to access slower main memory.

CPU – Control Unit

The **control unit** is a crucial part of the CPU that **directs the operation of the processor**. It **interprets instructions** from the program, **generates control signals**, and **coordinates the activities** of the CPU's other components, ensuring that instructions are executed in the correct sequence and timing.

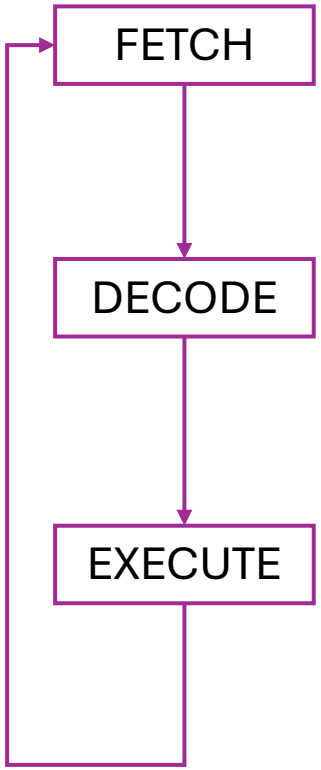
- **PC (Program Counter)**

The Program Counter (PC) is a register within the control unit that holds the **address of the next instruction to be executed**. It increments after each instruction is fetched, ensuring a sequential flow of program execution, unless a jump or branch instruction alters the flow.

- **IR (Instruction Register)**

The Instruction Register (IR) is a component of the control unit that **temporarily** holds the **current instruction being executed**. It stores the fetched instruction from memory, allowing the control unit to decode and process it, directing the necessary actions to other parts of the CPU.

Instruction Cycle



- **Fetch:**

The CPU retrieves the next instruction from memory, as indicated by the Program Counter (PC).

The instruction is loaded into the Instruction Register (IR).

- **Decode:**

The control unit interprets the fetched instruction in the IR.

It determines the operation to be performed and identifies the necessary operands.

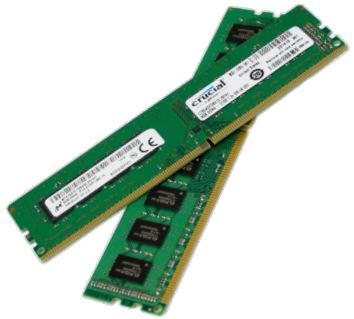
- **Execute:**

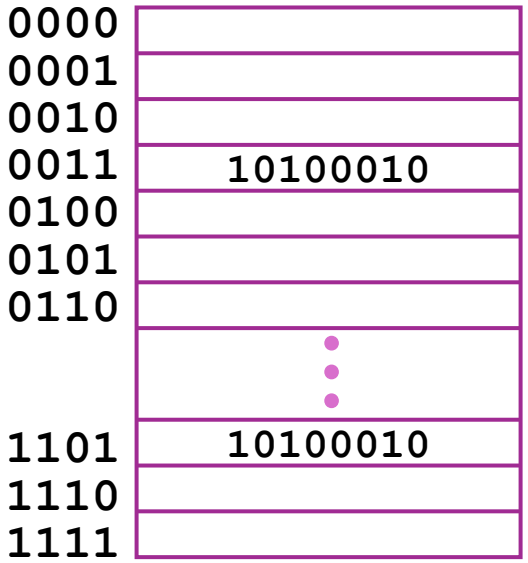
The CPU performs the operation specified by the instruction.

This may involve arithmetic or logical operations carried out by the ALU, data transfer, or control operations.

Memory/1

In the Von Neumann architecture, memory typically refers to **RAM (Random Access Memory)**, which is a critical component that **stores** both **data** and **program instructions temporarily** while the computer is running.





When a device is operating, both the **program** (e.g., Word application) and the **associated data** (e.g., the DOC document you are writing) are **stored in memory (RAM)**.

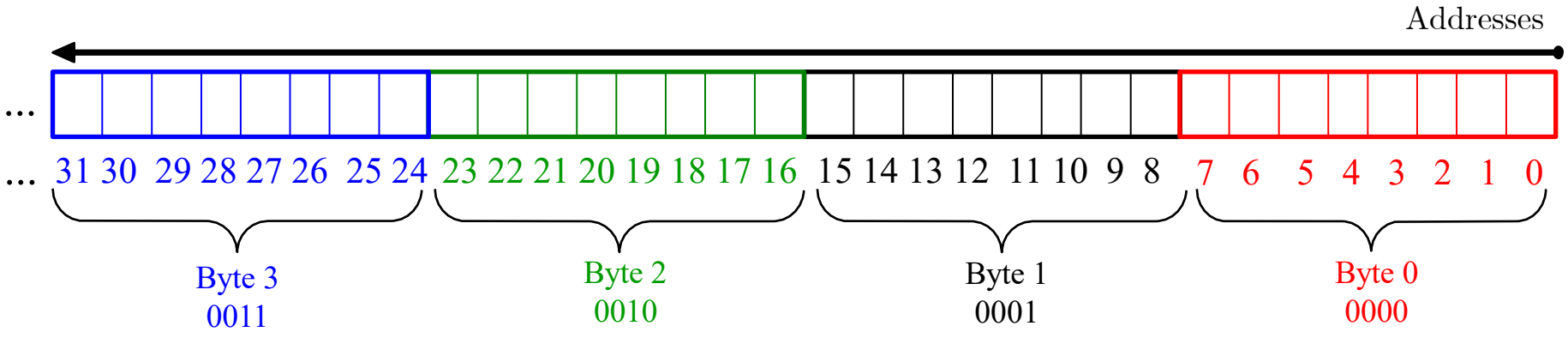
The **basic operations** that can be performed with memory include:

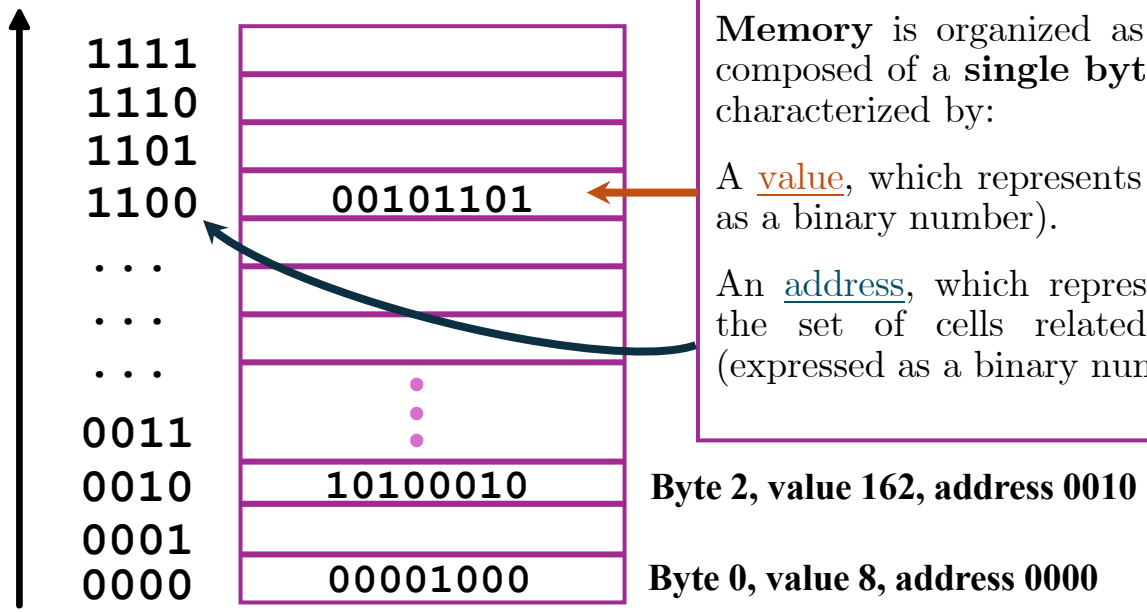
STORE: To save information.
FETCH: To retrieve information.

These operations are facilitated by the addressing mechanism known as **memory addresses**.

ICTAcademy
INFORMATION AND COMMUNICATIONS TECHNOLOGY

Memory/2





Memory is organized as a **sequence of cells**, each composed of a **single byte** (**8 bits**). A memory cell is characterized by:

A value, which represents the cell's content (expressed as a binary number).

An address, which represents the starting address of the set of cells related to a specific application (expressed as a binary number).

Byte 2, value 162, address 0010

Byte 0, value 8, address 0000

MDR and MAR Registers

- **MDR (Memory Data Register)**

The Memory Data Register (MDR), also known as the Memory Buffer Register (MBR), is a register within the CPU that temporarily **holds data being transferred to or from memory**.

When data is read from memory, it is first loaded into the MDR before being processed by the CPU. Similarly, when data is written to memory, it is placed in the MDR before being stored in the specified memory location. The MDR acts as a buffer, facilitating smooth data transfer between the CPU and memory.

- **MAR (Memory Address Register)**

The Memory Address Register (MAR) is a register within the CPU that **holds the address of the memory location** to be accessed next.

During the fetch stage of the instruction cycle, the MAR contains the **address of the instruction to be fetched**.

During data operations, the MAR holds the **address of the data to be read from or written to memory**. The MAR ensures that the correct memory location is accessed, enabling precise data retrieval and storage.

FETCH and STORE Operations

The operation of the Memory module involves the following two steps:

FETCH operation / To read from a location (A):

- Write the address (A) into the MAR.
- Send a "read" signal to the memory.
- Read the data from the MDR.

STORE operation / To write a value (X) to a location (A):

- Write the data (X) to the MDR.
- Write the address (A) into the MAR.
- Send a "write" signal to the memory.

INPUT and OUTPUT Devices

Input and Output (I/O) operations are crucial for allowing a computer system to **interact** with the **external environment** and **peripheral devices**.

Input:

- **Purpose:** To gather data from external sources and provide it to the computer for processing.
- **Devices:** Common input devices include keyboards, mice, scanners, microphones, and cameras.
- **Process:** Input devices convert user actions or external data into digital signals that the computer can process.

Output:

- **Purpose:** To present processed data from the computer to the external environment.
- **Devices:** Common output devices include monitors, printers, speakers, and projectors.
- **Process:** Output devices convert digital signals from the computer into a human-readable or perceivable form, such as text, images, sound, or physical output.

INPUT/OUTPUT Devices

Some devices function as **both input and output** devices, allowing for **two-way interaction** between the user and the computer system. These devices are versatile and can **both send data to** and **receive data from** the computer.

Examples of I/O Devices:

- **Touchscreen Monitors:**

- Input: The user can interact with the screen by touching it, which sends input signals to the computer.

- Output: The screen displays visual information from the computer.

- **External Hard Drives:**

- Input: Data can be written to the drive from the computer.

- Output: Data can be read from the drive to the computer.

- **Modems:**

- Input: Receive data from the internet.

- Output: Send data to the internet.

2. Computer Networks

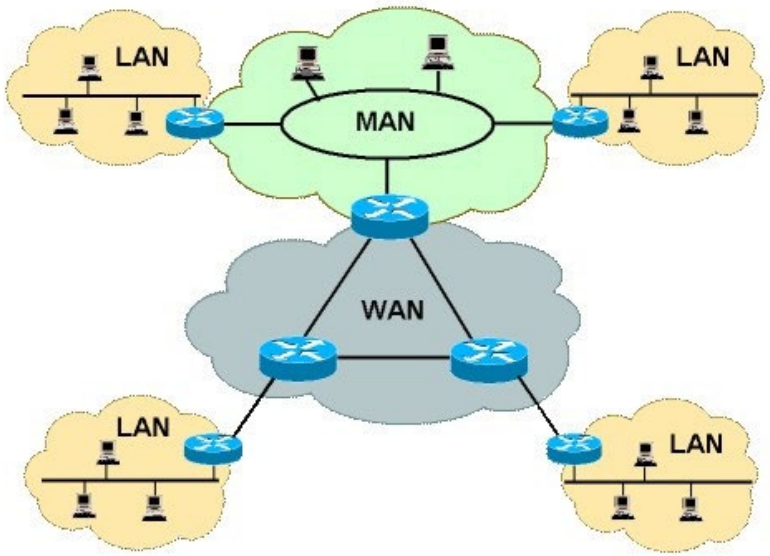
Protocols

Summary

- Internetwork
- Protocols
- Central Role of Software
- Network Communication
- Connection-oriented Services
- Connectionless Services

Internetwork

Internetwork or **Internet**: a collection of different networks connected via a gateway. Each network uses **different hardware** and, more importantly, **different protocols** to manage internal communication.



Protocols

Communication between various entities within networks is managed by **protocols**.

A **network protocol** is a formal, predefined **set of rules** that govern the interactions between two or more connected electronic devices to facilitate communication. These rules primarily address the software components of the devices.

The earliest computer networks were primarily focused on hardware, considering software merely as an adjunct. This approach is outdated. In modern contexts, **network software** is intricately structured and plays a **central role** in network functionality.

Central Role of Software/1

Feature Complexity:

Modern networks must support a wide range of services and applications, from simple data transmission to complex security operations, traffic management, performance optimization, and cloud integration. These functions require sophisticated software capable of dynamically managing various needs and configurations.

Adaptability and Scalability:

Today's networks need to be highly adaptable and scalable to swiftly respond to changing business or consumer demands. Network software enables this flexibility, allowing updates and modifications to be implemented much more quickly and at lower costs compared to hardware changes.

Security:

The security of modern networks heavily relies on advanced software to implement firewalls, intrusion detection systems, encryption, and other essential security measures to protect data and keep the network secure against increasingly sophisticated threats.

Central Role of Software/2

Network software is crucial not only for managing functionality but also for **ensuring compatibility** and **communication** among **diverse hardware** components.

By acting as an **intermediary**, it enables **different devices**, potentially from various manufacturers, to **work together** seamlessly.

This **interoperability** is key to building flexible and efficient network systems that can adapt to **new technologies** and **requirements**.

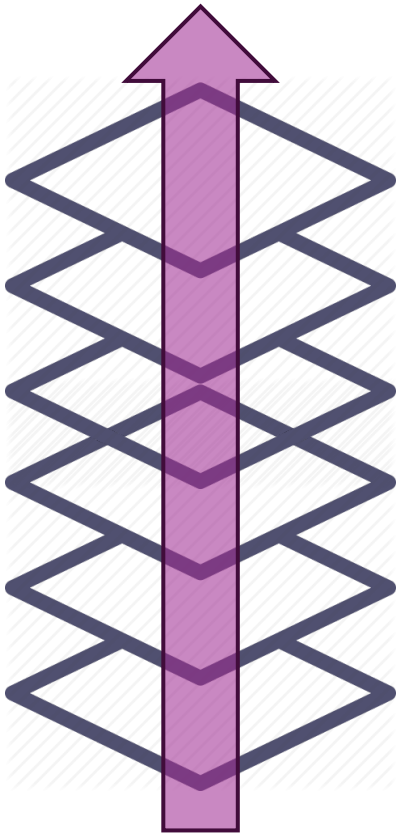
Stack of Layers

To simplify design, networks are structured into **layers** that are built upon one another.

Differences between networks can include:

- The number of layers
- The names of the layers
- The content of each layer
- The functionality of each layer

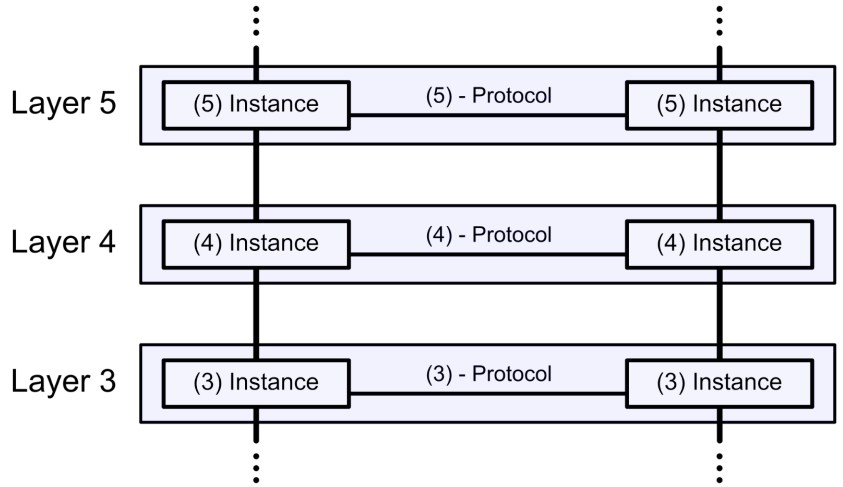
Each layer serves to provide **specific services** to the layers above it, while **abstracting the details** of how those services are implemented.



Layer Protocol

When the same network layer, referred to as **layer n**, communicates across different machines, the rules and conventions guiding this communication are known collectively as the **layer n protocol**.

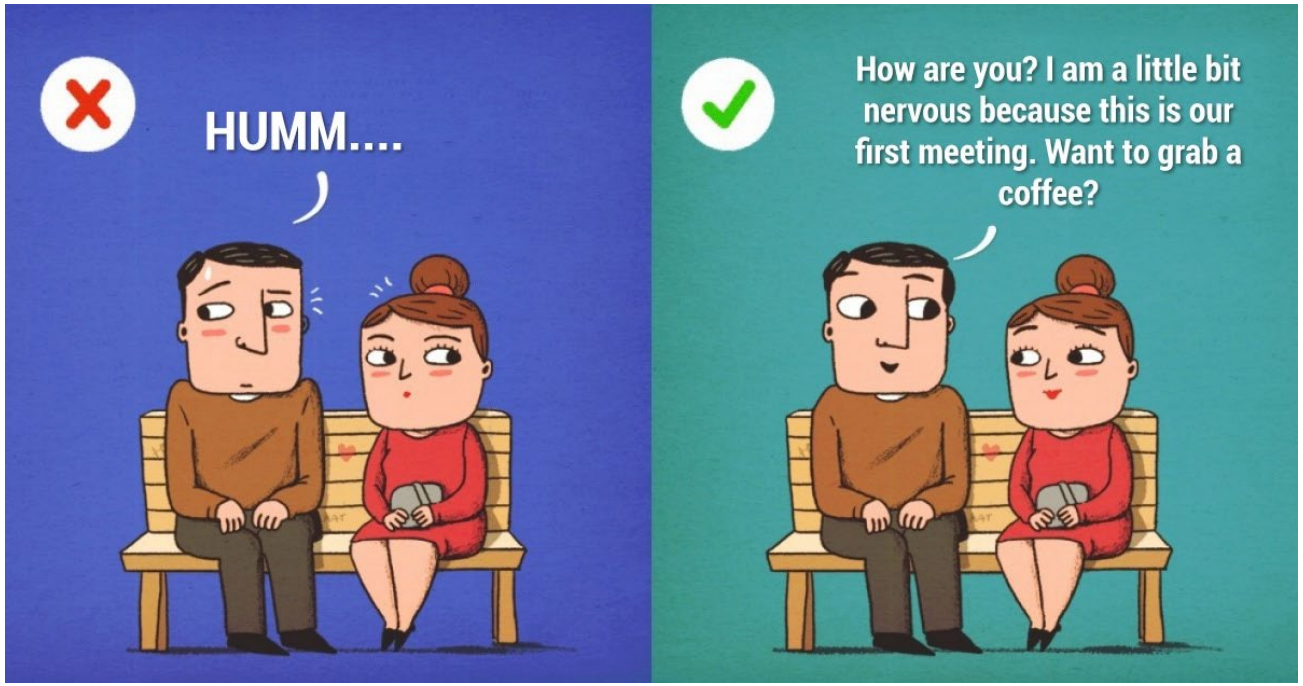
A **protocol** is essentially an agreement that dictates the **procedures for communication** between the parties involved.



Communication between the n-th layer of one machine and the n-th layer of another machine is enabled by the services offered by the underlying layer (n-1).

Protocols - Analogy

- Disregarding the protocol could complicate communication, potentially to the point of impossibility.



Network Communication - Analogy/1

Start and End Points:

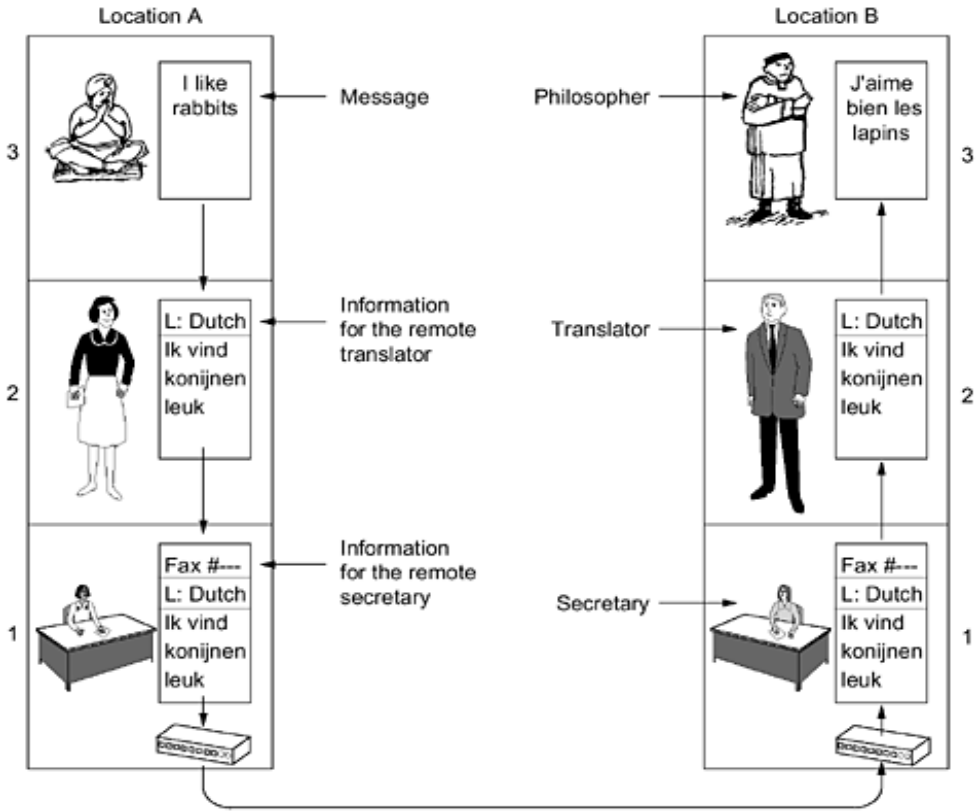
The process begins at Location A where a person (the sender) expresses a message, "I like rabbits." It ends at Location B with a philosopher (the receiver) receiving the message translated into French, "J'aime bien les lapins." In networking, these points represent the source and destination nodes.

Transmission and Translation Layers:

The message undergoes several steps before reaching the philosopher. Initially, it is translated into Dutch and passed through various intermediaries (secretary and translator) who handle and forward the information, akin to data passing through multiple network devices like routers and switches.

Data Packet Analogy:

The original message "I like rabbits" can be seen as data packets. As it moves from the source, it is processed and repackaged at various stages—translation into another language can represent encoding or encryption in network terms.



ICTAcademy
INFORMATION AND COMMUNICATIONS TECHNOLOGY

Network Communication - Analogy/2

Routing:

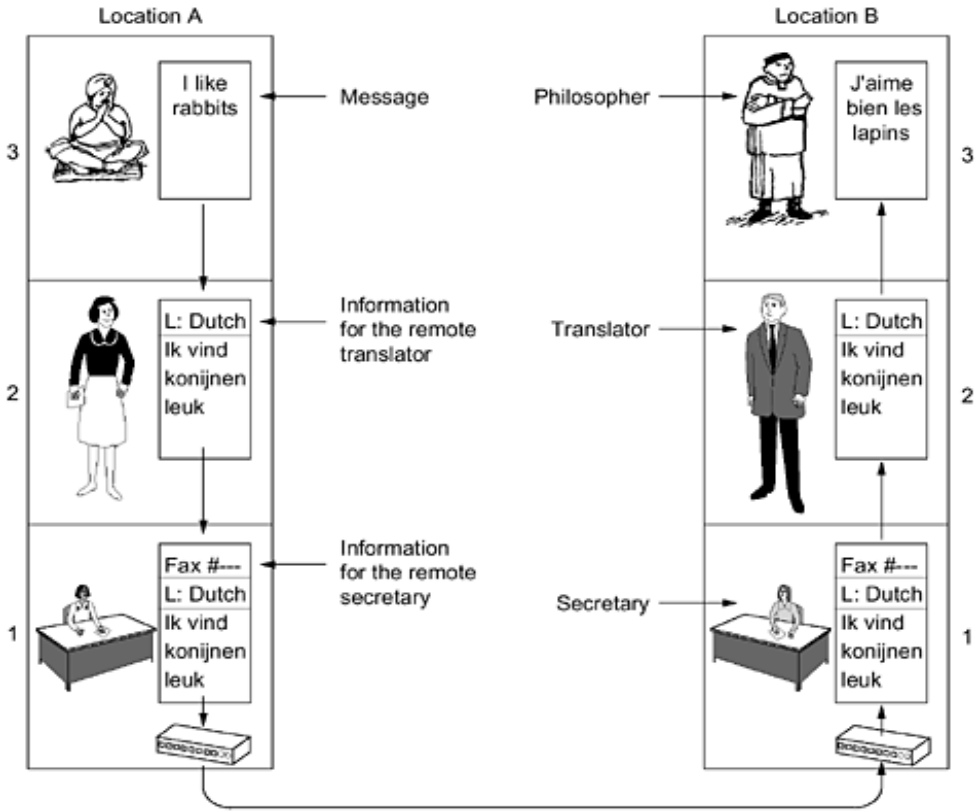
The use of faxes and different personnel for processing the information is similar to how data packets are routed through different network paths. Each node (person or device in the diagram) decides how and where to send the data next based on network protocols.

Protocol Layers:

Each step in the diagram where the message is handled and processed represents different layers in network protocols (like the OSI model). For example, the secretary could be seen as operating in the presentation layer, translating data formats, while the translator operates in the application layer, converting the message contents into different languages.

End-to-End Delivery:

Just as the message starts as a simple expression and ends up being understood by someone who speaks a different language, in networking, data starts from a source and is delivered to a destination across various network segments and possibly transformed or secured in the process, ensuring it is usable upon receipt.



Network Communication

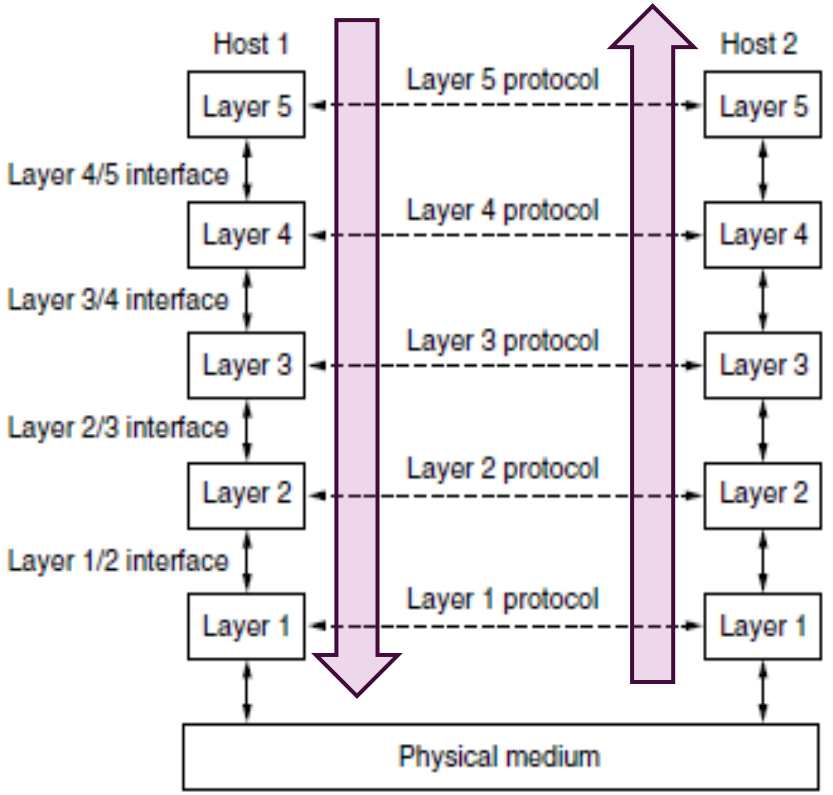
Data is not directly transferred from layer *n* on one machine to layer *n* on another machine.

- **Source (top-down):** each layer passes data and control information down to the immediately lower layer, continuing until it reaches the lowest layer.

- **Destination (bottom-up):** each layer passes data and control information up to the immediately upper layer, continuing until it reaches the highest layer.

Interface: Specifies the operations and services that the lower layer provides to the upper one.

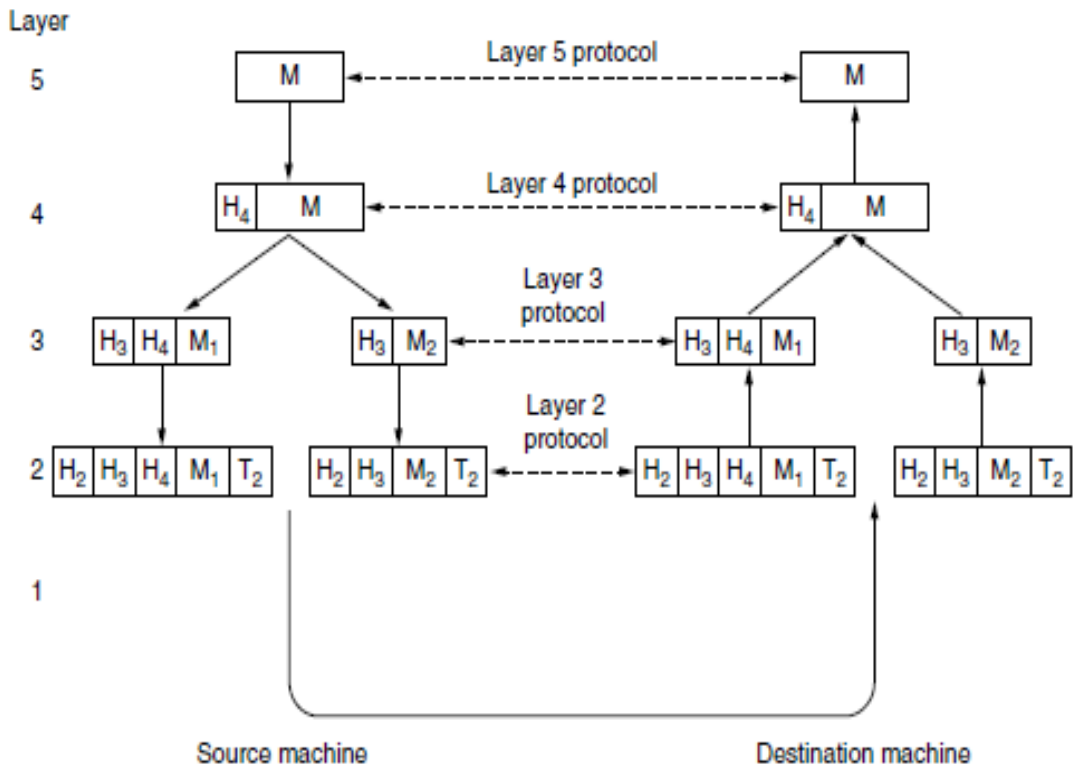
Below level 1, the **physical medium** facilitates data transfer from host 1 to host 2.



Network Communication – Technical Example/1

The application program needs to send the message M to its peer entity (i.e., the same application program on another machine).

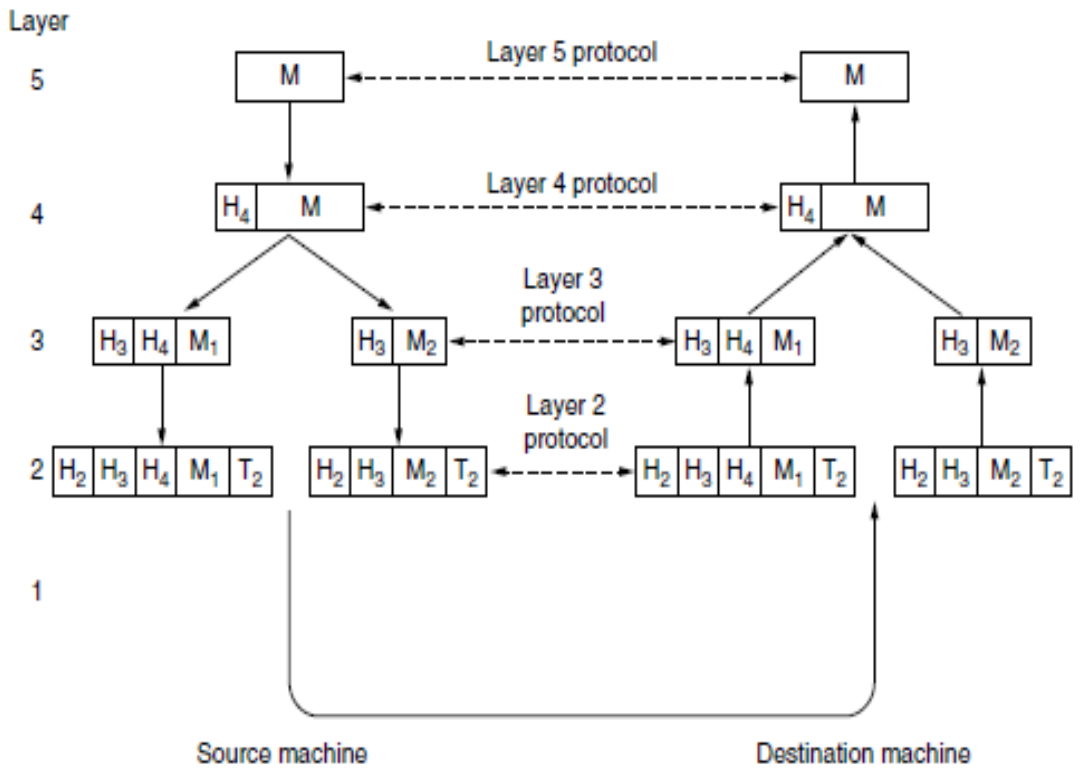
1. **Layer 5** delivers message M to **layer 4.**
2. **Layer 4** adds a header H_4 to the message. The headers contain control information such as sequence number, data offset, acknowledgment number, size, etc. The **layer 4** delivers message M to **layer 3**.
3. **Layer 3** divides the data into smaller units (packets) and adds a header H_3 to each packet. The headers contain source and destination IP addresses, total length, a unique identifier for the packet, header length, TTL etc. The **layer 3** delivers packets to **layer 2**.



ICTAcademy
INFORMATION AND COMMUNICATIONS TECHNOLOGY

Network Communication – Technical Example/2

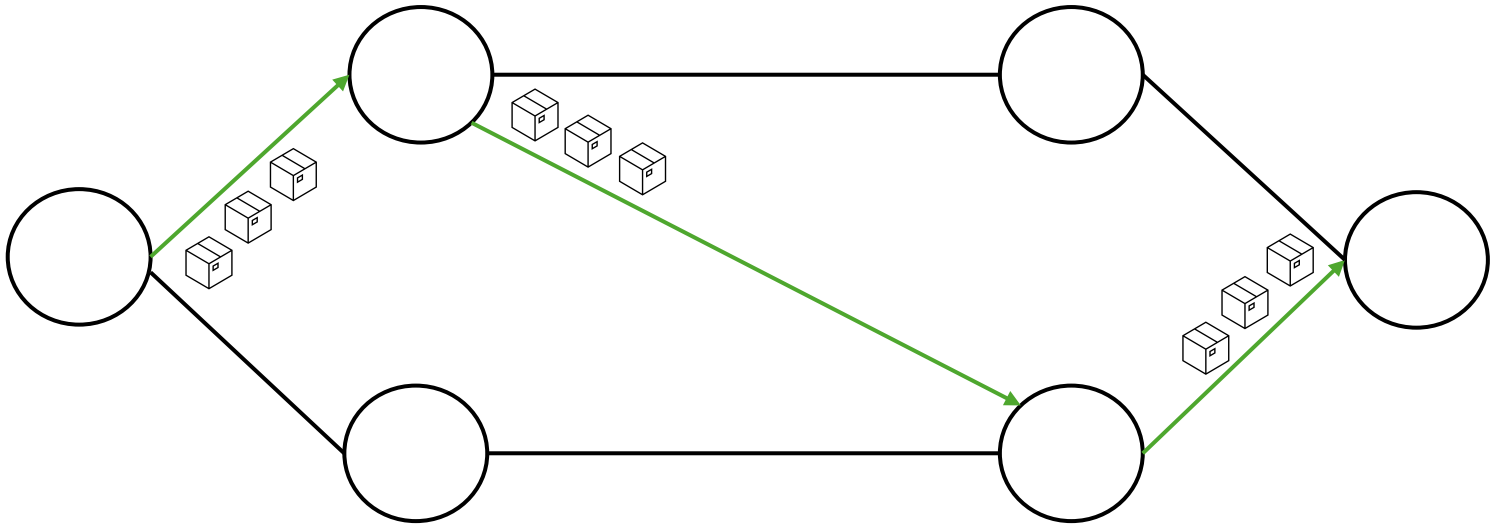
4. **Layer 2** adds a header H_2 and a trailer T_2 to each received packet referred as to frame. The headers contain source and destination MAC addresses, payload length, etc. The trailer is used for transmission error-checking. **Layer 2** delivers packets to **layer 1**.
5. **Layer 1**, the **physical layer**, transfers data to the destination machine.
6. **At the destination machine,** the message moves upward through each layer, with headers and trailers being removed at each step. The message is **fully reconstructed** and becomes readable once it reaches **layer 5**.



ICTAcademy
INFORMATION AND COMMUNICATIONS TECHNOLOGY

Connection-oriented Services/1

Connection-oriented services are those in which a **dedicated connection** is established between the two communication nodes before data transfer begins. This type of service ensures that data is **transmitted reliably** and in the **correct order**.



Connection-oriented Services/2

Key characteristics of connection-oriented services:

- **Reliability:**

Data transmission is guaranteed, and transmission errors are detected and corrected. Used in applications where reliability is more important than speed, such as email and online banking.

- **Ordering:**

Data arrives in the order it was sent.

- **Flow Control:**

Regulates the amount of data sent to prevent overwhelming the receiver.

- **Establishing and Closing the Connection:**

Requires an initial handshake process to establish the connection and a closing process to terminate it.

Connection-oriented Services/3

Handshaking (Opening the Connection)

1. **Connection request:**

Device A sends a request message to Device B to initiate the connection.

2. **Response:**

Device B responds to Device A, confirming that it has received the request and is ready to communicate.

3. **Confirmation:**

Device A sends a confirmation that it has received Device B's response.

Closing the Connection

1. Termination request:

When the communication is complete, Device A sends a termination request to Device B.

2. **Acknowledgment:**

Device B acknowledges the termination request, indicating it is ready to close the connection.

3. **Final confirmation:**

Device A sends a final confirmation, completing the termination process.

Connection-oriented Services/4

The **ACK** (**Acknowledgment**) service is a mechanism used to ensure that the data sent is correctly received by the recipient.

1. **Packet sending:**

Device A sends a data packet to Device B.

2. **Reception and acknowledgment:**

When Device B receives the packet, it sends an acknowledgment message (ACK) to Device A to indicate that the packet has been received correctly.

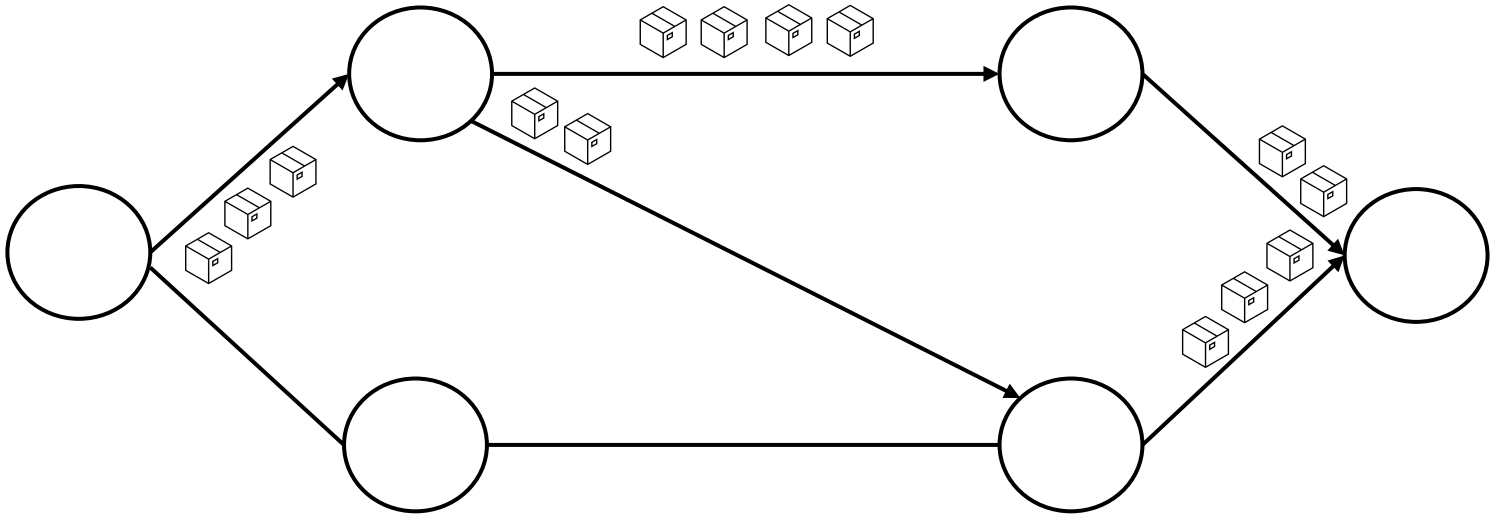
- **Retransmission in case of error:** If Device A does not receive the ACK within a certain period of time, it retransmits the packet, assuming it was lost or damaged during transmission.

This process continues **for each data packet** sent, ensuring that all packets are **received correctly** and in the **right order**.

Connectionless Services/1

Connectionless services do **not require** the establishment of a **dedicated connection** before sending data. Instead, each data packet is **treated independently** and contains all the necessary information to reach its destination.

Two packets with the same source and destination machines can travel through different paths and arrive at different times.



Connectionless Services/2

Key characteristics of connectionless services:

- **Simplicity and Speed:**

Does not require a handshake process, thus having lower latency compared to connection-oriented services.

- **No Delivery Guarantee:**

Packets can be lost, duplicated, or arrive out of order.

- **Packet Independence:**

Each packet is treated separately and contains all the necessary information to reach its destination.

- **Usage in Specific Applications:**

Used in applications where speed is more important than reliability, such as streaming and online gaming.