ANALYSIS AND MACHINE LEARNING TECHNIQUES WITH BASICS OF COMPUTER SCIENCE AND STATISTICAL LEARNING - ANALYSIS AND MACHINE LEARNING TECHNIQUES WITH BASICS OF COMPUTER SCIENCE AND STATISTICAL LEARNING

Prof. Loriano Storchi
loriano@storchi.org
https://www.storchi.org/

Contents



- Introduction to Informatics
  - What is a computer
  - Networks and TCP/IP
  - Digitalization and basics of data encryption

Informatics Core Concept & Flow

Informatics (Computer Science) is the science of representing and processing information.

It centers on the study of algorithms that describe, manipulate, and transform data to solve complex problems.

STEP 01



INPUT

Data capture & initial information entry

STEP 02



COMPUTER ELABORATION

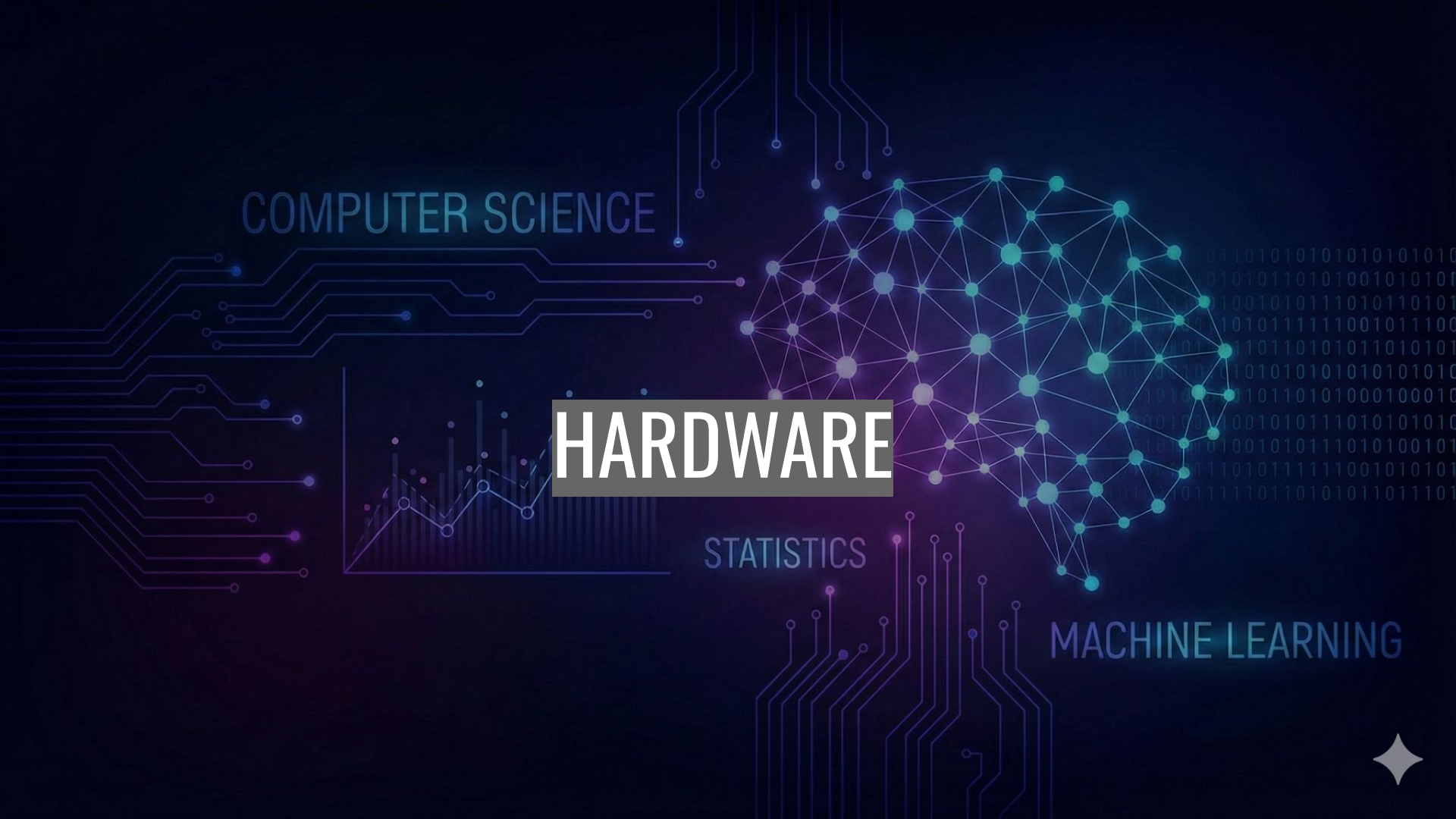
Algorithmic transformation & automated processing

STEP 03

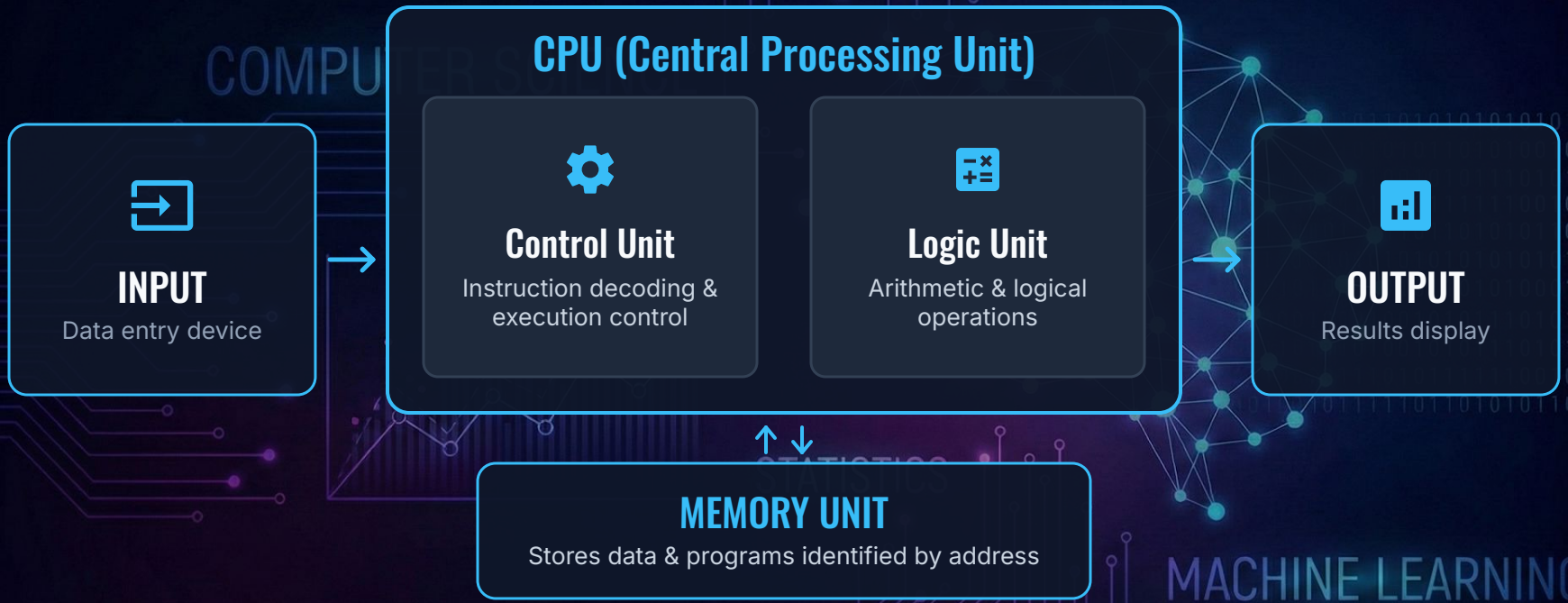


OUTPUT

Structured results & actionable insights


COMPUTER SCIENCE
HARDWARE
STATISTICS
MACHINE LEARNING

Von Neumann Machine (Zuse) Architecture & Subsystems



SYSTEM BUS: Interconnection line linking all subsystems | **HARVARD ARCHITECTURE:** Separation between data memory and program memory

Von Neumann Machine (Zuse) Core Architecture & Subsystems

The architecture described by Von Neumann (Zuse) is composed of four primary interconnected subsystems:

1. CPU (Central Processing Unit)

The CPU is the central processing unit responsible for executing program instructions and managing data operations across the system.

2. Memory Device

A dedicated memory storage device used to hold data and instructions, where each item can be uniquely identified by its address.

3. Input / Output (I/O)

Various Input/Output devices designed to facilitate interaction between the machine and external users or other computing systems.

4. System Bus

An interconnection communication line that links all individual subsystems together to transfer data and control signals.

Von Neumann Machine (Zuse)

Core Concept & Architecture Overview

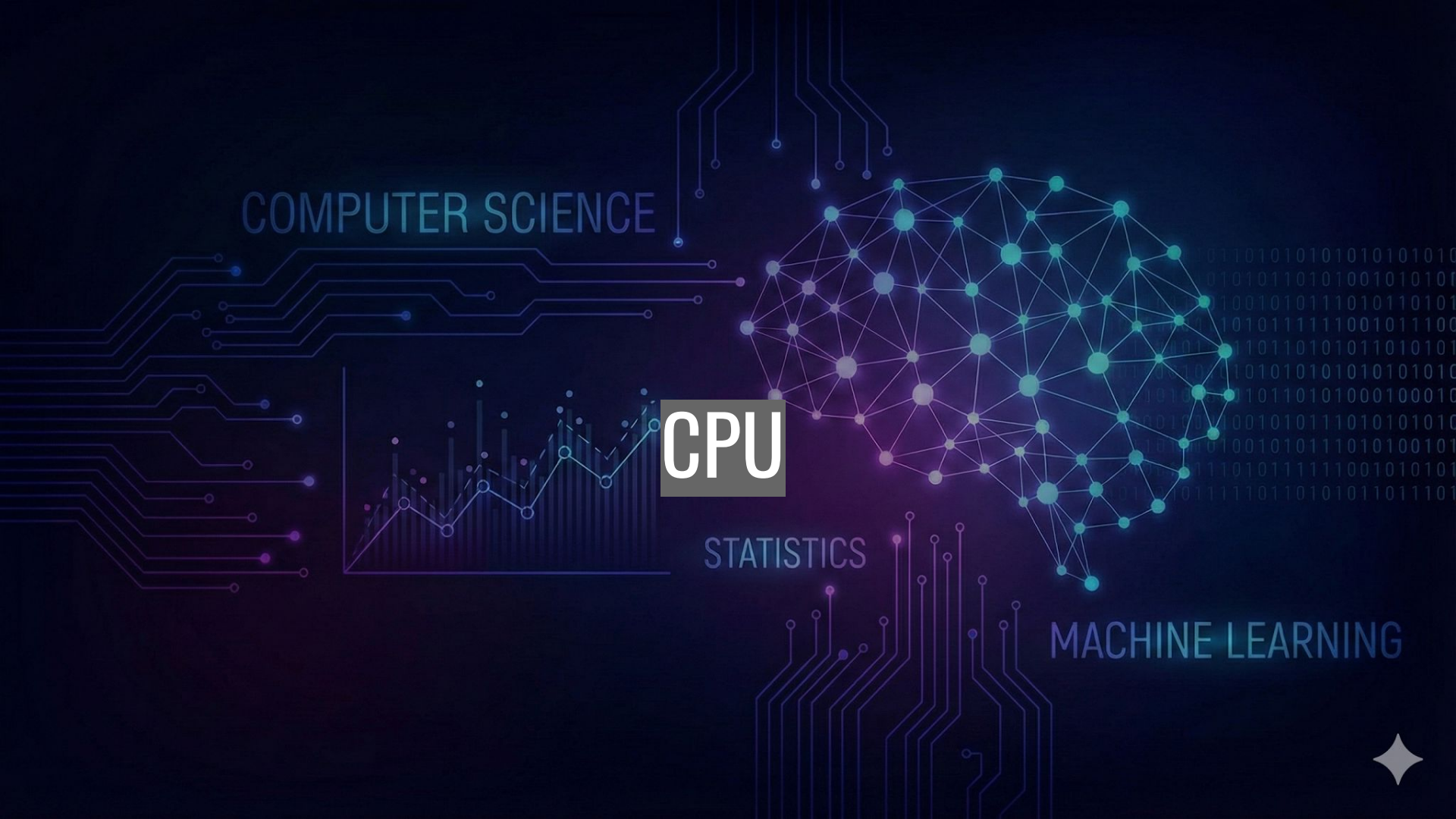
Hardware & Software Flexibility

A computer constructed in this way is an extremely flexible machine. The hardware provides basic functionality, while the software will then specialize the machine to perform specific tasks.

Modern Component Analysis

Below we will present the various basic components of the calculator with a “modern” perspective

KEY TAKEAWAY: Universal hardware adaptable to arbitrary tasks via programmable software instructions.


COMPUTER SCIENCE
CPU
STATISTICS
MACHINE LEARNING

CPU Central Processing Unit Overview

Core Management & Execution

CPU (Central Processing Unit) coordinates and manages all the various hardware devices to acquire, interpret, and execute program instructions.

Chip Architecture & Communication

Today, it consists of a single chip and communicates with the outside world via pins. Using these pins, it receives signals and sends electrical signals (information consisting of sequences of bits).

KEY TAKEAWAY: The central brain orchestrating hardware operations and processing bitstream signals through chip pins.

CPU Control Unit (CU) Overview

PRIMARY ROLE: The Control Unit (CU) is a core component of the CPU that directs the overall operation of the processor.



Orchestration

Acts like an orchestra conductor. It doesn't perform calculations (the ALU's job), but tells every component when and what to execute.



Data Flow Management

Manages and coordinates the flow of data between the CPU core, system memory, and input/output devices.



Core Enablement

Without the CU, high-speed Registers and the powerful ALU would remain silent and inactive.

KEY TAKEAWAY: The CU functions as the central command hub, directing timing and signal routing across all CPU resources.

CPU Arithmetic Logic Unit (ALU) Overview

PRIMARY ROLE: ALU (Arithmetic Logic Unit) performs logical and arithmetic operations.

Inputs

Operands (A and B) and an Opcode (Operation Code).

Outputs

The Result (Y) and Status Flags.

Role

It acts as the "calculator" of the CPU. Without it, the computer cannot process data, only move it.

REPRESENTATION: It is often represented by a "V" shape symbol in architectural diagrams.

CPU - ALU

Core Operations

ARITHMETIC

These operations handle numerical calculations. The complexity depends on the ALU design.

- **ADD / SUB:** Basic integer addition and subtraction.
- **INC / DEC:** Increment or decrement by 1.
- **MUL / DIV:** Often handled by specialized hardware in modern CPUs, but basic ALUs may use repeated addition/subtraction.

LOGIC

These operations treat data as individual bits rather than numbers, essential for decision making.

- **AND:** Used for masking (clearing specific bits).
- **OR:** Used for setting specific bits.
- **XOR:** Used for toggling bits or comparing values.
- **NOT:** Inverts all bits (1s Complement).

CPU Registers Overview

PRIMARY ROLE: High-speed internal memory within the CPU allowing rapid access to active data and system state.



Internal Memory

Allows access to data in a significantly faster way than standard memory.



Instruction Pointer

IP (or Program Counter PC) contains the pointer to the next instruction to execute.



Flags Register

Series of bits representing CPU state; e.g., Overflow flag set to 1 on result overflow.

REQUIREMENT: All CPUs always possess at least two registers: the Instruction Pointer (IP) and Flags Register.

CPU Clock & Timing Systems

CLOCK DEFINITION: Marks time intervals in which internal devices operate, determining overall processing speed as intervals per unit of time.



Clock Cycles

The CPU state changes with every pulse. Execution time of any given operation is measured in clock cycles.



Signal Circuits

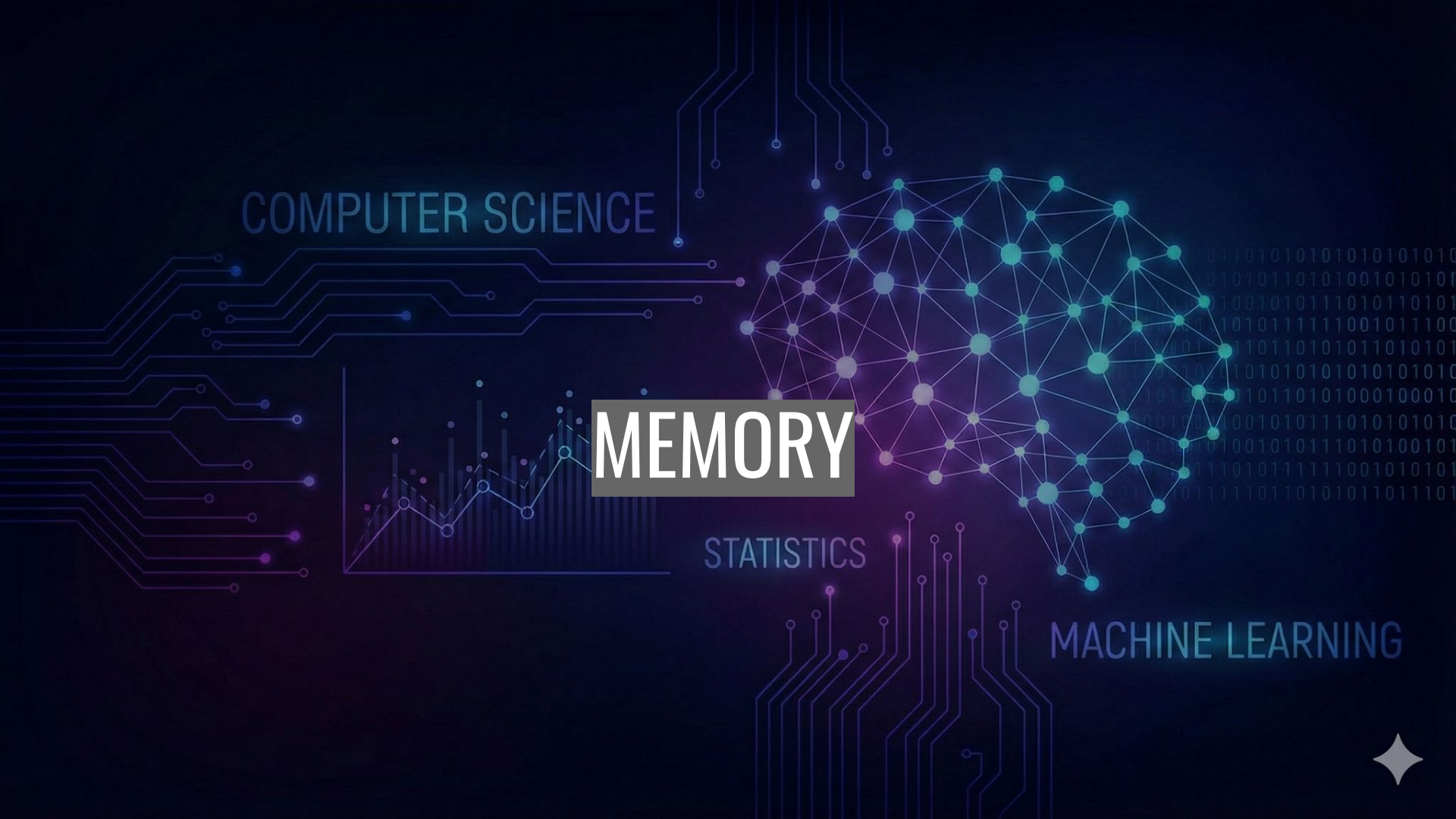
Dedicated distribution circuits propagate timing impulses reliably across all CPU components.



Critical Path

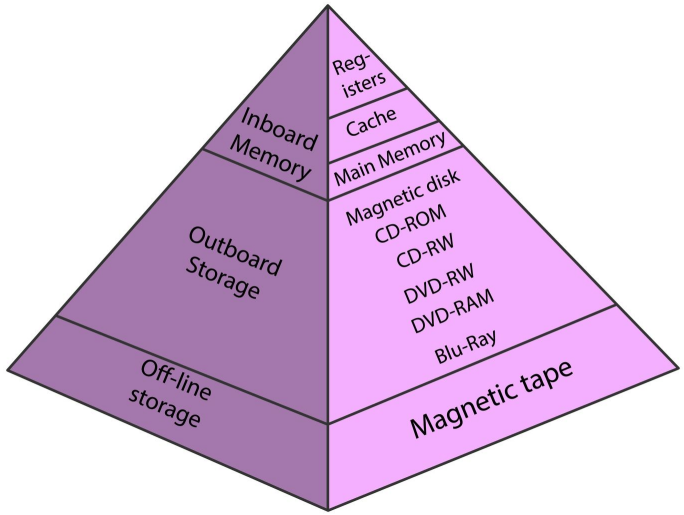
The longest path a signal must travel across the distribution circuit, establishing maximum frequency.

SPEED LIMITATION: No CPU can operate faster than the time it takes for the clock signal to traverse its critical path.


COMPUTER SCIENCE
MEMORY
STATISTICS
MACHINE LEARNING

Memory Hierarchy Speed vs. Capacity Trade-offs

HIERARCHY PRINCIPLE: The memory hierarchy in current processors is composed of several levels, each one characterized by data access speeds inversely proportional to their size: the larger these areas, the longer it takes to retrieve the data contained within them.



Access Speed vs. Size

Smaller memory units located closer to the CPU operate at peak speeds, while larger storage layers trade latency for massive capacity.

Tiered Architecture

Spans high-speed Inboard Memory (Registers & Cache), Main RAM, Outboard Storage (Disks/Optical), to Off-line Archives (Tape).

Registers The CPU's "Hands"

Location

Inside the CPU itself, sitting directly at the core of the processor execution unit.

Speed

The absolute fastest access speed possible in the system with virtually zero latency.

Capacity

Extremely small; holds only a few bits or words of data, like specific numbers currently being processed.

Role

Not standard "memory" in the storage sense; they function as the active working tools of the processor.

Cache High-Speed Buffer Memory

Cache Architecture & Levels

Generally, 1 to 3 levels of cache memory are installed (on-chip or off-chip), balancing access speed and size depending on cell technology. Use cat /proc/cpuinfo to inspect system cache details.

Principle of Locality

Programs tend to reuse recently accessed data and instructions. Cache performance is maximized when code effectively exploits both **spatial** and **temporal** locality.

The 90 / 10 Rule

A fundamental rule of thumb in computer execution:

90% of Execution Time

is spent executing only **10%** of the code instructions.

Cache Principle of Locality

Temporal Locality

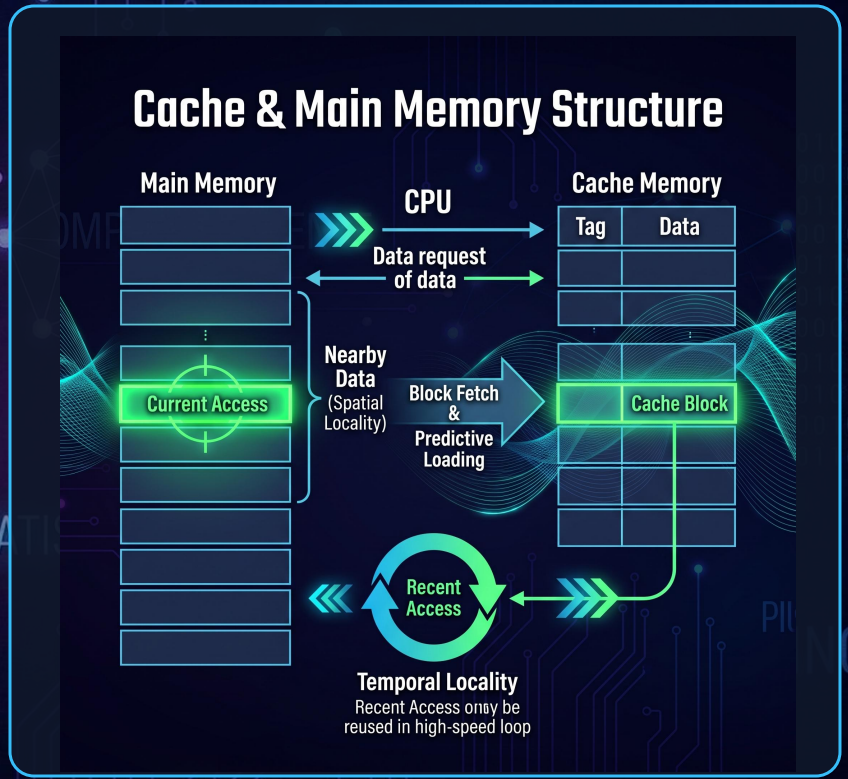
"If you used it recently, you'll likely use it again soon."

Impact: Keeps recently accessed data ready for quick reuse.
Example: A counter variable inside a loop.

Spatial Locality

"If you used this address, you'll likely use its neighbors."

Impact: Pulls in the entire "block" or "line" of surrounding memory when fetching data.



Central Memory

Primary Storage Overview

CPU Interface

Directly interfaced with the CPU and connected to the computer's motherboard.

Allows for a continuous flow of data to and from the CPU.

High Access Speed

Characterized by **extremely fast data access speed** to minimize CPU wait times.

Primary Storage

Also called **Primary Storage**.

Can be read-only like **ROM** or read-write like **RAM**.

Central Memory

Random Access Memory (RAM)

Memory Cells & Addressing

Divided into distinct memory cells, each assigned a unique address to read and write data.

Instructions & Data Storage

Holds execution instructions (opcodes) alongside the active data upon which those instructions operate.

Volatile

Temporary storage. All content is lost immediately when power is turned off.

High Speed

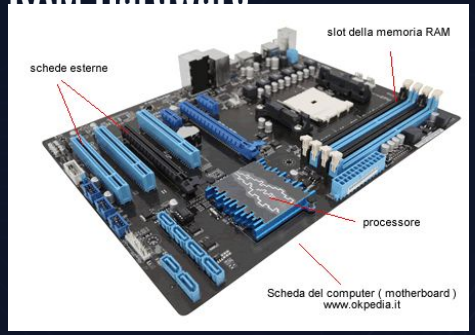
~100 clock cycles access time. Extremely fast performance, making it costly.

Small Capacity

Relatively small size compared to mass storage, typically on the order of a few GiB.

Central Memory RAM Technologies

RAM Hardware



RAM modules connect directly to motherboard slots to provide high-speed CPU access.

DRAM (Dynamic RAM)

Most commonly used technology nowadays.

Each bit is stored in a capacitor (**bit 1 or 0 depends on charge**).

Must be **periodically recharged** to prevent loss of charge and information.

Available in several technologically different variants.

SRAM (Static RAM)

Doesn't require continuous refresh.

Retains information for very long periods of time.

Offers **low power consumption** and **short access times**.

Expensive to build.

Generally used for **cache memory**.

Central Memory
Read Only Memory (ROM)

Permanent Storage (ROM)

Read-only memory offering higher access speed than mass storage. Stores data permanently without requiring continuous power.

Startup & Firmware

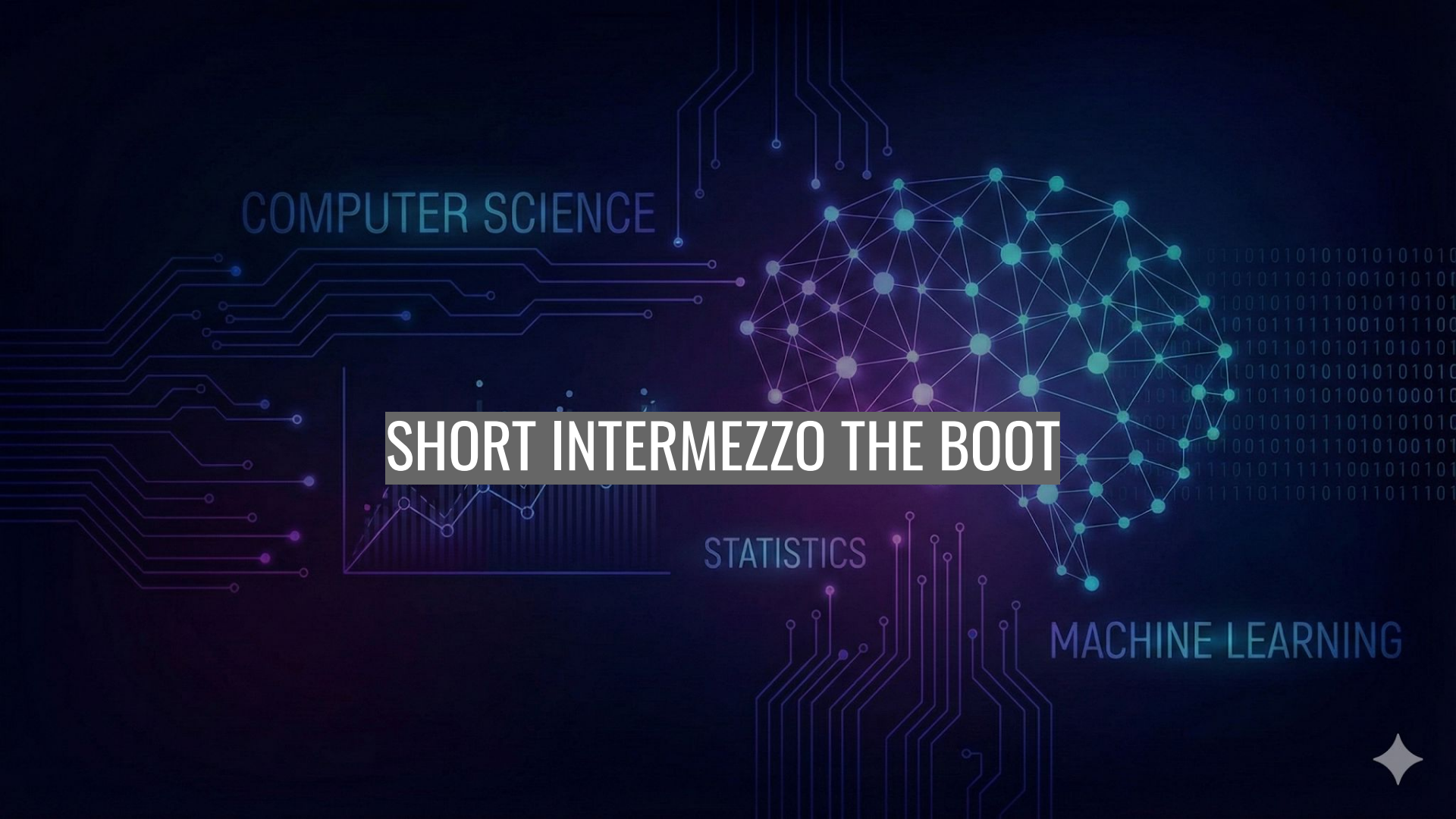
Holds code and data necessary for the computer startup procedure, system initialization, and core hardware routines (Firmware).

BIOS & UEFI Systems

Stores the BIOS (Basic Input/Output System). In modern computer systems, legacy BIOS is replaced by UEFI (Unified Extensible Firmware Interface).

EPROM Technology

Erasable Programmable Read Only Memory. Unlike standard ROM, EPROMs can be erased and rewritten a generally limited number of times.



SHORT INTERMEZZO THE BOOT

BOOTUP Process Stage 1: Power-On

STEP 01

Power-On

The initial stage of the boot process. Typically initiated directly by user interaction with the system.

FEATURE

Wake on LAN

Allows powering on over the network remotely, removing the requirement for physical presence or manual user action.

EXECUTION

ROM Code Execution

As soon as the CPU receives power, it immediately executes the code located in ROM on the motherboard.

BOOTUP Process Stage 2: POST & BIOS/UEFI

STEP 02

POST (Power-On Self Test)

Runs hardware diagnostic checks to verify that memory, hard drives, and system hardware are operational.

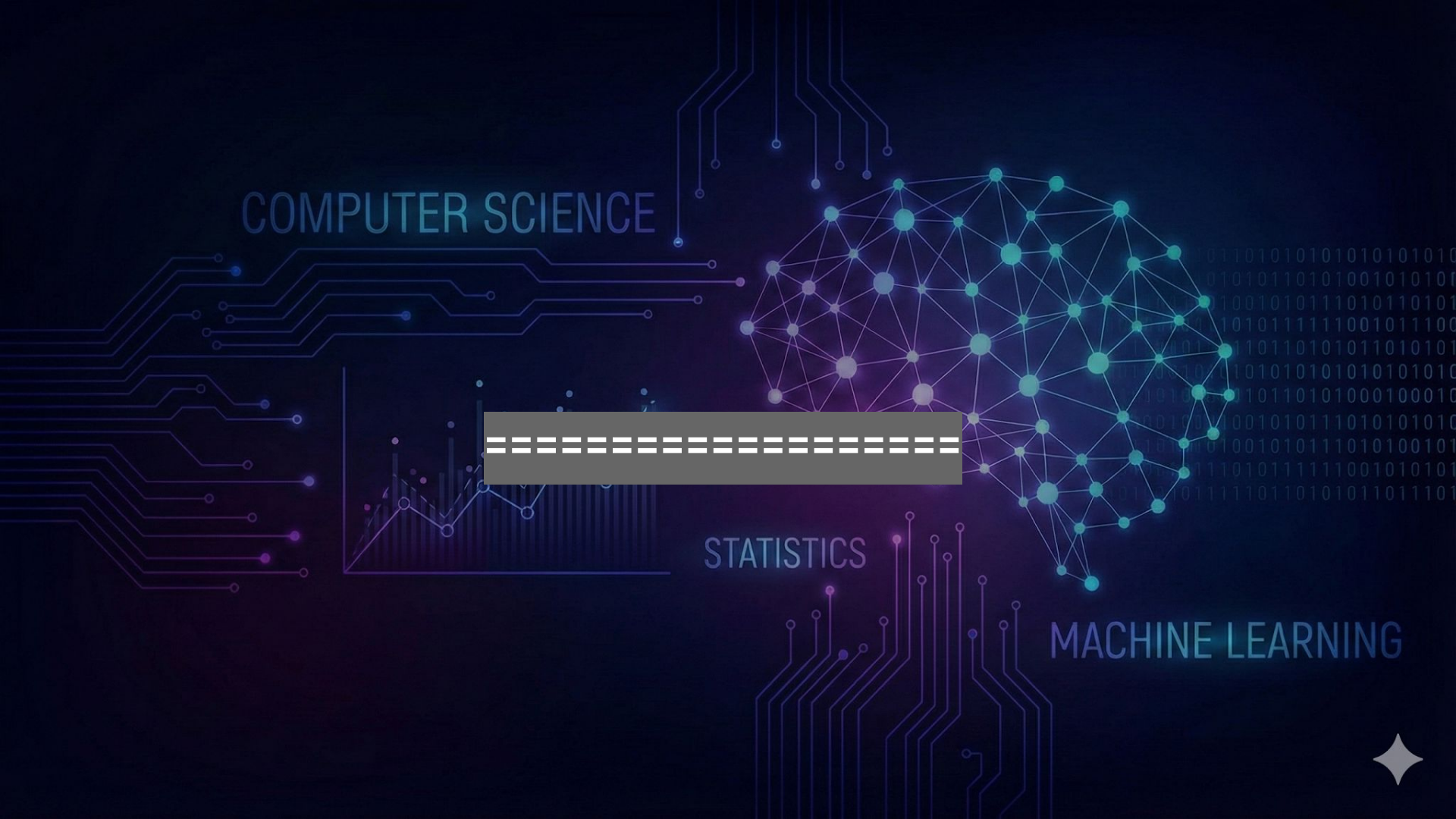
Once POST completes successfully, the system queries for the first device in the boot order list.

STEP 03

BIOS / UEFI Initialization

Searches through the boot device list to locate an active, bootable drive.

Provides core instructions for basic communication between peripherals and motherboard hardware.


COMPUTER SCIENCE
==========
STATISTICS
MACHINE LEARNING

Mass Storage – Secondary Storage

ACCESSIBILITY

Controller Communication

Unlike main memory, secondary storage is not directly accessible from the CPU, but communicates via a controller (I/O bus).

DATA FLOW

Main Memory Buffer

Data is transferred from secondary memory to main memory and read from there directly by the CPU.

HARDWARE

Physical Connection

Connected to the motherboard with high-speed cables or via external interface connections.



STORAGE MEDIA

Technology Types

Secondary storage devices can be made with magnetic, optical, or solid-state technology.

Mass Storage – Key Characteristics

MEDIA TYPES: Consisting of Hard Drives, CDs, DVDs, Solid State Drives, Tapes

PERSISTENCE

Non-volatile

Data remains stored even when the computer restarts or loses power.

PERFORMANCE

Slow Compared to RAM

Takes > 10,000 cycles; latency varies significantly by media type.

ECONOMICS

Averagely Cheaper

Much lower cost per byte compared to main DRAM or SRAM memory.

CAPACITY

Large Storage Scale

Provides vast storage capacity, currently ranging from hundreds of GiB to several TiB per drive.

ACCESS MODES

Read-Only or Read-Write

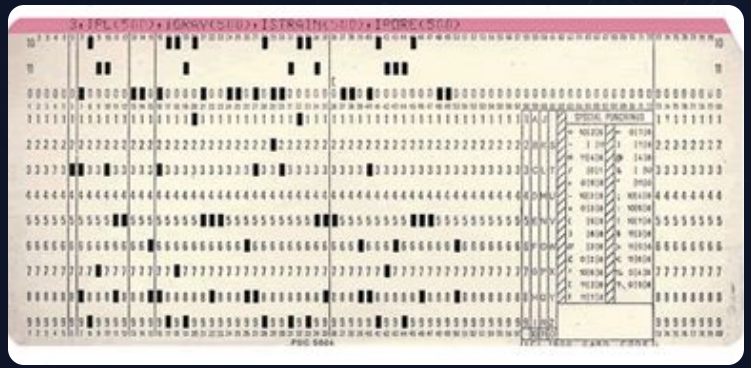
Supports diverse access operational modes, including immutable read-only media (DVD-ROM) and full read-write drives (HDDs/SSDs).

Mass Storage – The Early Days

FIRST SECONDARY MEMORY

Punched Cards

Punched cards are the first secondary memory in the history of computers. Data is stored on cardboard cards and recorded by punching holes, and then read by the computer.



CONTINUOUS MEDIA

Punched Tapes

Punched tapes are similar to punched cards in type, using long continuous strips of paper to record and read data sequentially through punched holes.



Mass Storage – Magnetic Tape

HISTORICAL USE

1970s & 1980s Mainframes

Data is stored on tape in magnetic form. Historically used in mainframes—large computers capable of performing very fast processing and storing large amounts of data.



MODERN APPLICATION

Data Backup & Archival

Despite being an early storage technology, magnetic tape cartridges are actively used today for secure, high-capacity offline data backup.



Mass Storage – Hard Disk

ARCHITECTURE & STRUCTURE

Magnetic Platters

Composed of several magnetic plates placed one on top of the other and rotating. Data is stored on both sides of individual platters and organized into tracks and sectors.

OPERATION & SPEED

Heads & Rotation (RPM)

Read/write heads access data across platter surfaces. The rotation speed of the platters (rpm) is one of the primary determining factors in reading and writing speed.

SOLID STATE EVOLUTION

Solid State Drives (SSD)

SSD solid state drives achieve significantly lower latencies and are less susceptible to hardware failures and mechanical shocks due to having no moving mechanical parts.

Mass Storage – Removable

OPTICAL STORAGE

CDs & DVDs

Data is stored and read using laser light (reflection or non-reflection identifies bits as 1 or 0). Available in both read-only and read-write formats.

FLASH MEMORY

USB Flash Drives

Highly portable solid-state storage. Can be easily removed and separated from the host computer.

LEGACY MEDIA

Floppy Disks

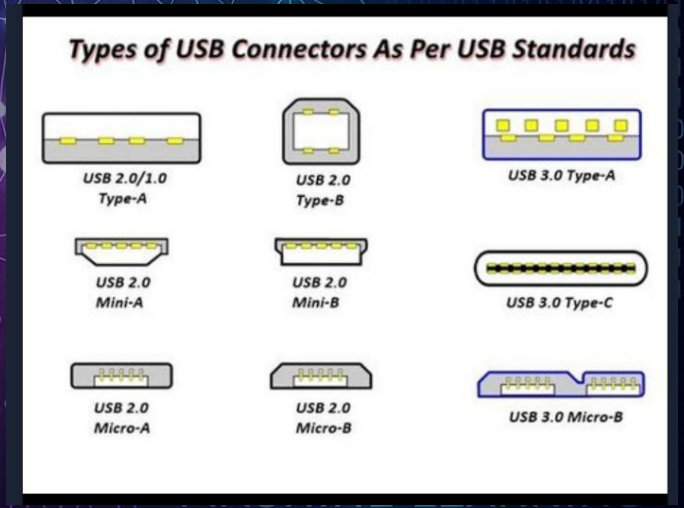
Early magnetic removable storage technology. Completely obsolete and no longer used.

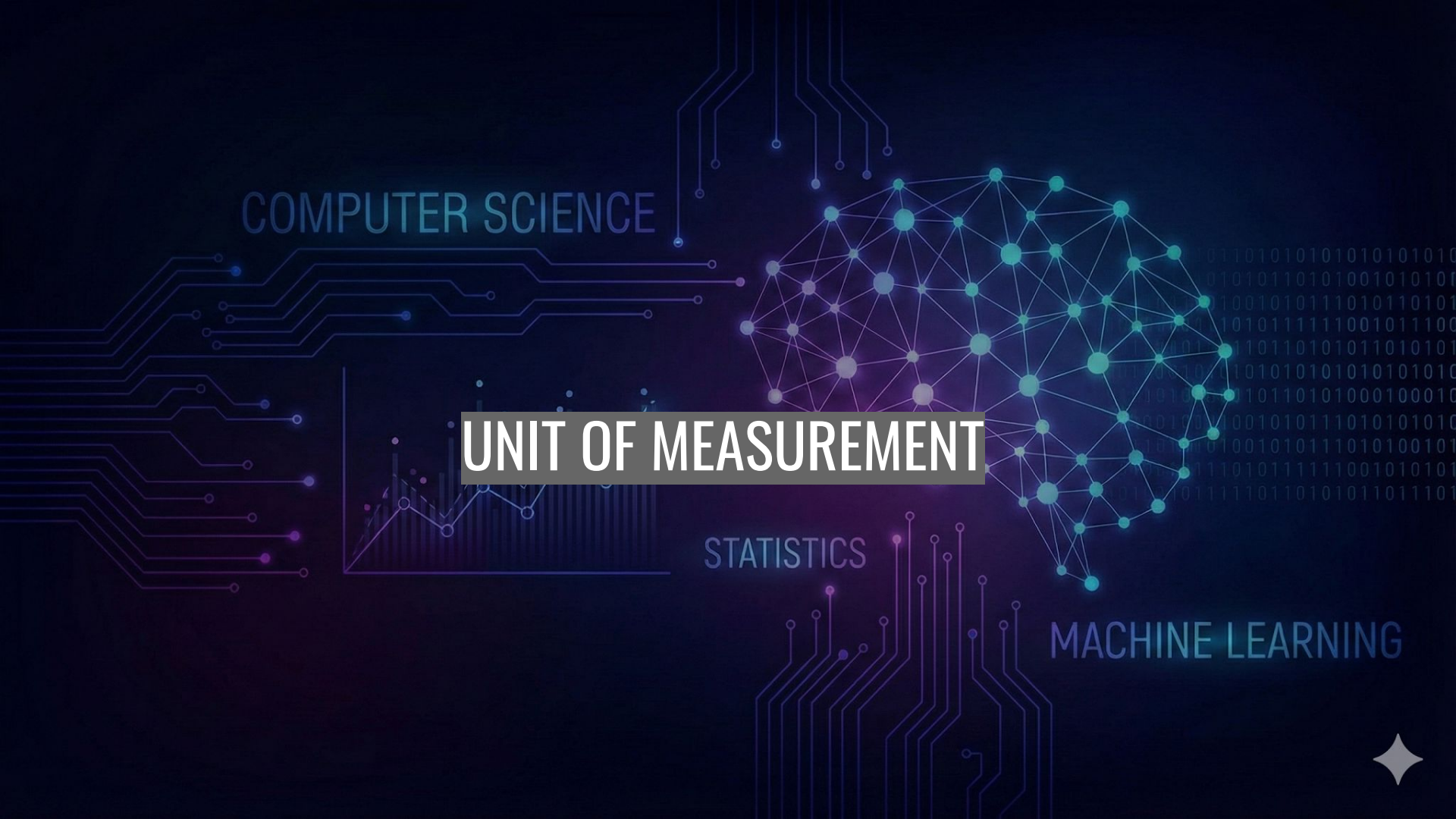


USB

GENERATION	PORT COLOR	MAX SPEED	MAX POWER
USB 2.0 High Speed	● Black / White	480 Mbps	2.5 W (500mA)
USB 3.0 SuperSpeed	● Blue	5 Gbps	4.5 W (900mA)
USB 3.1 / Type-C SuperSpeed+ / PD	● Teal / Any	10 Gbps+	100 W (Power Delivery)

Backwards Compatibility: USB 3.0 ports (Blue) can accept USB 2.0 devices, but speeds will drop to 480 Mbps.




COMPUTER SCIENCE
UNIT OF MEASUREMENT
STATISTICS
MACHINE LEARNING

Unit of Measurement of Information

BIT

The unit of measurement of information (from **"binary digit"**), defined as *the minimum quantity of information needed to distinguish between two equally likely events.*

BYTE = 8 BITS

Historically, characters were represented by 8 BITS, which is why **1 Byte remains the minimum addressable memory unit to this day**.

KiB

KiloByte
better "kibibyte"
2^{10}
= 1,024 Bytes

MiB

MegaByte
better "mebibyte"
2^{20}
= 1024 * 1024 Bytes

GiB

GigaByte
better "gibibyte"
2^{30}
= 1024 * 1024 * 1024 Bytes

TiB

TeraByte
better "tebibyte"
2^{40}
= 1024 * 1024 * 1024 * 1024 Bytes

Computer Performance

Core Objective: Increasing computer performance requires **decreasing execution time** needed to complete an operation.

EXECUTION TIME FORMULA

$$T_{execution} = T_{clock} \sum_{i=0}^{n} N_i \cdot CPI_i$$

T clock — **Clock Period**

Clock period of the machine (**frequency increase** reduces cycle duration).

CPI i — **Cycles Per Instruction**

Number of clock "shots" needed for instruction type i (**reduction of instruction complexity**).

N i — **Instruction Count**

Number of instructions of type i executed (**e.g., sums, jumps, logic operations**).

$$T_{execution} = T_{clock} * \sum(N_i * CPI_i)$$

Example: Calculating Execution Time

Given Variables:

- $T_{clock} = 0.5$ ns

Instruction Counts (N_i) and Cycles Per Instruction (CPI_i):

- Additions: $N_1 = 100M$, $CPI_1 = 1$
- Jumps: $N_2 = 20M$, $CPI_2 = 2$
- Loads/Stores: $N_3 = 50M$, $CPI_3 = 4$

$$T_{execution} = 0.5 \text{ ns} * [(100M * 1) + (20M * 2) + (50M * 4)]$$

$$T_{execution} = 0.5 \text{ ns} * [100M + 40M + 200M]$$

$$T_{execution} = 0.5 \text{ ns} * 340{,}000{,}000 \text{ cycles}$$

$$T_{execution} = 170{,}000{,}000 \text{ ns} = 0.17 \text{ seconds}$$

MIPS (Mega Instructions Per Second)

MIPS is an abbreviation for **Mega Instructions Per Second**, indicating the number of general-purpose instructions a CPU executes in one second.

MEASUREMENT & FORMULA

It serves as a general-purpose unit for measuring computer performance, comparing instruction throughput across architectures.

In comparison, FLOPS measures floating-point performance.

CALCULATION

$$\text{MIPS} = \text{Clock Frequency} / (10^6 \times \text{CPI})$$

CRITICAL LIMITATIONS

1. Ignores Cache Memory

Does not account for speed optimizations resulting from high-speed CPU cache systems.

2. Instruction Mix Variation

Overlooks relative percentages of different instruction types executed within real-world programs.

3. Architectural Discrepancies

Instruction set complexity varies across processors, making raw instruction count an inconsistent metric.

FLOPS (Floating Point Operations Per Second)

FLOPS is an abbreviation for **Floating Point Operations Per Second**, measuring the number of floating-point operations a CPU performs in one second. It is the standard performance metric in scientific computing.

DEFINITION & PURPOSE

Core Unit of Measure:
Serves as a fundamental metric to evaluate raw computational throughput, specifically for real-number floating-point calculations.

Scientific Computing Focus:
Highly critical in workloads like linear algebra, physics simulations, weather modeling, and deep learning neural network training.

Empirical Evaluation:
Measured by executing known matrix or vector algorithms and timing execution speed.

PRACTICAL EVALUATION EXAMPLE

Matrix Product Operation

For a classic matrix product with dimension N, exactly $2 \times N^3$ floating-point operations are performed.

FORMULA

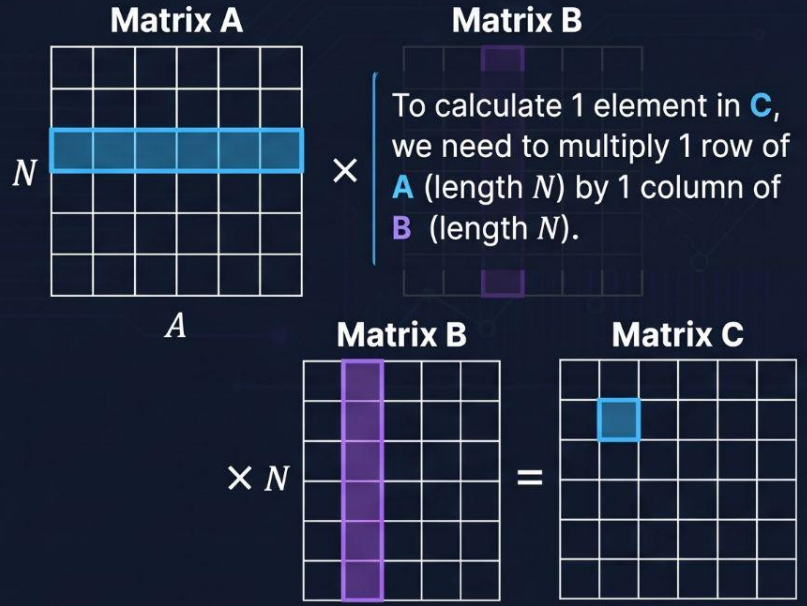
$$[\text{FLOPS}] = 2 \times N^3 \,/\, \text{Time}$$

Evaluate FLOPS directly by measuring time required to complete the multiplication.

MATRIX-MATRIX MULTIPLICATION OPERATIONS

Analyzing the count of **floating-point operations (FLOPs)** required to multiply two $N \times N$ matrices.

VISUALIZING THE OPERATION



COUNTING THE OPERATIONS

Dot Product of Row × Column:

- N multiplications (e.g., $a_{ik} * b_{kj}$)
- $N - 1$ additions (to sum the products)
- Total = $2N - 1 \approx 2N$ operations per element.

For All N^2 Elements in the Result Matrix:

- Total Operations = $N^2 \times (2N - 1)$
- Roughly equivalent to $2N^3$ total operations.

OPERATION COUNT

$$[\text{Total Operations}] \approx 2 \times N^3$$

For large N, the –1 is negligible, giving $2N^3$ floating-point operations.

SPEC (Standard Performance Evaluation Corporation)

SPEC is a non-profit organization that produces and maintains standardized computer benchmarks representative of real-world applications.

CORE PURPOSE & OBJECTIVE

Standardized Benchmarks:
Develops objective test programs that accurately reflect real-world computing workloads and applications.

Non-Profit Organization:
Ensures impartial, vendor-neutral performance measurement standards across the computer industry.

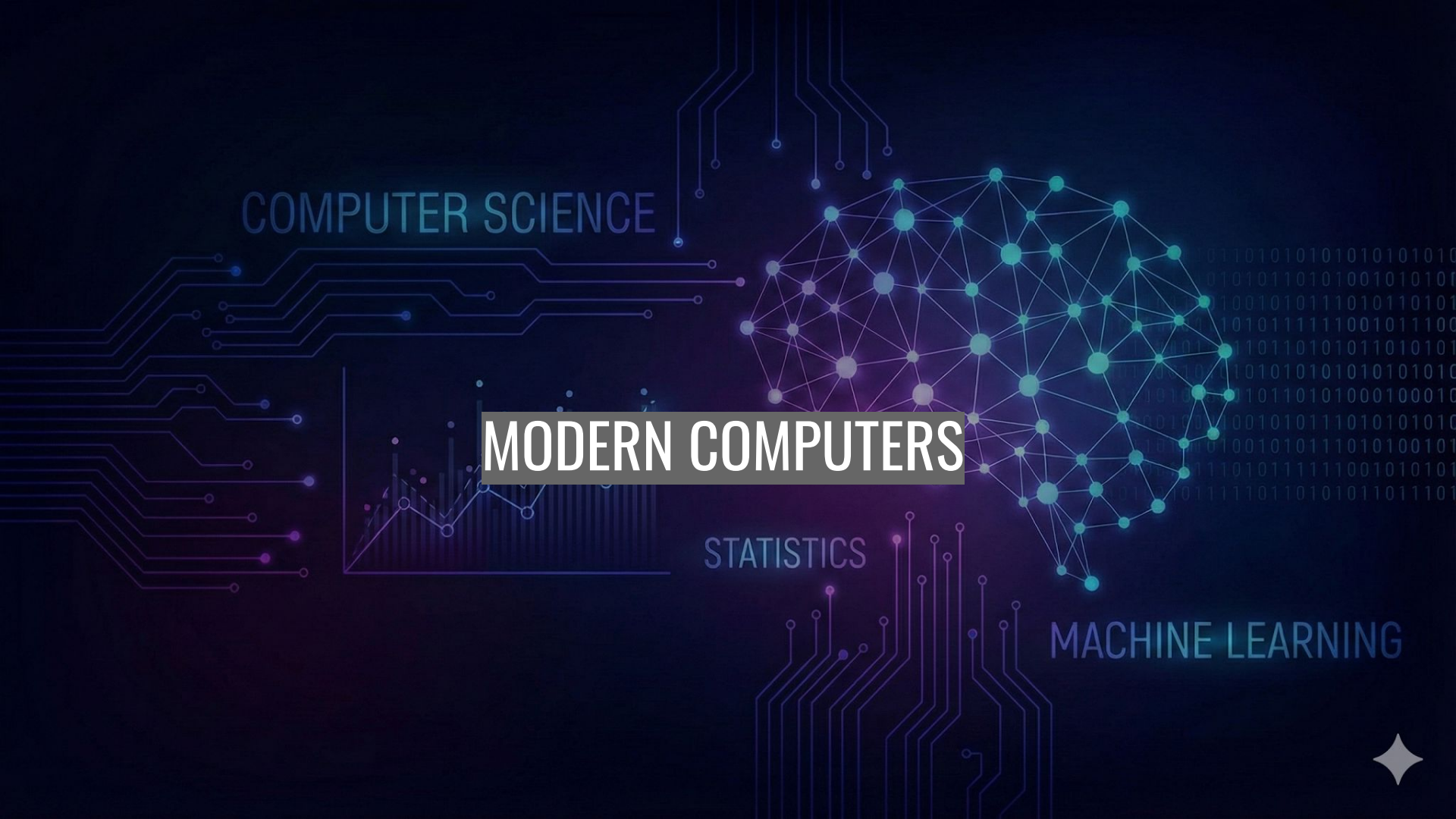
Performance Measurement:
Allows meaningful system comparisons by maintaining rigorous testing methodologies.

BENCHMARK SUITES & APPLICATIONS

Targeted Evaluations:
Maintains distinct benchmark sets specifically designed for various intended uses of computers.

Diverse Domains:
Covers specialized domains including CPU compute, graphics, high-performance computing (HPC), and cloud infrastructure.

Real-World Relevance:
Continually updates test suites to match evolving software demands and hardware architectures.


COMPUTER SCIENCE
MODERN COMPUTERS
STATISTICS
MACHINE LEARNING

FLOPs

Prefix	Scale	Notation	Typical Device
GigaFLOPS	Billion	10^{9}	Standard Laptop
TeraFLOPS	Trillion	10^{12}	PS5 / Gaming PC
PetaFLOPS	Quadrillion	10^{15}	Supercomputers (2010s)
ExaFLOPS	Quintillion	10^{18}	Frontier (Current)
ZettaFLOPS	Sextillion	10^{21}	Future AI Clusters

The Rigid Specialist

Finite State Machine

An **FSM** is a computational model that exists in exactly **one state** at any given time.

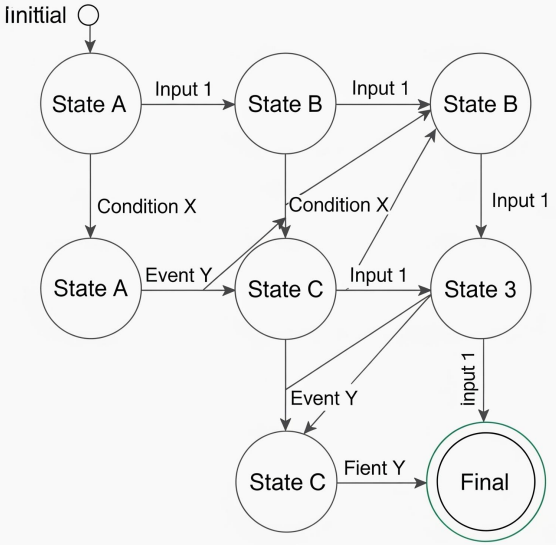
Logic Engine
State transitions are strictly triggered by specific inputs or conditions.

Storage Mechanism
Does not store a program in memory; logic is fixed directly into the circuit structure.

Real-World Example: Vending Machine

- Input: Insert Coins → State: Credit → Action: Dispense Soda

Bound by fixed rules; it cannot "learn" complex tasks like playing chess.



The Flexible Generalist

Von Neumann Machine

The **Von Neumann architecture** describes a computer where program instructions and data are held in the same memory storage.

Stored Program

The machine can change its behavior by loading a different software program into memory.

Execution Cycle

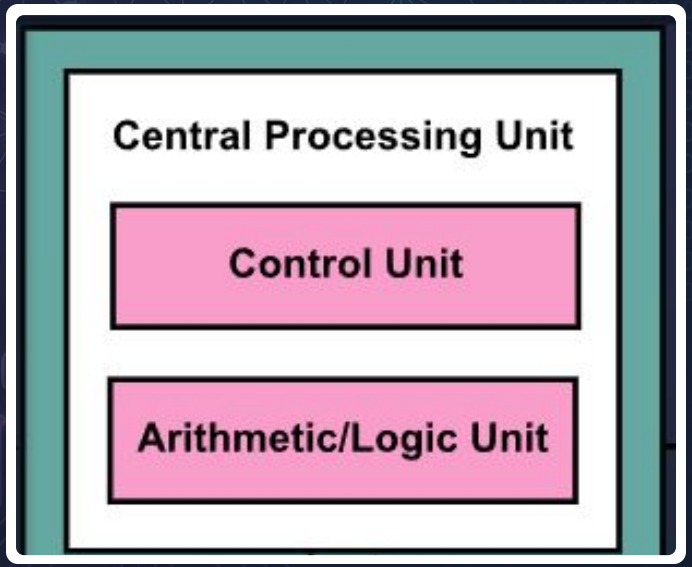
Uses a Central Processing Unit (CPU) to endlessly Fetch, Decode, and Execute instructions.

Flexibility & Versatility

Universal Hardware Capability

The same hardware can calculate taxes, play video games, or browse the web.

Architecture Diagram



Radio-Frequency Identification

RFID Systems

Architecture & Processing

Most tags are **Finite State Machines (FSM)** costing cents (~few MIPS). Smart tags (passports, credit cards) act as full computers with **Von Neumann / Harvard** architecture.

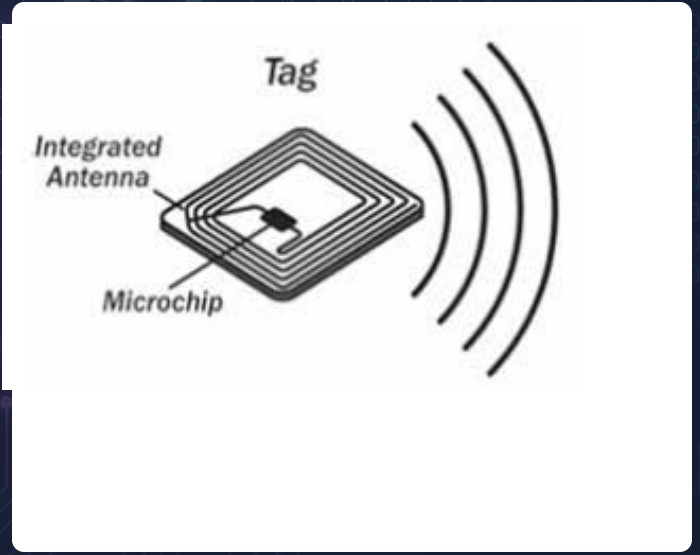
Identification Mechanism

Uses **electromagnetic fields** to automatically identify and track objects via embedded IDs.

Power Supply Categories

- **Passive:** Powered directly by energy emitted from the RFID reader (or battery-assisted).
- **Active:** Battery-powered; periodically broadcasts an autonomous signal.

RFID Tag Components



Hardware Spectrum

Embedded Systems & Mobile Devices

Low-Power & Dedicated

Embedded Systems

- **Applications:** Embedded calculators for watches, cars, household appliances, medical devices, and Android Boxes.
- **Cost & Scale:** Cost only a few dozen euros while delivering performance in the order of a few hundred MIPS.
- **Operating System:** Frequently equipped with a lightweight Linux OS.

High-Performance Portable

Smartphones & Tablets

- **Computing Power:** Systems offering significantly higher processing capability than standard embedded units.
- **Advanced Silicon:** Modern high-end flagships (e.g., iPhone 15 Pro, Galaxy S24) feature dedicated Neural Engines and GPUs.
- **Performance Scale:** Operates in the Teraflops (TFLOPS) range, going far beyond standard GFLOPS.

Hardware Spectrum

Game Consoles, PCs & Workstations

Consumer Gaming

Game Consoles

- **Overall Performance:** Reaches around **10 to 12 TFLOPS** (10,000 – 12,000 GFLOPS) considering GPUs.
- **Architecture:** Dedicated fixed-hardware systems optimized for gaming workloads.

Desktops & Laptops

Personal Computers

- **Wide Range:** Covers possibilities from a few hundred up to several thousand euros.
- **GPU Scalability:** Performance spans from hundreds up to **80,000 GFLOPS**.

Enterprise & HPC

Servers & Workstations

Designed for **High Performance Computing** and service centralization.

At €20,000 Budget Tier:

- **Workstation:** **100+ TFLOPS** for Graphics / AI.
- **Server:** **10–20 TFLOPS** for high-precision scientific simulation.

Hardware Spectrum

High Performance Computing (HPC)

Core Concept

Parallel Computing

- **Interconnection:** Coupling numerous computers with high-performance networks.
- **Objective:** Multiplies computing power beyond single-system hardware limits.

Scale & Architectures

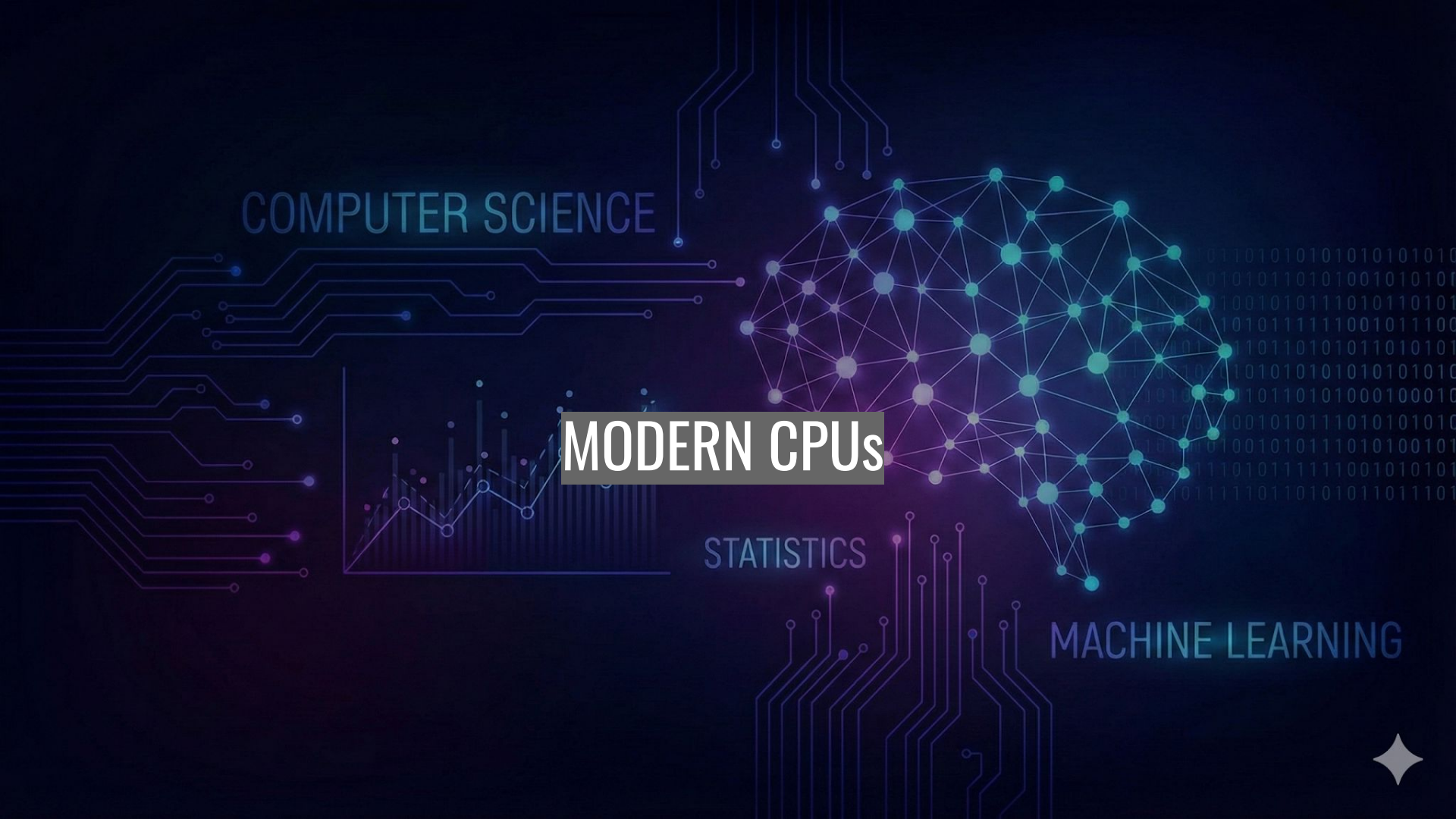
Clusters & Supercomputers

- **Workstation Clusters:** Distributed nodes working in tandem for compute tasks.
- **Exascale Frontier:** Top supercomputers (like Frontier) surpassed **1000 PFLOPS** (1 Exaflop).

Infrastructure & Overhead

Costs & Energy Scale

- **Proportional Scaling:** Production and maintenance costs increase in tandem with performance.
- **Power Consumption:** Extreme energy demand and specialized thermal management required.


COMPUTER SCIENCE
MODERN CPUs
STATISTICS
MACHINE LEARNING

	Prefix	Scale	Notation	Typical Device
	GigaFLOPS	Billion	10^{9}	 Standard Laptop
	TeraFLOPS	Trillion	10^{12}	 PS5 / Gaming PC
	PetaFLOPS	Quadrillion	10^{15}	 Supercomputers (2010s)
	ExaFLOPS	Quintillion	10^{18}	 Frontier (Current)
	ZettaFLOPS	Sextillion	$10^{\wedge 21}$	 Future AI Clusters

Processor Architecture

CISC vs RISC — Instruction execution speed is a core factor in CPU performance

Complex Instruction Set

CISC Architecture

- **Rich Instruction Set:** Designed with a comprehensive and feature-dense basic instruction set.
- **Multiple Clock Cycles:** Each instruction requires more clock cycles to fully execute.
- **Historical Impact:** Predominant architecture across market hardware in the 1970s and 1980s.

Reduced Instruction Set

RISC Architecture

- **Single Clock Cycle:** Streamlined execution where each instruction completes in a single cycle.
- **Instruction Volume:** Requires a higher total count of individual instructions to complete operations.
- **Modern Standard:** **Strong industry trend moving towards favor of RISC-type processors.**

Optimization Strategies

Strategies to Improve CPU Performance — A continuous trade-off between speed, cost, and power consumption

STRATEGY 01

Reduce Cycles per Instruction

- **Cycle Optimization:** Minimize the cycles required for individual operations.
- **Example:** Optimizing multi-cycle operations like hardware multiplication.

STRATEGY 02

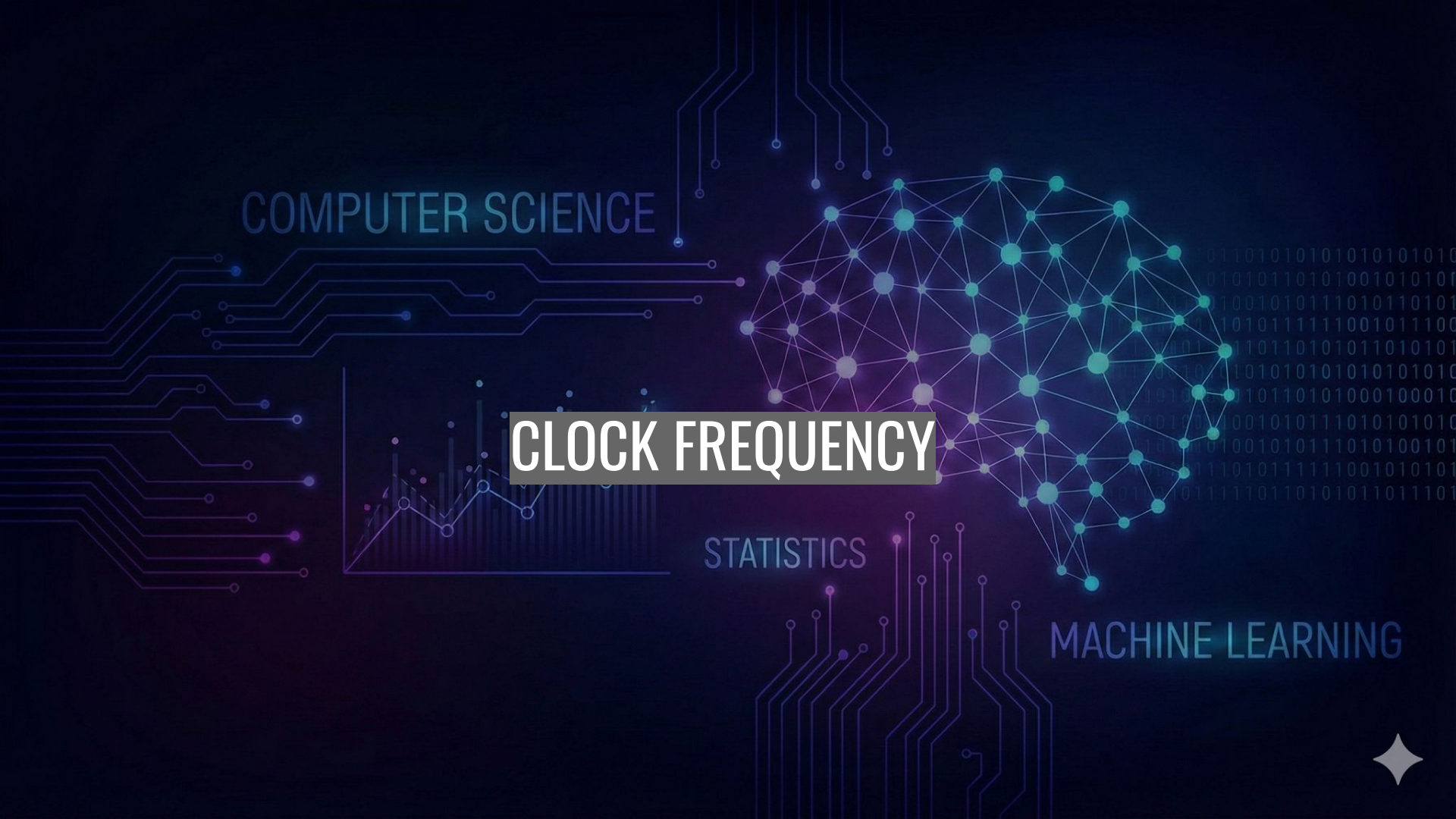
Increase Clock Frequency

- **Clock Scaling:** Speed up execution rates by raising clock frequency.
- **Physical Constraints:** Bounded by thermal limits and the speed of light.

STRATEGY 03

Parallel Execution

- **Instruction-Level:** Pipelines & superscalar designs to execute concurrent instructions.
- **Core-Level:** Scaling performance across multiple physical cores per CPU.


COMPUTER SCIENCE
CLOCK FREQUENCY
STATISTICS
MACHINE LEARNING

Clock Frequency Scaling

Physical Limits of Clock Speed — Why scaling CPU performance through clock frequency reached a wall

HISTORICAL TREND & LIMITS

The 4 GHz Physical Wall

- **Historical Alignment:** Until 2000, CPU gains scaled directly with clock frequency increases.
- **Frequency Ceiling:** Performance plateaued around ~4 GHz due to fundamental physical boundaries.
- **Speed of Light Limit:** At 1 GHz, an electrical impulse travels ~33 cm in 1 ns in vacuum—and much slower through copper interconnects.

PHYSICAL CONSTRAINTS

Key Bottlenecks

- **Heat & Thermal Dissipation:** High switching frequencies create substantial electrical noise and excessive heat to dissipate.
- **Signal Propagation Delays:** Finite travel speed causes latency issues across physical chip distances.
- **Bus Skew:** Signals traveling on different parallel lines move at varying speeds, risking synchronization errors.

Clock Frequency & Miniaturization

The Speed of Light Limit — Why signal propagation delay forces processor miniaturization

PHYSICAL BOUNDARY

Signal Propagation Delay

- **Finite Speed:** Inside copper or gold interconnects, electrical signals travel ~15 to 20 cm per nanosecond.
- **Signal Delay:** Electricity travels extremely fast, but it is not instantaneous across physical distances.
- **The Solution: To increase clock speed, components must be packed tightly together to shorten signal paths.**

REAL-WORLD SCENARIO

The 4 GHz CPU Limit

- **Cycle Timing:** A CPU running at 4 GHz has a cycle time of only 0.25 nanoseconds.
- **Travel Distance:** In 0.25 ns, a signal can physically travel only about 4 to 5 centimeters.
- **Scale Conflict:** If a chip were dinner-plate sized, clock cycles would end before signals reached the other side.

Transistor Capacitance & Miniaturization

The "Bucket" Problem — How miniaturization overcomes physical capacitance limits

Capacitance Analogy

Transistor as a Water Bucket

- **Capacitor Behavior:** A transistor acts like a tiny capacitor—think of it as a bucket of water.
- **Switching On ("1"):** To switch a transistor from "0" to "1" (Off to On), you have to fill that bucket with electrons.
- **Switching Off ("0"):** To switch back to "0", you have to empty it completely.

Scale & Frequency Impact

Large vs. Small Transistors

- **Large Transistors = Large Buckets:** Takes longer to fill. High frequency switching leaves them partially filled or empty, causing data corruption.
- **Small Transistors = Small Buckets:** You can fill and empty a thimble much faster than a bathtub.
- **The Solution:** **Miniaturization makes gate capacitance smaller, enabling switching billions of times per second without lagging.**

Clock Frequency & Miniaturization

Power and Heat (Power Density)

— How miniaturization overcomes CPU power consumption and thermal limits

POWER DYNAMICS

The Power Equation

$$P = C * V^2 * f$$

(Power = Capacitance × Voltage squared × Frequency)

- **Frequency Impact:** If you increase Frequency (f), Power (P) shoots up, generating massive heat.
- **Thermal Wall:** To stop the CPU from melting, you must lower the other variables.

COOLING BY DESIGN

Role of Miniaturization

- **Reduces Capacitance (C):** Miniaturization directly reduces Capacitance (C).
- **Lowers Voltage (V):** Miniaturization allows you to lower the Voltage (V).
- **The Result: By making things smaller, you require less energy to move electrons, compensating for heat generated by increased speed.**

Optimization Strategies

Strategies to Improve CPU Performance — A continuous trade-off between speed, cost, and power consumption

STRATEGY 01

Reduce Cycles per Instruction

- **Cycle Optimization:** Minimize the cycles required for individual operations.
- **Example:** Optimizing multi-cycle operations like hardware multiplication.

STRATEGY 02

Increase Clock Frequency

- **Clock Scaling:** Speed up execution rates by raising clock frequency.
- **Physical Constraints:** Bounded by thermal limits and the speed of light.

STRATEGY 03

Parallel Execution

- **Instruction-Level:** Pipelines & superscalar designs to execute concurrent instructions.
- **Core-Level:** Scaling performance across multiple physical cores per CPU.

Clock Frequency Scaling

Physical Limits of Clock Speed — Why scaling CPU performance through clock frequency reached a wall

HISTORICAL TREND & LIMITS

The 4 GHz Physical Wall

- **Historical Alignment:** Until 2000, CPU gains scaled directly with clock frequency increases.
- **Frequency Ceiling:** Performance plateaued around ~4 GHz due to fundamental physical boundaries.
- **Speed of Light Limit:** At 1 GHz, an electrical impulse travels ~33 cm in 1 ns in vacuum—and much slower through copper interconnects.

PHYSICAL CONSTRAINTS

Key Bottlenecks

- **Heat & Thermal Dissipation:** High switching frequencies create substantial electrical noise and excessive heat to dissipate.
- **Signal Propagation Delays:** Finite travel speed causes latency issues across physical chip distances.
- **Bus Skew:** Signals traveling on different parallel lines move at varying speeds, risking synchronization errors.

Clock Frequency & Miniaturization

The Speed of Light Limit — Why signal propagation delay forces processor miniaturization

PHYSICAL BOUNDARY

Signal Propagation Delay

- **Finite Speed:** Inside copper or gold interconnects, electrical signals travel ~15 to 20 cm per nanosecond.
- **Signal Delay:** Electricity travels extremely fast, but it is not instantaneous across physical distances.
- **The Solution: To increase clock speed, components must be packed tightly together to shorten signal paths.**

REAL-WORLD SCENARIO

The 4 GHz CPU Limit

- **Cycle Timing:** A CPU running at 4 GHz has a cycle time of only 0.25 nanoseconds.
- **Travel Distance:** In 0.25 ns, a signal can physically travel only about 4 to 5 centimeters.
- **Scale Conflict:** If a chip were dinner-plate sized, clock cycles would end before signals reached the other side.

Transistor Capacitance & Miniaturization

The "Bucket" Problem — How miniaturization overcomes physical capacitance limits

Capacitance Analogy

Transistor as a Water Bucket

- **Capacitor Behavior:** A transistor acts like a tiny capacitor—think of it as a bucket of water.
- **Switching On ("1"):** To switch a transistor from "0" to "1" (Off to On), you have to fill that bucket with electrons.
- **Switching Off ("0"):** To switch back to "0", you have to empty it completely.

Scale & Frequency Impact

Large vs. Small Transistors

- **Large Transistors = Large Buckets:** Takes longer to fill. High frequency switching leaves them partially filled or empty, causing data corruption.
- **Small Transistors = Small Buckets:** You can fill and empty a thimble much faster than a bathtub.
- **The Solution:** **Miniaturization makes gate capacitance smaller, enabling switching billions of times per second without lagging.**

The Limits of Miniaturization

Why atomic scale and quantum mechanics restrict further transistor shrinking

ATOMIC & MOLECULAR SCALE

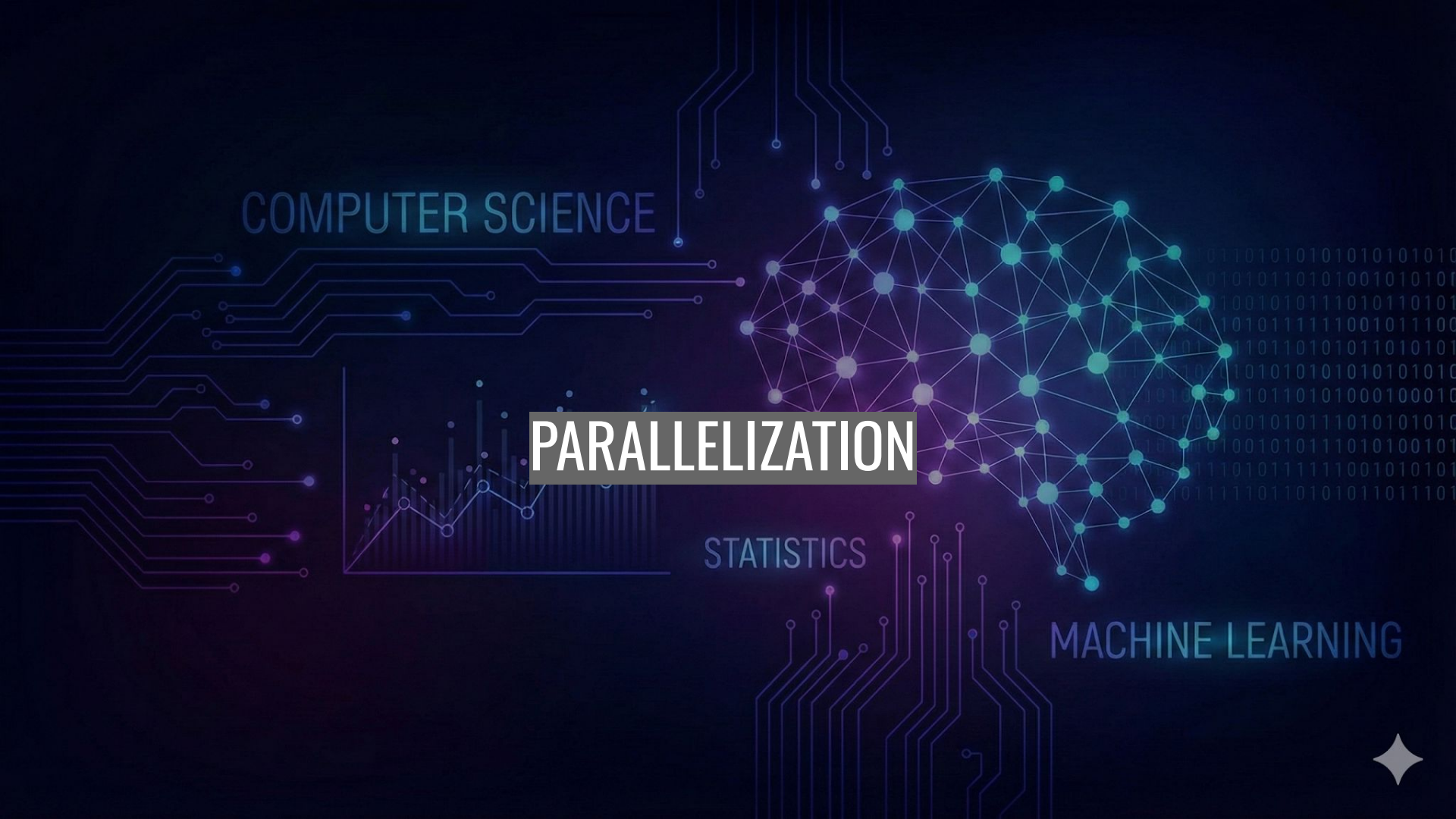
The Physical Size Limit

- **Discrete Matter:** Chips are physically built from individual atoms and molecules.
- **Few-Atom Gates:** Modern transistor features are only a few silicon atoms wide (~0.2 nm per atom).
- **Structural Wall:** We cannot shrink components smaller than the atoms they are made of.

QUANTUM EFFECTS

Quantum Tunneling

- **Barrier Breakdown:** At atomic scales, classical physics gives way to quantum mechanics.
- **Electron Tunneling:** Electrons pass directly through ultra-thin insulating barriers even when switched off.
- **Leakage & Errors:** Uncontrolled current leakage causes excess heat and loss of reliable 0 and 1 states.

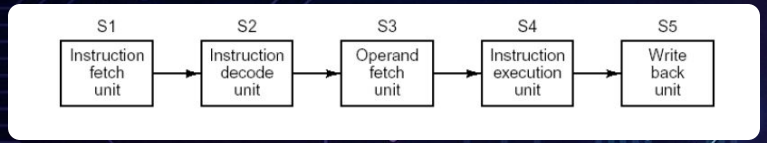

COMPUTER SCIENCE
PARALLELIZATION
STATISTICS
MACHINE LEARNING

PARALLELIZATION & CPU ARCHITECTURE

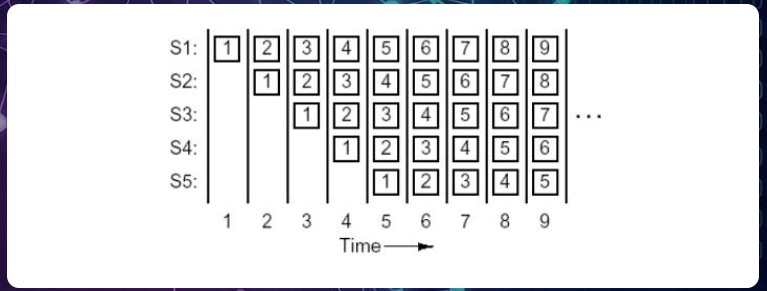
Improving Performance – Pipeline

Dividing instruction execution into dedicated hardware stages to achieve high throughput.

Each single instruction is divided into multiple stages (or phases) and each phase is handled by a dedicated piece of CPU (hardware):



- **Dedicated Units:** Each phase is handled by distinct CPU hardware modules operating simultaneously.
- **Sequential Flow:** S1 Fetch → S2 Decode → S3 Operand Fetch → S4 Execute → S5 Write Back.
- **Efficiency:** Hardware units remain continuously active across consecutive clock cycles.



- **Latency & Throughput:** After loading the pipeline, **n** operations run concurrently.
- **Independence Rule:** Instructions must be independent to avoid execution stalls.
- **Multi-Pipeline:** Multiple pipelines allow fetching & executing multiple instructions at once.

Clearly, **the operations must be independent**. In this case, after an initial time (**latency**) to load the pipeline, there will be n operations executed in parallel. Multiple pipelines can also be used, and therefore multiple instructions are read at a time and executed.

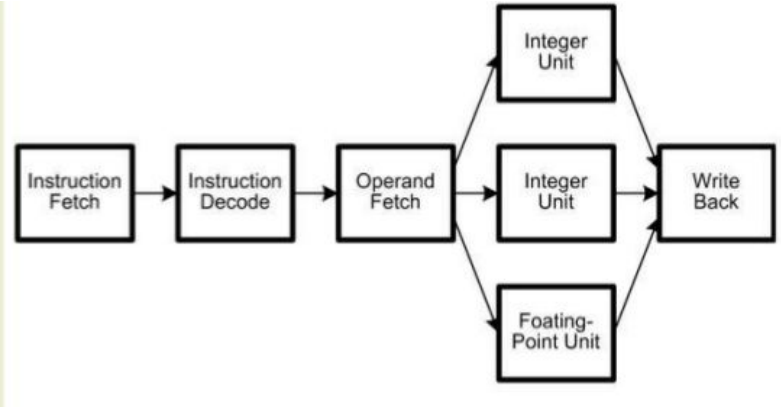
PARALLELIZATION & CPU ARCHITECTURE

Improving Performance – Superscalars

Executing multiple instructions simultaneously through duplicated execution hardware.

Different instructions process operands simultaneously on separate hardware units. Modern CPUs feature **multiple functional units of the same type** (e.g., several ALUs).

Superscalar Execution Pipeline



Multiple Execution Units

- **Scalar vs Superscalar:** Unlike scalar processors with a single pipeline completing at best 1 instruction per cycle, superscalar processors execute multiple instructions per cycle.
- **Hardware Duplication:** Integrates parallel functional units, such as multiple Integer ALUs, Floating Point Units (FPUs), and Load/Store units.
- **Concurrent Processing:** Allows independent instructions to execute in parallel across available hardware units during the same clock cycle.

PARALLELIZATION & CPU ARCHITECTURE

Improving Performance – Jump Prediction

Managing pipeline control hazards caused by conditional execution and branching.

Pipelining works particularly well with sequential instructions, but one of the basic constructs of programming is branching instructions, such as **IF...THEN...ELSE decisions**:

The Control Hazard Problem

- **Sequential Assumption:** CPU pipelines operate under the assumption that instructions are fetched sequentially.
- **Branching Disruption:** Conditional jumps change execution path based on runtime evaluation of conditions.
- **Pipeline Stalls:** Fetching the wrong instruction stream requires flushing partially processed stages.

Code Example (High-Level Branch)

```
if (a == b)
  printf("the two numbers are equal");
else
  printf("They are different \n");
```

PARALLELIZATION & CPU ARCHITECTURE

Improving Performance – Jump Prediction

Pre-fetching and predicting branch decisions to prevent pipeline stalls.

Modern CPUs can **(pre-fetch) try to guess** whether the program will jump to maintain instruction flow:

Static Prediction

- **Common Sense Criteria:** Uses fixed rules and assumptions without relying on past execution history.
- **Default Assumption:** For example, assuming that all jumps are executed by default.
- **Low Overhead:** Simple hardware implementation, but lacks dynamic adaptability to changing workloads.

Dynamic Prediction

- **Execution Statistics:** In practice, the CPU maintains a history table based on past execution statistics.
- **Runtime Adaptation:** Dynamically updates branch expectations as code executes over time.
- **High Accuracy:** Significantly minimizes pipeline stalls and misprediction penalties in complex loops.

PARALLELIZATION & CPU ARCHITECTURE

Improving Performance – Out-of-order and Speculative Executions

Overcoming instruction dependencies to maximize pipeline throughput and CPU efficiency.

CPU design is significantly simpler sequentially, but **instruction dependencies** require executing instructions out of strict order or predicting outcomes ahead of time.

Out-of-Order Execution

- **Temporarily Skip:** Modern CPUs can skip blocked instructions to increase performance.
- **Hold Dependencies:** Puts dependent instructions on hold while executing non-dependent ones.
- **Resource Utilization:** Keeps execution units busy rather than waiting sequentially for stalled data.

Speculative Execution

- **Early Execution:** Runs code paths ahead of time before knowing if they are actually needed.
- **Heavy Computation:** Focuses on executing particularly heavy code parts early to save latency.
- **Branch Anticipation:** Works alongside branch prediction to avoid pipeline latency penalties.

PARALLELIZATION & CPU ARCHITECTURE

CPU & GPU – Final Thoughts

Scaling modern performance through multi-core architectures and hardware acceleration.

Modern processors rely on **parallel execution** rather than raw clock speed increases to drive computational performance.

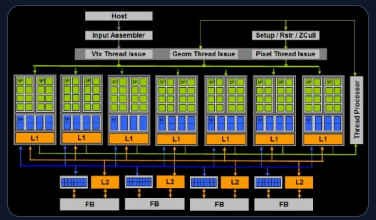
Multi-Core CPUs

- **Independent Processors:** Multiple full processing cores integrated into a single chip.
- **Dedicated Cache:** Each core features its own cache memory and high-speed interconnects.
- **Frequency Independent:** Increases theoretical performance without raising clock frequencies.
- **Hyper-Threading:** Simultaneous multithreading maximizes internal pipeline utilization.

GPU Acceleration

- **Graphics Processors (GPUs):** Increasingly harnessed today to accelerate general-purpose high-performance computing.
- **Massive Parallelism:** Thousands of smaller cores optimized for compute-heavy parallel workloads.





PARALLELIZATION & CPU ARCHITECTURE

CPU - Final Thoughts

Embedded systems, IoT, and Mobile CPUs (including Raspberry Pi)

ARM refers to a prominent family of RISC architectures developed by Arm Holdings. Uniquely, Arm licenses architecture to manufacturers like **Apple, Qualcomm, Samsung, NVIDIA, & Broadcom** rather than manufacturing chips directly.

Architecture & Scaling

- **Energy Efficiency:** Designed to optimize the balance between high performance and low power consumption.
- **SoC Production:** Produced as System-on-Chips tailored for mobile, IoT, and embedded devices like Raspberry Pi.
- **High Scalability:** Scaled beyond mobile devices to power modern laptops, servers, and supercomputers.

Power Consumption Comparison

- **ARM Core (Mobile/IoT):** Typically consumes **100 mW to 500 mW** under normal load (e.g., Cortex-A55).
- **Desktop x86 CPU:** Can consume **over 200 W** at peak performance (e.g., Intel Core i9 / AMD Ryzen 9).
- **Efficiency Delta:** High-performance x86 CPUs draw over 400x to 2000x the power of a low-power ARM core.

INSTRUCTION SET ARCHITECTURE

Why RISC Consumes Less Power Than CISC

Understanding the decoding penalty and architectural efficiency.

The Decoding Penalty (The Main Culprit): Instruction decoding is the single biggest factor causing power variance between CISC and RISC architectures.

CISC (e.g., Intel x86)

- **Complex Instructions:** Instructions act like complex sentences performing multiple tasks at once.
- **Variable Lengths:** Instruction size varies widely, anywhere from 1 byte to 15 bytes long.
- **High Decoding Cost: Requires a massive, power-hungry Decoder circuit to parse instruction boundaries and translate them.**

RISC (e.g., ARM)

- **Simple Commands:** Instructions behave like simple, individual commands for direct execution.
- **Fixed Length:** Uniform instruction length (typically 32 bits) simplifies pipeline fetching.
- **Hardware Advantage:** Uses a tiny, "hardwired" decoder requiring minimal power because instruction bounds are fixed.

INSTRUCTION SET ARCHITECTURE

Why RISC Consumes Less Power Than CISC

Architectural convergence and ongoing efficiency differences.

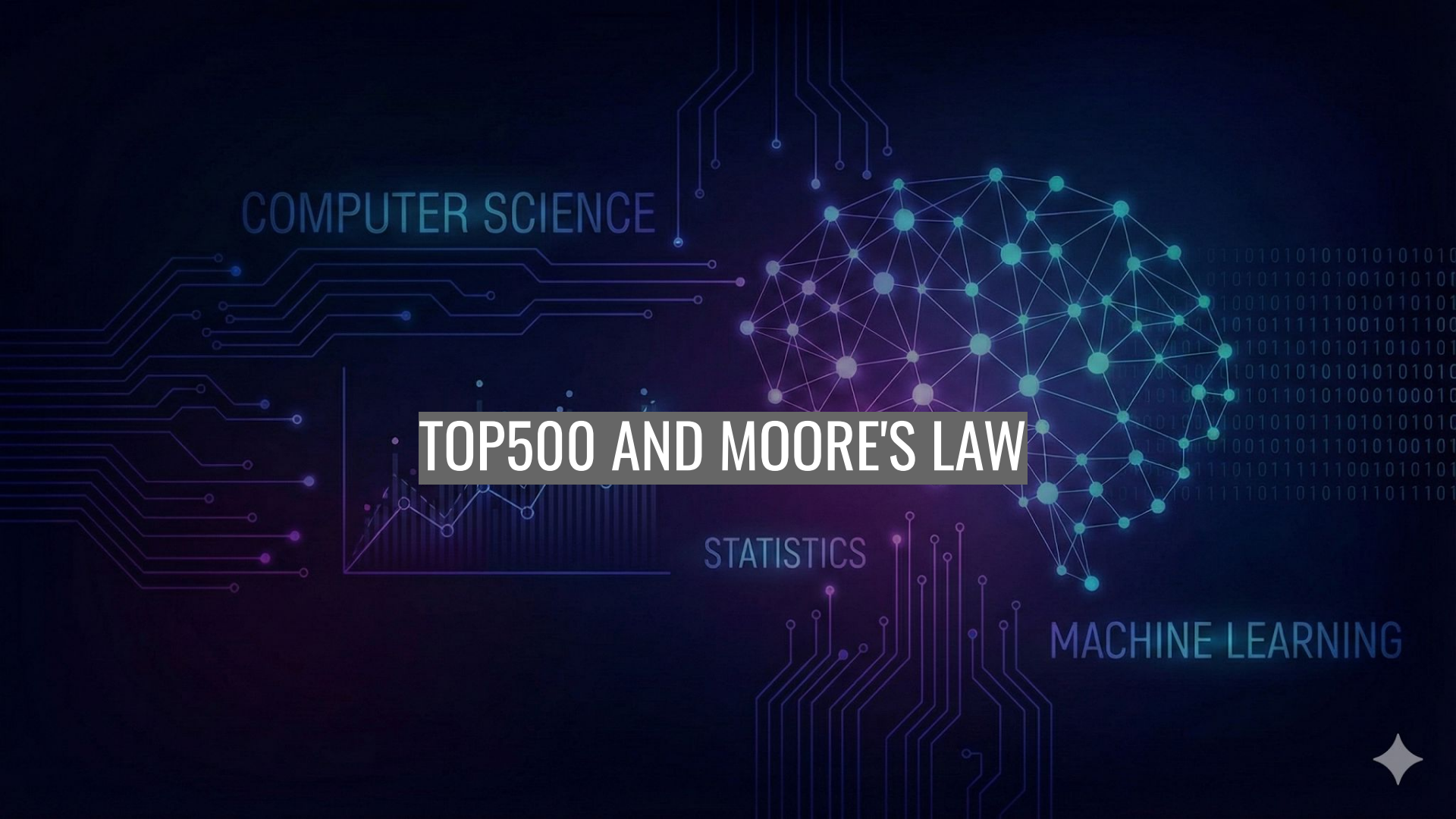
Today, the line is somewhat blurred: Modern CISC and RISC architectures borrow techniques from each other, yet key efficiency gaps remain.

Modern CISC (e.g., Intel Core i9)

- **Hardware Translation:** CISC processors "cheat" by using a hardware translator to turn CISC instructions into RISC-like "micro-operations" internally.
- **Power Overhead:** That translation step itself burns power constantly, creating a continuous energy tax.

Modern RISC (e.g., Apple M3, Snapdragon)

- **Evolving Complexity:** Becoming more complex to achieve higher performance and faster processing speeds.
- **Fundamental Efficiency:** Retains intrinsic instruction set efficiency, running cooler and consuming less battery than CISC counterparts.



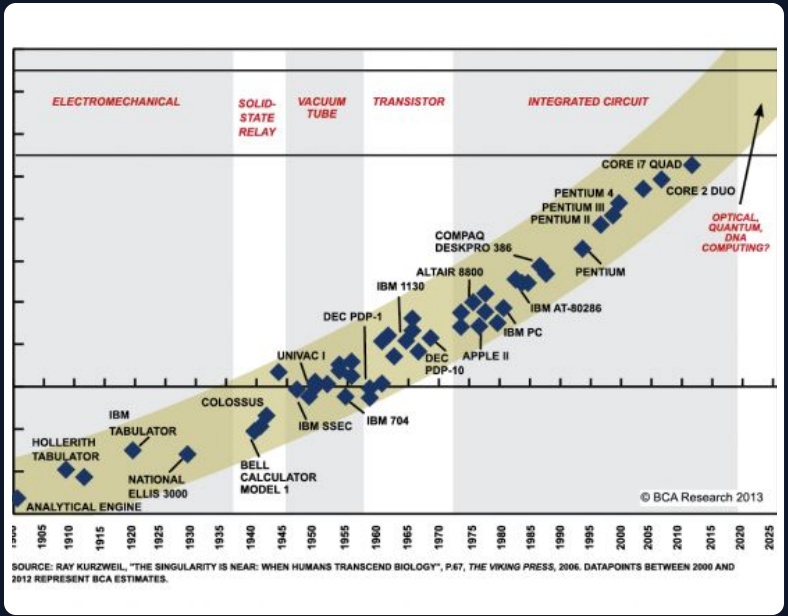
TOP500 AND MOORE'S LAW

COMPUTING ARCHITECTURE & PERFORMANCE

Moore's Law & The Modern Scaling Paradigm

Historical trajectory of transistor density and the shift toward architectural innovation.

Historical Growth (Calculations per $1000)



Key Takeaways & Future Outlook

- **The Classic Principle:** Moore's Law originally stated microcircuit complexity (transistors per chip) doubles roughly every 18 months.
- **Slowing Cadence:** While accurate for decades, physical limits have extended the doubling period to **2.5–3 years** .
- **New Growth Drivers:** Current performance gains rely on specialized hardware (AI accelerators), 3D chiplet stacking, and architectural efficiency over raw transistor count.

COMPUTING ARCHITECTURE & PERFORMANCE

Moore's Law: Is It Still True Today?

In the 2020s, Moore's Law is slowing down—or, according to some (like Nvidia's CEO), is officially "dead."

01 Quantum & Thermal Barriers

Physical Limits

We are rapidly approaching the fundamental size of atoms, with leading-edge transistors now measured down to **3nm and 2nm**.

At atomic scales, **quantum tunneling** and extreme **heat dissipation** create critical operational roadblocks that conventional silicon scaling cannot solve.

02 Moore's Second Law

Economic Limits

While packing more transistors remains technically possible, the capital required to build fabrication facilities (fabs) doubles with each node.

A modern semiconductor fab now costs **over $20 Billion**, making it exponentially harder to sustain the original promise of "cheaper and faster" computing.

COMPUTING ARCHITECTURE & PERFORMANCE

TOP500 & Supercomputer Benchmarking

Evaluating sustained vs. peak performance across the world's most powerful systems.

01 Performance Metrics

Sustained vs. Peak

A fundamental distinction in supercomputing analysis is the difference between **sustained performance** and **peak performance**.

Peak represents theoretical maximum capacity, while sustained measures actual execution speed on real workloads.

02 Evaluation Standard

Standard Benchmark

To objectively evaluate the performance of a computer system, you need a reference test and standard benchmark.

The most prominent industry example is **Linpack**, which measures a system's ability to solve dense systems of linear equations.

03 Global Index

TOP500 Project

The **TOP500** tracks and publishes rankings of the 500 most powerful computer systems in the world.

Explore details and historical rankings at top500.org.

Top 500 list November 2025

GLOBAL TOP 10 SUPERCOMPUTERS

Ranking based on HPL Performance (Rmax)

Source: Top500.org (Novembe

#	SYSTEM	CORES	RMAX (PFLOP/S)	POWER (KW)
1	El Capitan NEW #1 *USA (LLNL)*	11.3M	**1,809.00**	29,685
2	Frontier *USA (ORNL)*	9.1M	**1,353.00**	24,607
3	Aurora *USA (ANL)*	9.3M	**1,012.00**	38,698
4	JUPITER Booster NEW *Germany (EuroHPC)*	4.8M	**1,000.00**	15,794
5	Eagle *USA (Microsoft Azure)*	2.1M	561.20	N/A
6	HPC6 NEW *Italy (Eni S.p.A)*	3.1M	477.90	8,461
7	Fugaku *Japan (RIKEN)*	7.6M	442.01	29,899
8	Alps *Switzerland (CSCS)*	2.1M	434.90	7,124
9	LUMI *Finland (EuroHPC)*	2.8M	379.70	7,107
10	Leonardo *Italy (EuroHPC)*	1.8M	241.20	7,494

COMPUTING ARCHITECTURE & PERFORMANCE

Human Brain vs. Supercomputer Scale

Contextualizing human cognitive capacity against Top500 historical performance growth.



01 CONSERVATIVE THEORETICAL MODEL

A Few Dozen Petaflops

An estimate of the **computing power of the human mind** is a few dozen petaflops or more.

This baseline is considered correct but conservative, derived from older theoretical synapse models.

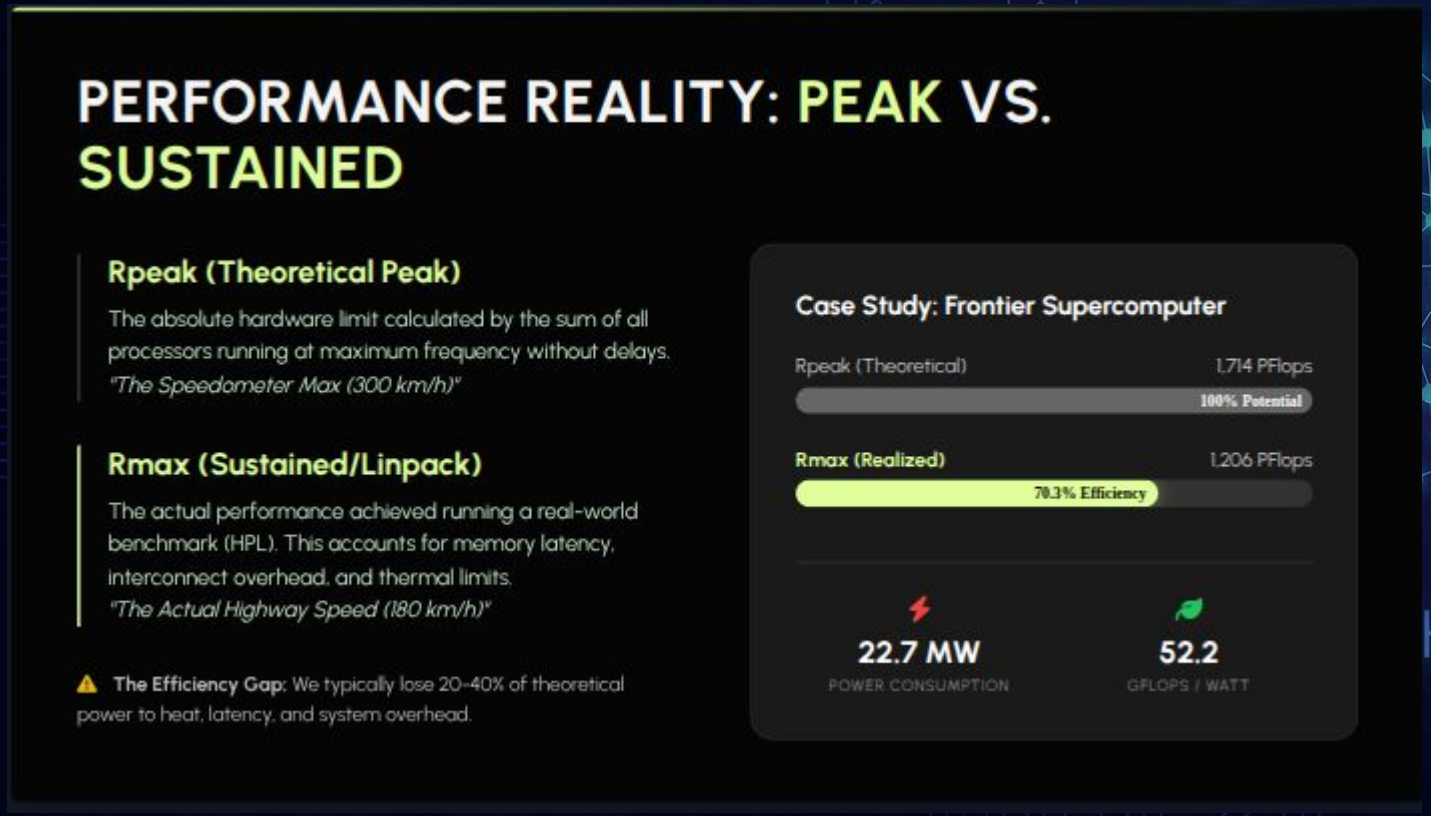
02 MODERN NEUROSCIENCE CONSENSUS

Exaflop Scale (1,000+ Petaflops)

Many modern neuroscientists and computer engineers now believe the true number is significantly higher.

Current research places human brain processing potential firmly in the **Exaflop (1,000 Petaflops) range**.

Top500 list historical chart



In real applications the gap is much worst

HINE LEARNING

ORIGINS OF THE COMPUTATIONAL ESTIMATE

Where Does the "Petaflop" Estimate Come From?

Popularized by futurists Hans Moravec and Ray Kurzweil in the late 1990s and early 2000s.

01 THE BIOLOGICAL FORMULA

10^{14} Synapses × 10 Signals/Sec

The classical calculation breaks down brain capacity using simple hardware mechanics:

- **Synapse Count:** Estimated **10^{14}** total connections.
- **Firing Rate:** ~**10 signals/sec** per synapse.
- **Baseline Output:** Yields **10^{15} ops/sec** (1 Petaflop).

02 FUTURIST RANGE ESTIMATE

1 to 50 Petaflops Target

Popularized by Hans Moravec and Ray Kurzweil in the late 1990s and early 2000s.

Target benchmark estimated for human-equivalent computational throughput.

03 KEY MODEL ASSUMPTION

Binary Switch Abstraction

If the brain works simply (like a binary switch), **"a few dozen petaflops"** is correct.

This serves as the conservative theoretical baseline for human mind processing power.

RE-EVALUATING BRAIN COMPLEXITY

The Modern View: The "Exaflop" Estimate

Recent research suggests the brain is far more complex than a simple system of binary switches.

01 MOLECULAR MACHINE

Non-Linear Processing

A synapse isn't just a simple on/off switch; it functions as a complex molecular machine.

Processes chemical interactions non-linearly across dendrites to replicate biological brain function.

02 REVISED REQUIREMENT

1 Exaflop to 1 Zettaflop

Simulating fine-grained chemical interactions jumps computational demand to 10^{18} (1 Exaflop) or 10^{21} (1 Zettaflop).

Represents the updated threshold required for full molecular-level brain simulation.

03 SCALE MAGNITUDE

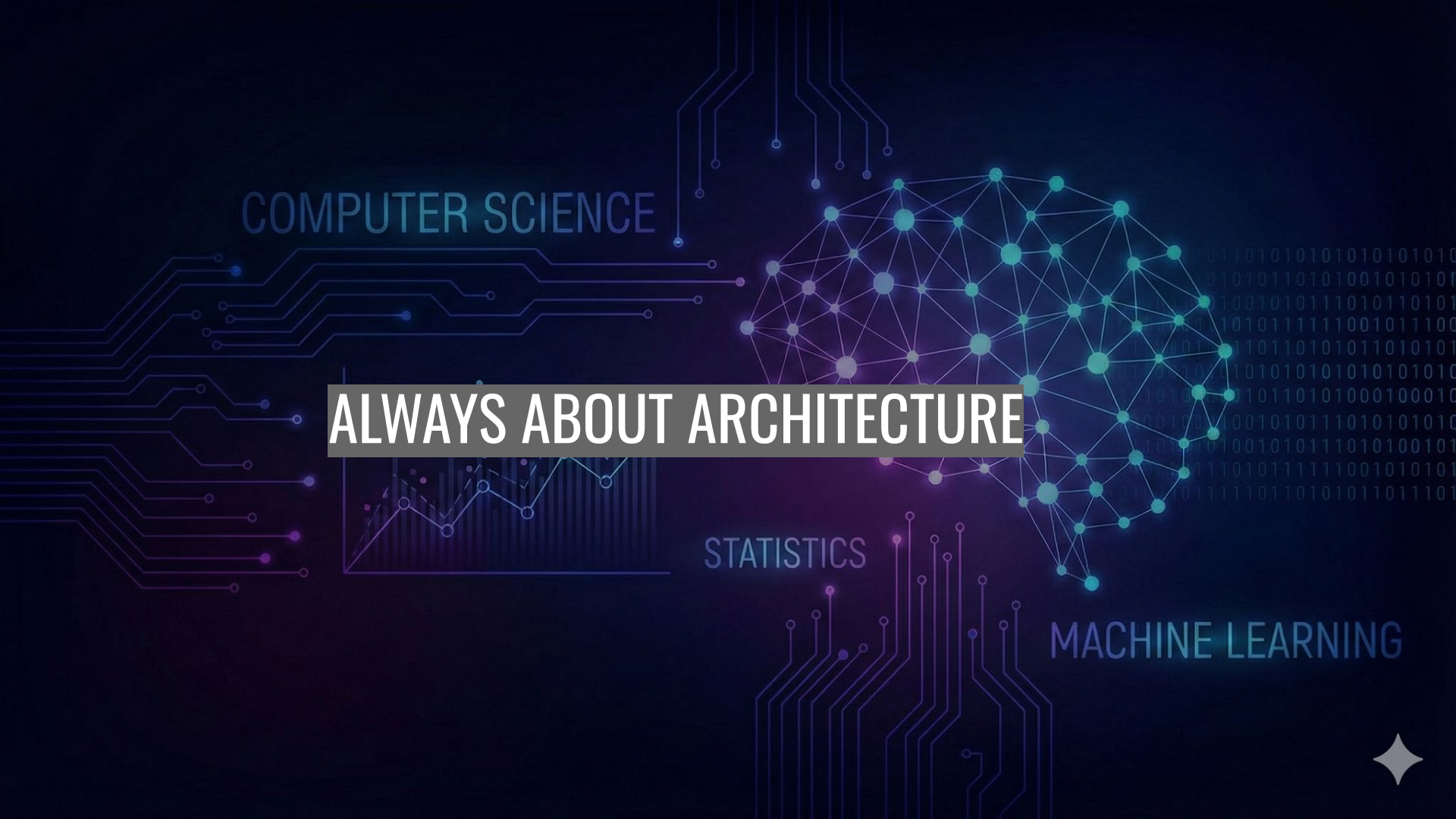
100× to 1,000× Gap

By modern biophysical standards, previous hardware estimates are drastically flawed.

The classic "few dozen petaflops" baseline is too low by a factor of **100 to 1,000**.

Top500 list historical chart

Feature	Human Brain	El Capitan (Nov 2025 #1 Supercomputer)
Speed (Est.)	~1 Exaflop (Variable)	1.8 Exaflops (Sustained)
Parallelism	Massive (Billions of threads)	High (Millions of cores)
Clock Speed	Slow (~200 Hz / 0.0002 GHz)	Fast (1.8 GHz - 3.0 GHz)
Power Usage	**~20 Watts** (A dim lightbulb)	**~30,000,000 Watts** (A small city)
Cooling	Blood flow	Massive liquid cooling plant


COMPUTER SCIENCE
ALWAYS ABOUT ARCHITECTURE
STATISTICS
MACHINE LEARNING

FPGA & ASIC

Field Programmable Gate Arrays & Application Specific Integrated Circuits

01. Architecture

Field Programmable Gate Array

An integrated circuit whose functions are programmable via software after manufacturing.

Core Components:

- Based on an array of Configurable Logic Blocks (CLBs).
- Connected via programmable interconnects.

02. Flexibility

Reprogrammability

Can be reconfigured based on application or functionality requirements post-manufacturing.

Primary Types:

- **SRAM-based:** Dominant type that can be continuously reprogrammed as design evolves.
- **OTP:** One-Time Programmable FPGAs for fixed deployments.

03. FPGA vs ASIC

Application-Specific ICs

ASICs are custom-built for specific design tasks rather than software reconfigurable.

Key Distinction:

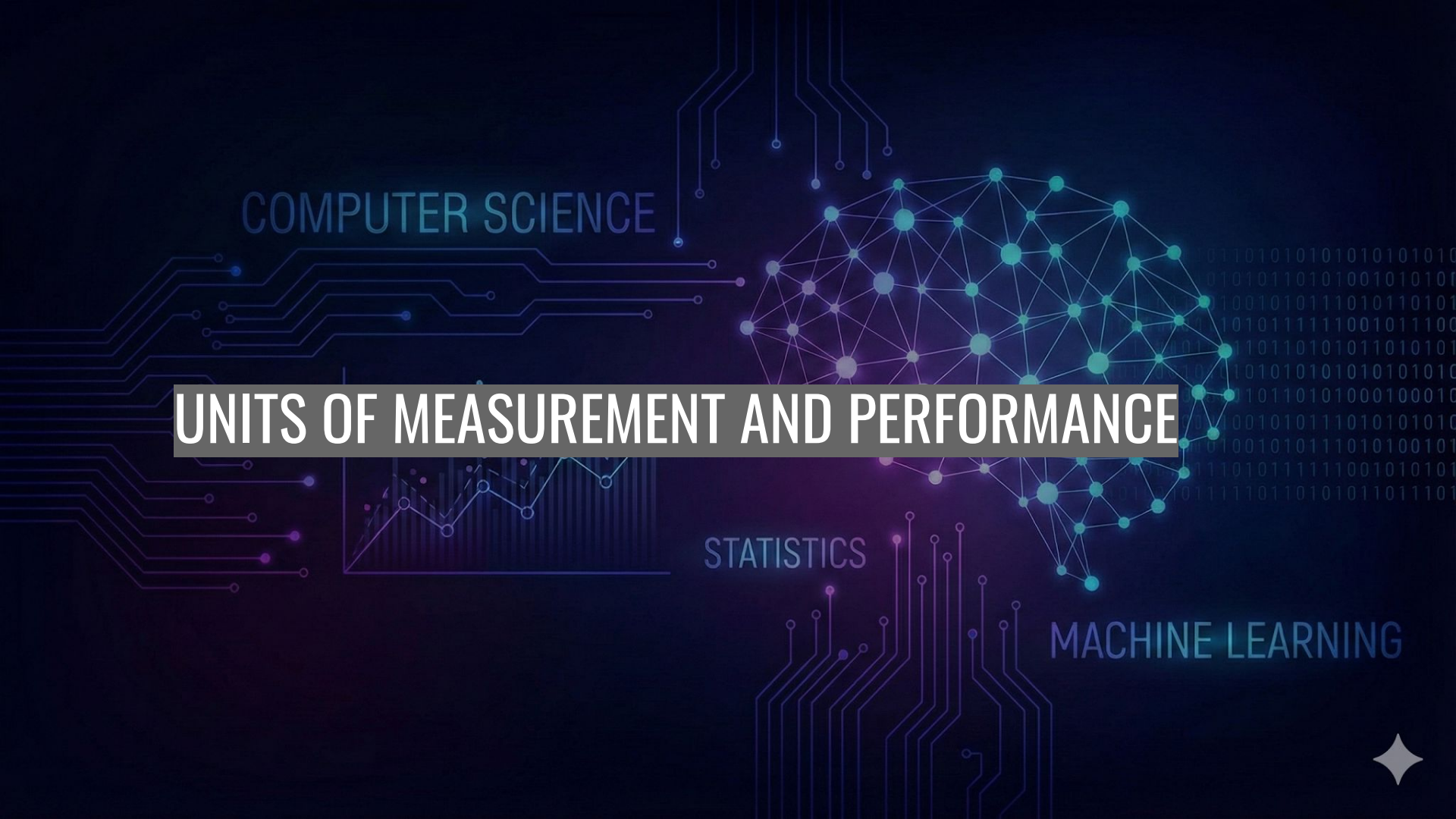
- **FPGAs:** Flexible, field-programmable software logic for evolving requirements.
- **ASICs:** Permanently fixed hardware optimized for target applications.

Contents

- Introduction to Informatics
  - What is a computer
  - Networks and TCP/IP
  - Digitalization and basics of data encryption







UNITS OF MEASUREMENT AND PERFORMANCE

Performance

Understanding Network Data Transmission Metrics

CAPACITY

Bandwidth

The **maximum amount of data** (number of bits) that can be transmitted in a channel.

Often used as an **approximation of actual throughput**.

Unit of Measurement **Mbps / Gbps**

ACTUAL RATE

Throughput

The **actual amount of data** (number of bits) transmitted on the channel in a certain period of time.

Direct measure of real-world **Performance** under current network conditions.

Unit of Measurement **Mbps**

Performance

Key Metrics: Latency and Round Trip Time (RTT)

ONE-WAY DELAY

Latency

The time taken by a message to go from one point to another.

- **Propagation delay:** Speed of signal in transmission medium & distance.
- **Processing time:** Overhead required to process packet headers.

Unit of Measurement **ms (1 ms = 1/1000 s)**

TWO-WAY DELAY

Round Trip Time (RTT)

The time it takes for a message to go from **point A to B** and back again from **B to A**.

Measures total end-to-end responsiveness across a bidirectional communication link.

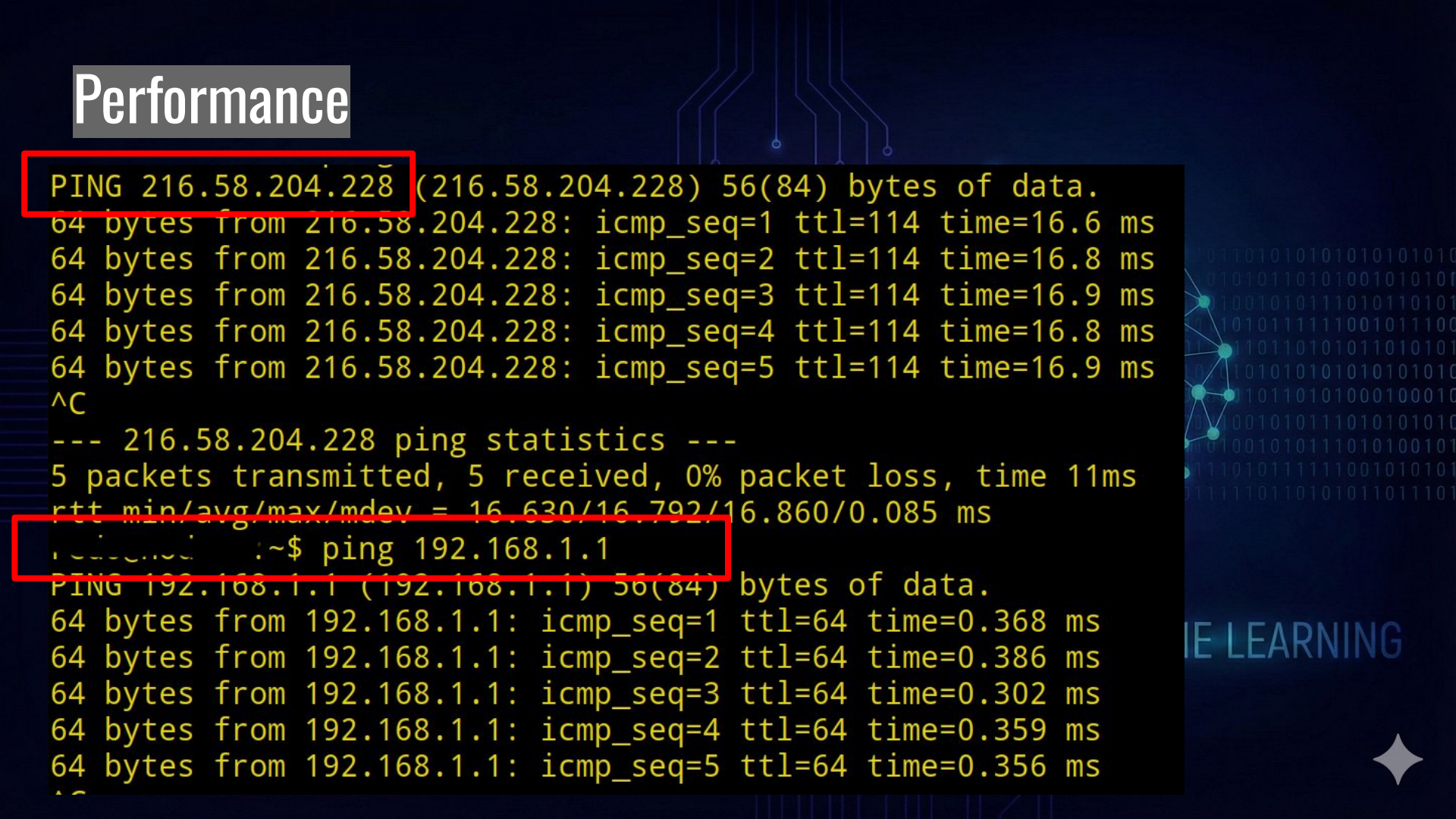
Diagnostic Tools **ping / traceroute**

Performance

```
traceroute to www.google.com (216.58.209.36), 30 hops max, 60 byte packets
 1  telecomitalia.it (192.168.1.1)  0.338 ms  0.203 ms  0.262 ms
 2  * * *
 3  172.18.25.0 (172.18.25.0)  3.892 ms  4.317 ms 172.18.25.4 (172.18.25.4)  3.046 ms
 4  172.18.25.212 (172.18.25.212)  3.731 ms 172.18.25.220 (172.18.25.220)  3.957 ms 172.18.25.212 (172.1
.25.212)  4.022 ms
 5  172.19.184.18 (172.19.184.18)  6.530 ms 172.19.184.22 (172.19.184.22)  7.524 ms 172.19.184.18 (172.1
.184.18)  7.087 ms
 6  172.19.177.72 (172.19.177.72)  10.343 ms 172.19.177.46 (172.19.177.46)  8.037 ms 172.19.177.72 (172.
9.177.72)  9.729 ms
 7  172.19.177.8 (172.19.177.8)  19.077 ms  18.159 ms 172.19.177.2 (172.19.177.2)  16.852 ms
 8  195.22.205.116 (195.22.205.116)  18.488 ms 195.22.192.144 (195.22.192.144)  20.639 ms 195.22.205.98
195.22.205.98)  20.207 ms
 9  72.14.204.72 (72.14.204.72)  22.495 ms 72.14.243.190 (72.14.243.190)  18.924 ms 72.14.219.236 (72.14
219.236)  18.668 ms
10  * 66.249.95.89 (66.249.95.89)  17.472 ms *
11  142.250.211.20 (142.250.211.20)  16.206 ms 142.251.235.174 (142.251.235.174)  14.762 ms 142.251.235.
72 (142.251.235.172)  16.842 ms
12  108.170.232.169 (108.170.232.169)  18.552 ms mil07s12-in-f4.1e100.net (216.58.209.36)  18.694 ms 192
178.104.214 (192.178.104.214)  20.536 ms
```

Performance

```
PING 216.58.204.228 (216.58.204.228) 56(84) bytes of data.
64 bytes from 216.58.204.228: icmp_seq=1 ttl=114 time=16.6 ms
64 bytes from 216.58.204.228: icmp_seq=2 ttl=114 time=16.8 ms
64 bytes from 216.58.204.228: icmp_seq=3 ttl=114 time=16.9 ms
64 bytes from 216.58.204.228: icmp_seq=4 ttl=114 time=16.8 ms
64 bytes from 216.58.204.228: icmp_seq=5 ttl=114 time=16.9 ms
^C
--- 216.58.204.228 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 11ms
rtt min/avg/max/mdev = 16.630/16.792/16.860/0.085 ms
                :~$ ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=0.368 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=0.386 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=0.302 ms
64 bytes from 192.168.1.1: icmp_seq=4 ttl=64 time=0.359 ms
64 bytes from 192.168.1.1: icmp_seq=5 ttl=64 time=0.356 ms
```



Performance

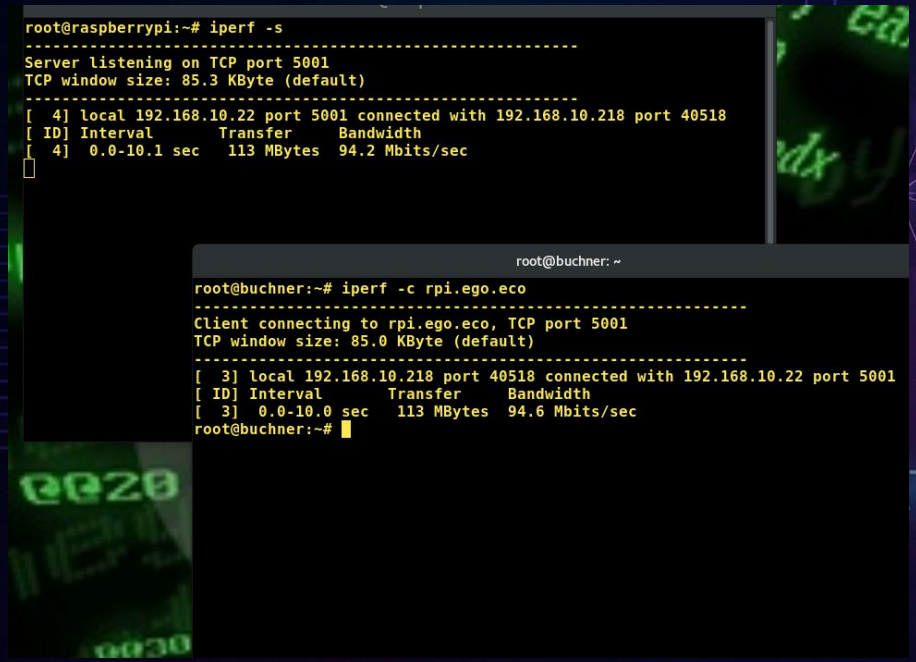
```
   Speedtest by Ookla

     Server: TIM SpA - Pescara (id = 36728)
        ISP: TIM
    Latency:     2.85 ms   (0.12 ms jitter)

   Download:   940.69 Mbps (data used: 682.2 MB)

     Upload:   313.48 Mbps (data used: 141.4 MB)
Packet Loss:     0.0%
```

Performance



iPerf Network Benchmark

Server: root@raspberrypi (port 5001)

Client: root@buchner (rpi.ego.eco)

Protocol: TCP window size 85.0–85.3 KByte

Measured Results

Data Transferred: 113 MBytes in 10.0–10.1 s

Average Throughput: ~94.4 Mbits/sec

(Operates near maximum capacity for 100 Mbps Fast Ethernet link)

Unit of Measurement

Formula: Data transmission speed = amount of information / transfer time

Units & Conversion

In general, speed is expressed in bits per second, i.e., **bit/s** (or **bps**, also called **bit rate**). The byte per second, **byte/s** (or **Bps**), is also used.

Converting from bps to Bps is simple, just divide by 8.

Example: ADSL 10 Mbps = 10 Mbps/8 = 1250 KBps

Standard Decimal Prefixes

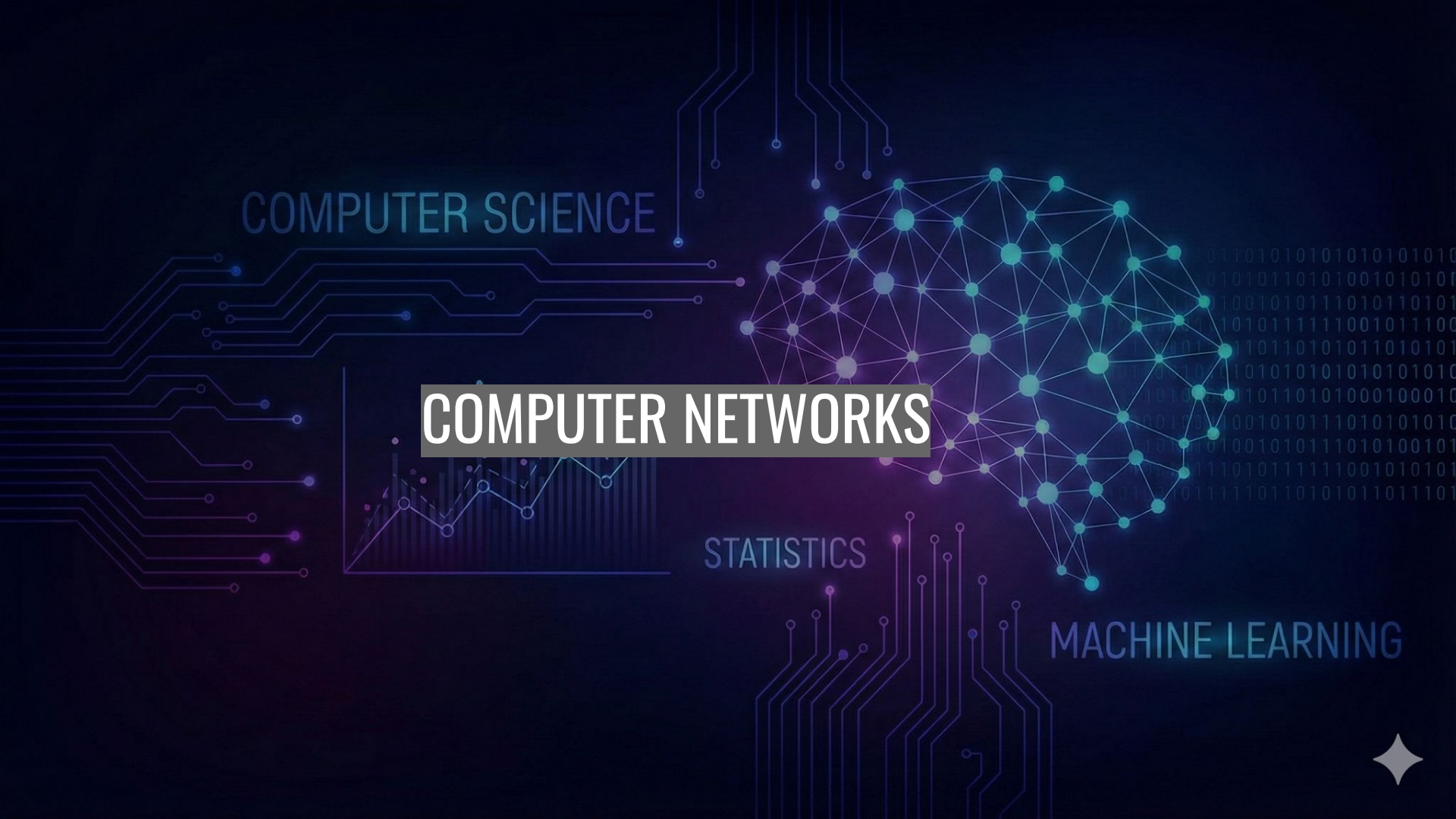
We use standard decimal prefixes, therefore not the approximations based on powers of two that are used in computer science:

- **k** (=kilo 10^3 = 1,000)
- **M** (=mega 10^6 = 1,000,000)
- **G** (=giga 10^9 = 1,000,000,000)

Calculation Example

Transferring a **10 MiB** file over a **5 Mbps** line will take approximately:

`(10 * 1024 * 1024 * 8) / (5 * 10^6) = 16.8 s` (ignoring latency)


COMPUTER SCIENCE
COMPUTER NETWORKS
STATISTICS
MACHINE LEARNING

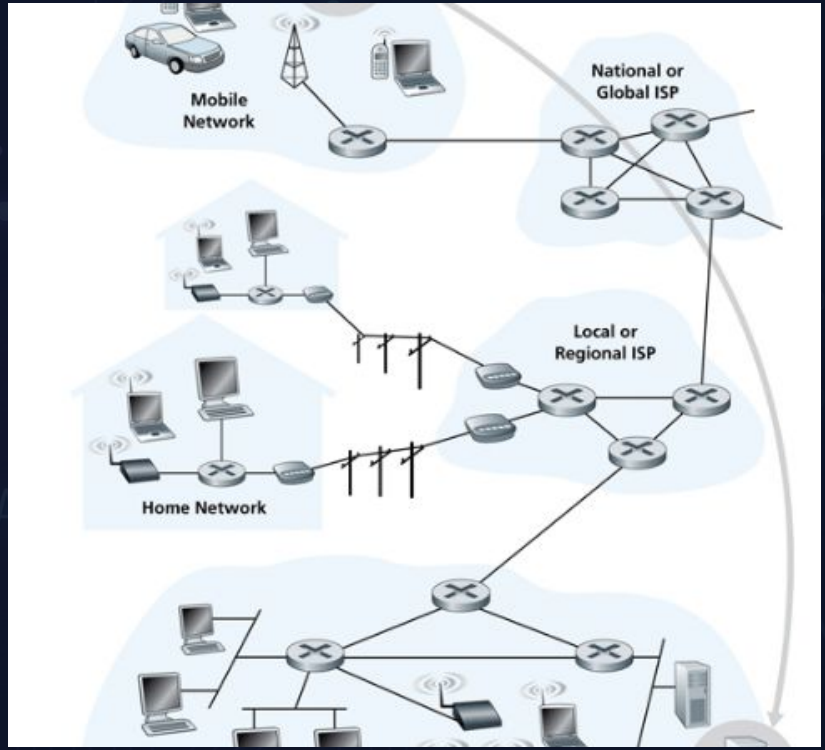
Computer Networks

Network Definition

A set of connected autonomous computers, the network is seen as providing **logical channels** through which various applications can communicate with each other.

Key Attributes

- **Autonomous Hosts:** Devices operate independently.
- **Logical Channels:** Abbreviated abstraction for app-to-app communication.
- **Interconnection:** Physical & wireless media links.



Computer Networks

Nodes (Hosts)

Autonomous computers connected within the network environment.

Device Types:

- Servers
- Desktop PCs
- Mobile devices & other endpoints

Connections

Communication channels allowing nodes to exchange data.

Transmission Media:

- Twisted pair telephone wire
- Optical fiber
- Wireless radio channels

Topology

Structural linking options available for network nodes.

Connection Modes:

- **Direct Links:** Point-to-point connections between hosts.
- **Indirect Links:** Channels mediated via switches.

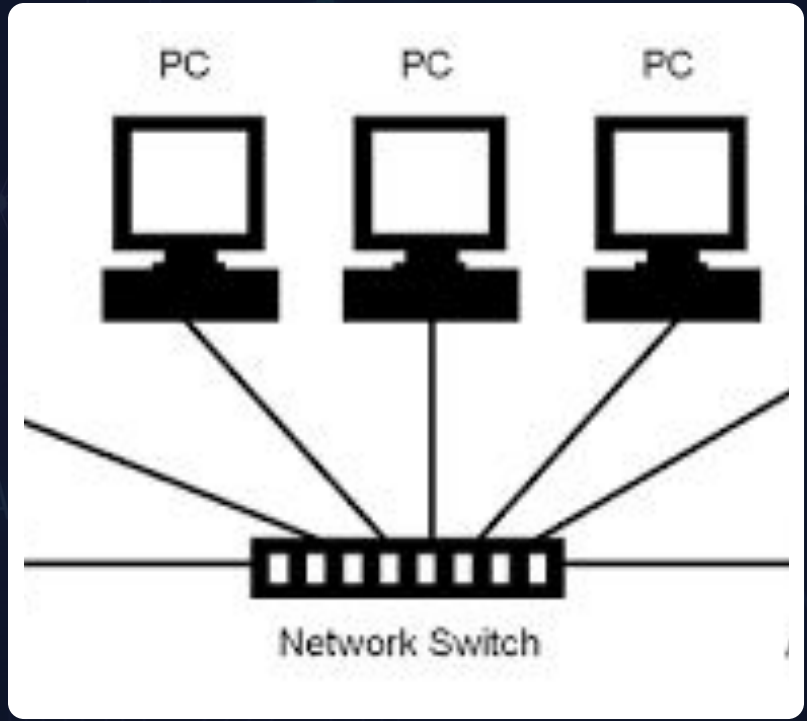
Switch

Definition

A network switch (also called a switching hub or bridging hub) is a network device that connects computer devices together using packet switching to receive, process, and forward data to the destination device.

Key Functions

- **Packet Switching:** Efficiently routes network traffic.
- **Data Handling:** Receives, processes, and forwards incoming data packets.
- **Device Interconnection:** Links nodes like PCs, servers, printers, and modems.



Computer Networks

I can characterize networks based on their extension

LAN

Local Area Network

Local area networks extend to the level of a single room or at most a building.

- Single room or office
- Entire building coverage
- High-speed local connection

MAN

Metropolitan Area Network

Networks designed to span a town or city, interconnecting multiple LANs.

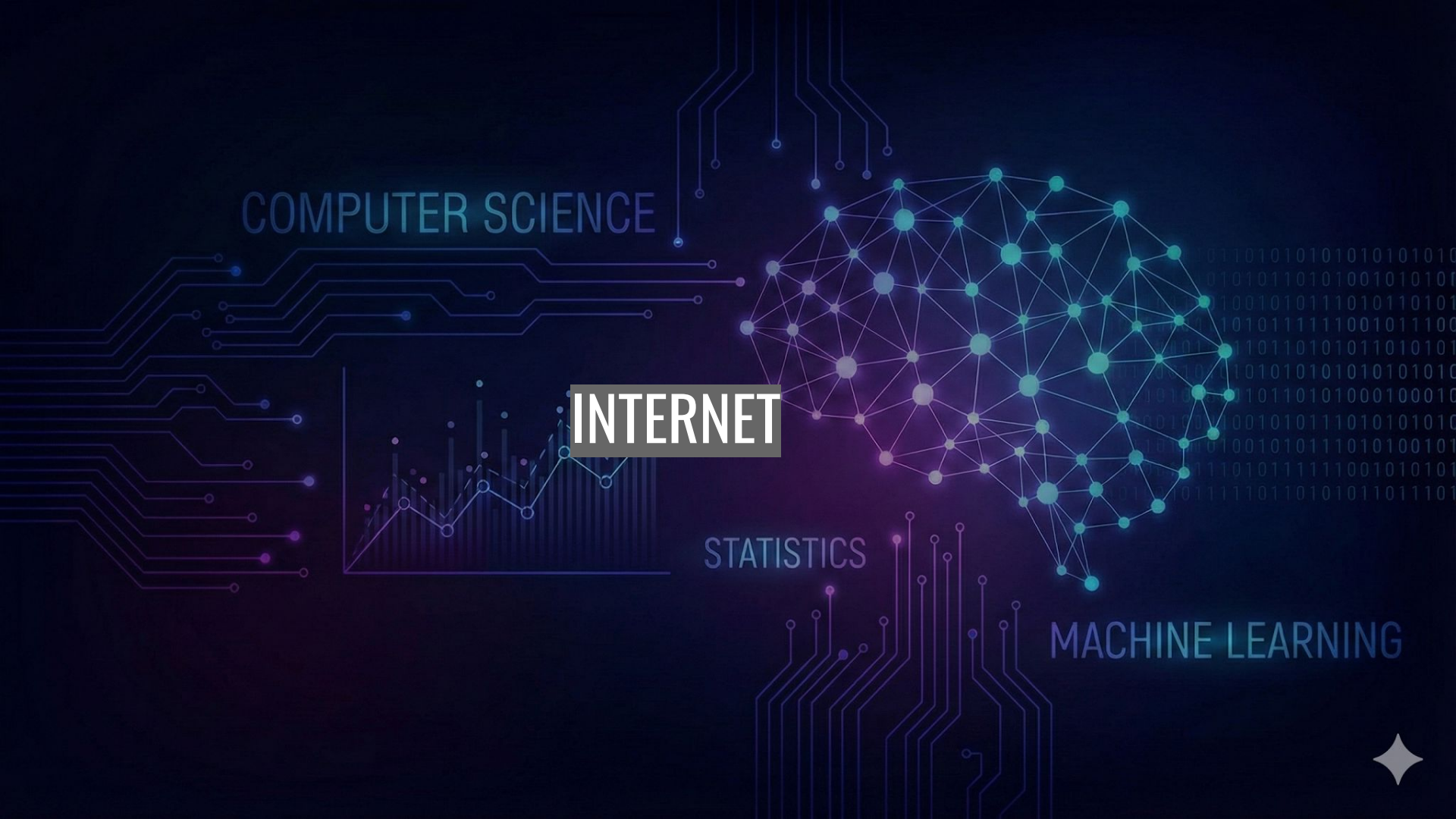
- Connects multiple LANs
- Spans town or city scale
- Campus-wide connectivity

WAN

Wide Area Network

Networks extending across large geographical areas, connecting LANs and MANs.

- Broad geographical scale
- Interconnects LANs & MANs
- Internet is WAN par excellence


COMPUTER SCIENCE
INTERNET
STATISTICS
MACHINE LEARNING

Internet

Basic components and network architecture

Basic Components

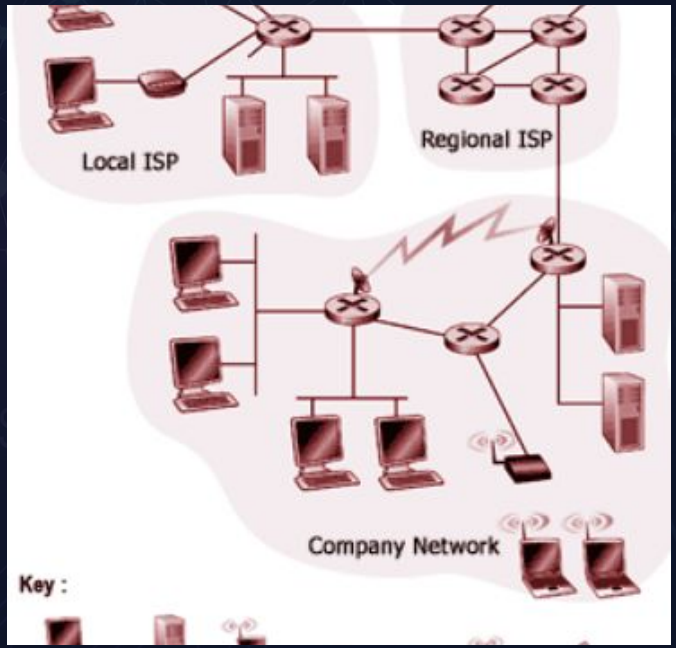
We can describe the internet first of all from the point of view of the basic components.

Hosts & End Systems

The Internet interconnects millions of devices around the world, including traditional desktop PCs, mobile devices, and servers.

- Desktop PCs & Traditional Workstations
- Mobile Devices & Handhelds
- Servers storing and transmitting data (HTML, emails)

Component Infrastructure Overview



Internet

Communication channels, routers, and core networking protocols

Communication Channels

The various hosts are interconnected via communication channels. These channels can be of very different nature:

- Coaxial cables
- Copper wires
- Optical fibers
- Radio links

Routers & Switching

Generally, hosts are not directly interconnected, but there are "switching" devices called routers.

- Used to route data traffic
- Take information arriving from one channel
- Send information to another channel

Interconnected Networks

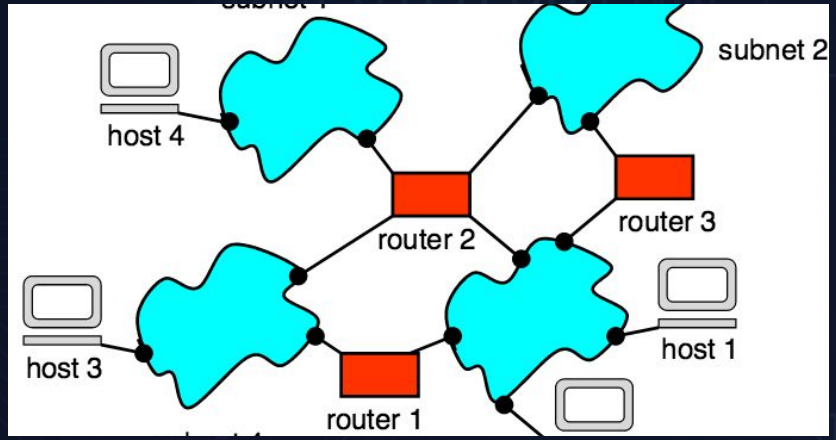
The Internet is a set of interconnected networks. Hosts and infrastructure devices communicate using common protocols:

- **IP:** Internet Protocol
- **TCP:** Transmission Control Protocol

Internet

The Internet today interconnects thousands of subnets

Subnet Network Architecture



Visual representation of interconnected subnets, routing nodes, and host endpoints forming network topology.

Host

Computer connected to the Internet; can be either a client or a server at the application level.

Router

Node used to route traffic (a Layer 3 network gateway device).

Subnet

A set of hosts between which there is a Layer 2 connection (for example, a LAN).

Billions of Hosts

Interconnected globally across subnets and gateways.

Internet: A Brief History

Key milestones in early computer networking and protocol development (1969–1979)

1969

ARPANET Design

Design of the ARPANET military network began. Built to withstand nuclear attacks by connecting devices via alternate paths during network failure.

1972

Early Services

Saw the birth of fundamental communication tools:

- Electronic Mail (e-mail)
- File transfer via FTP
- Remote login services

1974

Core Protocols

The IP (Internet Protocol) and TCP (Transmission Control Protocol) standard protocols were officially introduced, establishing the bedrock of modern networking architecture.

1979

CSNet Network

Birth of the CSNet network. Created to interconnect universities and academic research centers across the United States.

Internet: A Brief History

Milestones in network expansion and the birth of the World Wide Web (1982–1990)

1982

Official Birth of Internet

ARPANET and CSNet are connected. This is considered the official birth date of the Internet in some ways.

1990

NSFNet Transition

NSFNet, a network connecting supercomputers, replaces the ARPANET. This paves the way for civilian and commercial uses of the Internet.

1990

Invention of HTML

Tim Berners-Lee at CERN in Geneva invented HTML (Hyper Text Markup Language), allowing management of text, images, and information—the first step towards the World Wide Web.

Internet: A Brief History

The rise of web browsers, global expansion, and modern internet scale (1993–Present)

1993–1998

Browsers & Search

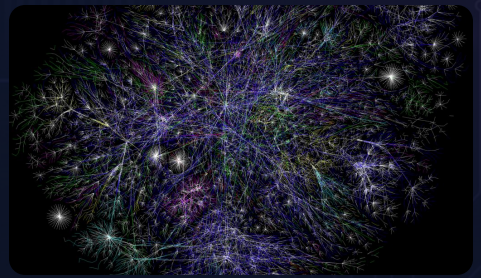
- **1993:** Mosaic, the first web browser, was created.
- **1994:** Yahoo!, the first search engine, was born.
- **1998:** Rapid expansion of web engines led to the birth of Google.

SCALE

Global Infrastructure

Contains ~10 billion computers, averaging billions of transistors per device.

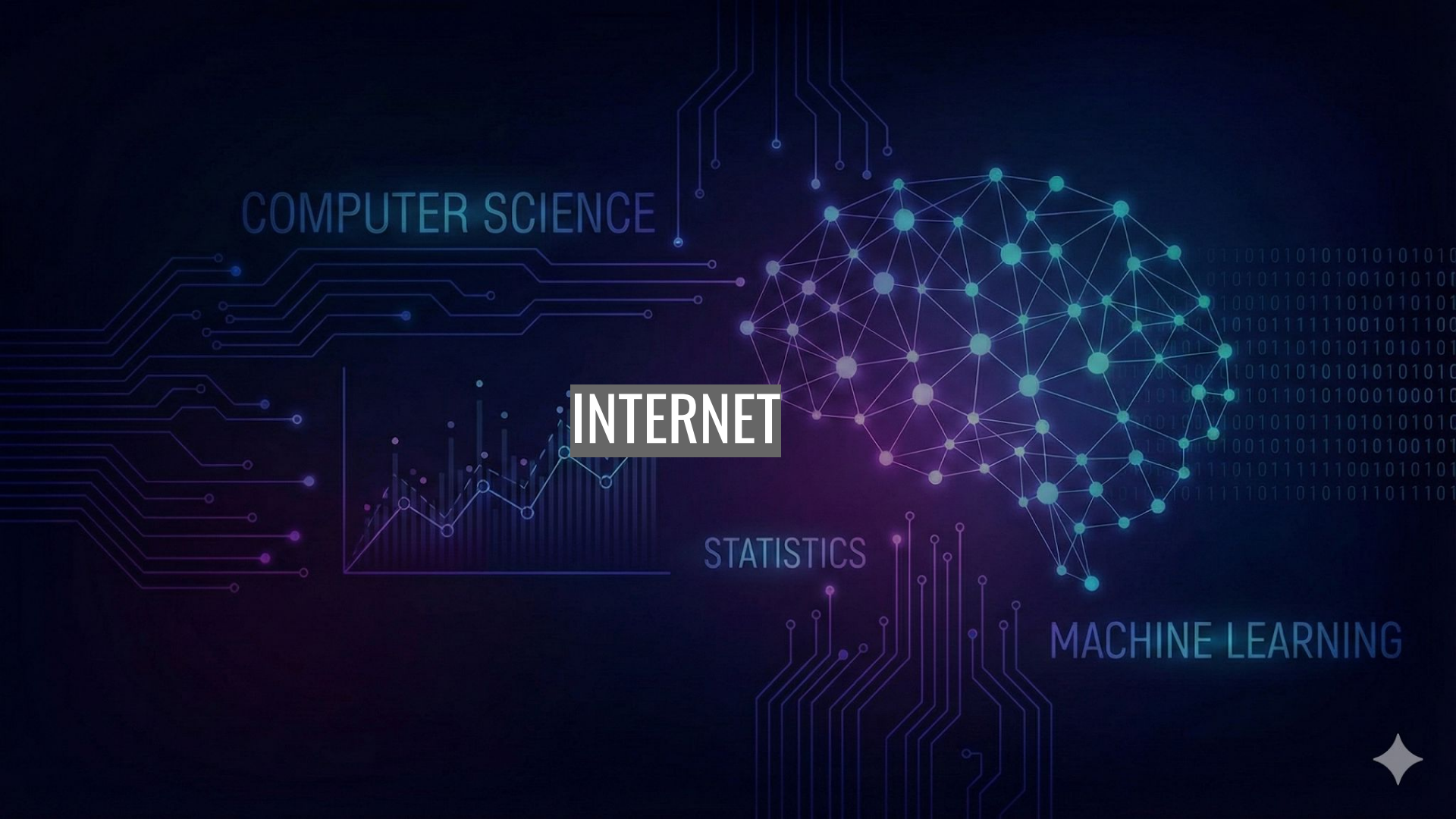
Interconnects 1019 transistors—10,000x more than synapses in the human brain.



TODAY

Modern Applications

- **Commerce:** Selling goods and services globally.
- **Media:** Seamless audio and video transmission.
- **Connectivity:** Online collaboration, work, gaming, and social interaction.


COMPUTER SCIENCE
INTERNET
STATISTICS
MACHINE LEARNING

Communication Protocol

Foundational principles of network interaction and data exchange

DEFINITION

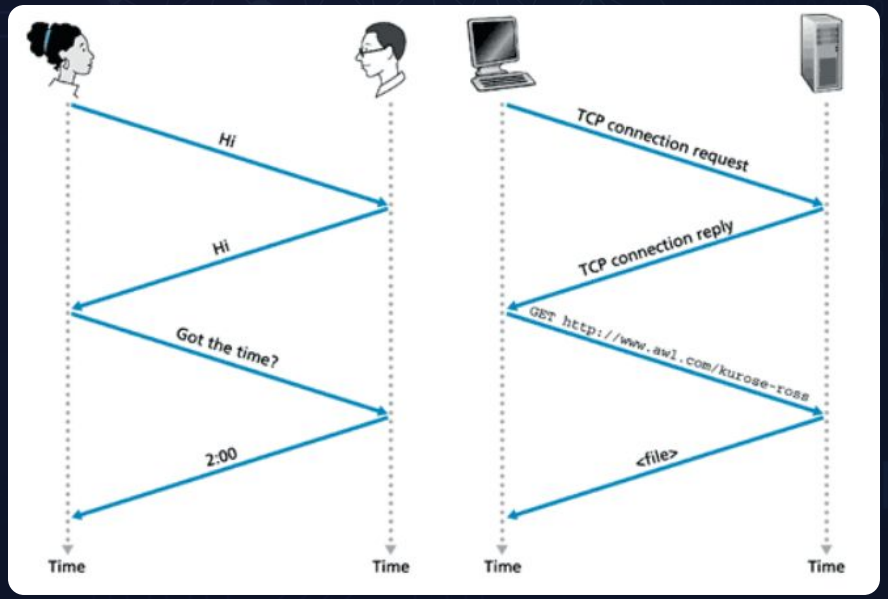
Core Specification

A set of rules (formally described) that define the methods of communication between two or more entities.

- **Syntax:** Format, structure, and encoding of transmitted data.
- **Semantics:** Action logic and control information for responses.
- **Timing:** Speed matching, sequencing, and message ordering.

PROTOCOL IN ACTION

Human Dialogue vs. Machine Handshake (TCP/HTTP)



Network Model: Client-Server

Centralized interaction mode and architecture for network service delivery

DEFINITION & ROLE

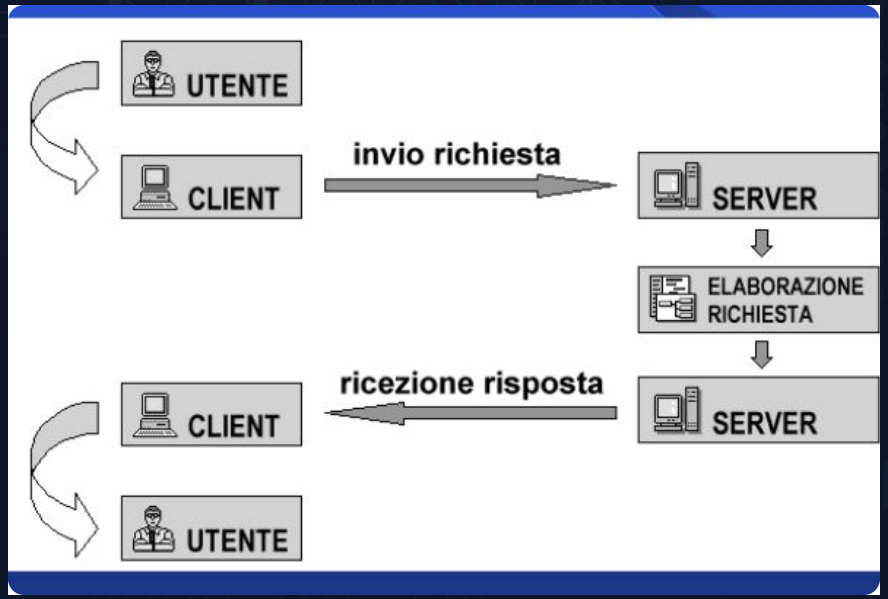
Core Interaction Model

Most of the telematic services offered by the Internet are based on the **client/server interaction mode** (different from P2P).

- **Client Capability:** Equipped with special client software capable of sending service requests to a specific server.
- **Protocol Standardization:** Formats requests appropriately so they are understandable to the server, using specific protocols (e.g., web browsers).

ARCHITECTURE IN ACTION

Request - Processing - Response Workflow



Network Model: P2P Peer-to-Peer

Decentralized architecture and direct node-to-node resource sharing

DEFINITION & ROLE

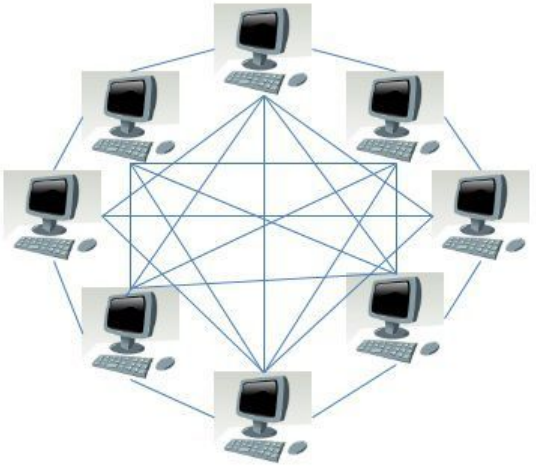
Equal Node Capabilities

In a Peer-to-Peer network, **there is no distinction between client and server**.

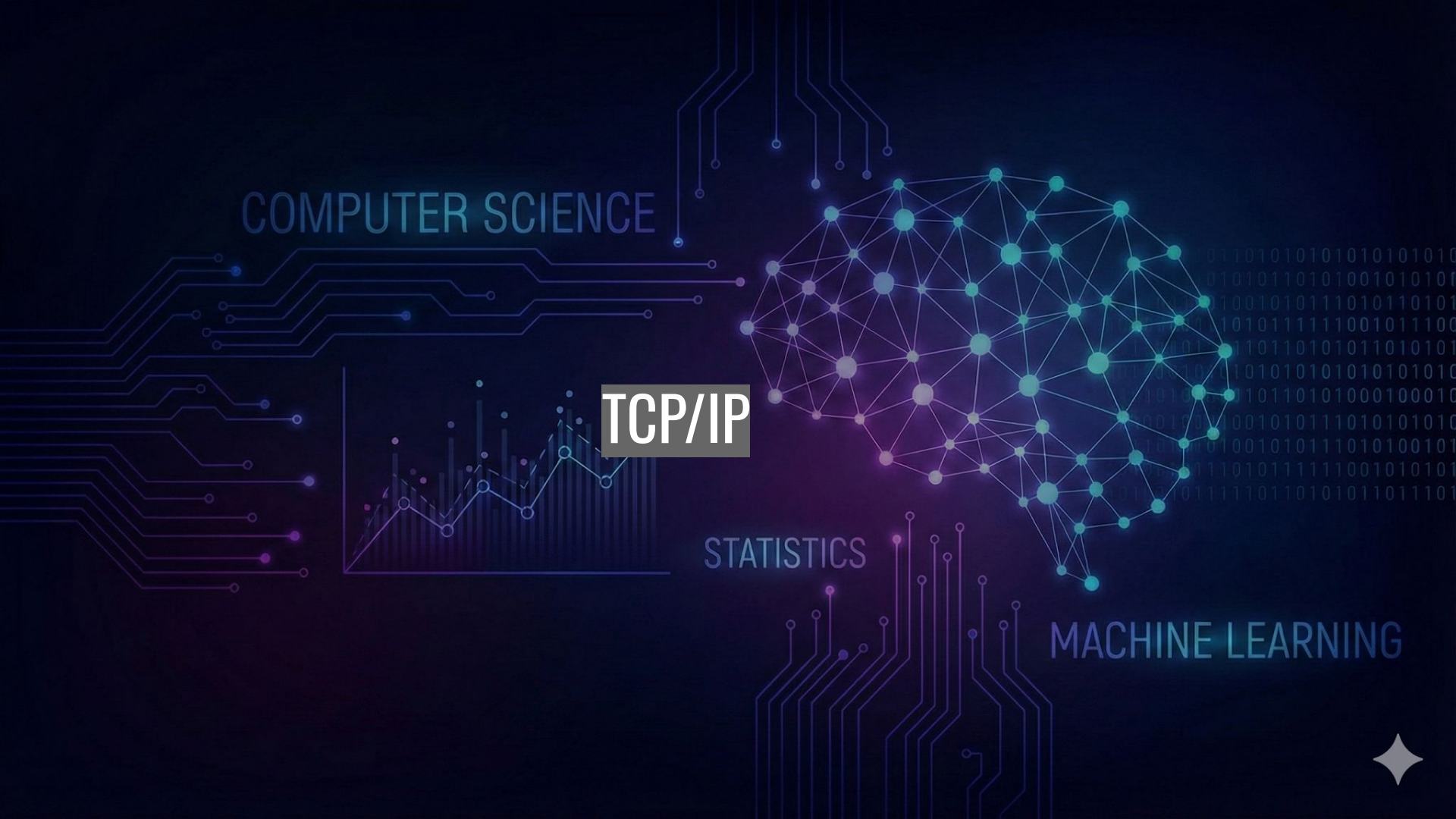
- **Dynamic Roles:** Each node can act as either client or server depending on whether it requests or provides data.
- **Network Joining:** Nodes must start providing services to request services from others (e.g. BitTorrent).

TOPOLOGY IN ACTION

Decentralized Mesh Architecture



Peer-to-Peer Network Model


COMPUTER SCIENCE
TCP/IP
STATISTICS
MACHINE LEARNING

TCP/IP: Basic Operation

Rules of communication and step-by-step interaction protocol example

DEFINITION & CONCEPT

What is a Protocol?

A protocol is a set of rules that the sender (sender) and the recipient (recipient) must follow.

A trivial example of a (non-formal) "protocol" that two people follow when they meet and one asks the time:

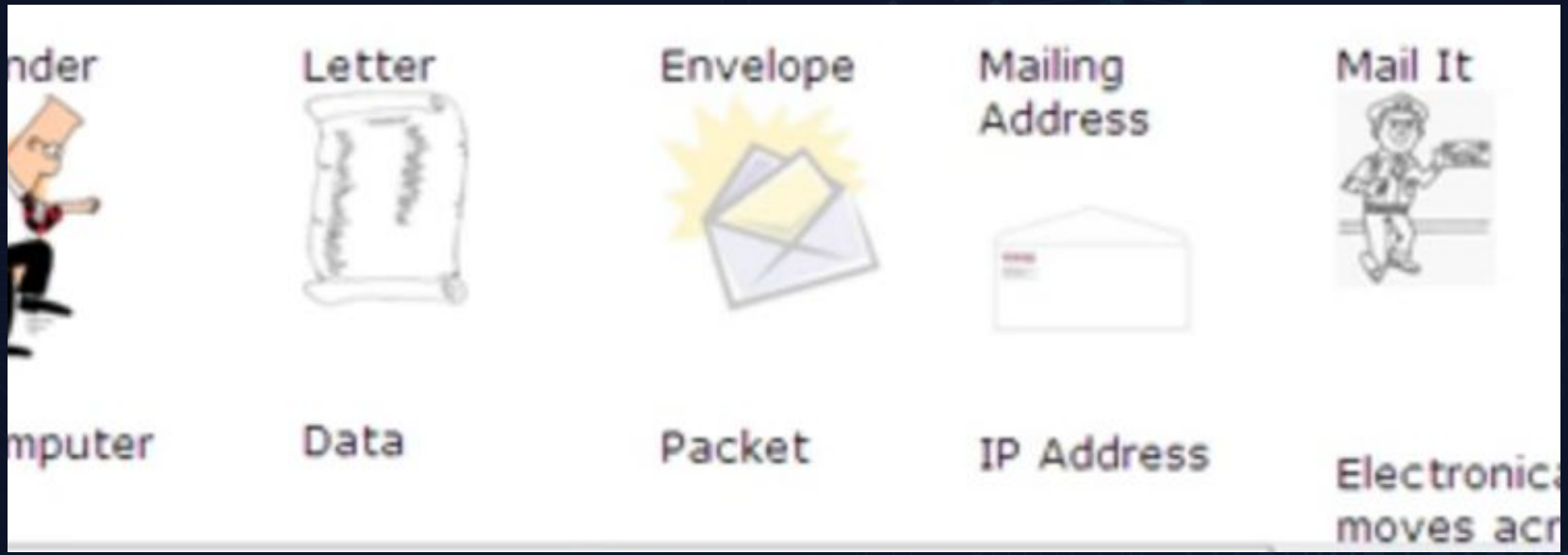
PROTOCOL IN ACTION

Sequence of Communication Steps

01. Bob says goodbye to Alice
02. Alice says goodbye to Bob
03. Bob asks the time
04. Alice tells the time
05. Bob thanks and says goodbye
06. Alice says hello

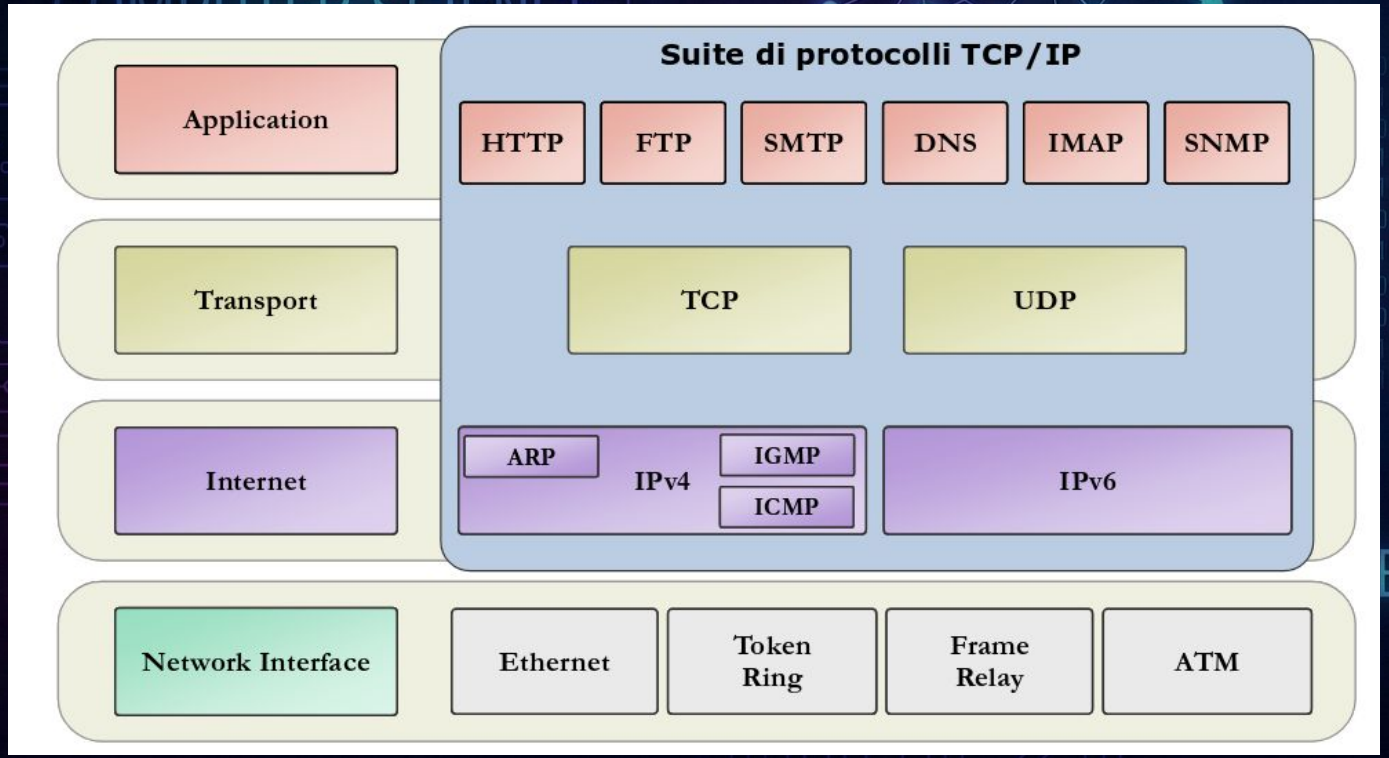
TCP/IP: Basic Operation

The postal mail analogy for data packet transmission



ACKNOWLEDGEMENT And finally a receipt **(ACK)** is sent back to confirm delivery.

TCP/IP Architecture



TCP/IP: Basic Operation

Key mechanisms behind network data transmission and layering

01

Layering Concept

The HTTP protocol is built on top of TCP.

A browser doesn't need to handle TCP implementation details, only that it functions reliably.

02

Packetization

Data from application layer is fragmented into smaller packets.

Packets reassemble at destination, though each packet can follow different network paths.

03

TCP Headers

TCP attaches control metadata to every packet.

Includes sequence numbers to verify packet ordering and complete data integrity.

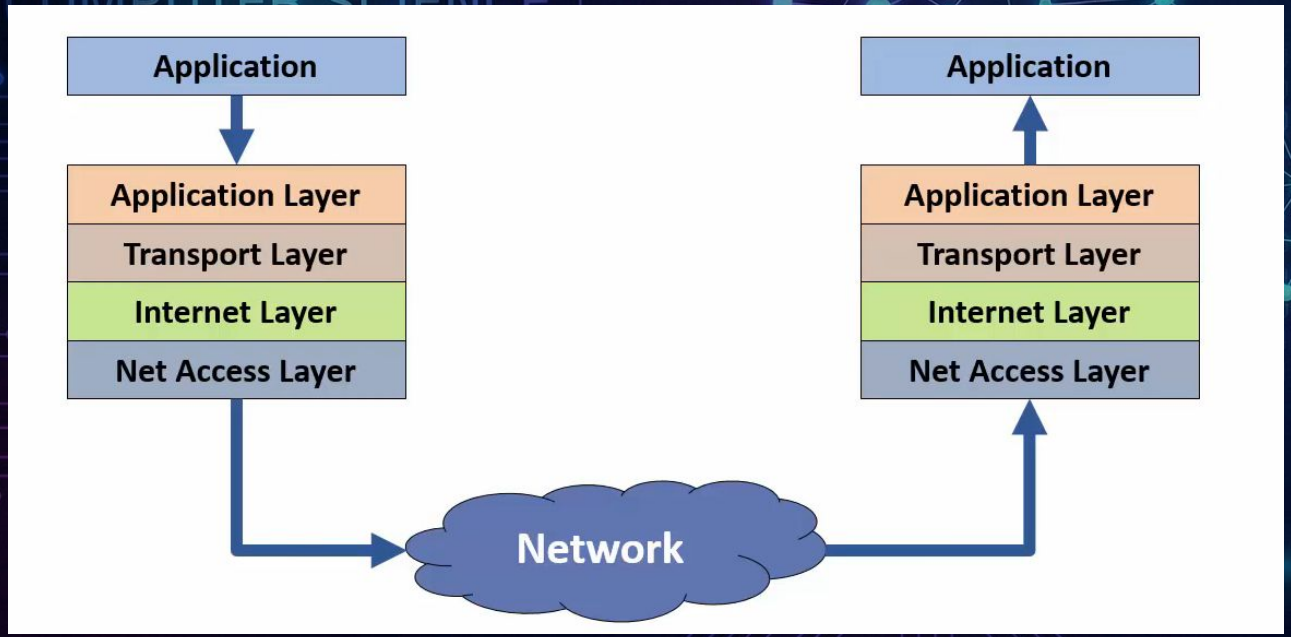
04

IP Routing

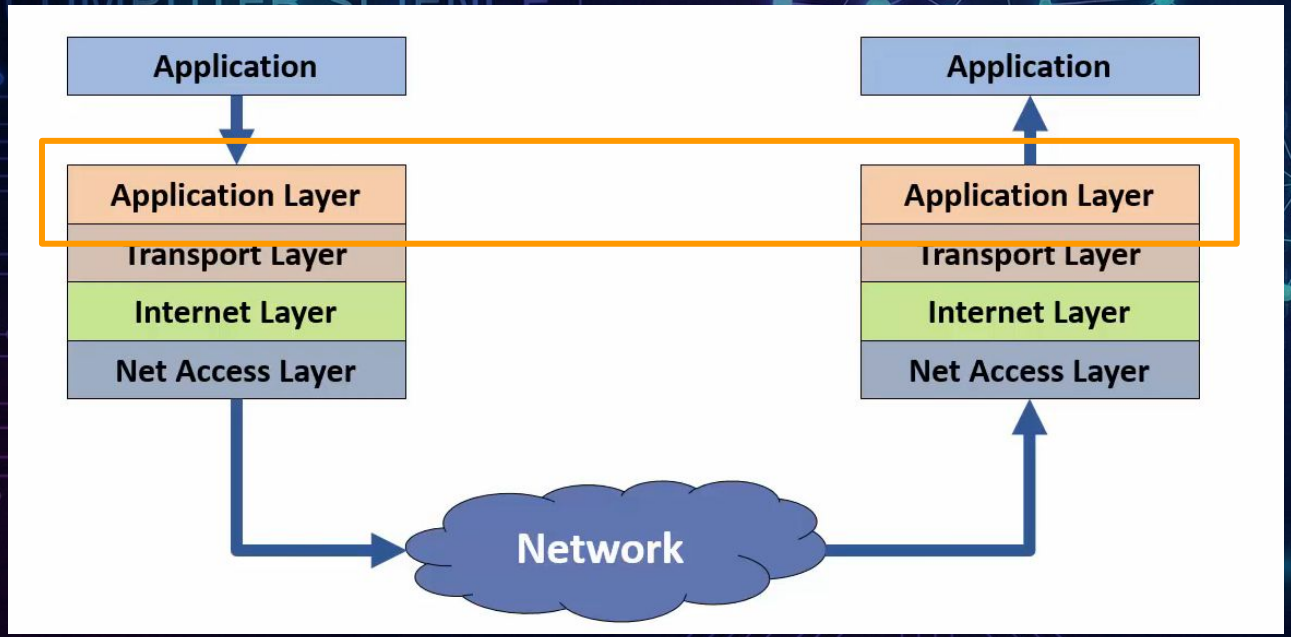
Packets pass down to the network layer.

IP routes each packet dynamically across nodes to reach the destination host efficiently.

TCP/IP


Application
Application Layer
Transport Layer
Internet Layer
Net Access Layer
Network
Application
Application Layer
Transport Layer
Internet Layer
Net Access Layer

TCP/IP


Application
Application
Application Layer
Application Layer
Transport Layer
Transport Layer
Internet Layer
Internet Layer
Net Access Layer
Net Access Layer
Network

TCP/IP Protocol Suite

Application Layer

Protocols & Data Exchange

Provides direct high-level network services for applications to communicate seamlessly.

Key Protocols:

HTTP • FTP • DNS • SMTP

At this layer, applications exchange **messages** directly without managing underlying delivery details.

User Interface & Sessions

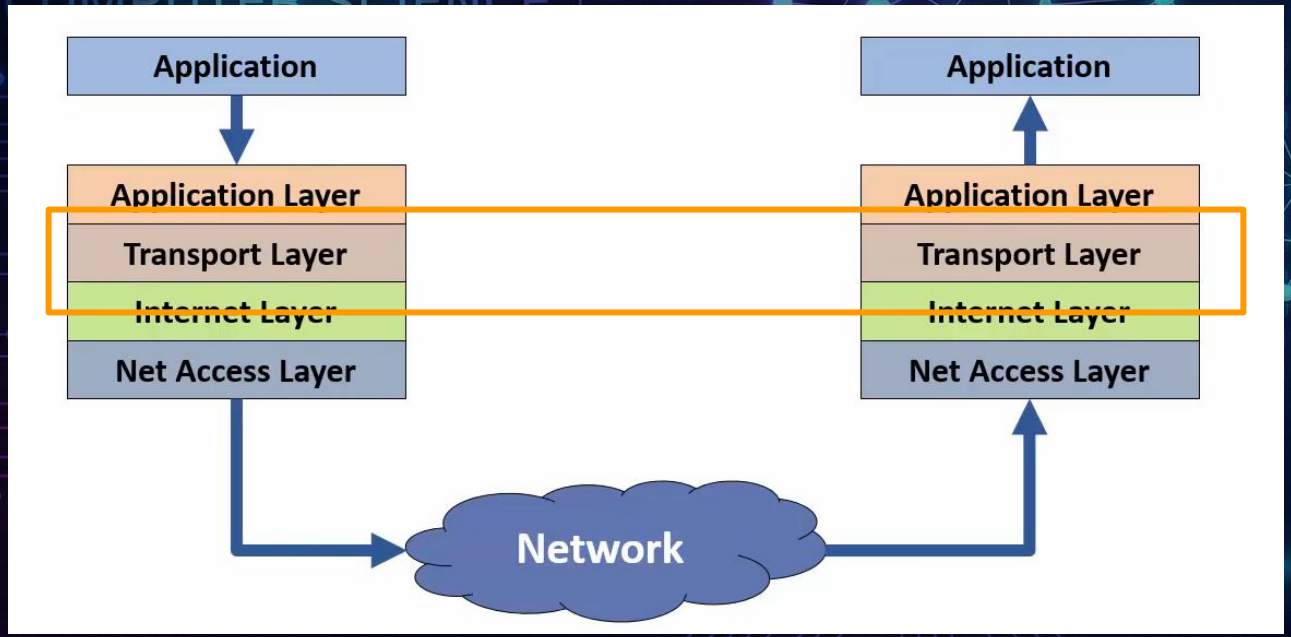
Acts as the primary interface between end-user software and network services.

Example Interaction:

Consulting a web page triggers protocols to manage sessions between browser (client) & web server.

Handles process-to-process dialogue and user-facing network operations.

TCP/IP


Application
Application Layer
Transport Layer
Internet Layer
Net Access Layer
Network
Application
Application Layer
Transport Layer
Internet Layer
Net Access Layer

TCP/IP Protocol Suite

Transport Layer

The Transport Layer is responsible for **process-to-process delivery** of data. It takes data from the Application layer, breaks it down, and ensures it gets to the correct application on the destination host.

Data Units

Segments

The unit of data exchanged at this level is called a **Segment** (or **Datagram** in UDP).

Service Provider

Upper Layer Support

Offers transport services directly to the upper **Application Layer** (HTTP, FTP, etc.).

Core Protocols

TCP & UDP

- **TCP:** Transmission Control Protocol
- **UDP:** User Datagram Protocol

TCP/IP PROTOCOL SUITE

Transport Layer & TCP Operations

Transport Layer: At this level two hosts exchange **segments** to offer transport service to the application layer. To manage **multiple sessions active at the same time**, TCP and UDP use different port numbers (logical ports).

CORE PROTOCOLS & SERVICES

TCP & UDP Protocols

- **Basic Protocols:** At this level we find two main protocols, **TCP** and **UDP**.
- **Data Exchange:** Two hosts exchange data formatted into segments.
- **Logical Ports:** Uses unique port numbers to route data for active sessions simultaneously.

RELIABLE TRANSMISSION

TCP Mechanism & ACK

- **Time Counter:** For each window of packets sent, TCP starts a time counter.
- **ACK Signal:** The recipient sends an **ACK** if it has received the packet successfully.
- **Data Resending:** If the sender does not receive an ACK before the timer expires, it takes care of resending data.

TCP/IP PROTOCOL SUITE

Transport Layer: Port Addressing & Multiplexing

Ports are 16-bit numbers (0–65535) used to direct network traffic to specific applications or processes on a host. They enable **multiplexing**—allowing multiple application sessions (e.g., Web, Email) to run concurrently.

PORT RANGES

Classification

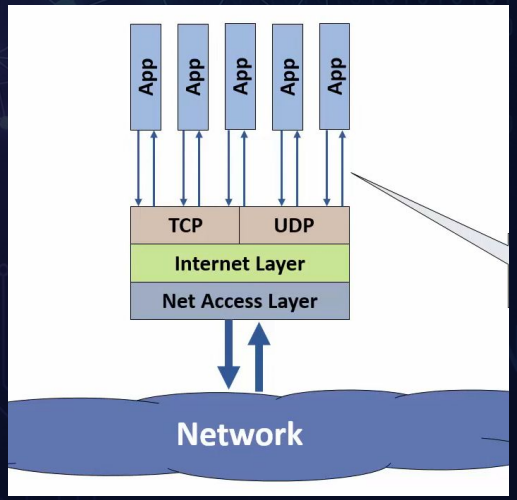
- **Well-Known (0–1023):**
  Reserved for system services.
  • HTTP: 80 | HTTPS: 443
  • SSH: 22 | FTP: 21
- **Registered (1024–49151):**
  Assigned to user apps & services (e.g., MySQL 3306).
- **Dynamic (49152–65535):**
  Ephemeral ports used for client connections.

MULTIPLEXING

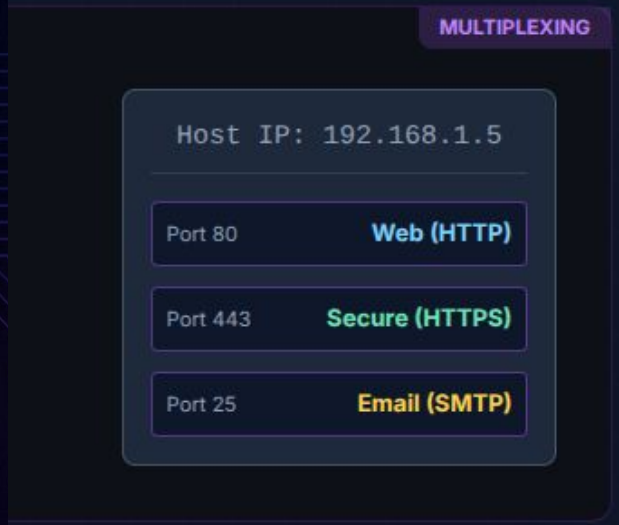
Sockets & Routing

- **Network Socket:**
  Combination of **IP Address + Port Number** (e.g., 192.168.1.5:80) to pinpoint host processes.
- **Multiplexing:**
  Gathers data from different sockets, encapsulates it into TCP/UDP segments.
- **Demultiplexing:**
  Delivers received segments to the correct application socket via port number.

ARCHITECTURE DIAGRAM



The TCP/IP STACK: Transport Layer



How does a computer run a web browser, an email client, and a multiplayer game all at the same time on a single network connection?

The Apartment Analogy

The **IP Address** is like the street address of an apartment building (the Host).

The **Port Number** (Logical Port) is like the specific apartment number inside that building (the Application Session).

THE TCP/IP STACK: TRANSPORT LAYER

TCP Reliability Mechanisms

TCP (Transmission Control Protocol) is designed for absolute reliability. It ensures that every single segment arrives correctly and in the right order.

01

Windows

TCP sends packets in batches called "windows".

02

Time Counter

For each window/packet sent, TCP starts a countdown timer.

03

ACK

The recipient **MUST** send an ACK back if it successfully receives the packet.

04

Retransmission

If the sender doesn't receive an ACK before the time expires, it assumes the packet was lost and resends the data.

THE TCP/IP STACK: TRANSPORT LAYER

UDP Protocol & Mechanics

UDP (User Datagram Protocol) favors speed over reliability. It does not set up a connection, does not track timers, and does not require ACKs.

01

Speed Focus

UDP favors speed over reliability and operates without connection setup overhead.

02

No Timers / ACKs

Does not track countdown timers and does not require ACKs from the recipient.

03

"Fire & Forget"

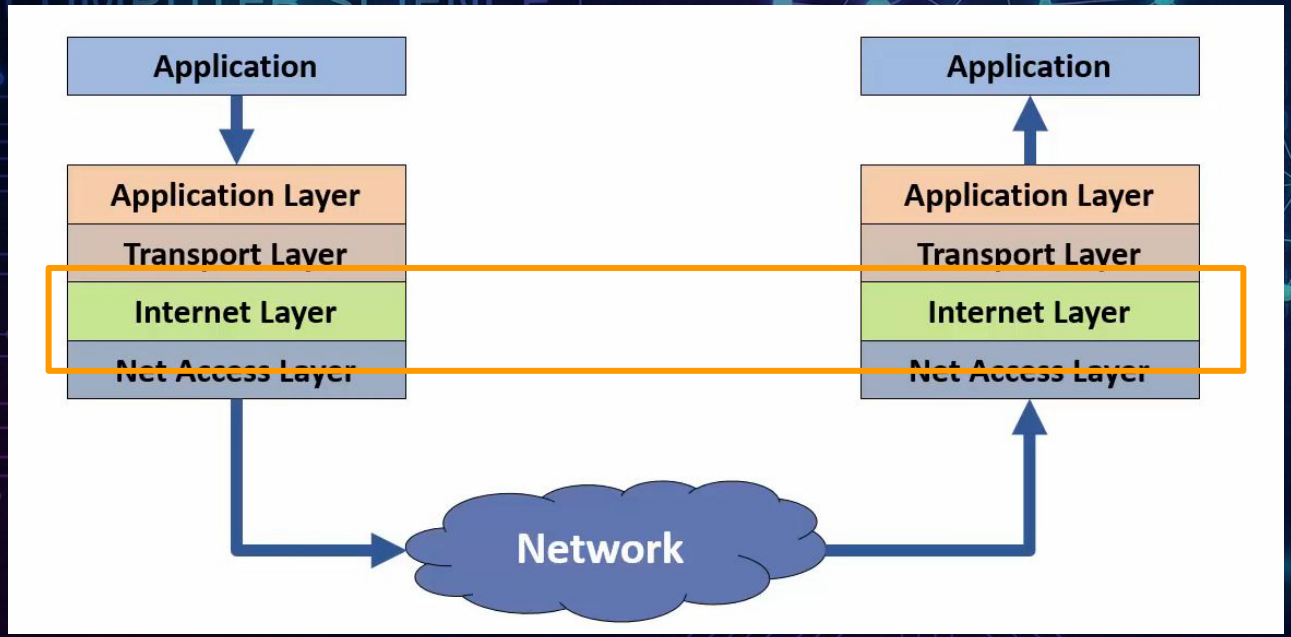
UDP simply blasts the data segments out to the recipient as fast as possible.

04

No Resending

If a packet gets lost along the way, it stays lost. The sender never resends it.

TCP/IP


Application
Application Layer
Transport Layer
Internet Layer
Net Access Layer
Network
Application
Application Layer
Transport Layer
Internet Layer
Net Access Layer

THE TCP/IP STACK: NETWORK LAYER

IP Protocol & Core Responsibilities

IP Protocol (Internet Protocol) operates at this layer. While Transport layer manages sessions between applications, Network layer handles actual delivery of data from one physical machine to another across multiple networks.

01

Addressing

Assigning a unique identity to every node. Ensures each connected device on the network can be unambiguously located.

02

Routing

Finding the best path through the internet maze. Dynamically selects path to forward data packets across multiple interconnected routers.

THE TCP/IP STACK: NETWORK LAYER

IP Address Fundamentals

Analogy: Just like the postal system needs a valid street address to deliver a letter, the internet needs IP Addresses.

01

Unambiguous Identification

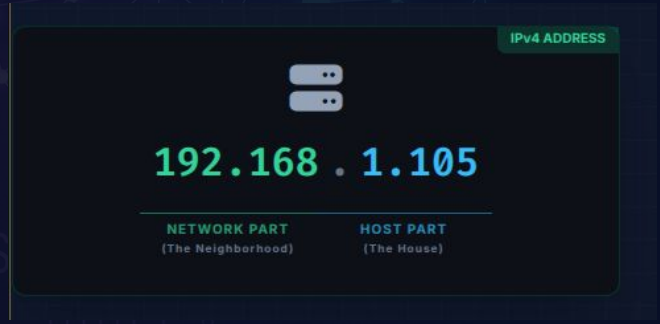
Each node (computer, phone, server, smart fridge) connected to a network is unambiguously identified by a unique IP address.

02

Structured Hierarchy

The address isn't random. It contains information about which specific sub-network the device is on, allowing routers to know which general direction to send the data.

IPv4 ADDRESS ANATOMY



IP Addresses IPv4 vs IPv6 Architecture

IPv4 Standard

32-bit address space divided into four 8-bit octets

Example IPv4 Format:

`192 . 167 . 12 . 66`

- **Structure:** 4 groups of 1 byte (0 to 255 per octet).
- **Hierarchy:** Initial octets specify the Subnet, while the final number identifies the specific Host machine.
- **Total Capacity:** 256^4 = ~4.29 Billion distinct addresses.

IPv6 Next Gen

128-bit massive capacity standard

Address Expansion:

2^{128} Addresses

(~3.4×10^{38} total)

- **128-bit Width:** Provides exponential scaling over IPv4.
- **IoT Ready:** Built for Internet of Things, smart devices, and modern web connectivity.

IP addresses

ADDRESSING: IPV4 VS. IPV6

IPv4 (Legacy)

- **Structure:** 32-bit address (4 bytes).
- **Notation:** Dotted decimal (e.g., 192.168.1.1).
- **Capacity:** 2^{32} addresses (~4.3 Billion).
- **Limit:** Exhausted (Allocations depleted in 2011).

IPv6 (Future Standard)

- **Structure:** 128-bit address (16 bytes).
- **Notation:** Hexadecimal (e.g., 2001:0db8::).
- **Capacity:** 2^{128} addresses.
- **Scale:** 3.4×10^{38} (Undecillion).



The Subnet Mask (The Divider)

How does a computer know exactly where the Network part ends and the Host part begins? It uses a secondary number called the **Subnet Mask**.

- **The Filter:** The mask acts like a digital stencil placed over the IP address.

  The 255s: Wherever the subnet mask has a `255`, it tells the computer: *"This section belongs to the Network."*

  The 0s: Wherever the subnet mask has a `0`, it tells the computer: *"This section belongs to the Host."*

Standard Home Setup:

```
IP:   192.168.1.105
Mask: 255.255.255.0
```



Example: A Typical Home Network

Observe how the **Network Part** remains identical across all devices on the same Wi-Fi, while the **Host Part** uniquely identifies each piece of hardware.

Device	Full IP Address	Network Part	Host Part
Wi-Fi Router	`192.168.1.1`	`192.168.1`	`.1`
Your Phone	`192.168.1.105`	`192.168.1`	`.105`
Your Laptop	`192.168.1.106`	`192.168.1`	`.106`
Smart TV	`192.168.1.110`	`192.168.1`	`.110`

Example 2: A Corporate Network

SUBNET MASK: 255.255.0.0

10.50.120.5

NETWORK PART HOST PART

← The Shifted Boundary

By changing the mask to `255.255.0.0`, the dividing line moves to the left. Now, only the first two numbers define the network.

Massive Capacity

This leaves two full numbers for the Host Part. Instead of just 254 devices, this large corporate network can hold over **65,000** unique devices!

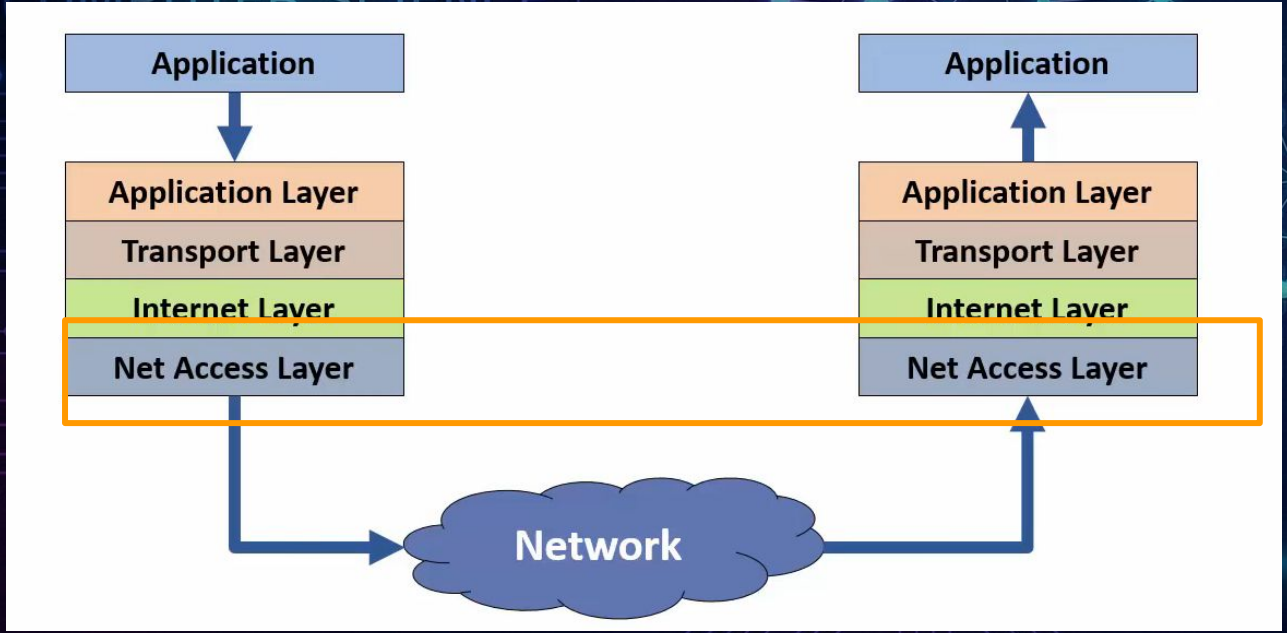
THE TCP/IP STACK: NETWORK LAYER

Routing & Dynamic Path Selection

Routing is the feature that allows the network to select the best path to follow to get data packets from the sender to the recipient across multiple interconnected routers.



TCP/IP


Application
Application Layer
Transport Layer
Internet Layer
Net Access Layer
Network
Application
Application Layer
Transport Layer
Internet Layer
Net Access Layer

THE TCP/IP STACK: NETWORK ACCESS LAYER

Link Layer & Physical Signal Transmission

Layer Specification: The TCP/IP model only specifies that below the IP layer there must be a network access layer that actually takes care of sending packets.

01. LINK LAYER

At the link layer, protocols decide how the message should be transferred for each stage of the path.

For example, how to get from the first host to the first router and so on.

IDENTIFICATION & ADDRESSING

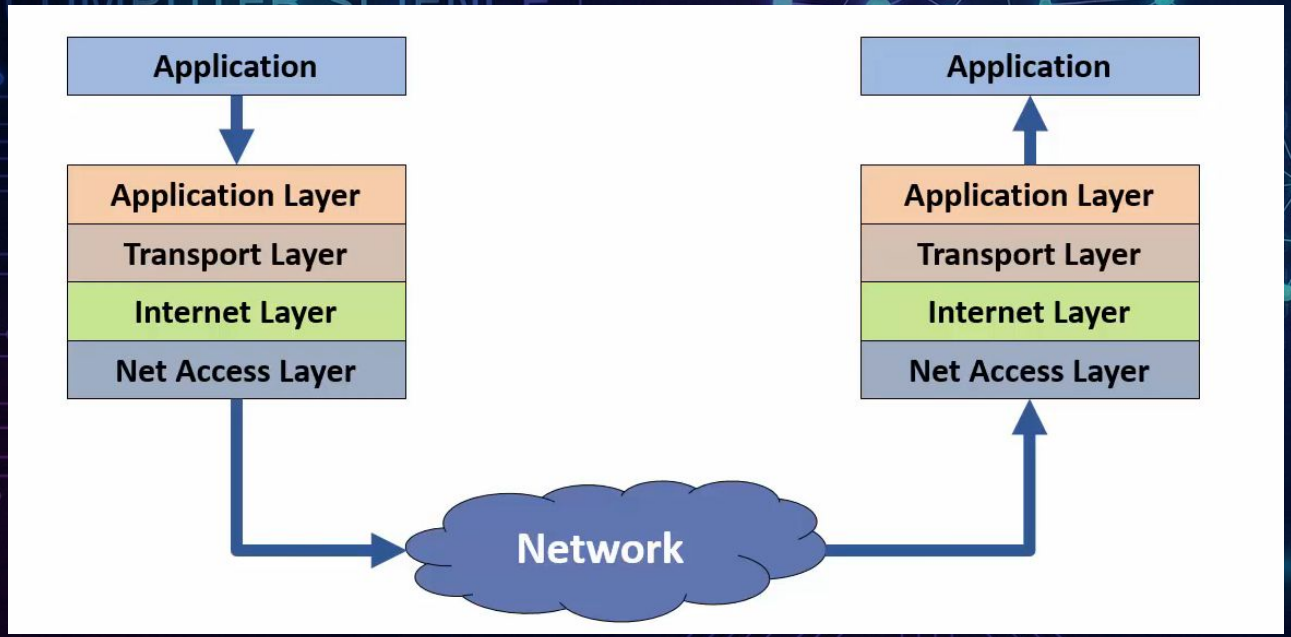
MAC and Ethernet addresses

02. PHYSICAL LEVEL

At the physical level, the data is then converted into physical transmission signals across hardware media:

- **Electrical Signals** (e.g. copper cabling)
- **Electromagnetic Signals** (e.g. wireless / RF)
- **Optical Signals** (e.g. fiber optics)

TCP/IP


Application
Application Layer
Transport Layer
Internet Layer
Net Access Layer
Network
Application
Application Layer
Transport Layer
Internet Layer
Net Access Layer

THE TCP/IP STACK

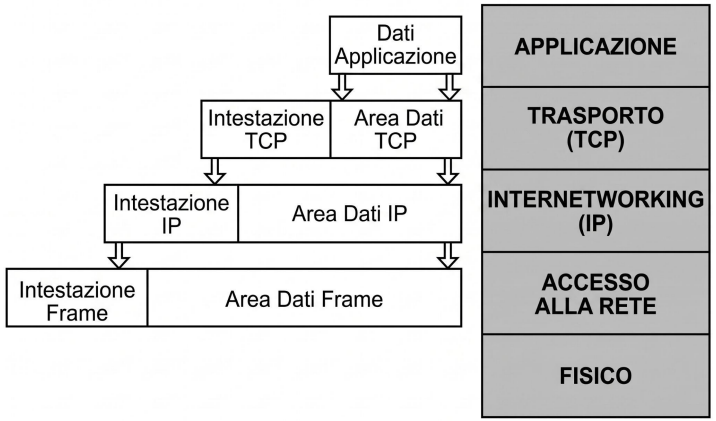
Data Encapsulation & Frame Creation

Encapsulation Process: When an application sends data, it is passed down through each layer one at a time until reaching the physical network. Each layer adds header information, creating a "network frame".

LAYER ADDITIONS

1. **Application:** Raw Data
2. **Transport:** + TCP Header
3. **Internet:** + IP Header
4. **Network Access:** + Frame Header

Each layer wraps the payload from above, building up headers sequentially before physical transmission.



TRANSPORT LAYER PROTOCOL

TCP Header & Logistics

The Logistics Layer: The TCP header is the inner layer managing specific conversation streams between applications, ensuring reliable and ordered data transfer.

KEY FUNCTIONS

1. Inner Layer Conversation

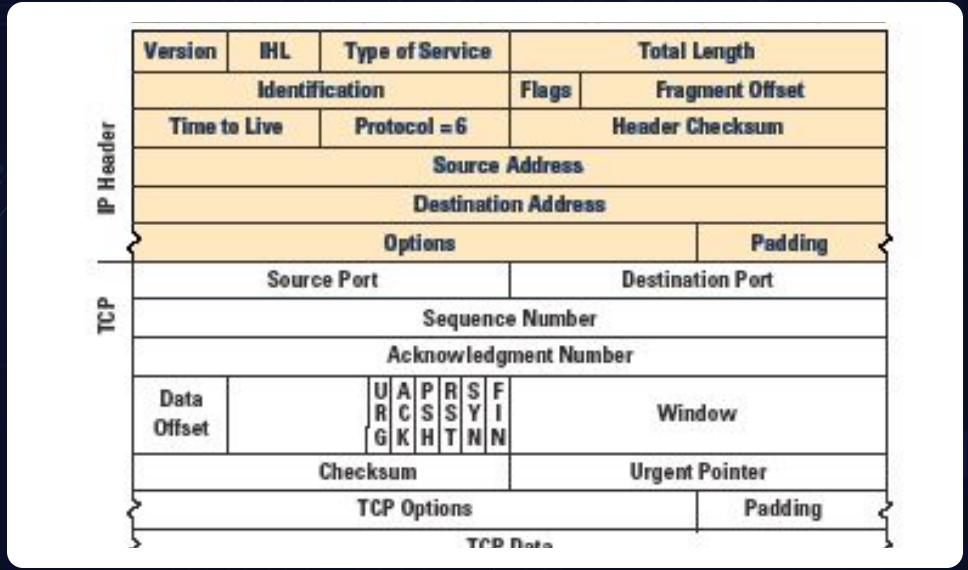
Manages end-to-end communication channels directly between active applications.

2. Port Numbers

Directs incoming data packets to the correct application (e.g., Chrome vs. Spotify).

3. Sequence Numbers

Reassembles packets back into exact order if network delivery arrives scrambled.



NETWORK LAYER PROTOCOL

IP Header & Network Routing

The IP Header (The Address): This is the outer layer that handles routing across the network. It contains the Source and Destination IP Addresses to ensure the packet finds the correct physical machine in the vastness of the internet.

KEY FUNCTIONS

1. Outer Layer Routing

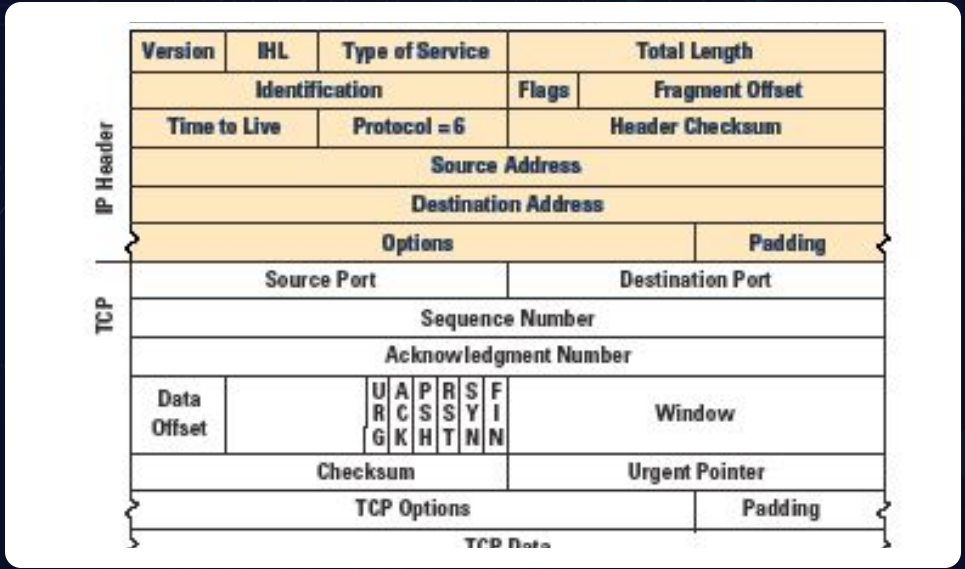
Encloses transport segments and manages routing across network infrastructure.

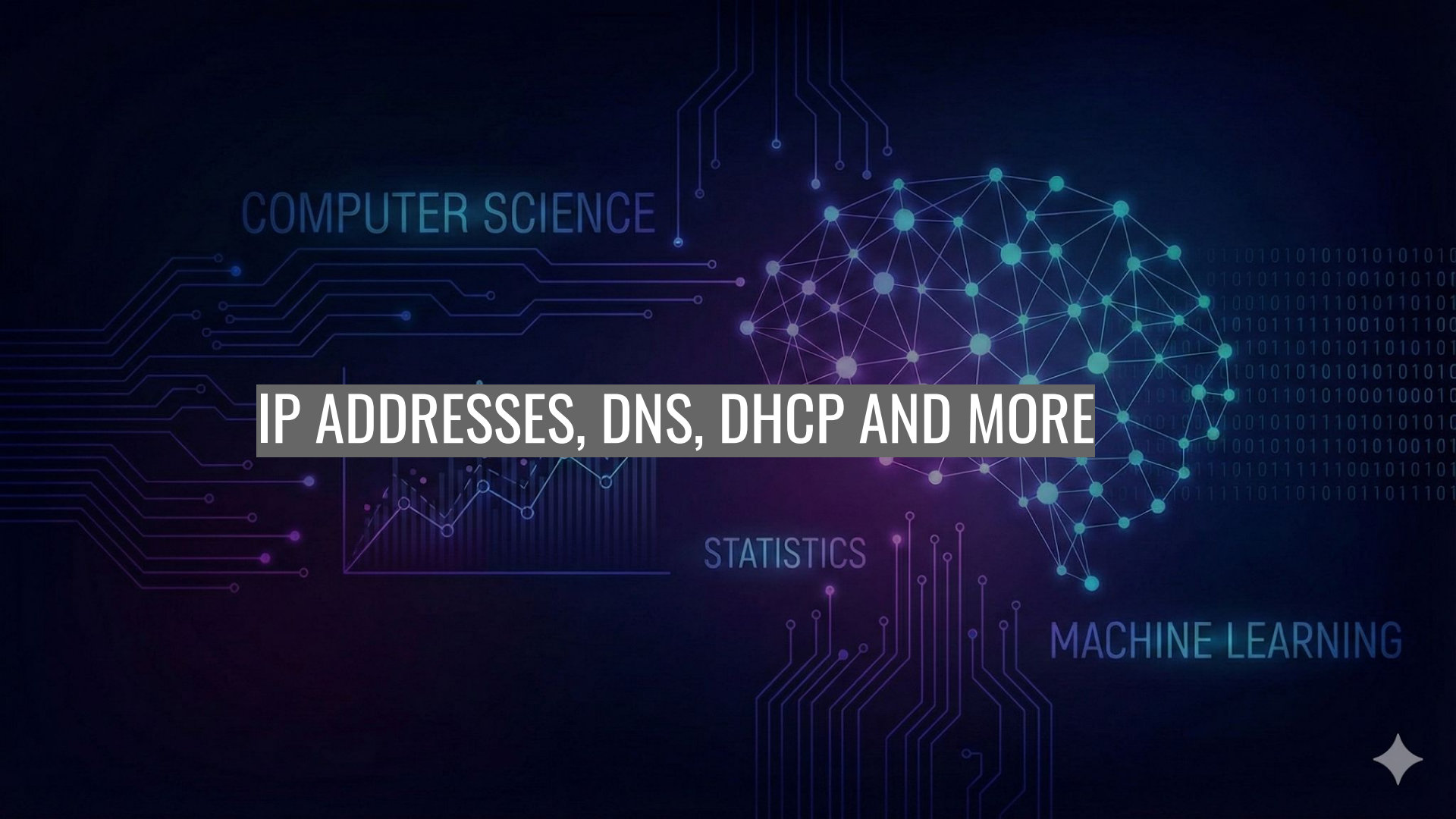
2. IP Addressing

Holds Source and Destination IP Addresses to target physical machines.

3. Global Packet Delivery

Ensures accurate end-to-end packet delivery through the internet.





IP ADDRESSES, DNS, DHCP AND MORE

NETWORK ARCHITECTURE & ROUTING

IP Addresses: Gateway

Default Gateway Role: Serves as the access point that forwards local network traffic to external destinations across remote networks and the internet.

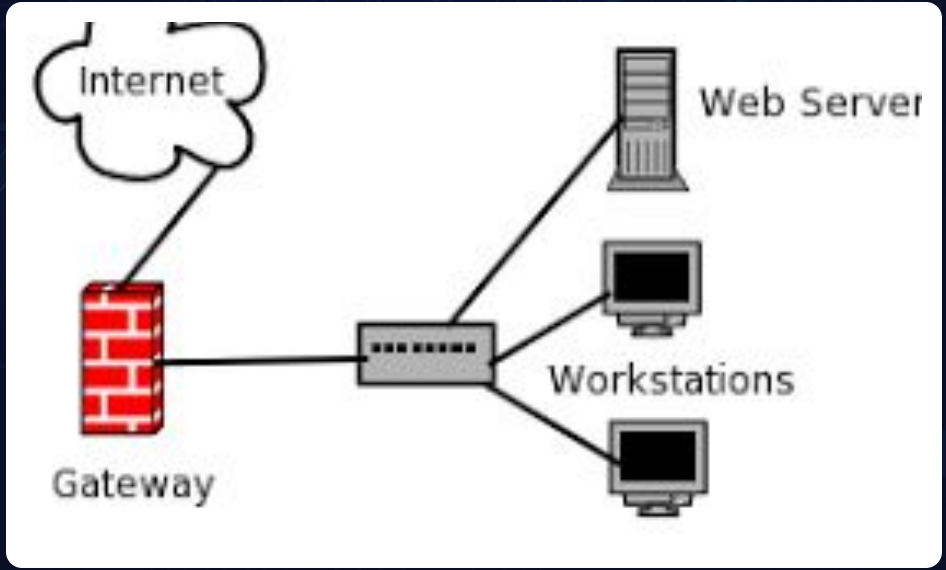
KEY FUNCTIONS

1. Packet Routing

The default gateway is actively used to route network packets to external destinations.

2. Automatic Configuration

DHCP is almost always used to automatically provide the client with the default gateway IP address.



Two Domains of Addressing

Public IP

The global identity of a network. It is completely unique across the entire Internet and allows external devices to find and communicate with your network.

- Assigned by your **ISP** (Internet Service Provider).
- Routable on the global internet.
- Think of it as your home's **Mailing Address**.

Example Output:
`203.0.113.42`

Private IP

The internal identity within a Local Area Network (LAN). It is only unique inside your specific home or office network, allowing local devices to talk to each other.

- Assigned by your local **Router** (via DHCP).
- Hidden from the outside internet.
- Think of it as your specific **Room Number**.

Common Range:
`192.168.1.105`

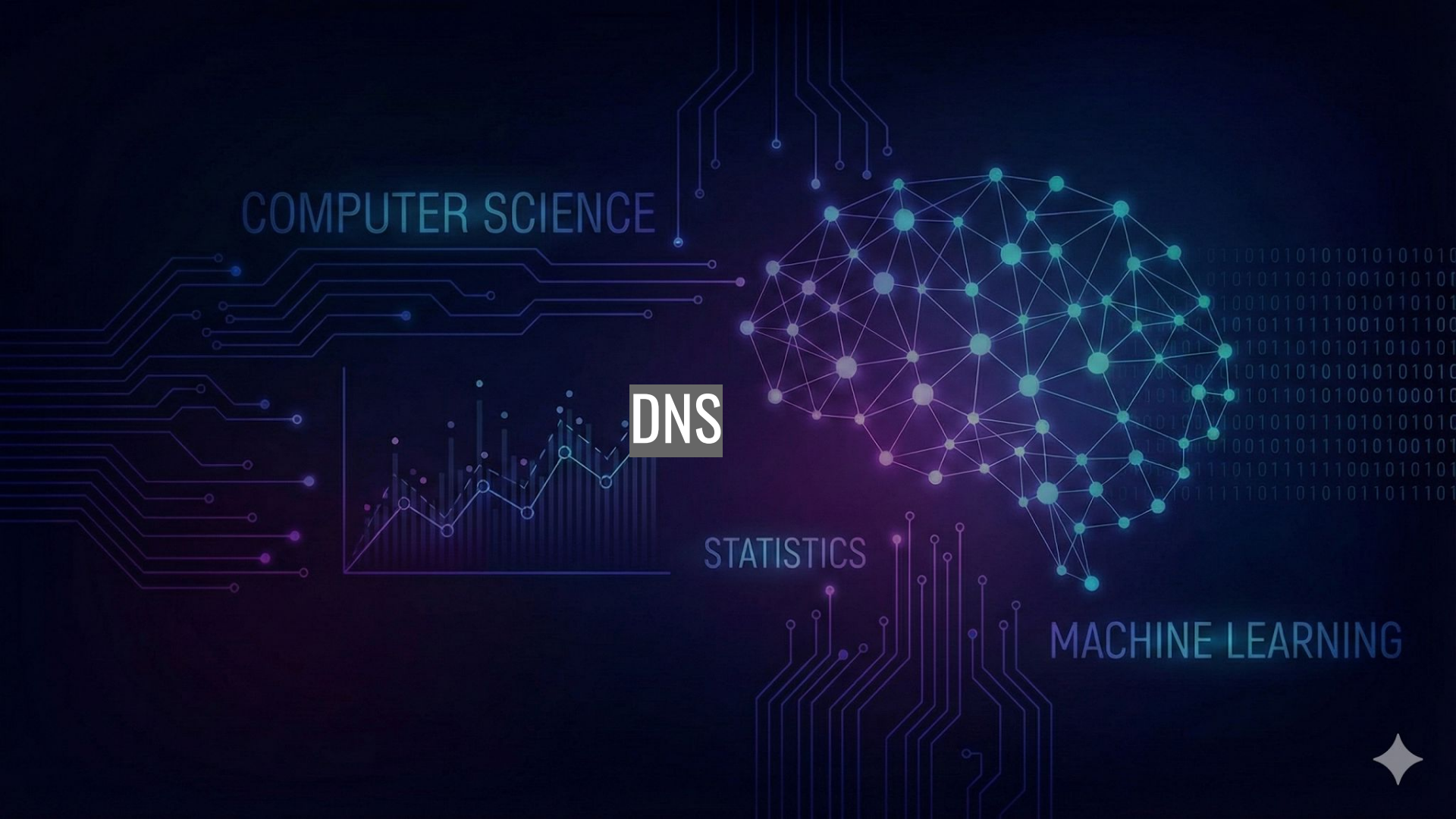
The Bridge: Network Address Translation (NAT)

COMPUTER SCIENCE

If Private IPs are hidden from the internet, how does your phone load a webpage? The answer is **NAT**.

- **The Translator:** Your router acts as a gateway. It takes requests from your internal `Private IP` and translates them to its single external `Public IP`.
- **Inherent Security:** Because external internet traffic only sees the router's Public IP, your individual devices remain shielded behind the NAT firewall.
- **IPv4 Conservation:** NAT allows dozens of devices in a home to share a single Public IP, drastically slowing the depletion of global IPv4 addresses.




COMPUTER SCIENCE
DNS
STATISTICS
MACHINE LEARNING

Networking Fundamentals

DNS (Domain Name System)

Core Concept

It's difficult for humans to memorize numbers, but much easier to memorize names. DNS bridges this gap by automatically translating human-friendly names to IP addresses back and forth.

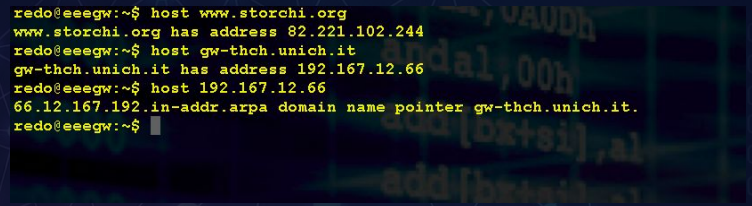
HUMAN MEMORY

www.storchi.org

MACHINE ADDRESS

82.221.102.244

Terminal — `host` command output



Bi-Directional Resolution

1. Forward Lookup: Maps domain name to server IPv4.
2. Reverse Lookup: Resolves IP address back to domain pointer.

Networking Fundamentals

DNS: Naming & Domain Hierarchy

Domain Name Structure

Hosts are identified by symbolic hierarchical names that uniquely locate them within a domain space.

HOSTNAME BREAKDOWN

gw-thch.unich.it

- **.it** — Top-Level Domain (ccTLD)
- **unich** — Subdomain / Organization
- **gw-thch** — Host Name within domain

Hierarchical Management

Names are assigned uniquely and managed administratively in a structured, multi-tier hierarchy.

Main Domain Categories

Generic / Organizational TLDs

.gov, **.edu**, **.com** — Associated primarily with organization types.

Country Code TLDs (ccTLDs)

.it, **.uk**, **.fr**, **.de** — National domains designated per country.

Networking Fundamentals

DNS: History & Core Architecture

Historical Context

Before DNS: The SRI-NIC Era

Before the DNS system was introduced, matching IP addresses to hostnames relied on a centralized, flat-file resolution method.

MANAGING ENTITY

SRI-NIC

CENTRAL RECORD FILE

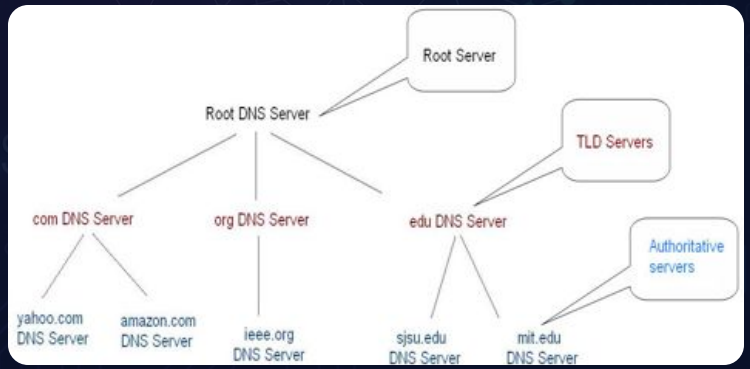
hosts.txt

All host-to-IP mappings were maintained in a single, centrally updated text file distributed across the network.

Modern Architecture

Distributed Database

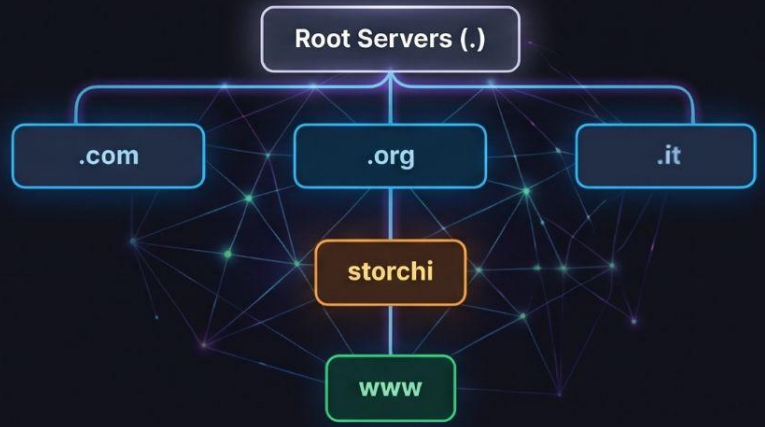
DNS is a **distributed database**. Information is partitioned across many servers, with each server responsible for a specific portion of the namespace called a **domain**.



The Hierarchical Tree Structure

Servers in the DNS system are not just a random flat list. They are organized in a strict **hierarchical tree structure**, read from right to left.

- **Root Level (.):** The top of the tree. They know where the TLD servers are.
- **TLD Level:** Top-Level Domains like `.org`, `.com`, `.it`.
- **Authoritative Level:** The specific domain (e.g., `storchi`). These hold the actual IP records.
- **Subdomains:** The final specific host (e.g., `www`).



NETWORKING FUNDAMENTALS

DNS Resolution: Step-by-Step Request Flow

DNS servers are organized in a **hierarchical tree structure**. Here is the flow when requesting **www.storchi.org**:

STEP 01

Request & Cache Check

Local Network / ISP

Your device asks your local DNS resolver. The resolver checks its local cache first for a recent answer.

STEP 02

The Root Server

Root DNS Server (.)

If uncached, the resolver queries a Root Server, which knows which servers manage the .org extension.

STEP 03

The TLD Server

Top-Level Domain (.org)

The Root server directs the resolver to the specific Top-Level Domain (TLD) Server for .org domains.

STEP 04

Authoritative Server

storchi.org DNS

The TLD server directs the resolver to the Authoritative Server registered specifically for storchi.org.

STEP 05

The Final Answer

IP Delivered & Cached

The Authoritative server returns the exact IP for www. Resolver caches it and passes it to your browser.

NETWORKING FUNDAMENTALS

High-Level Protocols and URFL

Different types of protocols are used, each designed for a specific network service:

WEB BROWSING

HTTP / HTTPS

HyperText Transfer Protocol

Provides access to hypertext pages (WEB) within the WWW (including encrypted HTTPS).

FILE MANAGEMENT

FTP

File Transfer Protocol

Dedicated protocol used to transfer and copy files across systems on a network.

EMAIL SERVICES

SMTP / POP3 / IMAP

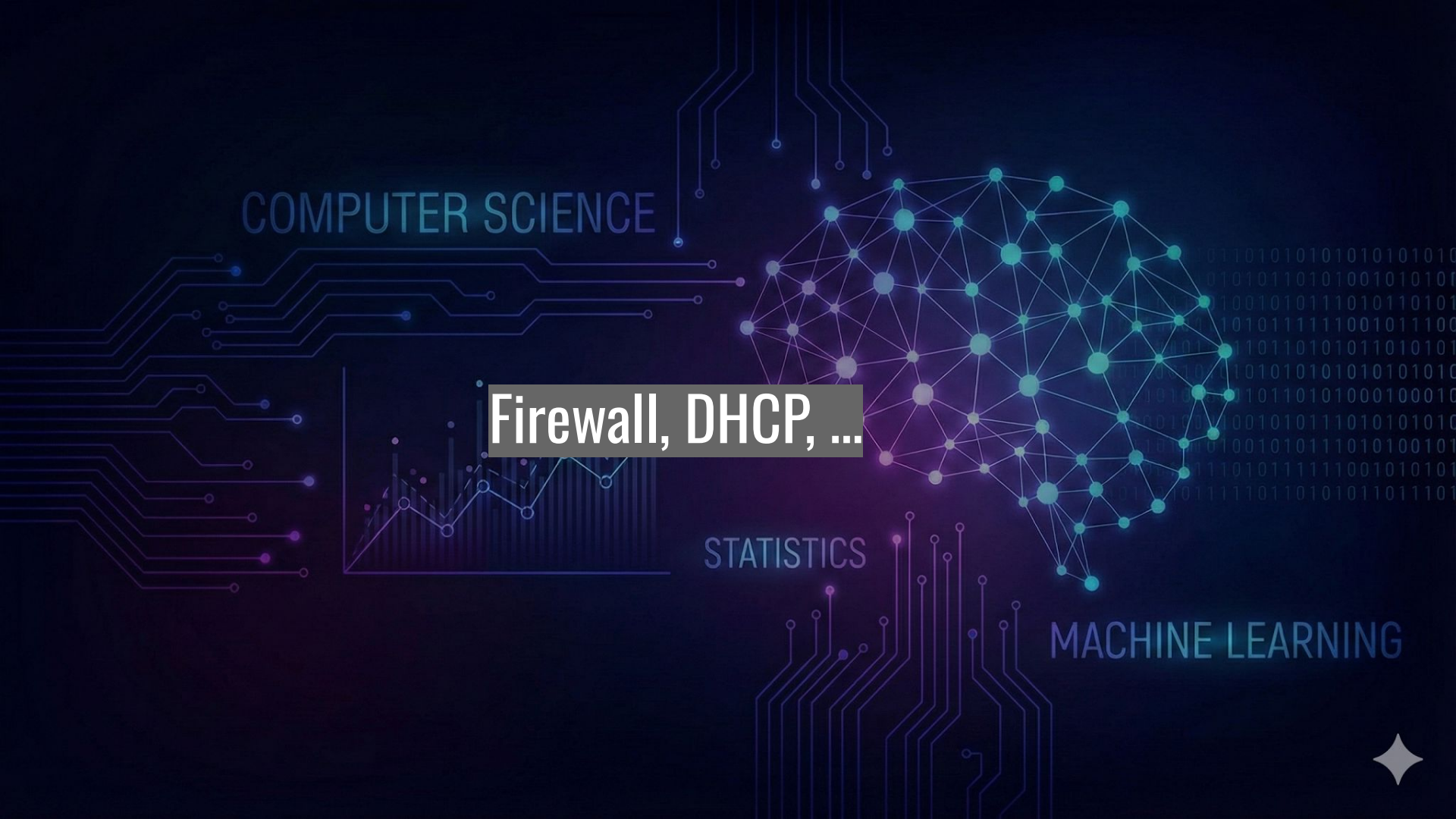
Mail Transfer & Retrieval

SMTP for sending emails. POP3 for downloading messages to a computer. IMAP for checking messages across multiple devices.

RESOURCE ADDRESSING

A resource on the network is therefore "identified" by the URL:

http://hostname.it/index.html


COMPUTER SCIENCE
Firewall, DHCP, ...
STATISTICS
MACHINE LEARNING

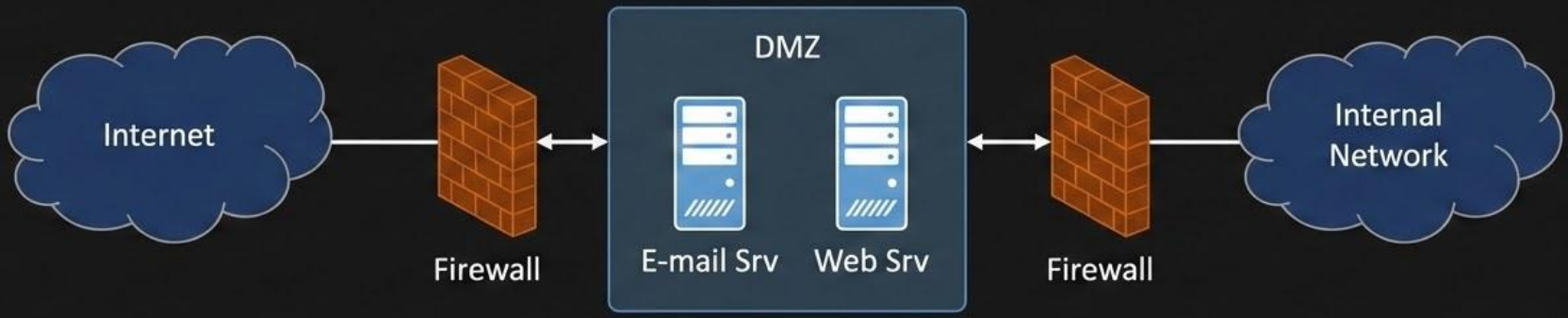
IP ADDRESSES: FIREWALL



The firewall is a network security system that controls all incoming and outgoing traffic according to well-defined rules.

IP Addresses: DMZ

A DMZ (demilitarized zone) is a physical or logical subnetwork that contains and exposes an external organization's services to an unprotected network, usually a larger network such as the Internet.



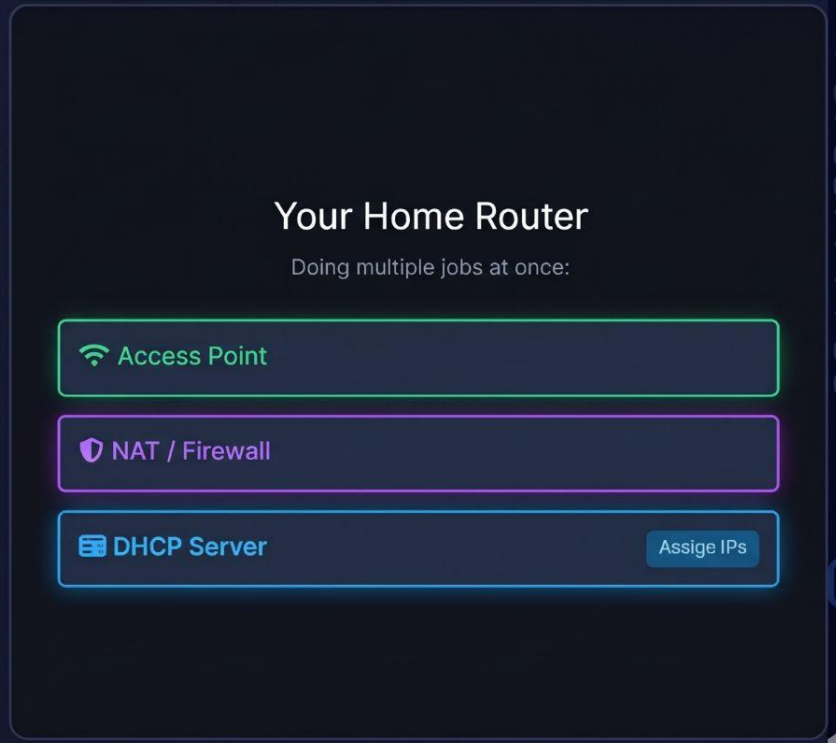
DHCP: The Home Router Example

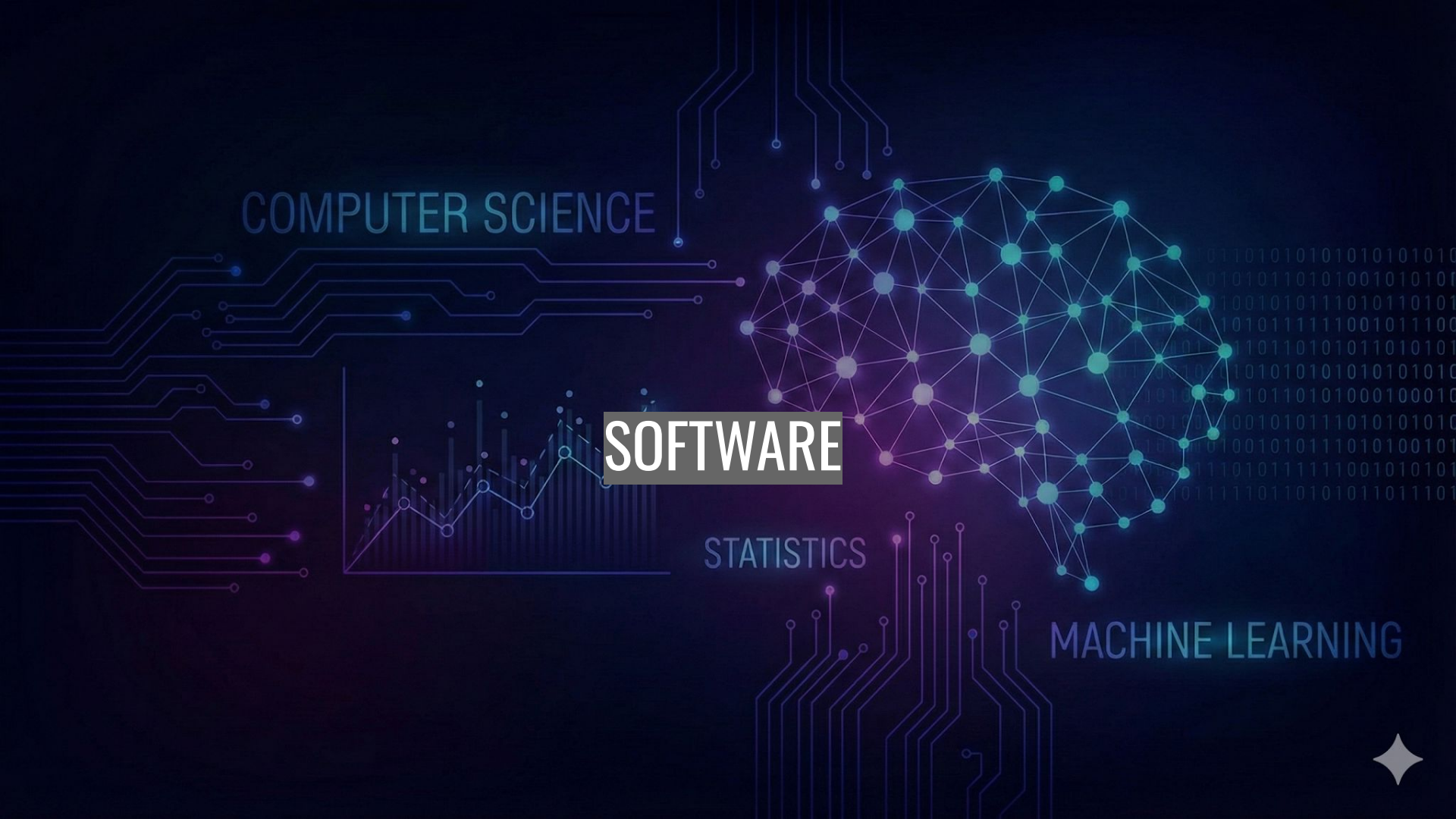
The Problem

Every device on a network needs an IP address. Manually typing an IP address into your phone, smart TV, and laptop every time you connect to Wi-Fi would be a nightmare.

The DHCP Solution

- The **ADSL/Fiber Router you have at home** has a built-in DHCP Server.
- It maintains a "pool" (database) of available Private IP addresses (e.g., 192.168.1.2 to 192.168.1.254).
- When a new device connects, the router automatically **leases** an IP address to it so it can access the network instantly.




COMPUTER SCIENCE
SOFTWARE
STATISTICS
MACHINE LEARNING

SOFTWARE CATEGORIES

Understanding the core distinction between system management and user applications

System Level

Basic Software

Dedicated to the management of the computer itself, for example the Operating System.

Key Functions & Examples

- Operating System (OS) control
- Hardware & peripheral management
- Low-level system utilities

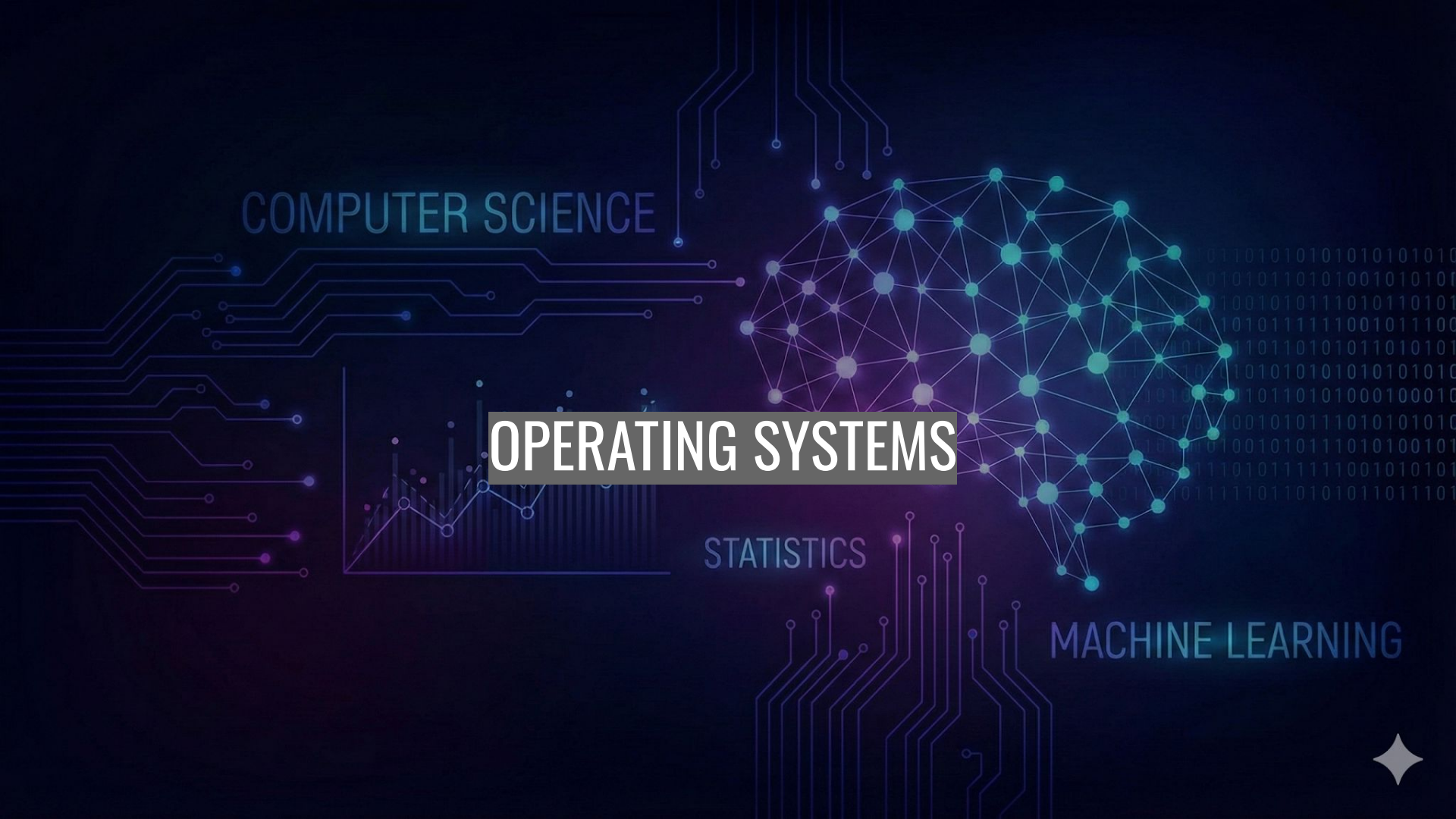
User Level

Application Software

Dedicated to the creation of specific applications, such as internet browsing, word processing, or other.

Key Functions & Examples

- Web browsers & networking tools
- Word processing & productivity suites
- Custom end-user software solutions


COMPUTER SCIENCE
OPERATING SYSTEMS
STATISTICS
MACHINE LEARNING

OPERATING SYSTEM

The core software bridge connecting users, applications, and system hardware

CORE DEFINITION

Low-Level Software Layer

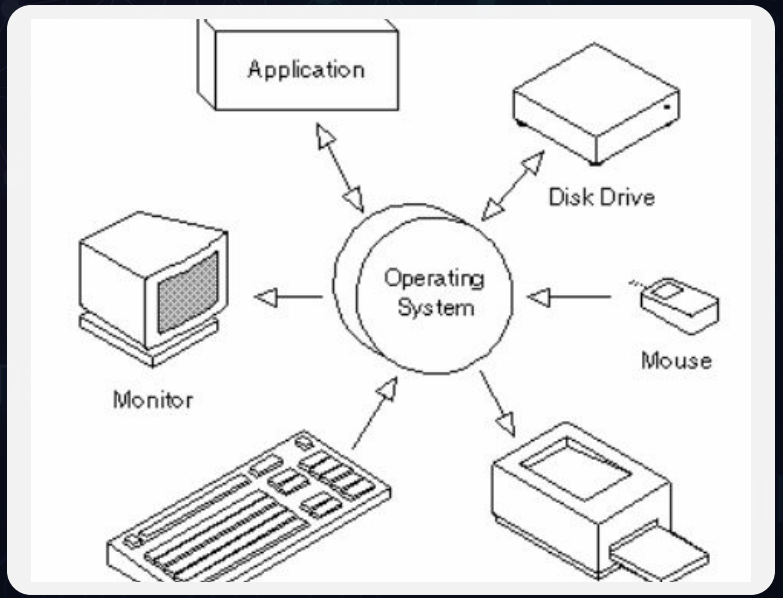
Essential system software that enables users and high-level applications to interact directly with hardware and managed data.

BASIC OPERATIONS

Key Responsibilities

- Keyboard input & display output handling
- File and directory management on disk
- Peripheral control (printers, input devices)

SYSTEM INTERACTION ARCHITECTURE



OPERATING SYSTEM SERVICES

Core functions providing execution environments, hardware abstraction, and connectivity

EXECUTION

Running Programs

OSs provide an environment where the user can run application programs without having to worry about memory allocation or CPU scheduling.

HARDWARE

I/O Operations

Every application requires some input and produces some output. The OS hides the details needed to handle these types of operations from the underlying hardware.

CONNECTIVITY

Communication

There are situations where two applications need to communicate with each other, whether on the same computer (different processes) or on different computers. The OS is responsible for managing this type of inter-process communication.

OPERATING SYSTEMS

They make it easier to write application programs that do not have to worry about specific hardware characteristics.



OPERATING SYSTEM ARCHITECTURE

Layered view of hardware & software components — provides an easy-to-use API & hides hardware details

USER LEVEL

Application Programs (Programmi Applicativi)

End-user applications, software suites, and graphical user interfaces

DEVELOPER LEVEL

System Libraries & Tools (Librerie e Tool di Sistema)

Provides programmers with standard API calls, user-space utilities, and system tools

KERNEL LEVEL

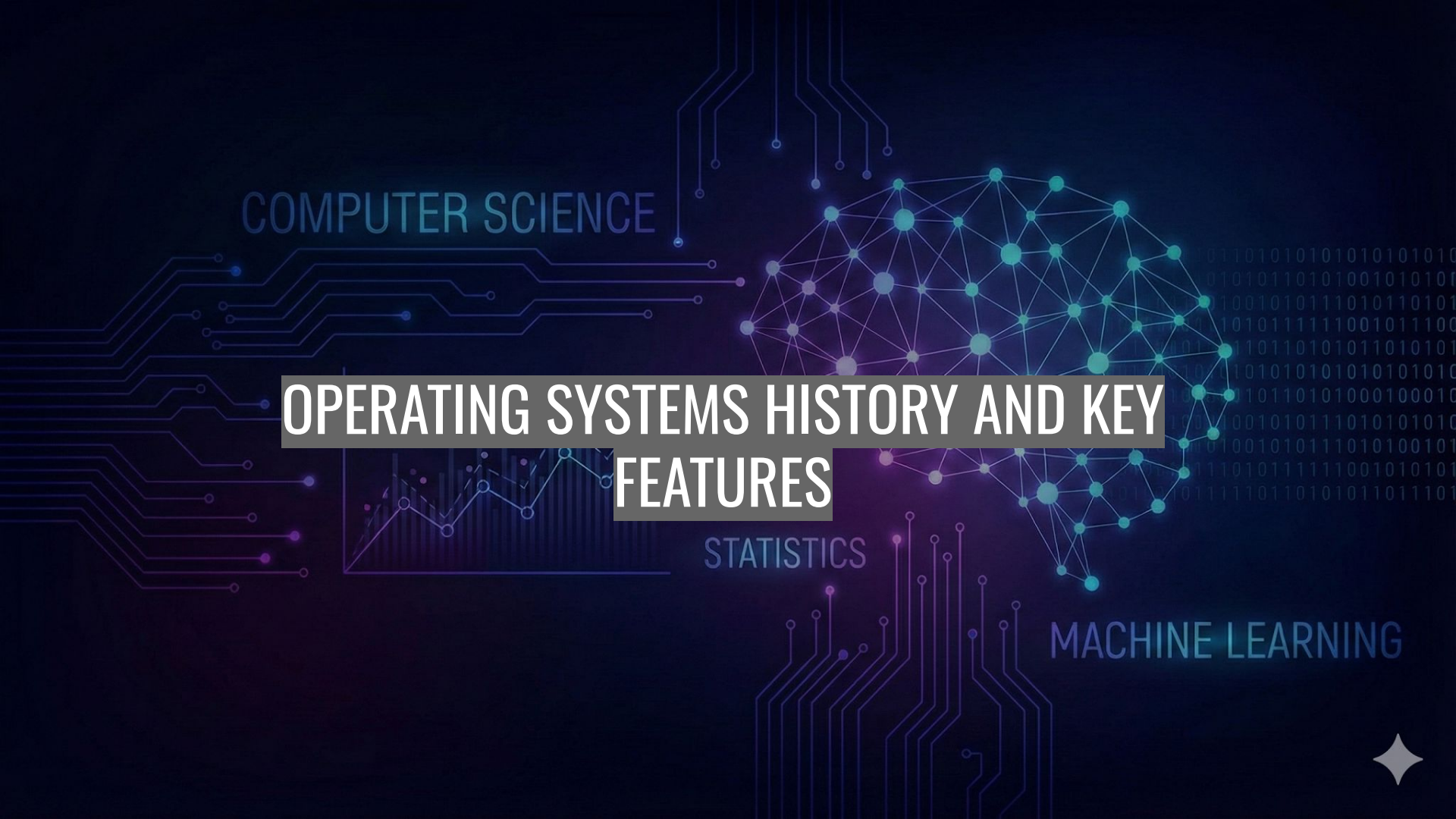
Operating System Core (Sistema Operativo)

Manages process scheduling, memory allocation, storage, and hides complex hardware details

PHYSICAL LEVEL

Computer Hardware (Hardware)

Physical CPU, RAM, storage devices, and I/O communication channels



OPERATING SYSTEMS HISTORY AND KEY FEATURES

OPERATING SYSTEM: EARLY HISTORY

From mechanical computing concepts to first-generation vacuum tube machines (1792–1955)

1792 – 1871 | MECHANICAL ERA

Babbage & Lovelace

- **Charles Babbage:** Tried to build an analytical and programmable mechanical machine designed without an operating system.
- **Lady Ada Lovelace:** The first female programmer in history (daughter of Lord Byron), who developed early algorithms for the machine.

1944 – 1955 | FIRST GENERATION

Valve Machines Era

- **Single Team Design:** Machines were designed, built, and programmed by a single group of people.
- **Machine Language:** Programmed directly in raw machine code, restricted solely to numerical calculations.
- **No OS Layer:** No distinction existed between designer, programmer, and operator.
- **Sequential Single-User:** Operator writes code, loads it, loads data, awaits output, then resets for the next job.
- **Lab Limited:** Computers were not expected to ever expand beyond specialized research laboratories.

OPERATING SYSTEM: BATCH PROCESSING ERA

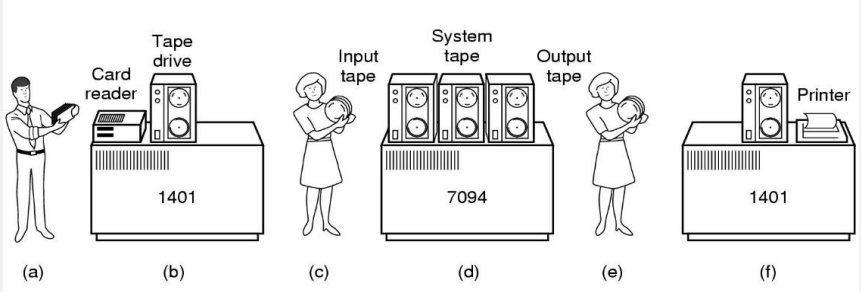
Introduction of magnetic tapes to minimize downtime between job executions

AUTOMATED JOB FLOW

Tape-Based Job Storage

- **Downtime Reduction:** Introduced to eliminate idle processor time between job setups.
- **Batch Loading:** Card readers transfer batches of jobs sequentially onto magnetic tapes.
- **Off-Line I/O:** Fast tape drives handle input/output, freeing mainframes from slow peripherals.

HISTORICAL BATCH PROCESSING WORKFLOW



OPERATING SYSTEM: TRANSISTORS ERA

Transistors, early high-level languages, and the first Resident Monitors (1955–1965)

1955 – 1965 | SECOND GENERATION

Hardware & Language Shifts

- **Transistors:** Enabled more reliable and cheaper computing machines.
- **Expanded Utility:** Started handling tasks beyond basic numerical calculations.
- **Role Separation:** Machine builders separated from programmers/users (user == programmer).
- **New Tooling:** Assembly & FORTRAN introduced along with punched cards.

EARLY OPERATING SYSTEMS

Resident Monitors

The earliest examples of operating systems created to automate setup and execution:

- **Machine Control:** Primary control over the machine hardware is handed directly to the monitor.
- **Job Handoff:** Control/check is passed sequentially to the job to be executed.
- **Automatic Return:** Once a job completes execution, control automatically reverts back to the monitor.

OPERATING SYSTEM: INTEGRATED CIRCUITS ERA

Integrated circuits (1965–1980) and the emergence of modern operating system features

1965 – 1980 | THIRD GENERATION

Languages & User Interface

- **Operator Disappears:** The figure of the operator as an interface towards the computer disappears.
- **High-Level Languages:** Programming is done primarily using high-level languages such as C.
- **Direct Operation:** Text editors and terminals arrive to allow direct operation.

MODERN OPERATING SYSTEM FEATURES

Core Architecture Innovations

- **Interactive:** Operating systems enable direct, real-time user interaction.
- **Multi-programming:** Multiple programs are kept in memory for efficient processing.
- **Time Sharing:** CPU time quanta slicing enables concurrent multi-user execution.

OPERATING SYSTEM: MULTIPROGRAMMING

Maximizing CPU utilization by keeping multiple active jobs in memory

EXECUTION & SCHEDULING

CPU Efficiency

- **Concurrent Memory:** Keep multiple jobs in memory at the same time.
- **Overlap Processing:** Use the processor while other jobs perform I/O operations.
- **OS Scheduler:** Manages CPU usage across jobs, eliminating CPU idle dead time.

SYSTEM REQUIREMENTS

OS Responsibilities

- **I/O Routines:** I/O subroutines and drivers must be provided directly by the OS.
- **Memory Allocation:** The operating system must manage and allocate memory for multiple concurrent jobs.
- **Resource Management:** The OS allocates I/O devices and system resources between active processes.

OPERATING SYSTEM: TIME-SHARING

Extending multiprogramming through CPU time slicing and multi-user protection

EXECUTION & SCHEDULING

Time Quanta & Context Switching

- **Multiprogramming Extension:** Essentially an extension of the multiprogramming concept.
- **Time Quanta Slicing:** CPU time is divided into intervals (quanta); process switches when quantum expires.
- **Context-Switching:** Fast and transparent, giving users the impression of parallel execution.

SYSTEM & RESOURCE MANAGEMENT

Protection & Memory Swapping

- **Multiple Users:** Processes concurrently running generally belong to different users.
- **System Protection:** Multiple users require robust memory and file system protection mechanisms.
- **Job Swapping:** Jobs are transferred dynamically from disk to memory and vice versa.

OPERATING SYSTEM: MULTI-USER

Enabling simultaneous access, dynamic resource sharing, and strict security isolation

CONCURRENCY & ACCESS

Simultaneous User Access

- **Concurrent Terminals:** Multiple users interact with the system at the same time via independent terminals.
- **Resource Sharing:** CPU time, main memory, and peripherals are dynamically allocated across users.
- **Independent Sessions:** Each user operates within an isolated session context, unaware of others.

SECURITY & ISOLATION

Protection & Authentication

- **User Authentication:** Secure access control mechanisms using credentials and role-based permissions.
- **Memory Isolation:** Prevents one user's processes from reading or overwriting another user's memory space.
- **File System Security:** Strict access permission modes (read, write, execute) to safeguard private data.

OPERATING SYSTEM: HISTORY

Key evolutionary milestones from early time-sharing to modern system foundations

1965 • MIT

CTSS

- **Full Title:** Compatible Time-Sharing System
- **Multiprogramming:** Early implementation of multi-programming concepts
- **Time-Sharing:** Pioneered CPU time-sharing across terminals

1965 • MIT / BELL / GE

Multics

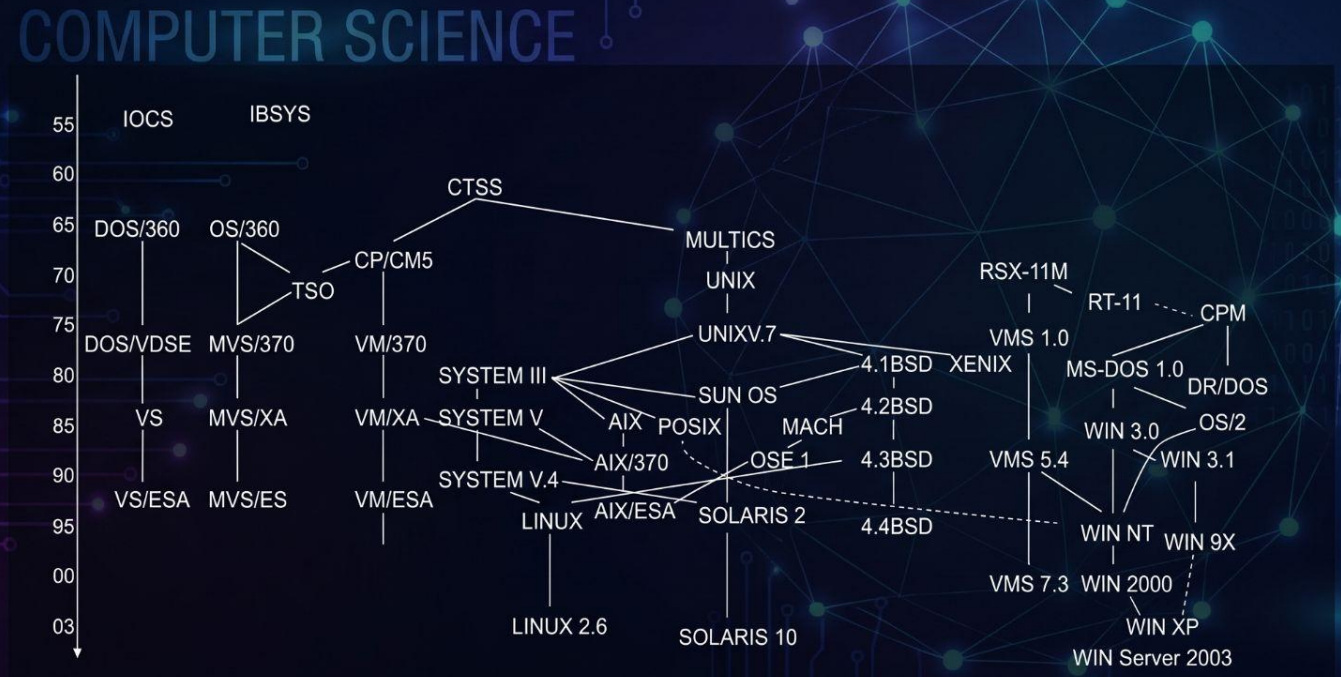
- **Process Abstraction:** Introduced the key concept of a system process
- **Core Influence:** Advanced OS structure and dynamic utility computing
- **OS Heritage:** Directly laid technical groundwork for Unix

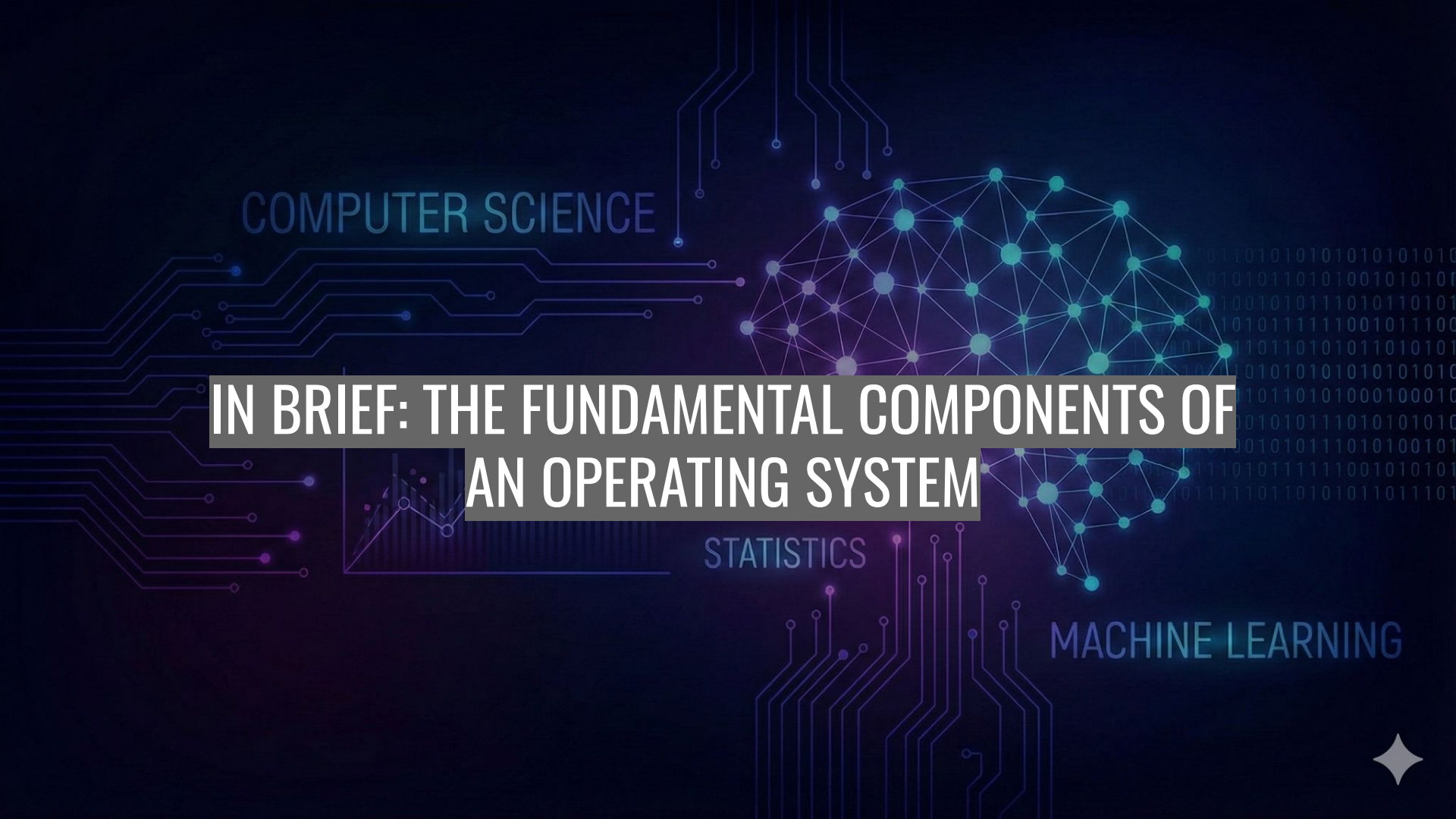
1970 • BELL LABS

Unix

- **Origins:** Derived directly from Multics & CTSS concepts at Bell Labs
- **C Transition:** Evolved from pure Assembly implementation into C
- **Architecture:** Expanded across hardware architectures, defining modern OS

Operating Systems: History





IN BRIEF: THE FUNDAMENTAL COMPONENTS OF AN OPERATING SYSTEM

Operating System: Process Management

What is a Process?

A process is basically **a running program, which therefore uses resources**

The OS must be able to:



01. Process Creation

Create and terminate the processes themselves



02. Communication

Manage communication between the processes themselves



03. Execution State

Suspend and resume processes

Operating System: Main Memory Management

What is Main Memory?

Main memory is essentially an **array of individually addressable bytes** , shared between CPU and I/O devices.

The OS must be able to:



01. Memory Tracking

Keep track of which memory areas are used and by which process



02. Space Allocation

Decide which processes to allocate memory areas to when free



03. Deallocation

Allocate and deallocate to free up memory space as needed

Operating System: Secondary Memory Management

What is Secondary Memory?

Computers are equipped with **secondary, non-volatile memory** of large size (larger than volatile main memory).

The OS must be able to handle the following activities:

01. Space Allocation & Deallocation

Allocate the required space and free it when no longer used

02. Device Access Scheduling

Manage device access scheduling (e.g. hard disks, CDs/DVDs, or USB sticks)

Operating System: Filesystem Management

What is a File & Filesystem?

A **file** is an abstract container for information. A **filesystem** is composed of files and directories that reference other files.

The OS must be able to handle the following tasks:



01. File & Directory Ops

Create and delete files and directories, and manipulate them



02. Filesystem Encryption

Encrypt the filesystem on secondary storage for data protection



03. Supported Standards

Support key formats such as ext3, ext4, NTFS, FAT32 ...

Operating System: Filesystem Management

Filesystem Role & Core Responsibilities

The **filesystem** manages disk space allocation, tracks data storage locations, and maintains essential file characteristics and metadata.

Key OS Management Functions & Data Structure:

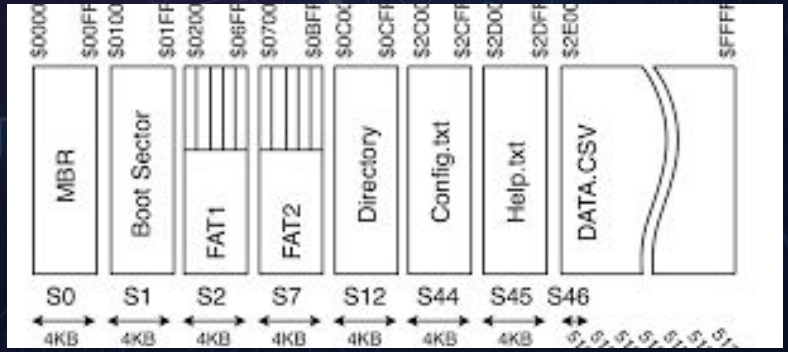
01. Disk Space Allocation

Manages allocation of disk space and maintains an index of stored file data locations.

02. File Attributes & Metadata

Tracks characteristics including access dates, user permissions, filenames, and properties.

03. Filesystem Layout & Index Structure Example



Operating System: I/O Management

I/O Architecture & Hardware Management

The **OS** must manage I/O operations and govern the interaction between software and hardware devices.

Key I/O Subsystem Responsibilities:



01. Device Driver Interface

Common interface for managing various device drivers uniformly



02. Driver Kernel Modules

Dedicated drivers (kernel modules) tailored for specific hardware



03. Buffering & Caching

Buffering & caching system to optimize data transfers

Operating System: Protection

Protection Architecture & Resource Control

The **software protection mechanism** manages and controls the access of various processes to shared system resources (e.g., memory).

Key Protection Responsibilities & Goals:

01. Usage Authorization

Distinguish access between authorized and unauthorized use across processes and users.

02. Protection Enforcement

Provide basic mechanisms required to specify controls and enforce system security policies.

Operating System: Networking

Networking Architecture & Inter-Process Communication

Networking is an essential component of an **OS**, allowing computers (or processes) to communicate and share resources.

Key Networking Capabilities & Protocols:



01. Basic Protocols

Provides foundational communication protocols such as TCP/IP, UDP for system transport.



02. High-Level Services

Delivers high-level communication services such as shared file systems (SMB, NFS).

Operating System: Command Interpreter

User Interface & System Control

Operating systems provide a **user interface** to start or end programs and interact with basic OS components, such as the filesystem.

Two Essential Interface Types:



01. Graphical Interface

Graphics-based user experience incorporating visual icons, windows, and graphical elements.



02. Command Line

Textual interface allowing direct command input, shell scripting, and text-based system interaction.

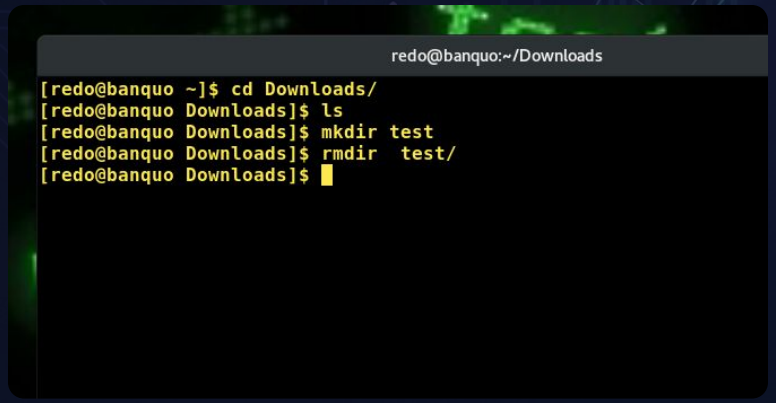
Operating System: Command Interpreter

User Interface & System Control

Operating systems provide two distinct user interfaces to execute programs, manipulate files, and interact with the core filesystem.

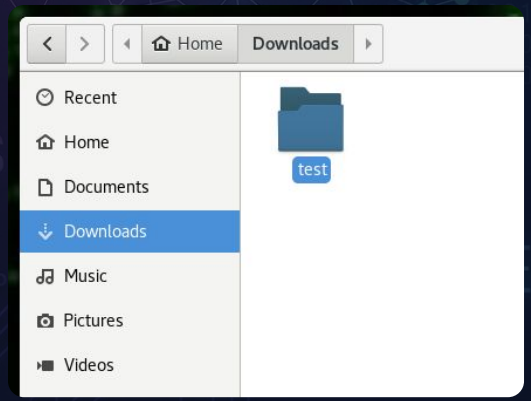
01. Command Line Interface (CLI)

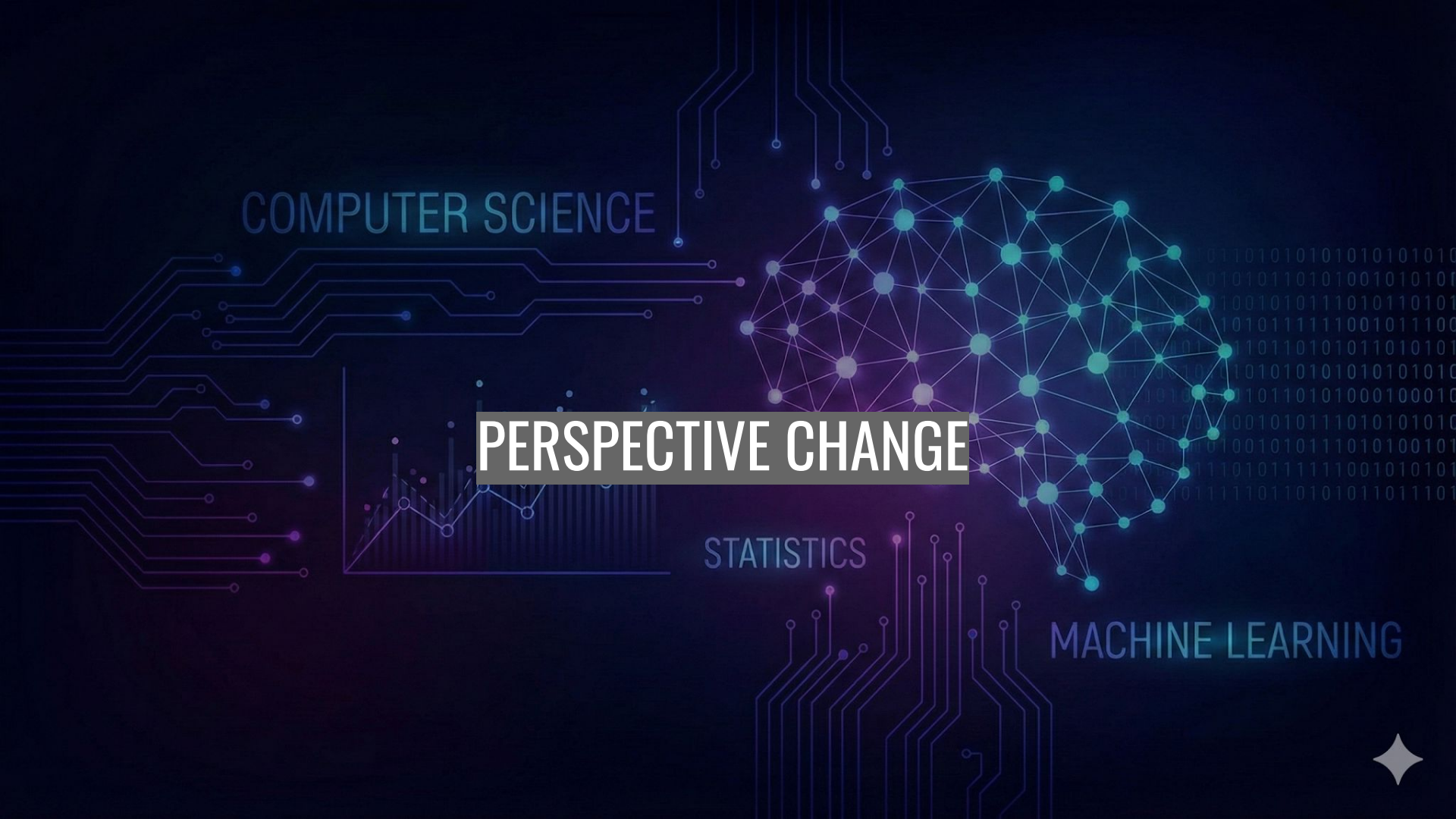
Textual interface allowing direct command input, shell scripting, and fast text-based system interaction.



02. Graphical User Interface (GUI)

Graphics-based user experience incorporating visual icons, windows, and intuitive point-and-click interaction.




COMPUTER SCIENCE
PERSPECTIVE CHANGE
STATISTICS
MACHINE LEARNING

Operating System: Programmer's Perspective

SYSTEM CALL INTERFACE

System calls provide the fundamental API mechanism for user programs to request kernel services, varying across platform implementations such as UNIX and Win32.

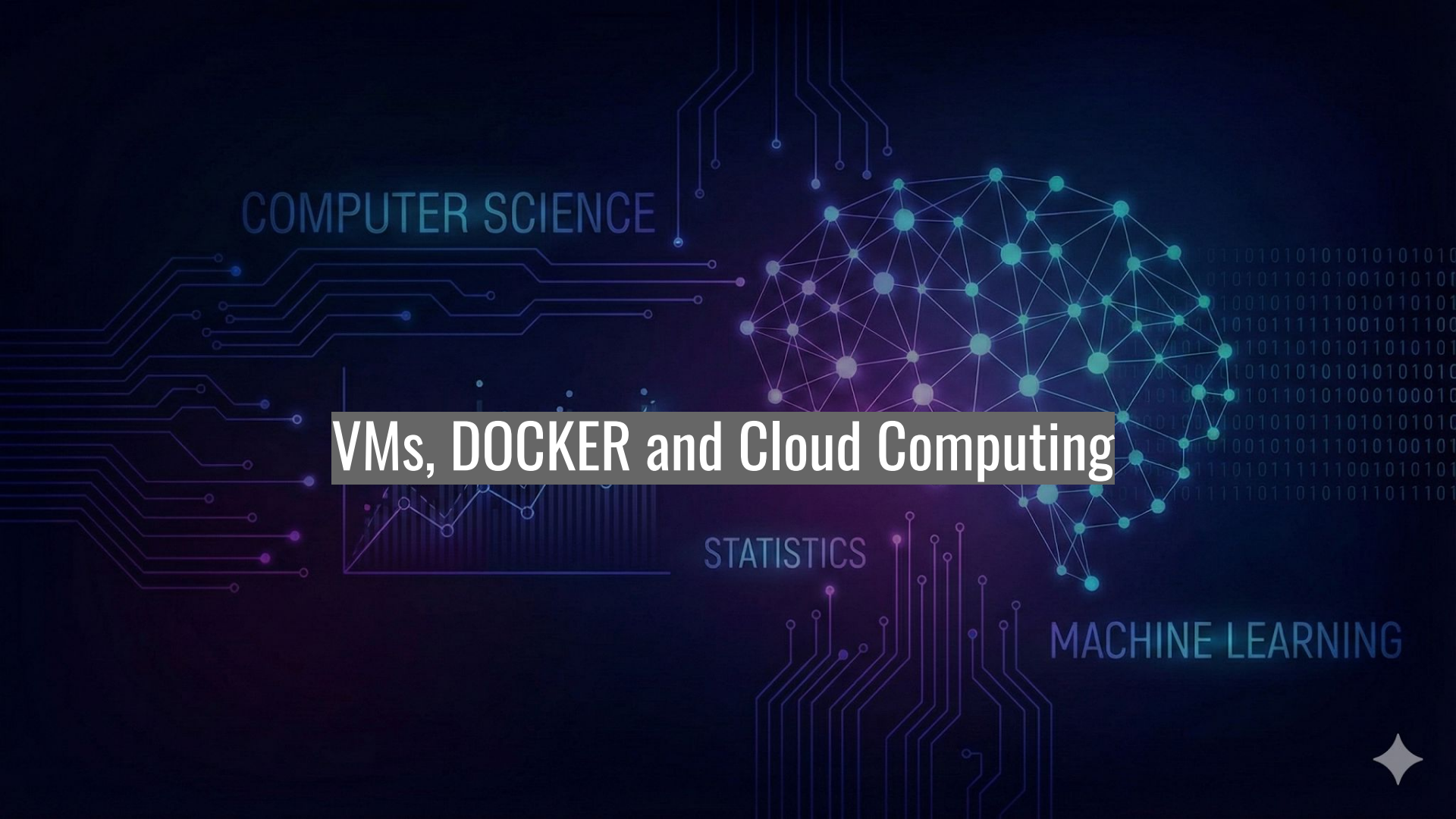
01. Platform System Call Mappings

Direct functional comparison of standard POSIX/UNIX system calls against their Win32 API equivalents.

UNIX	WIN32	UNIX	WIN32
fork	CreateProcess	mkdir	CreateDirectory
waitpid	WaitForSingleObject	rmdir	RemoveDirectory
execve	-	link	-
exit	ExitProcess	unlink	DeleteFile
open	CreateFile	mount	-
close	CloseHandle	umount	-
read	ReadFile	chdir	SetCurrentDirectory
write	WriteFile	chmod	-
lseek	SetFilePointer	kill	-
stat	GetFileAttributesEx	time	GetLocalTime

Key API Categories

- **Process Control**
  fork, execve vs CreateProcess
- **File Management**
  open, read, write vs CreateFile
- **Directory & Attributes**
  mkdir, stat vs GetFileAttributesEx



VMs, DOCKER and Cloud Computing

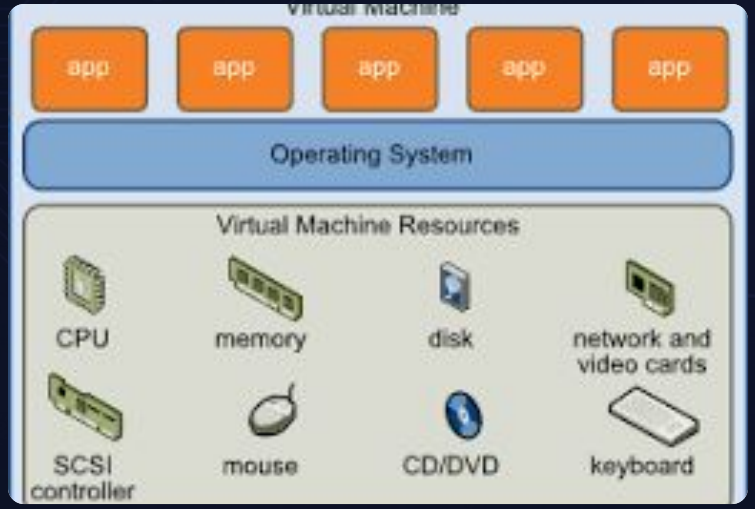
Cloud computing

VIRTUALIZATION ARCHITECTURE

Virtual Machine; turning one physical server into many virtual servers.

01. Virtual Machine Architecture & Resources

Hardware emulation layer providing isolated operating environments on shared infrastructure.



System Definition & Capabilities

A Virtual Machine (VM) is a software emulation of a physical computer that runs its own completely isolated operating system and applications, functioning exactly like a separate physical machine while sharing the underlying hardware of a host server.

- **Full OS Isolation**

  Executes dedicated guest operating system and isolated applications.

- **Hardware Abstraction**

  Virtualizes CPU, memory, storage disks, network, and input controllers.

Cloud computing

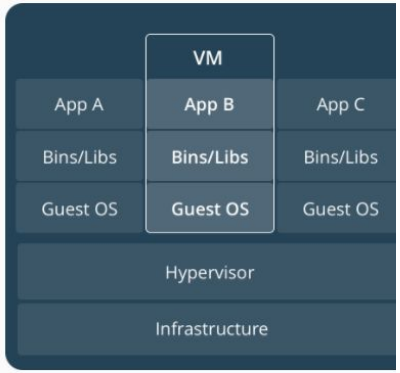
CONTAINERIZATION VS VIRTUALIZATION

Docker vs Virtual Machines (VM)

Architectural Comparison

Key structural differences between hardware-level virtualization and OS-level containerization.





action at the app layer that packages code and
. Multiple containers can run on the same machine and
h other containers, each running as isolated processes in
take up less space than VMs (container images are typically
l start almost instantly.

VIRTUAL MACHINES

Virtual machines (VMs) are an abstraction of physical hardv
into many servers. The hypervisor allows multiple VMs to r
Each VM includes a full copy of an operating system, one o
binaries and libraries - taking up tens of GBs. VMs can also

Kernel & Efficiency Breakdown

While a Virtual Machine (VM) emulates an entire computer system with its own heavy operating system kernel, a Docker container packages only the application and its dependencies, sharing the host system's kernel to run much faster and more efficiently.

- **Virtual Machines**
  Full hardware emulation with dedicated OS & kernel per instance.
- **Docker Containers**
  Lightweight app isolation sharing host kernel for instant boot.

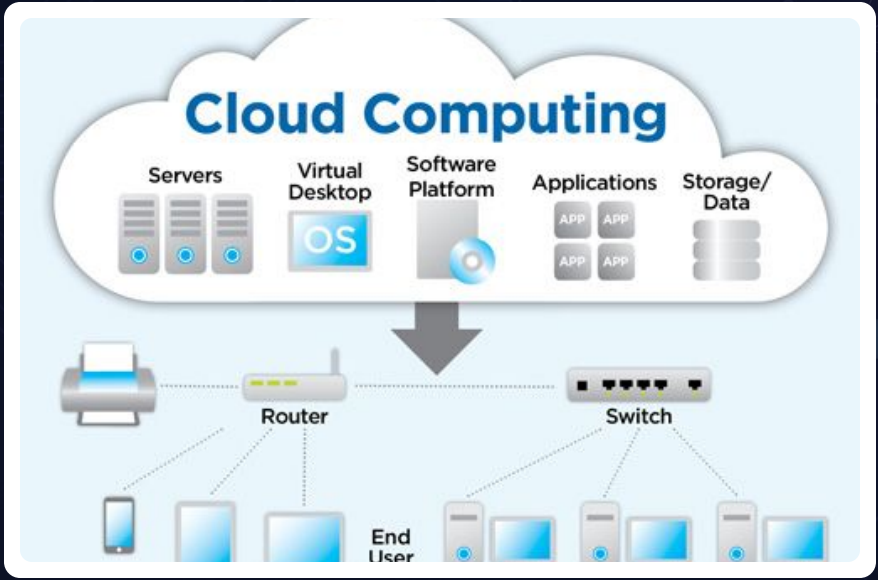
Cloud computing

VIRTUAL INFRASTRUCTURE OVERVIEW

A virtual infrastructure for delivering on-demand computing services

On-Demand Service Delivery

Access technology resources as a pay-as-you-go utility over the Internet



Core Concept & Benefits

Cloud computing is the on-demand delivery of computing services—such as servers, storage, databases, and software—over the Internet.

- **Pay-As-You-Go Utility**

  Access resources flexibly on demand without high upfront costs.

- **Infrastructure Management**

  Eliminates the need to own and maintain physical data centers.

Contents

- Introduction to Informatics
  - What is a computer
  - Networks and TCP/IP
  - Digitalization and basics of data encryption





Binary representation of numbers

POSITIONAL NUMBERING SYSTEMS

Core principles governing base values, digit symbols, and positional weights

1. Base & Symbol Definition

In a positional numbering system, the given **base** directly defines the total number of unique **symbols (digits)** available to represent numbers.

Decimal System Example (Base 10)

Uses exactly 10 distinct symbols:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

2. Positional Value & Weight

Each digit's total value depends explicitly on its position, represented as increasing powers of the base.

Polynomial Expansion (423)

$$423 = (4 \times 10^2) + (2 \times 10^1) + (3 \times 10^0)$$

Breakdown: 4 Hundreds + 2 Tens + 3 Ones

Binary representation of numbers (Short Digression)

GENERAL FORMULATION OF NUMBER SYSTEMS

Polynomial expansion for integer & fractional components across any base b

1. Base b Representation

Given a base **b**, there are **b symbols (digits)**. An integer N is represented as:

$$N = c_n \cdot b^n + c_{n-1} \cdot b^{n-1} + \dots + c_0 \cdot b^0$$

Similarly, for a fractional number **N = 0.c1c2...cn**:

$$N = c_1 \cdot b^{-1} + c_2 \cdot b^{-2} + \dots + c_n \cdot b^{-n}$$

2. Unsigned Integers (Naturals)

Using a numbering system of **base b** with **n digits** allows representing a total capacity of:

Maximum of b^n distinct numbers

Valid Range: **0 to (b^n - 1)**

Concrete Example (Base 10):

With **2 digits** in Base 10, represent numbers from **0 to 99**:

10^2 = 100 distinct values

The Binary Base

BASE 2 NUMERATION SYSTEM

Core capacity and range formulas for n-digit binary representation

1. Base Definition & Capacity

Uses base 2 with only two available symbols (**0** and **1**).

TOTAL CAPACITY (n DIGITS)

2^n distinct numbers

REPRESENTABLE RANGE

0 to $(2^n - 1)$

2. Concrete Example: 2 Digits (n = 2)

Capacity = 2^2 = **4 distinct numbers** (Range: 0 to 3)

00 = $0 \times 2^1 + 0 \times 2^0 = 0$

01 = $0 \times 2^1 + 1 \times 2^0 = 1$

10 = $1 \times 2^1 + 0 \times 2^0 = 2$

11 = $1 \times 2^1 + 1 \times 2^0 = 3$

The Positional Weights

Understanding how binary values are calculated from right to left

In binary, every position represents a power of 2. We read the number from right to left, starting at As we move left, the power increases. 2^0.

← Increases Left ←

Position (Power) ...	2^4	2^3	2^2	2^1	← 2^0
Base	2	2	2	2	2
Exponent ...	4	3	2	1	0
Value (Decimal)	16 $2 \times 2 \times 2 \times 2$	8 $2 \times 2 \times 2 \times 2$	4 $2 \times 2 \times 2 \times 2$	2 $2^1 = 2$	1 $2^1 = 2$
Binary Digits	1	0	1	0	1

From right to left ←

Conversion Example

1011

BASE 2 (BINARY)

The Addition Process

Let's map the binary number to `1011` positional weights and add them up:

$$(1 \times 8) + (0 \times 4) + (1 \times 2) + (1 \times 1)$$

$$8 + 0 + 2 + 1 = 11$$

Decimal to Binary

Translating human numbers into machine logic using the **Divide-by-2 Method**.

The Divide-by-2 Method

1. The Algorithm

- ÷ Divide your starting decimal number by 2.
- Write down the **remainder** (it will always be 0 or 1).
- Take the new quotient and divide it by 2 again.
- ↑ Repeat until the quotient is 0. **Read the remainders from bottom to top.**

2. Example: Convert 13

```
13 ÷ 2 = 6   ...  Remainder 1 (LSD) ↑
 6 ÷ 2 = 3   ...  Remainder 0       |
 3 ÷ 2 = 1   ...  Remainder 1       |
 1 ÷ 2 = 0   ...  Remainder 1 (MSD) ↑

Result: 1101
```

Why Binary Matters

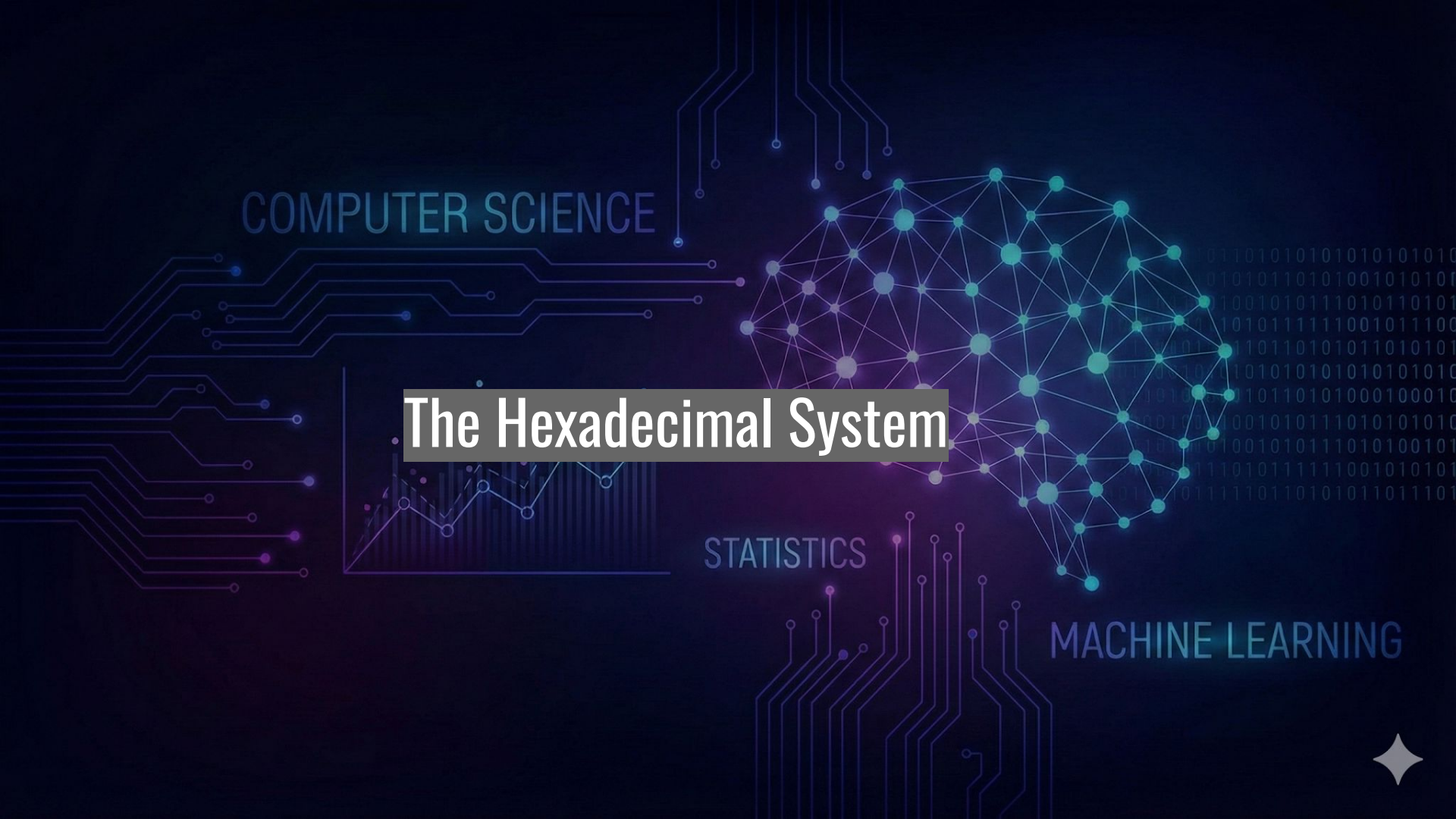
Physical Hardware Limitations

Computers do not understand abstract numbers. At a physical hardware level, a CPU is comprised of billions of microscopic transistors.
These transistors only have two reliable, measurable physical states regarding electricity:

- `1` = High Voltage (Switch ON)
- `0` = Low Voltage (Switch OFF)

Binary is the absolute lowest level of abstraction. Everything a computer does—from displaying an image to playing a video game—is ultimately translated into streams of 1s and 0s.





The Hexadecimal System

What is Base-16?

The Standard 10

Humans naturally count in **Decimal (Base-10)** because we have ten fingers.

This system uses exactly ten distinct symbols to represent all possible numbers before rolling over and adding a new column:

`0, 1, 2, 3, 4, 5, 6, 7, 8, 9`

The Extra 6

Hexadecimal (Base-16) requires sixteen distinct symbols.

Since we run out of standard numeric digits after 9, we borrow the first six letters of the English alphabet to represent values 10 through 15:

`A, B, C, D, E, F`

The Magic of 16

How binary logic perfectly aligns with hexadecimal

16

VALUES PER DIGIT

The Perfect Binary Match

Why do computer scientists use Base-16 instead of Base-10? Because of its perfect mathematical alignment with binary logic.

$$16 = 2^4$$

4 binary digits (bits)

This simple equation means that one single hexadecimal digit perfectly represents exactly 4 bits of binary data (known as a "nibble").

Consequently, it takes exactly two hex digits to represent a full 8-bit Byte.

The Translation Table

Observe how a single Hex character maps flawlessly to a 4-bit binary sequence.

DECIMAL (BASE-10)	BINARY (BASE-2)	HEXADECIMAL (BASE-16)
0	0000	0
5	0101	5
10	1010	A
15	1111	F
255 (Full Byte)	1111 1111	FF

Real-World Applications

Web Colors

Digital colors mix Red, Green, and Blue light from 0 to 255. In CSS and design software, this is written as a 6-digit hex code.

`#FF0000` (Pure Red)

Memory & Pointers

When debugging software or viewing memory dumps, RAM addresses are always displayed in Hex to keep the output readable.

`0x7FFF5FBFF8A0`

Networking

Hardware MAC addresses and the new standard of IPv6 internet addresses use Hex to manage massive numeric identities.

`00:1A:2B:3C:4D:5E`

Visual Data Compression

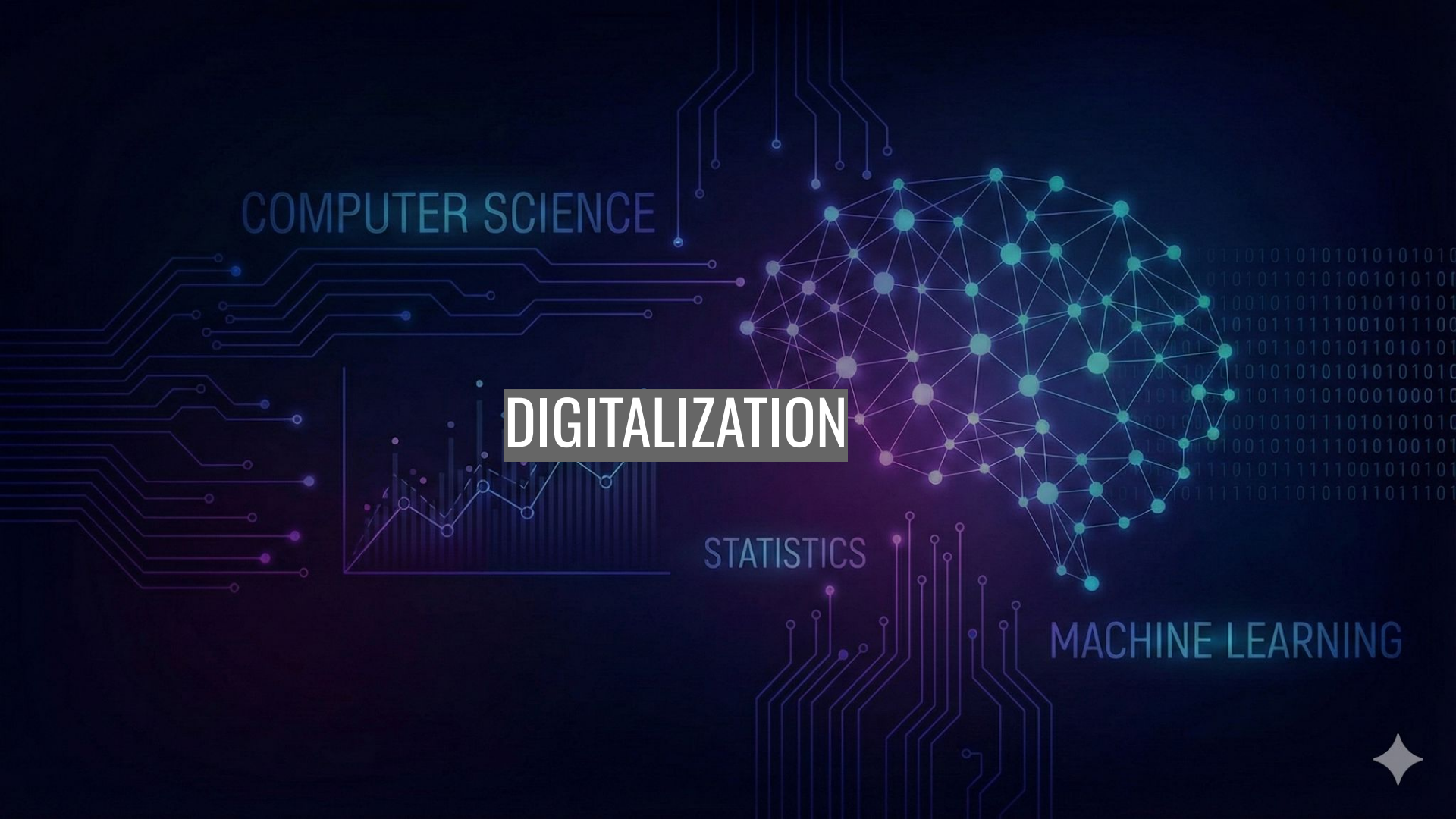
For the Human Eye.

Reading endless streams of raw 1s and 0s is virtually impossible for human programmers.

Hexadecimal acts as a visual compression algorithm. By grouping bits into blocks of 4, we reduce the physical length of binary strings by **75%**.

It does not change how the computer stores the data underneath; it simply changes how we view it.




COMPUTER SCIENCE
DIGITALIZATION
STATISTICS
MACHINE LEARNING

THE BINARY BASE

Fundamental Unit

1 BYTE = 8 BITS

The core element of computer addressability.

A byte (a morsel) represents a sequence of 8 bits and has historically become the basic unit of measurement for digital information.

Numeric Values

2^8 = 256 VALUES

Supports unsigned and signed integers.

- **Unsigned:** 0 to 255
- **Signed:** -128 to 127
- Sign bit (0 = positive, 1 = negative), spanning 10000000 to 01111111.

Text Representation

256 CHARACTERS

Symbol and character mapping.

Alternatively, an 8-bit byte can map directly to 256 unique text characters or control symbols in standard encodings.

ASCII ENCODING

US-ASCII (7-Bit)

ORIGINAL ASCII

The standard 7-bit character encoding set developed in the 1960s.

- **Capacity:** 128 characters (0 to 127)
- **Control Codes:** 33 non-printable codes (NUL, LF, CR, ESC...)
- **Printable:** 95 characters (A-Z, a-z, 0-9, symbols)

8-Bit Expansion

EXTENDED ASCII

Utilizes the full 8th bit to expand the available character repertoire.

- **Capacity:** 256 characters (0 to 255)
- **Extra Range:** 128 additional codes (128–255)
- **Includes:** Accented letters, graphic symbols, math operators

Character Map

ASCII TABLE

	033	!	066	B	099	c	132	ä	165	Ñ	198				
er (SOH)	034	"	067	C	100	d	133	à	166	ª	199				
(STX)	035	#	068	D	101	e	134	å	167	º	200				
(ETX)	036	$	069	E	102	f	135	ç	168	¿	201				
mission (EOT)	037	%	070	F	103	g	136	ê	169	®	202				
	038	&	071	G	104	h	137	ë	170	¬	203				
(ACK)	039		072	H	105	i	138	è	171	½	204				
	040	(	073	I	106	j	139	ï	172	¼	205				
S)	041	)	074	J	107	k	140	î	173	¡	206				
	042	*	075	K	108	l	141	ì	174	«	207				
	043	+	076	L	109	m	142	Ä	175	»	208				
	044	,	077	M	110	n	143	Å	176	▒	209				
F)	045	-	078	N	111	o	144	É	177	▒	210				
rn (CR)	046	.	079	O	112	p	145	æ	178	▓	211				
	047	/	080	P	113	q	146	Æ	179	│	212				
	048	0	081	Q	114	r	147	ô	180	┤	213				
pe (DLE)	049	1	082	R	115	s	148	ö	181	Á	214				
	050	2	083	S	116	t	149	ò	182	Â	215				
	051	3	084	T	117	u	150	û	183	À	216				
	052	4	085	U	118	v	151	ù	184	©	217				
	053	5	086	V	119	w	152	ÿ	185	╣	218				
nowledge (NAK)	054	6	087	W	120	x	153	Ö	186	║	219				
Idle	055	7	088	X	121	y	154	Ü	187	╗	220				
mission Block	056	8	089	Y	122	z	155	ø	188	╝	221				
	057	9	090	Z	123	{	156	£	189	¢	222				
n	058	:	091	[	124	|	157	Ø	190	¥	223				
	059	;	092	\	125	}	158	×	191	┐	224				
	060	<	093	]	126	~	159	ƒ	192	└	225				
	061	=	094	^	127 (DEL)	⌂	160	á	193	┴	226				
ator	062	>	095	_	128	Ç	161	í	194	┬	227				
ator	063	?	096	`	129	ü	162	ó	195	├	228				
	064	@	097	a	130	é	163	ú	196	─	229				
	065	A	098	b	131	â	164	ñ	197	┼	230				

ASCII ENCODING

CLI TERMINAL INSPECTION

Binary & Hex Analysis

```
└─ $ cat testo.txt
Hello World!
(base) |redo@banquo /home/redo/Downloads
└─ $ xxd -b testo.txt
00000000: 01001000 01100101 01101100 01101100 01101111 00100000  Hello
00000006: 01010111 01101111 01110010 01101100 01100100 00100001  World!
0000000c: 00001010                                               .
(base) |redo@banquo /home/redo/Downloads
└─ $ od -A x -t x1 -v testo.txt
000000 48 65 6c 6c 6f 20 57 6f 72 6c 64 21 0a
00000d
```

Sample File: testo.txt containing string **"Hello World!"**

Demonstrates three perspectives of the same underlying byte stream stored on disk, showing ASCII text conversion into raw binary bits and hex bytes.

COMMAND BREAKDOWN

Decoding the Commands

1. cat testo.txt Prints raw string output. Translates stored ASCII codes back to readable characters.

2. xxd -b testo.txt Binary dump. Displays exact 8-bit sequence for each character (e.g., 01001000 = 'H').

3. od -A x -t x1 testo.txt Hexadecimal dump. Shows byte values in hex (48 65 6c...), representing character offsets.

INFORMATION CODING

DUAL INTERPRETATION

BIT SEQUENCE CONVENTION

An encoding is an agreed convention. The exact same binary sequence yields different meanings based on interpretation.

- Bit Sequence: **01001100**
- Text Context: **Character 'L' (Uppercase)**
- Numeric Context: **Decimal Number 76**

VISUAL ENCODING

1-BIT PIXEL REPRESENTATION

Images are composed of individual pixels. Bit depth determines the number of expressible states per pixel.

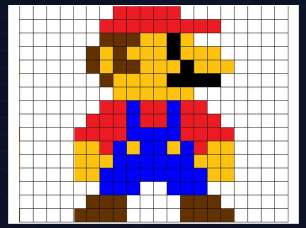
- Structure: **Pixel Grid System**
- 1-Bit Limitation: **Black & White only**
- State Mapping: **1 = Black Pixel**, **0 = White Pixel**

INFORMATION CODING

If I use more bits to represent each pixel, I can instead obtain ranges of grays or colors. And from there, sounds, videos...

FOUNDATIONAL CONCEPTS

Pixel & Resolution



PIXEL ELEMENT

The pixel represents the smallest autonomous element of the image. Each pixel is characterised by its own position.

GRID RESOLUTION

The total number of pixels in a digital image is called its **resolution**.

Example: A 10 × 10 grid results in an image made up of 100 pixels.

DISPLAY METRICS & DEPTH

Density & Color Range

1. DPI (Dots Per Inch)

Number of dots per inch. For example, a typical monitor displays **72 pixels per inch**.

2. Color / Bit Depth

In the case of a grayscale image, 8 bits can be used for each pixel, providing 2^8 **= 256 shades of gray** available.

INFORMATION CODING

RGB COLOR MODEL

PRIMARY COLOR CHANNELS

In color images, each individual pixel is characterized by three distinct color scales corresponding to the primary colors: **RGB (Red, Green, Blue).**



BIT DEPTH & COLOR CAPACITY

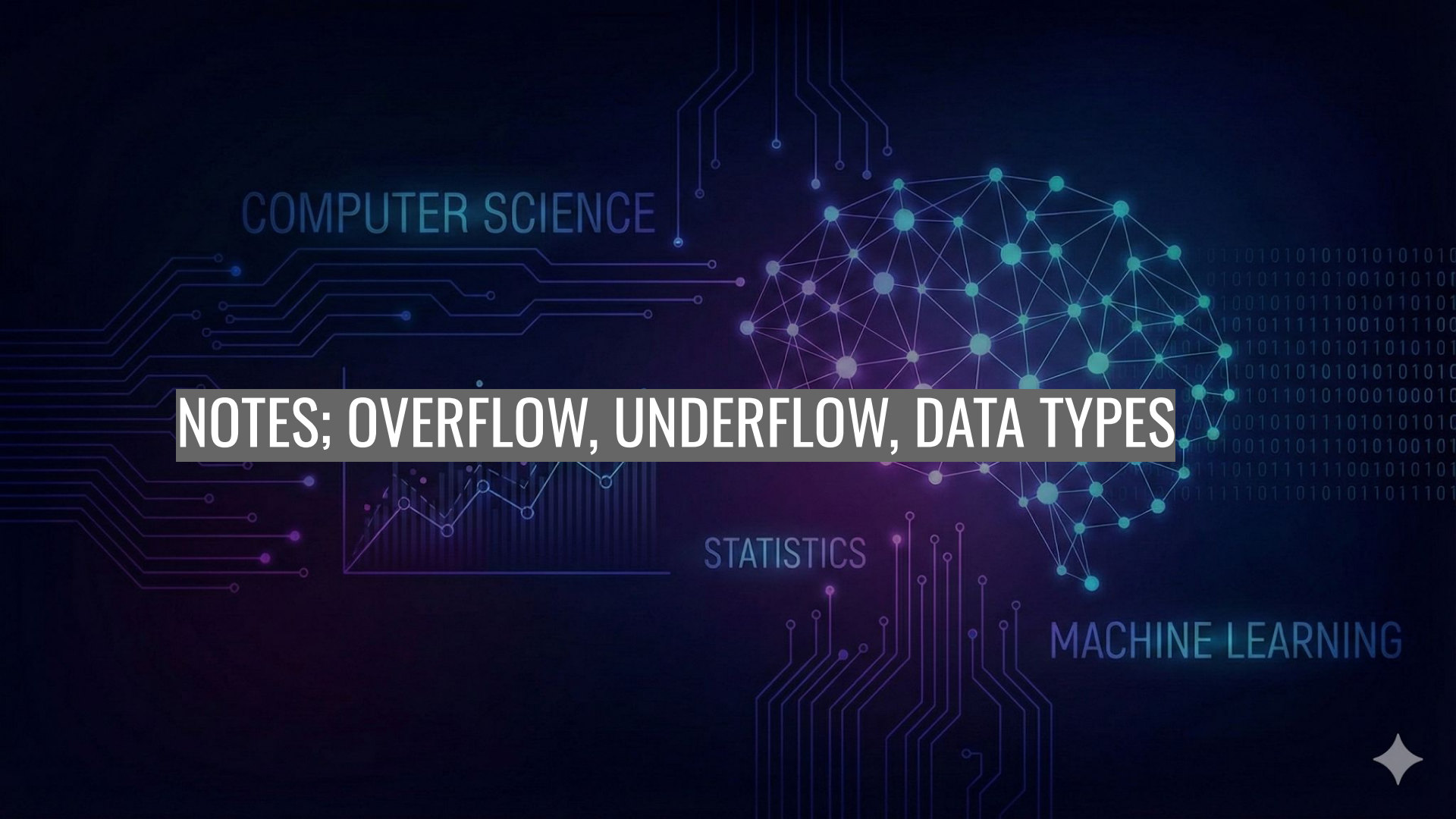
COLOR SPECTRUM RANGES

1. 8-Bit Standard Image

256 shades per primary color (R, G, B).
Total combinations: **16,777,216 possible colors.**

2. 12-Bit Professional Image

2^{12} = 4,096 shades per primary color.
Total combinations: **68,719,476,736 possible colors per pixel.**



NOTES; OVERFLOW, UNDERFLOW, DATA TYPES

THE COMPUTER'S MEMORY IS RUNNING OUT

MEMORY LIMITATIONS

OVERFLOW & UNDERFLOW

1. Overflow

Occurs when there are not enough bits available in memory to represent the final calculated result.

2. Underflow

Occurs when a number is too small or fractional to be stored accurately (e.g., **3 / 2 = 1.5**, but integer memory can only represent **1 or 2**).

PRACTICAL DEMONSTRATION

8-BIT OVERFLOW EXAMPLE

With 8 bits, positive integers range from **0 to 255**. What happens when adding 1 to 255?

```
  1 1 1 1 1 1 1 1  (255)
+ 0 0 0 0 0 0 0 1  (  1)
-----------------------
1 0 0 0 0 0 0 0 0  (256)
```

Result requires 9 bits!

Standard 8-bit memory drops the leading bit, leaving **00000000 (0)**.

THE COMPUTER'S MEMORY IS RUNNING OUT

UNDERFLOW CONCEPT

INTEGER TRUNCATION

1. Precision Loss

Occurs when a calculation yields a fractional value, but the integer data type cannot store decimal places, forcing the result to be rounded down or truncated.

2. Zero Result Underflow

When performing integer division where the denominator is larger than the numerator (e.g., **1 / 2 = 0.5**), integer memory truncates the decimal completely, leaving **0**.

PRACTICAL DEMONSTRATION

INTEGER DIVISION 1 / 2

In standard 8-bit integer memory, what happens when we divide **1 by 2**?

```
Mathematical: 1 / 2 = 0.5
Binary Exact: 0.5 (Base 2)
Integer Store: 00000000 (0)
```

Fractional Bits Dropped!

Integer formats cannot represent fractional binary places. The value **0.5** underflows to **0**, resulting in a total loss of mathematical precision.

DATA TYPES & MEMORY STRUCTURE

TYPING CONCEPTS

TYPE SYSTEMS & SPECS

1. Type Classification

Understanding the distinction between strongly vs. non-strongly typed languages, and dynamic vs. static typing.

2. Native Data Types

Programming languages provide built-in types such as integers, floating-points, booleans, and characters.

3. Sizes & Ranges

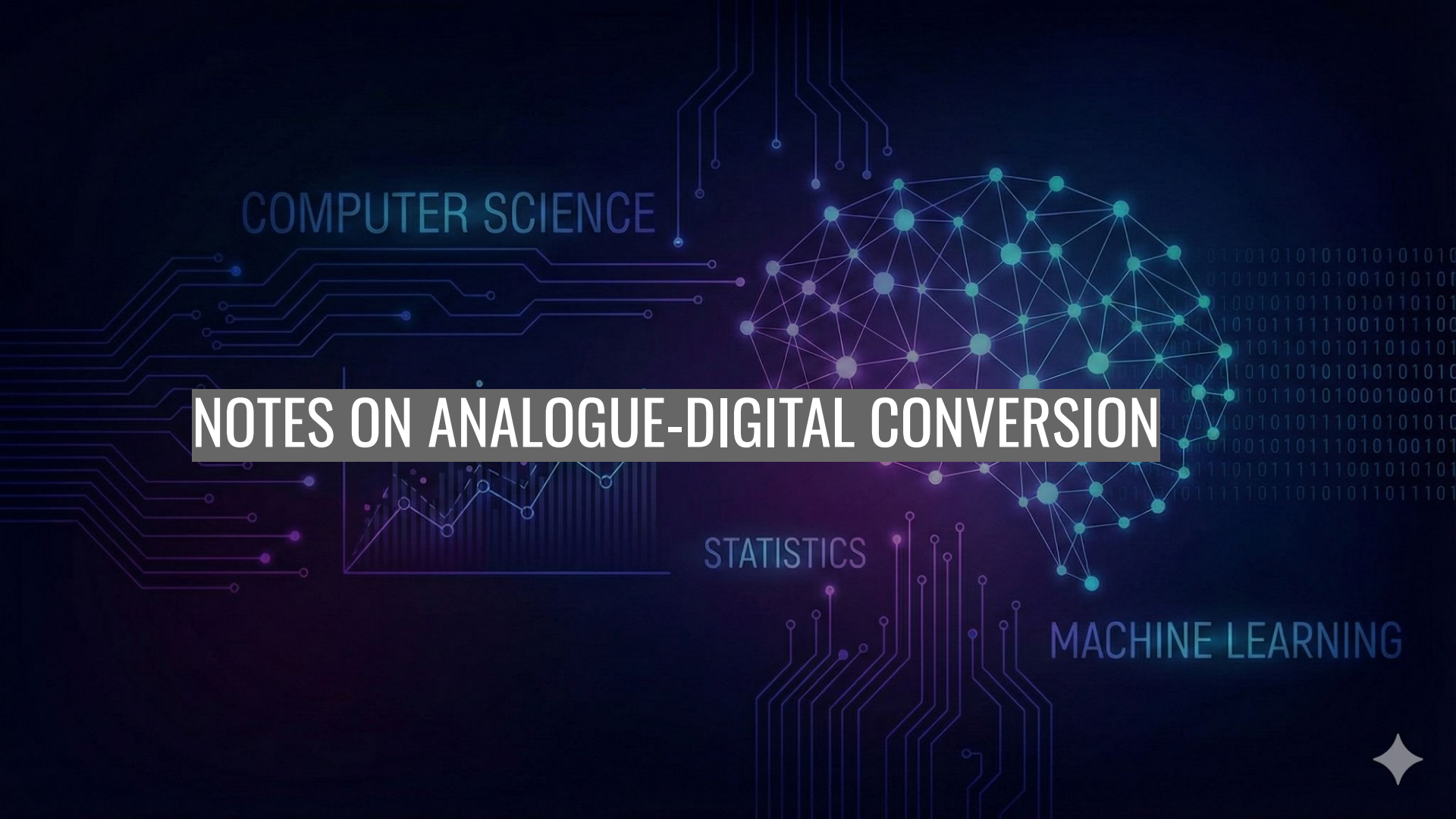
Different types allocate specific memory sizes, governing exact vs. non-exact algebraic behavior.

MEMORY ALLOCATION

TYPE SIZE & RANGE TABLE

Standard byte sizes and value bounds across common primitive data types:

label	size (bytes)	smallest value	largest value
byte	1	-128	127
short	2	-32768	32767
int	4	-2147483648	2147483647
long	8	-9223372036854775808	9223372036854775807
char	2	0	65535



NOTES ON ANALOGUE-DIGITAL CONVERSION

ANALOG TO DIGITAL CONVERSION

ADC PROCESS

CONVERSION STEPS

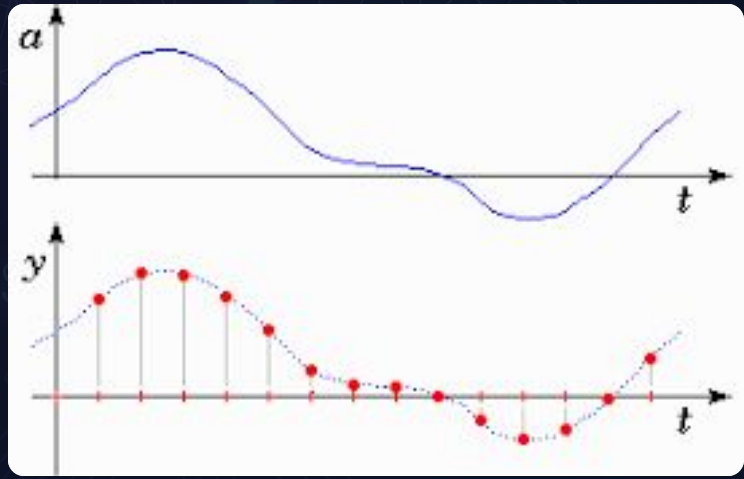
1. Sampling
Time discretization (Nyquist-Shannon sampling theorem)

2. Quantization
Discretization of the amplitude

3. Coding
Using binary “words” to express the value of the signal

VISUALIZATION

SAMPLING & QUANTIZATION



ANALOG SIGNAL FIDELITY

SIGNAL RECONSTRUCTION

KEY FIDELITY DRIVERS

1. Sampling Frequency

Increasing the number of samples per unit of time captures fast signal changes more accurately.

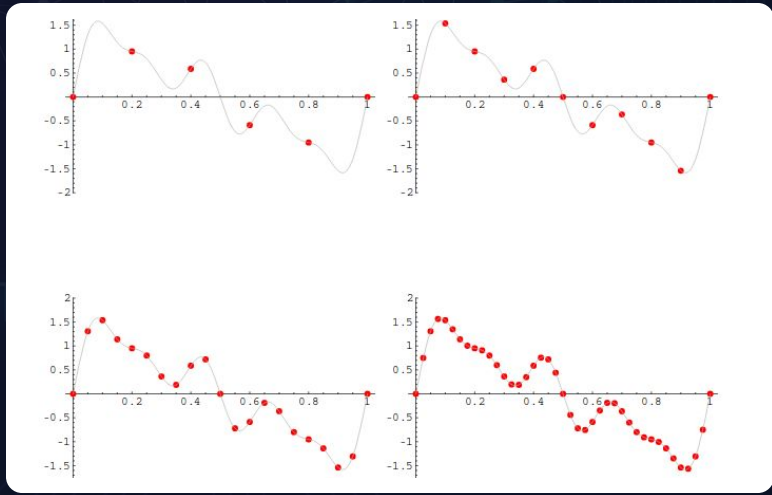
2. Quantization Levels

Expanding amplitude levels reduces rounding errors between discrete steps.

Fidelity improves continuously as both sampling rate and quantization resolution increase.

VISUAL COMPARISON

RESOLUTION IMPACT



DIGITAL AUDIO SPECIFICATIONS

AUDIO FUNDAMENTALS

SAMPLING & QUANTIZATION

1. Sampling Frequency

Human hearing ranges from 20 Hz to 20,000 Hz. The sampling theorem dictates a minimum sampling rate of 40 kHz to accurately reproduce sound.

2. Quantization Depth

Quantization levels far exceed 256. Modern digital audio standard typically utilizes 16-bit depth (65,536 distinct discrete amplitude levels).

REAL-WORLD CASE STUDY

COMPACT DISC (CD) AUDIO

Standard Configuration:

2 Channels (Stereo) 44.1 kHz Sampling Rate 16-bit Resolution

Bit Rate Calculation

44,100 samples/s 16 bits 2 channels

44100*16*2 = 1,411,200 bits/s (divide by 8 = 176,400 Bytes/s)

Storage Required (1 Minute of Music)

60 s 176,400 Bytes/s = 10,584,000 Bytes

10.1 MiB per minute

FINAL CONSIDERATIONS

SENSORY ELIMINATION

PERCEPTUAL LIMITS

1. Visual & Auditory Sensitivity

Compression of images and sound (and therefore also video) is essentially based on the elimination of information to which the human ear and the human eye are not very sensitive.

Psychoacoustic and psychovisual models discard data imperceptible to human senses.

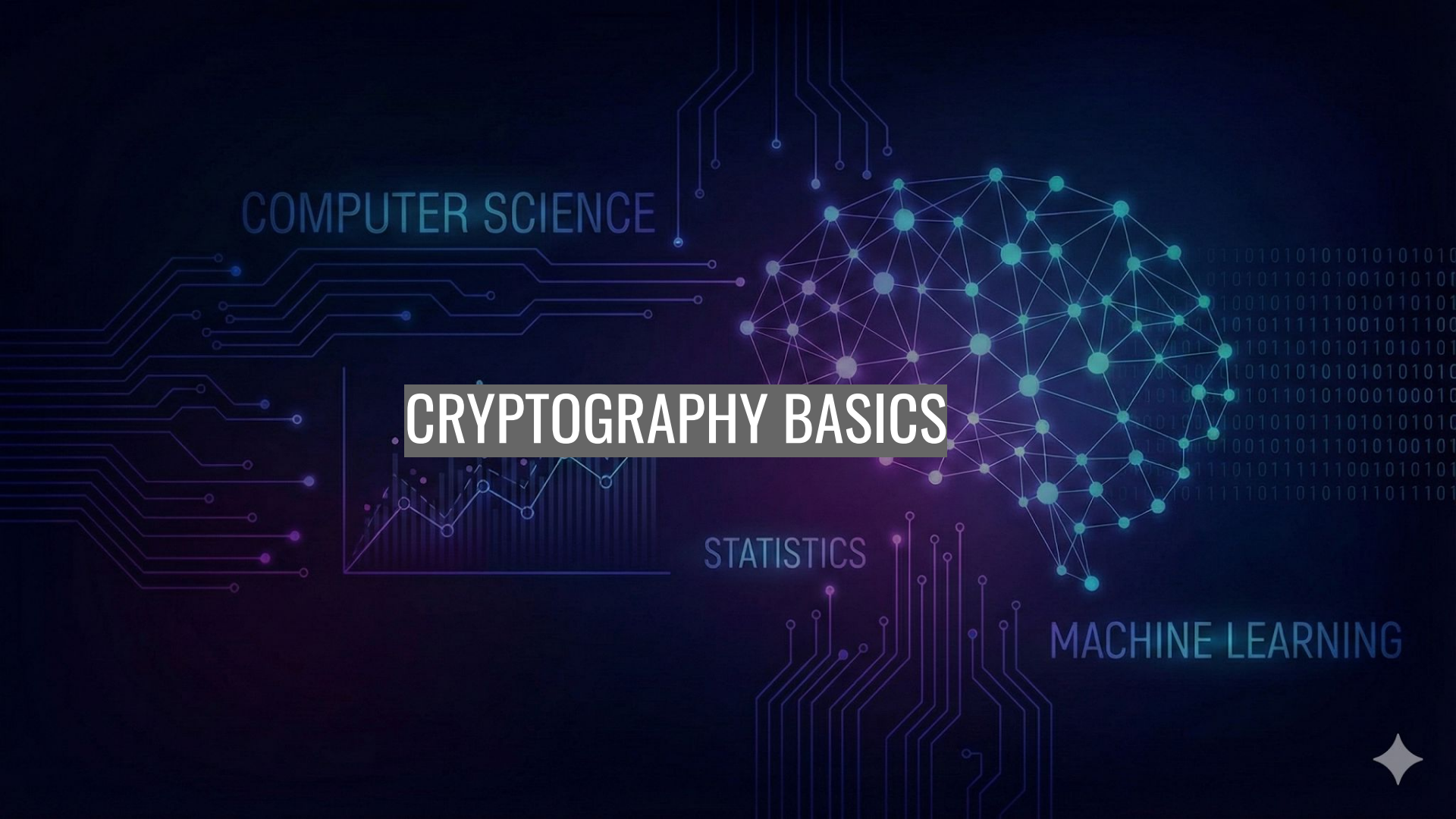
BEYOND PERCEPTION

BUT NOT ONLY THAT...

2. Statistical & Structural Redundancy

Compression algorithms also heavily exploit spatial, temporal, and mathematical redundancies in data, ensuring optimal bitrate reduction beyond human perception limits.

Combines perception filtering with lossless entropy & pattern encoding.


COMPUTER SCIENCE
CRYPTOGRAPHY BASICS
STATISTICS
MACHINE LEARNING

CRYPTOGRAPHY BASICS

CORE MECHANISM

In computers, information is stored as sequences of bits. Cryptographic techniques modify these sequences (strings) to obtain different sequences that can be transmitted and transformed back by the recipient into the original sequence using secret keys.

HISTORICAL METHOD

Symmetric Encryption

Single Secret Key

Used since the Egyptians and the ancient Romans. The sender and recipient share a single identical key to encrypt and decrypt the message payload.

MODERN PARADIGM

Asymmetric Encryption

Public & Private Key Pair

Dates back to the 70s. Employs mathematically linked key pairs, allowing secure data transfer without needing to exchange secret keys beforehand.

MESSAGE LIFECYCLE

Core Terminology

To understand how secure messaging works, we must define the two states of a message's lifecycle.

Plaintext

ORIGINAL STATE

The original, intelligible message or data that is fed into the algorithm. It is completely readable by anyone who intercepts it.

Example:
"MEET ME AT NOON"

Ciphertext

ENCRYPTED STATE

The scrambled, seemingly random output of the encryption algorithm. It is entirely unreadable without the correct mathematical key.

Example:
"X#9!Qp2L@Z$8"

HISTORICAL CRYPTOGRAPHY

Symmetric Encryption

CORE MECHANISM

Shared Secret Key

The key used for encryption and decryption is unique and shared between sender and recipient. Both parties must possess this exact secret key.



CIPHER TYPES

Classic Examples

Caesar Cipher

Each character is replaced by another character offset by **k** places (the key is the value of **k**).

Monoalphabetic Cipher

Uses a fixed substitution mapping across the alphabet:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
m	n	b	v	c	x	z	a	s	d	f	g	h	j	k	l	p	o	i	u	y	t	r	e	w	q

CRYPTOGRAPHY CONCEPTS

Symmetric Encryption

KEY DISTRIBUTION

Secret Key Exchange

A secure method must be established to exchange the unique secret key between sender and recipient before encrypted communication can take place.

SECURITY RISK

Brute-Force Attacks

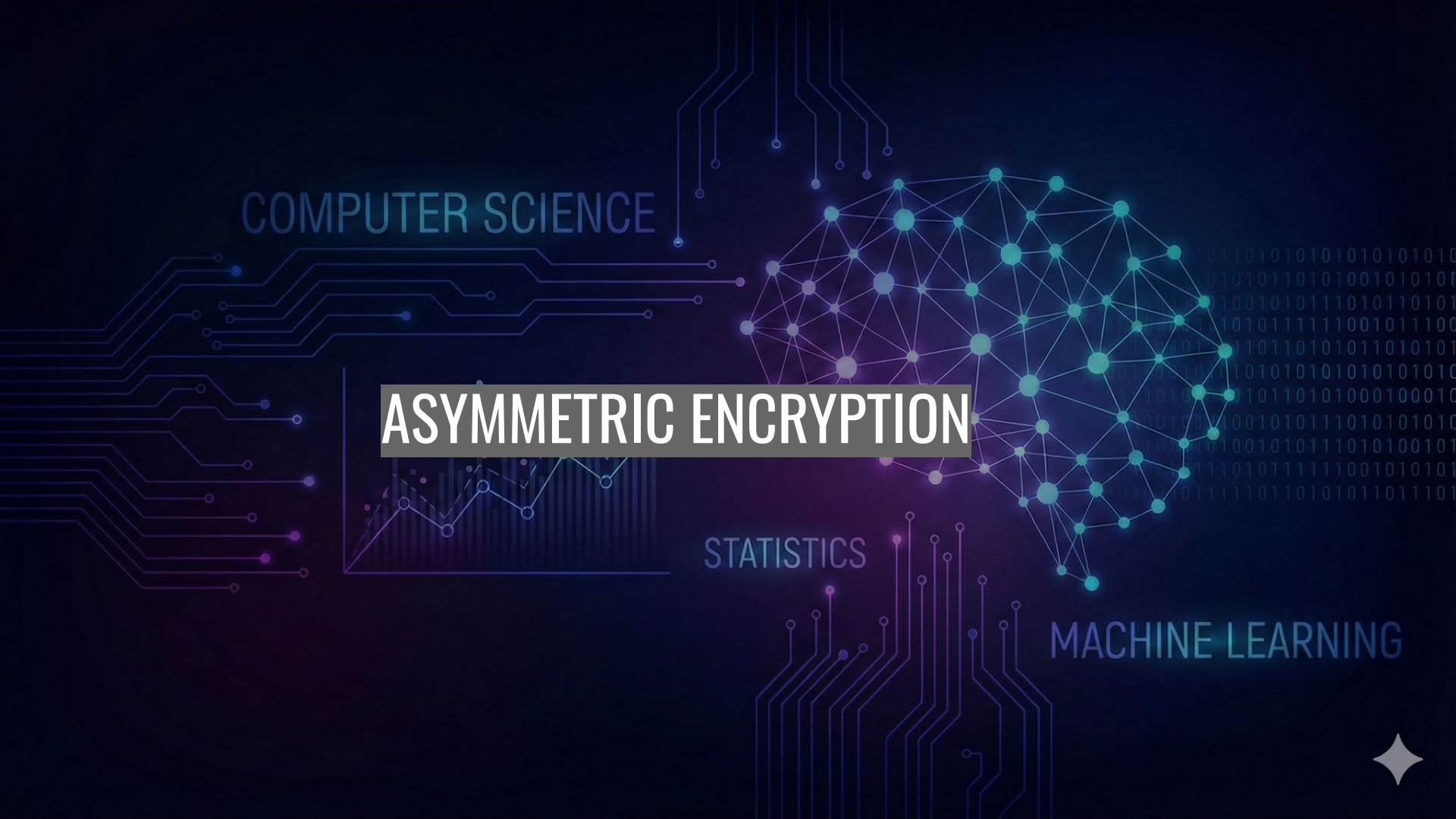
Attackers systematically test all possible key values (e.g., in Caesar ciphers) until coherent words and sentences are revealed.

MODERN STANDARDS

Robust Algorithms

Modern ciphers rely on difficult mathematical problems that are computationally infeasible to solve without the key:

- **Standard AES**
- Blowfish & Twofish
- Standard DES / Triple DES


COMPUTER SCIENCE
ASYMMETRIC ENCRYPTION
STATISTICS
MACHINE LEARNING

CRYPTOGRAPHY CONCEPTS

Asymmetric Encryption

CORE MECHANISM

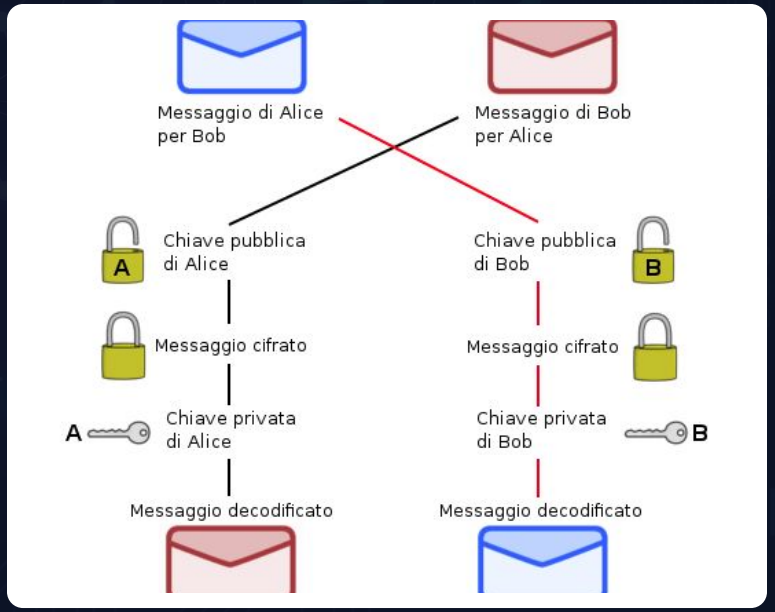
Key Pair Principle

The keys used for encryption and decryption are separate and mathematically linked:

Public Key
Distributed openly and used exclusively to **encrypt** messages.

Private Key
Kept strictly secret and used to **decrypt** and recover original messages.

MESSAGE EXCHANGE WORKFLOW



CRYPTOGRAPHY CONCEPTS

Asymmetric Encryption

KEY MANAGEMENT

Key Pair Distribution

Private Key (KS)

Must be strictly guarded by the user (e.g. stored in a smartcard which performs encryption internally).

Public Key (KP)

Distributed openly to allow anyone to encrypt messages intended for the owner.

MESSAGE WORKFLOW

Confidential Exchange

1. Encryption by Bob

Bob uses Alice's **Public Key (KP)** to create an encrypted cryptogram.

2. Decryption by Alice

Only Alice, holding her matching **Private Key (KS)**, can reveal the clear text.

EXTENDED APPLICATIONS

Authentication & Identity

Identity Verification

Asymmetric key architecture is fundamental for authenticating users and system processes.

Hardware Isolation

Smartcards securely store private keys and execute crypto operations on-board.

CRYPTOGRAPHY CONCEPTS

Asymmetric Encryption

CORE MECHANISM

Public & Private Keys

Asymmetric encryption uses a mathematically linked pair of keys to secure communication:

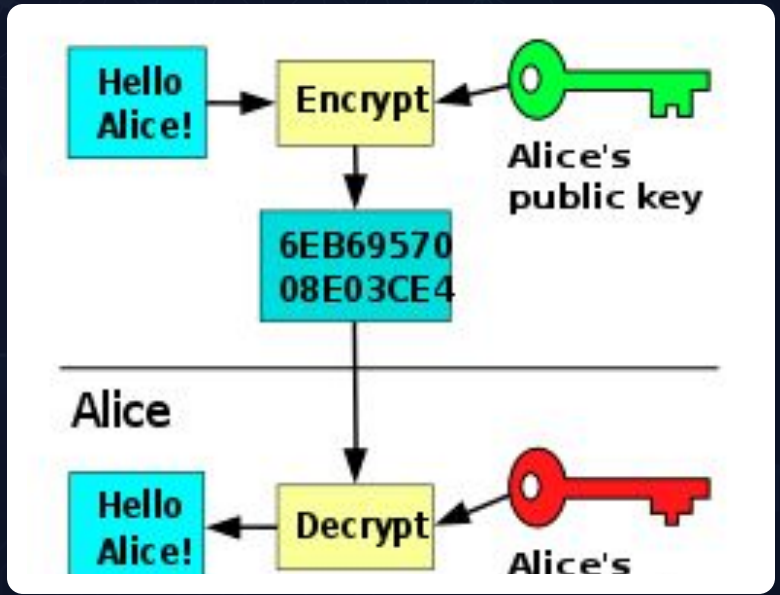
Public Key (Encryption)

Bob uses Alice's public key to **encrypt** the plaintext message.

Private Key (Decryption)

Only Alice's private key can **decrypt** and recover the original plaintext.

ENCRYPTION WORKFLOW



CRYPTOGRAPHY CONCEPTS

Asymmetric Encryption: RSA

ALGORITHM OVERVIEW

RSA Standard

Foundational Algorithm

The most well-known algorithm, named after inventors Rivest, Shamir, and Adleman.

Versatile Applications

Used for encryption, authentication, and ensuring document integrity via digital signatures.

CORE MATHEMATICS

Prime Numbers

Mathematical Basis

Based on prime numbers divisible only by 1 or themselves (e.g. 2, 3, 5, ..., 19249 · 213018586 + 1).

Key Derivation

Secret (KS) and public (KP) keys are mathematically derived from these prime factors.

SECURITY & QUANTUM

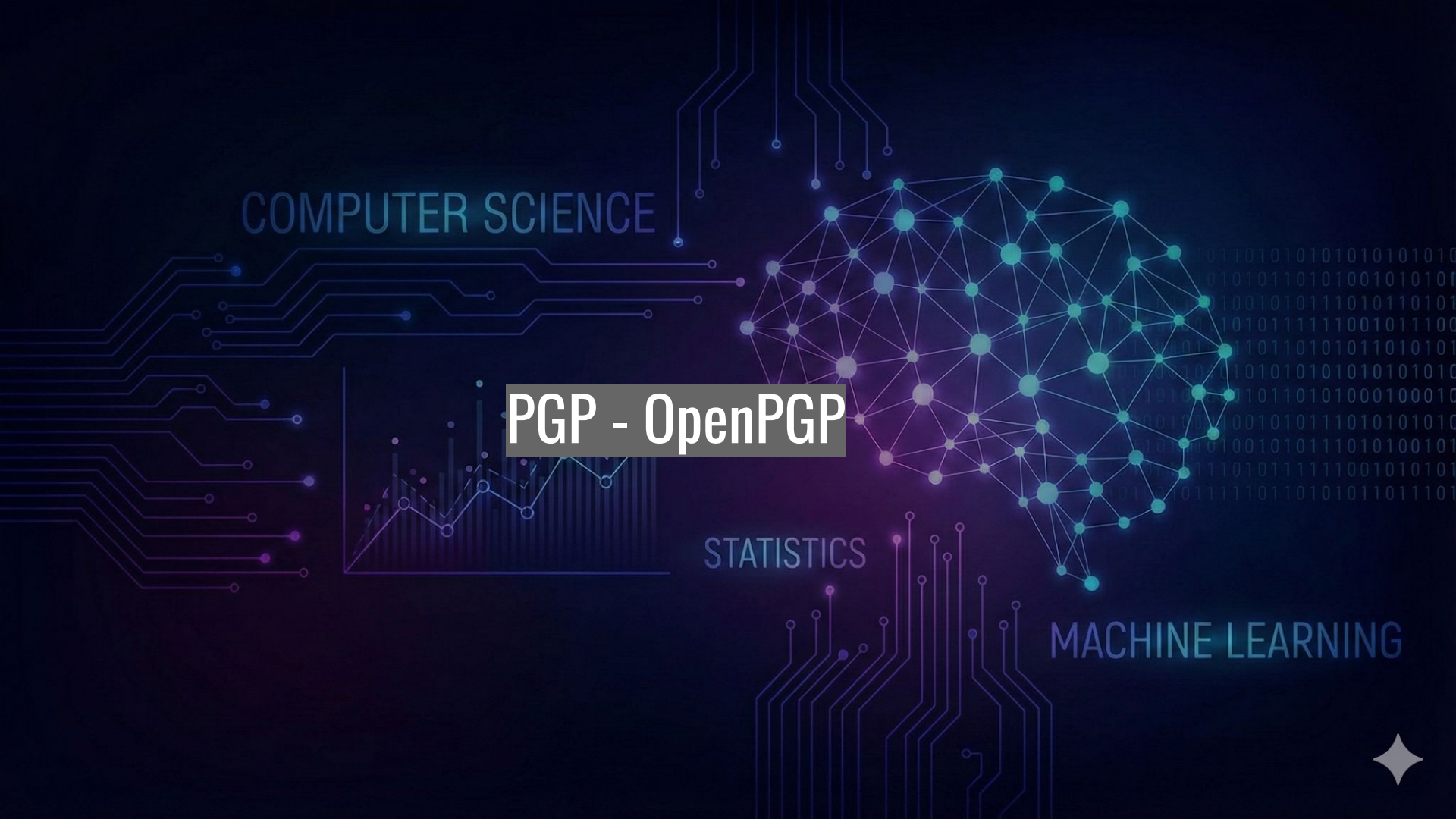
Prime Factorization

Computational Difficulty

Security relies on the extreme difficulty of finding the prime factors of a very large modulus.

Quantum Computing

High interest in factorization algorithms such as Shor's quantum algorithm.


COMPUTER SCIENCE
PGP - OpenPGP
STATISTICS
MACHINE LEARNING

CRYPTOGRAPHY CONCEPTS

OpenPGP Overview & Architecture

FOUNDATIONS

PGP vs. OpenPGP

Software to Standard

PGP began as software and evolved into a standard protocol generally known as OpenPGP.

Algorithm Distinction

RSA is an algorithm (for encryption & signatures), whereas OpenPGP defines the broader protocol.

FUNCTIONALITY

Secure Messaging

Data Element Formats

Defines structured formats for data elements that enable secure communication.

Core Operations

Supports encryption, digital signatures, and essential management operations like key distribution.

CRYPTOGRAPHY

Algorithms

Algorithm Reliance

As a flexible protocol, OpenPGP relies on a wide variety of cryptographic algorithms.

RSA Integration

Among the key asymmetric algorithms supported and utilized by OpenPGP is RSA.

CRYPTOGRAPHY CONCEPTS

OpenPGP Development & History

ORIGINS

Pretty Good Privacy

Created by P. Zimmermann

Philip R. Zimmermann created Pretty Good Privacy (PGP) as an email encryption software package.

Human Rights Tool (1991)

Originally conceived as a human rights tool, PGP was released free of charge on the Internet in 1991.

LEGAL CHALLENGES

Export Investigation

Three-Year Inquiry

Releasing PGP online made Zimmermann the target of a three-year criminal investigation.

Export Restrictions

The government believed U.S. export restrictions on cryptographic software were violated as PGP spread worldwide.

OPEN REPLACEMENT

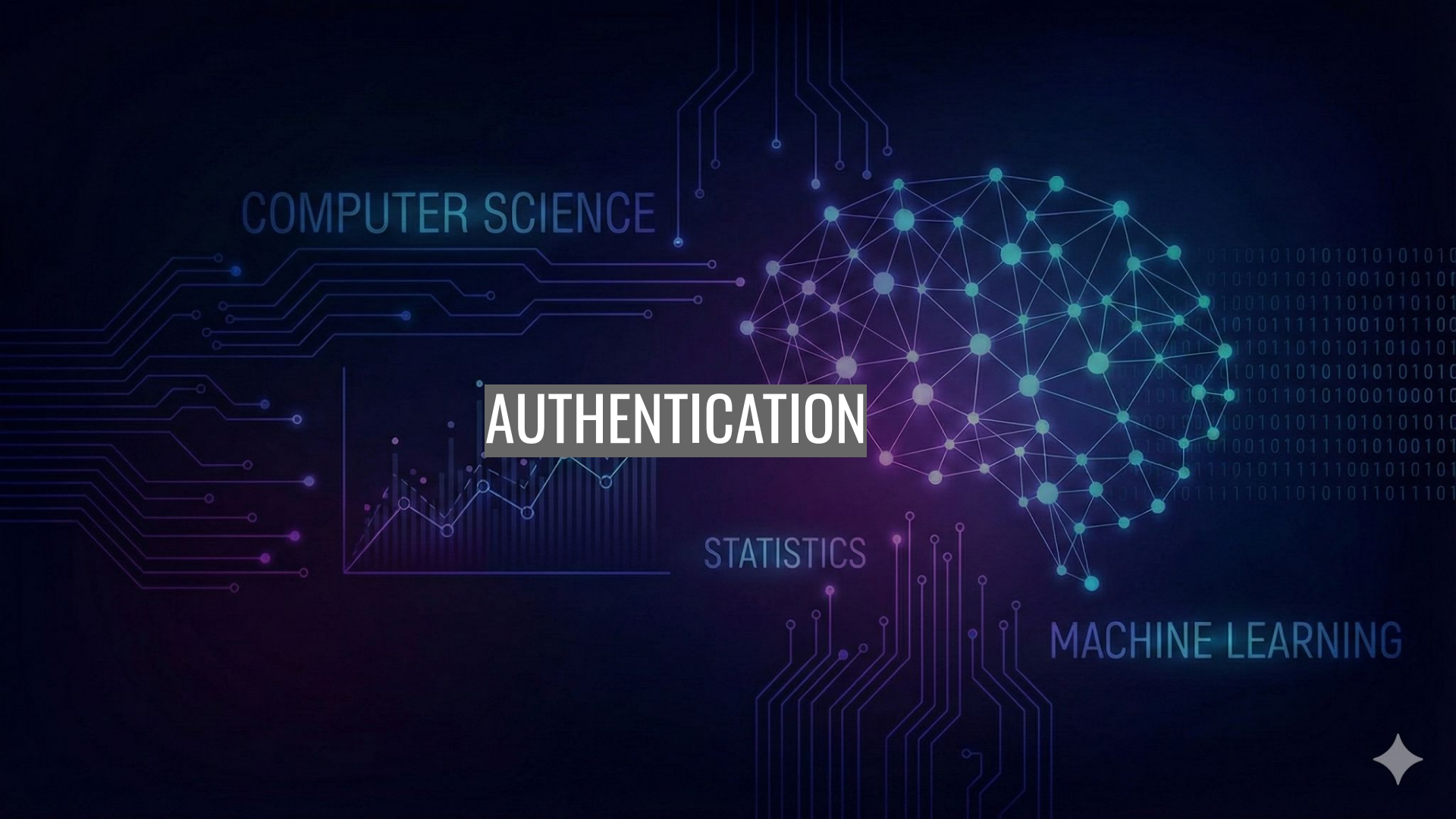
GNU Privacy Guard

Free Software Suite

GNU Privacy Guard (GnuPG or GPG) was developed as a free software implementation.

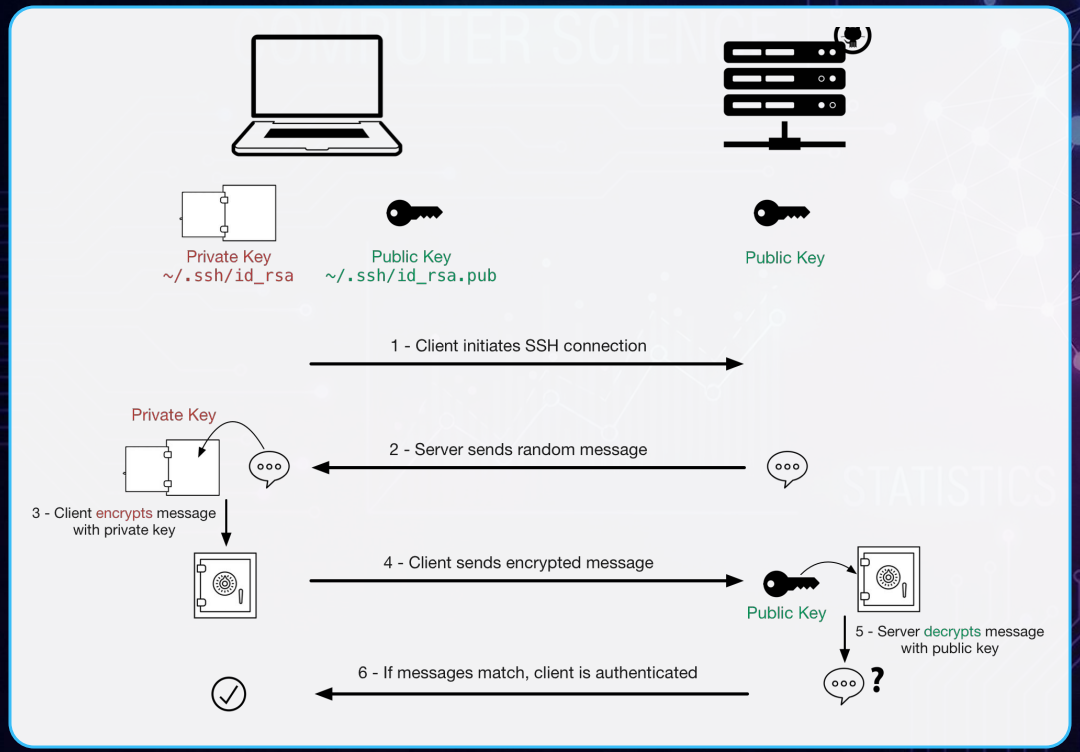
PGP Replacement

Designed specifically to replace the original PGP cryptographic suite while ensuring open standards.


COMPUTER SCIENCE
AUTHENTICATION
STATISTICS
MACHINE LEARNING

CRYPTOGRAPHY & AUTHENTICATION

SSH Public Key Authentication Example



WORKFLOW BREAKDOWN

Challenge-Response Steps

1. Connection & Challenge

Client initiates SSH. Server returns a random challenge message.

2. Private Key Encryption

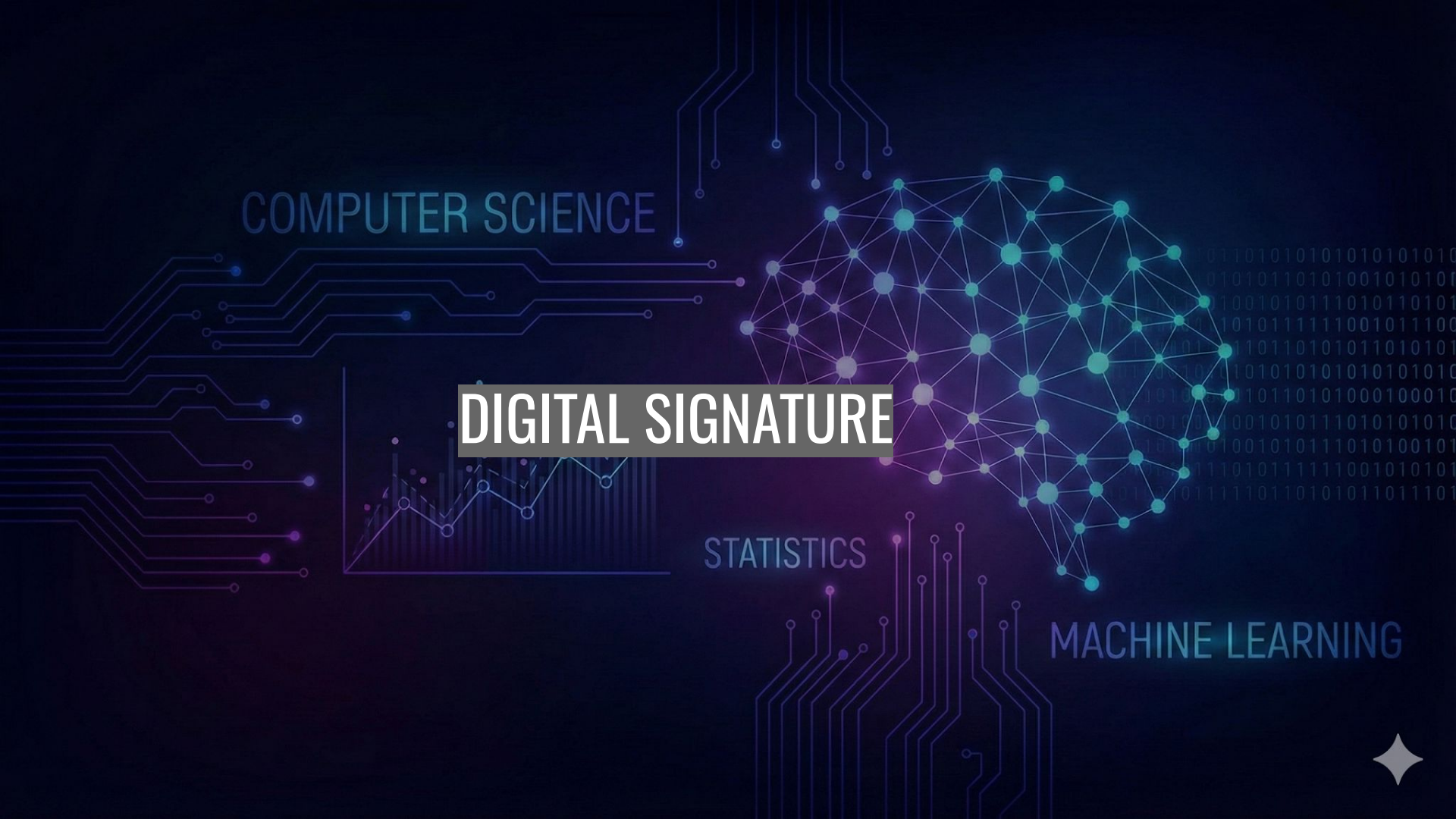
Client encrypts message using local private key (~/.ssh/id_rsa).

3. Public Key Decryption

Server decrypts payload using pre-stored public key (id_rsa.pub).

4. Verification

If original & decrypted messages match, access is granted.


COMPUTER SCIENCE
DIGITAL SIGNATURE
STATISTICS
MACHINE LEARNING

CRYPTOGRAPHY & SECURITY

Digital Signature Fundamentals

PRIMARY PURPOSE

Core Security Guarantees

1. Authenticity

Guarantees the origin of the message and verifies sender identity.

2. Integrity

Ensures the message contents have not been modified or tampered with in transit.

3. Non-Repudiation

Prevents the original signer from denying having signed and sent the message.

PROPERTIES & TECH

How It Works

Sender Specificity

Only the legitimate sender possesses the private key to produce that specific signature.

Universal Verification

Anyone can verify signatures across documents, text, sound, images, and videos.

Core Building Blocks

Powered by **asymmetric encryption** algorithms combined with **cryptographic hash functions**.

DIGITAL SIGNATURE FORMATS

Signature Methods

CASDES

pdf.p7m extension

The file has the **pdf.p7m extension** and can be read with signature software (File Protector, DiKe, etc.).

PADES

PDF extension

The file is signed with signature software and read directly with **Acrobat Reader**.

XADES

xml.p7m extension

The XML file is **automatically read** by the software that receives it.

What is a Hash Function?

A hash function is a fundamental mathematical algorithm that takes an input (often called a "message") of any length and transforms it into a fixed-size string of characters.

This output is known as the **hash value**, **digest**, or simply the **hash**.

- **Variable Input:** You can hash a single word, a 4k image, or the entire encyclopedia.
- **Fixed Output:** The resulting hash will always be the exact same length, regardless of the input size.



Essential Hash Properties

For a hash function to be considered cryptographically secure, it must strictly adhere to three rules.



Deterministic

Feeding the exact same input message into the algorithm will always, without fail, yield the exact same hash output. It involves no randomness.



Pre-Image Resistance

It is a one-way street. It must be computationally impossible to reverse-engineer or reconstruct the original input message starting from the hash output alone.



Collision Resistant

It must be astronomically improbable to find two completely different input messages that produce the exact same resulting hash output.

Hashing vs. Encryption

These terms are often confused, but they serve entirely different cryptographic purposes.

Encryption

Two-Way Process (Reversible)

Encryption is designed to hide information in transit. It uses mathematical *keys* to scramble data, and requires those same keys to unlock and read the data later.

Purpose: Confidentiality and Privacy.

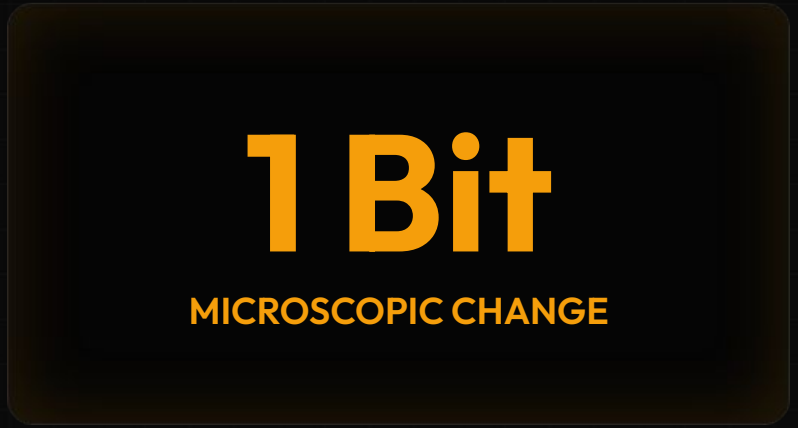
Hashing

One-Way Process (Irreversible)

Hashing does not use keys. Once data is hashed, it cannot be un-hashed. It is used to generate a unique digital fingerprint of the data to prove it hasn't changed.

Purpose: Verification and Integrity.

The Avalanche Effect



Total Randomization

A crucial feature of strong hash functions is the **Avalanche Effect**.

If you make a microscopic change to the input file—even flipping a single binary bit, altering one pixel in an image, or changing an uppercase letter to lowercase—it causes a catastrophic cascade inside the algorithm.

The resulting output hash will be completely unrecognizable compared to the previous hash. This prevents attackers from guessing the input by analyzing patterns in the output.

Algorithm Example: SHA-256

Secure Hash Algorithm (256-bit)

Developed by the NSA, SHA-256 is one of the most secure and widely used hashing algorithms in the world. It always outputs a 64-character hexadecimal string regardless of input size.

Input 1: **"Hello"**

```
185f8db32271fe25f561a6fc938b2e264306ec304eda518007d1764826381969
```

Input 2: **"hello"** (lowercase 'h')

```
2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824
```

Avalanche Effect: Notice how changing just one single letter completely scrambles the output hash!

Real-World Applications



Password Storage

Databases never store your actual password. They only store the *hash* of your password. If a hacker breaches the database, they only steal useless, irreversible hashes.



Digital Signatures

When downloading software, companies provide a "Checksum" (a hash). You hash the downloaded file on your computer; if it matches, the file is safe and uncorrupted.



Blockchains

Cryptocurrencies link blocks of data together by placing the hash of the previous block inside the new one. Tampering with past data destroys the mathematical chain.

Digital Signature: HASH

Hash Algorithms: MD5 • SHA-1 • RIPEMD • SHA-256

Fixed-Sized Output

A mathematical function that, given a variable-sized bitstream, returns a fixed-sized string of letters or numbers (acting as a unique one-time stamp).

Unique Identifier

The generated string is a unique identifier. Changing just 1 single bit of the source stream produces a completely different HASH.

One-Way Invertibility

It is strictly non-invertible, meaning it is mathematically impossible to determine or reconstruct the original flow from the returned string.

Digital Signature: HASH

Hash Algorithms: MD5 • SHA-1 • RIPEMD • SHA-256

```
└─ $ ▶ date
dom 22 mar 2026, 16:28:42, CET
(base) |redo@rbuchner --> /home/redo/Downloads
└─ $ ▶ date  | sha256sum
a57ea038ba20eb66ba187ba674c3bc095dda473df43438c3c321409b196b0462  -
(base) |redo@rbuchner --> /home/redo/Downloads
└─ $ ▶ date  | sha256sum
5a6989fb0c3d9dc115e2791f2a948c330b1ae9ebdefb3c73c4fd32e143718b7f  -
(base) |redo@rbuchner --> /home/redo/Downloads
```

Digital Signature: How It Works

Asymmetric Cryptography Workflow • Hash Digest & Key Pairs

1. Signing Process (Alice)

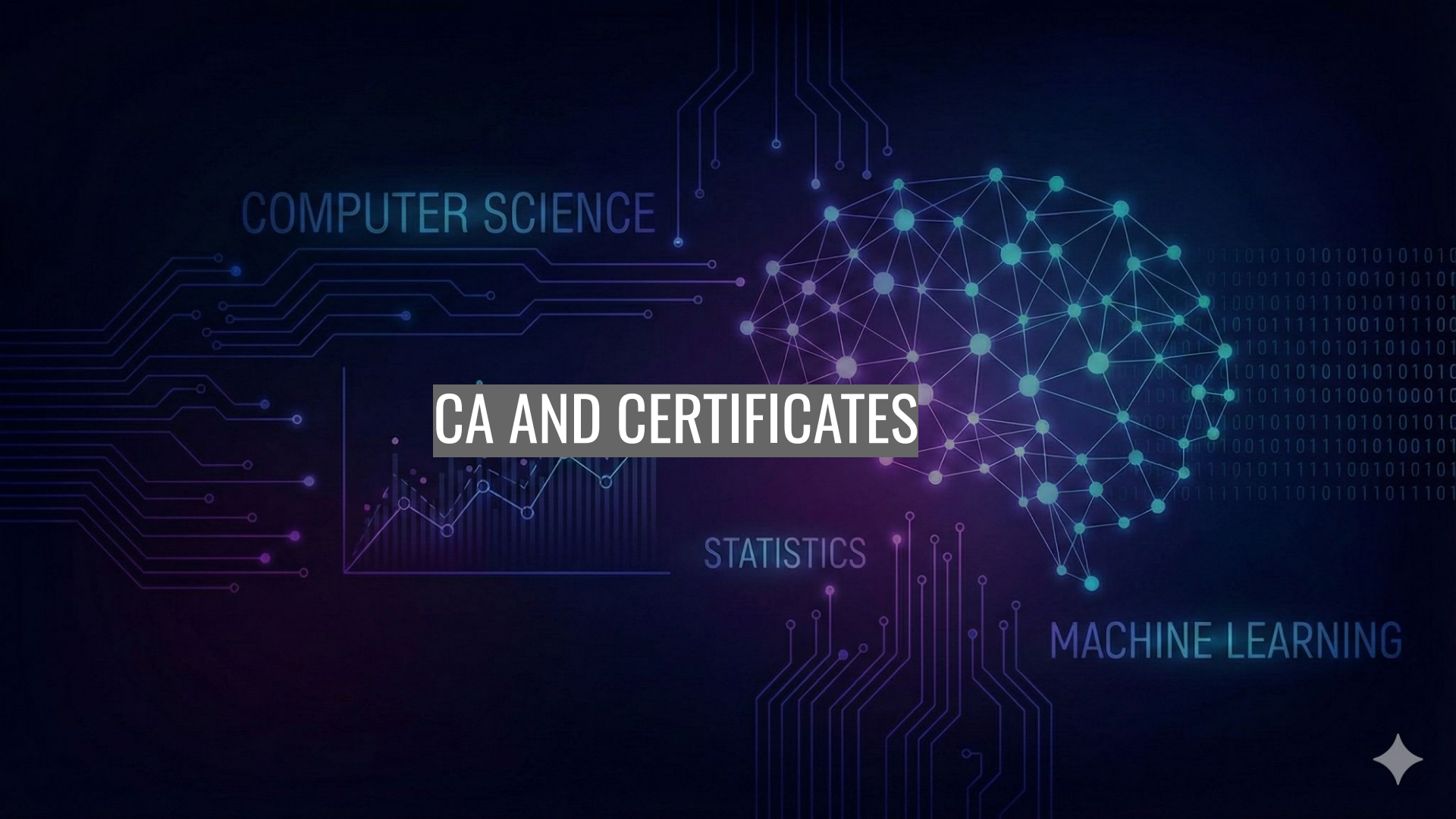
Alice signs document/object O:

1. **Calculate HASH of O** (also called digest)
2. **Encrypt HASH** with Alice's private key
3. **Append encrypted HASH** (signature F) and public key to object O

2. Verification Process (Bob)

Bob verifies Alice's signature:

1. **Calculate HASH of O** independently
2. **Decrypt signature F** using Alice's public key
3. **Check values match** — if identical, the signature is valid


COMPUTER SCIENCE
CA AND CERTIFICATES
STATISTICS
MACHINE LEARNING

CA and Certificates

Public Key Infrastructure • Identity Verification & Trust Model

The Fundamental Challenge

How can I be sure that the signature used actually belongs to the signer? How can we guarantee the user–public key association?

Digital Certificates

Purpose & Mechanism

Digital certificates serve this purpose by binding public keys to verified user identities.

They contain key data, owner details, and cryptographic metadata.

Functions like physical paper certificates to provide verified user info.

Certification Authority

The Trust Anchor (CA)

The CA guarantees the association between the Public Key and the owner's identity.

Acts as a trusted third-party issuer recognized across the network.

Validates identity claims prior to issuing signed digital certificates.

Key Association

Identity Verification

Prevents impersonation by securely linking public keys to real-world identities.

Ensures recipients can trust signature validity without out-of-band checks.

Forms the backbone of PKI trust chains and secure communication.

CA and Certificates

X.509 Standard • Certificate Anatomy & Trust Model

Digital Certificates (X.509) Consist Of:

- Owner's public key
- **His identity** (name, surname, date of birth, etc.)
- Public key expiration date
- Name of the CA that issued it
- **Digital signature of the CA that issued the certificate**

Verification Principle

If we trust the **CA (Certification Authority)** we can verify its signature and therefore the identity of the signatory.

The CA's signature acts as a seal of authenticity, binding the owner's identity to their public key.

CA and Certificates

Public Key Infrastructure • Trust Chain & Validation Model

Certification Authority

Identity Guarantee

Guarantees the binding between a digital signature and the owner's true identity.

Trusted Issuer

Acts as an authoritative third-party entity recognized across the network to establish digital trust.

Signing Process

Authenticity Certification

Signed directly by the CA to certify authenticity and integrity of the document.

Private Key Encryption

Executed by attaching CA contact metadata and encrypting the entire document with the CA's private key.

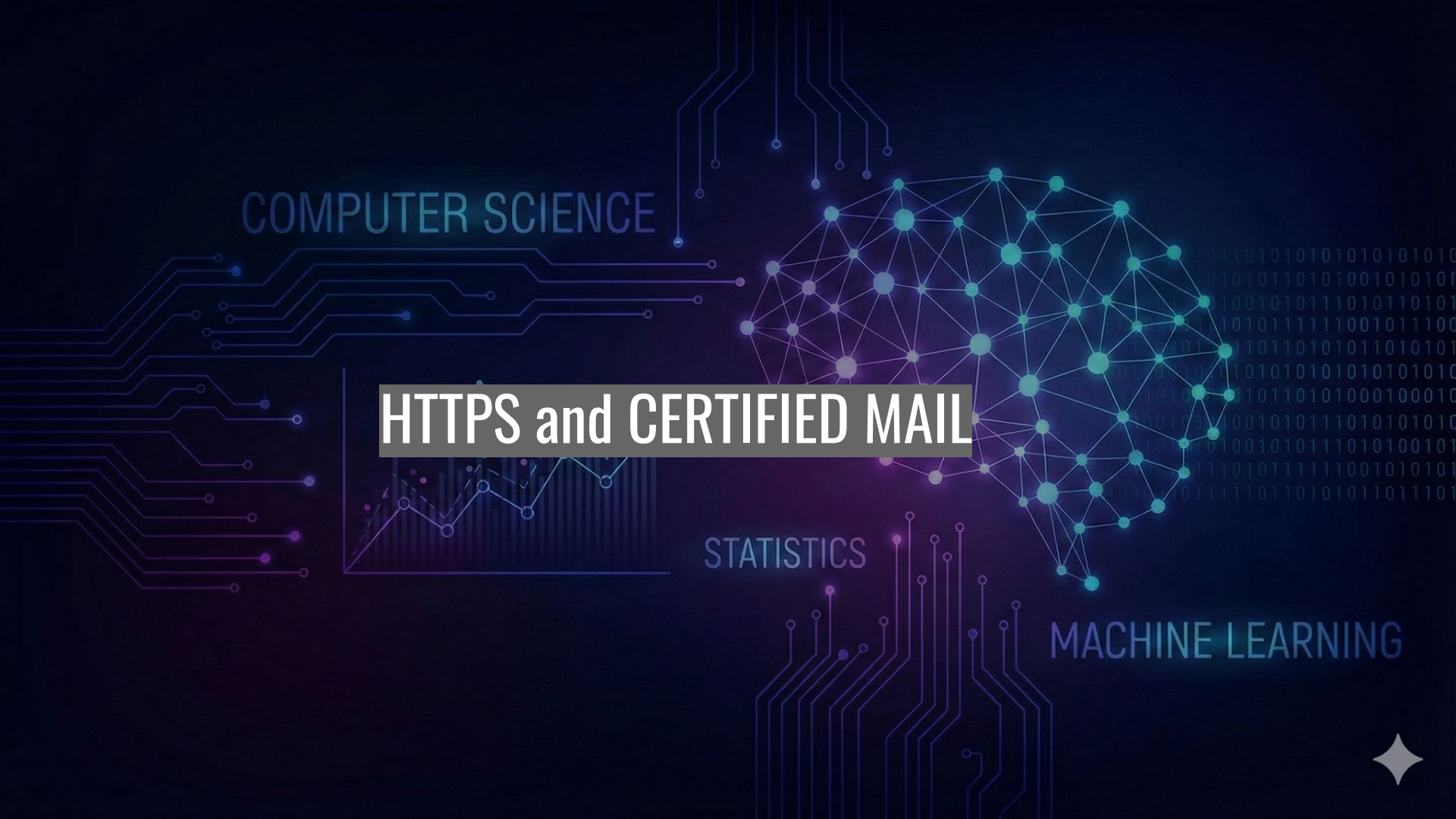
Chain of Trust

Hierarchical Verification

If an issuing CA is not locally trusted, the system verifies that CA's own certificate issued by a higher authority.

Root CA Validation

Repeats iteratively up the chain until reaching an explicitly trusted Root CA, validating the full path.


COMPUTER SCIENCE
HTTPS and CERTIFIED MAIL
STATISTICS
MACHINE LEARNING

HTTPS Protocol

Secure Web Communications • Fundamental for E-Commerce & Online Banking

1. Server Identity

Public Key Certificate

When connecting via HTTPS, the server declares its identity by sending its public key certificate guaranteed by a CA.

Certificate Authority

Ensures digital trust through cryptographically signed credentials from an accredited CA.

2. Client Verification

Domain Matching

My browser (client) verifies Hostname/Domain matches the identity contained in the certificate.

Integrity Validation

Prevents impersonation attacks by ensuring exact cryptographic alignment before establishing session trust.

3. TLS 1.3 Key Exchange

Diffie-Hellman Method

In modern HTTPS (TLS 1.3), client and server use Diffie-Hellman to generate the same key independently.

Perfect Forward Secrecy

The key never travels across the network, so even if the server's Private Key is stolen later, past conversations remain safe.

Certified Mail (PEC)

Step A: Submission & Acceptance (Ricevuta di Accettazione)

PEC System

Accredited Providers

Unlike standard email where you just send data to a server, in the PEC system both Sender's Provider and Recipient's Provider must be certified and accredited by a government body (in Italy, AgID).

Guarantor Role

They act as the official guarantors of the communication.

Submission Flow

1. Send Email

You (Sender) send the email to your PEC provider.

2. Security Check

Your Provider checks the email for viruses and compliance.

3. Issue Receipt

Your Provider issues a Ricevuta di Accettazione (Acceptance Receipt) back to you.

What It Proves

Digital Signature

This receipt is digitally signed by your provider.

Timestamp & Recipient Proof

It legally proves when (timestamp) you sent the message and who you sent it to.

Legal Protection

Even if the recipient claims they never got it, you have legal proof you sent it.

Certified Mail (PEC)

Step B: The Transport Envelope (Busta di Trasporto)

1. Packaging

Digital Envelope

Your Provider does not just forward your email. It wraps your original message inside a new digital package called a Busta di Trasporto (Transport Envelope).

Original Message

Preserves the exact content and attachments intact inside the outer container.

2. Provider Signature

Digital Signature

Your Provider digitally signs this newly wrapped envelope before sending it onward.

Trust Anchor

Confirms the communication originates from an officially accredited PEC provider.

3. What It Proves

Message Integrity

Guarantees Integrity. The recipient's provider can verify that the message was not altered in transit.

MitM Prevention

Protects against Man-in-the-Middle tampering during transmission between providers.

Certified Mail (PEC)

Step C: Delivery (Ricevuta di Consegna)

1. Delivery Flow

Envelope Transfer

Your Provider sends the envelope to the Recipient's Provider.

Verification

Recipient's Provider verifies digital signature (checking valid PEC provider).

Inbox Placement

Recipient's Provider puts the message into recipient's inbox.

2. Delivery Receipt

Receipt Issuance

Recipient's Provider issues a Ricevuta di Consegna (Delivery Receipt) and sends it back to You (Sender).

Critical Confirmation

This is the most critical receipt in the certified email process, completing the legal chain of transmission.

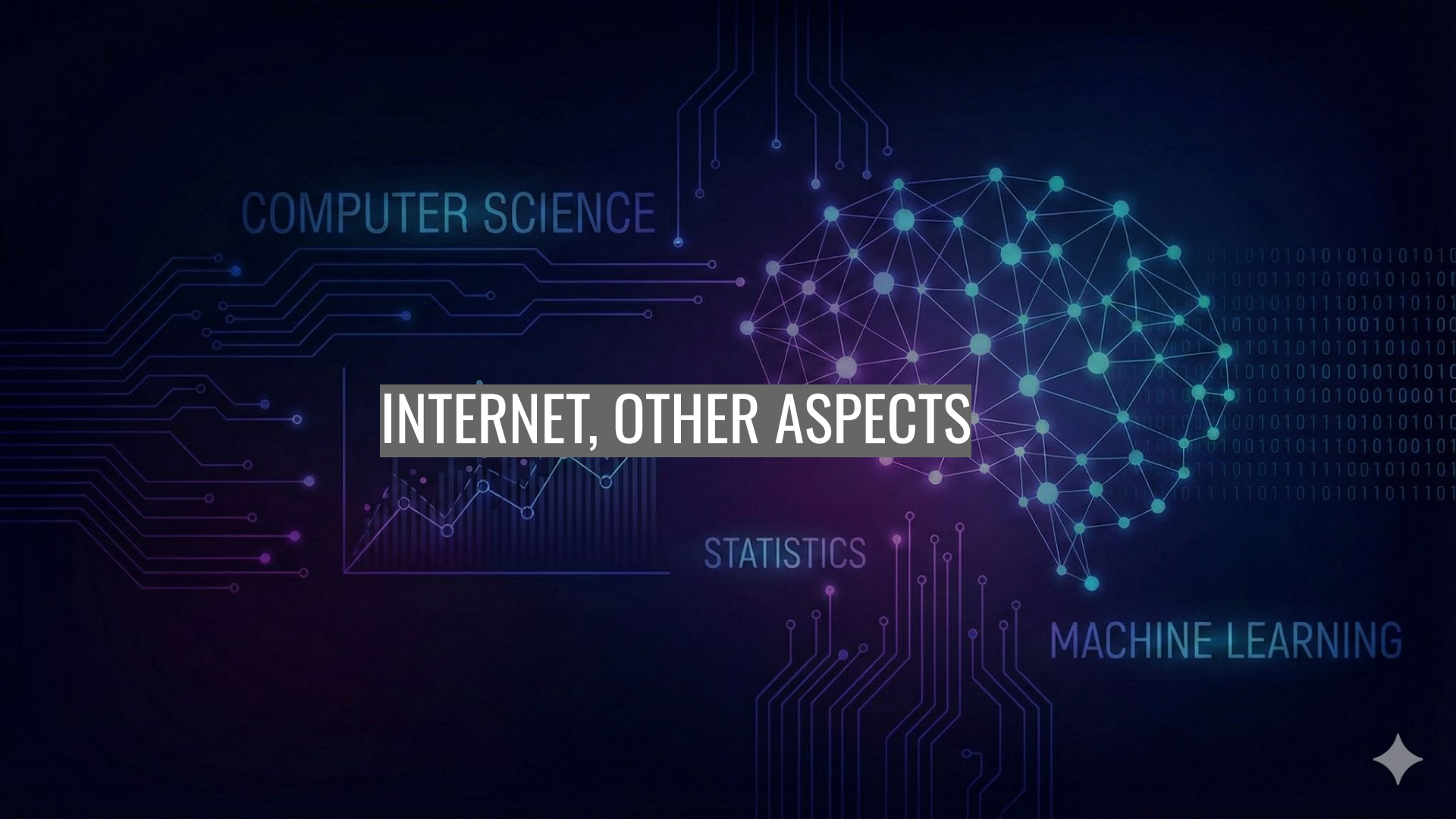
3. What It Proves

Legal Proof of Reach

Legally proves the message reached the destination address. Equivalent to return receipt (cartolina di ritorno) in traditional registered mail.

Delivery vs. Reading

Proves delivery to inbox, not that user read it (legally served upon mailbox delivery even if unopened).


COMPUTER SCIENCE
INTERNET, OTHER ASPECTS
STATISTICS
MACHINE LEARNING

DEEP WEB, TOR

ONION ROUTING: A VOLUNTEER NETWORK FOR ANONYMITY



- I can reach “hidden” services, but also normal servers via exit nodes.

Deep Web & Tor Network

Anonymity through volunteer-operated relay nodes and multi-layered encryption

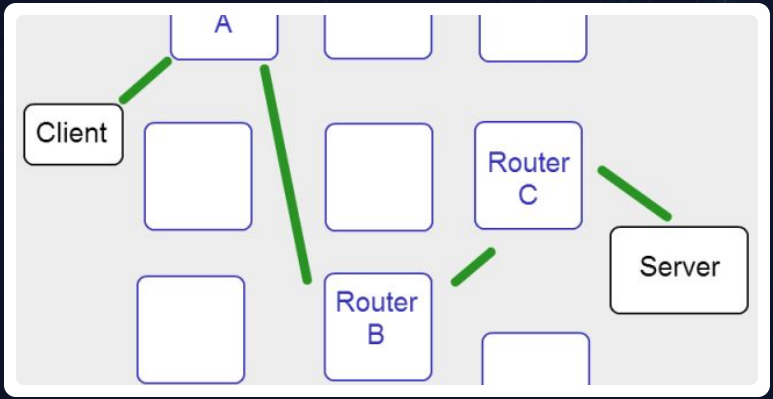
Volunteer Node Network

The Tor network is powered by volunteer computers acting as relay nodes, masking source origin and routing path.

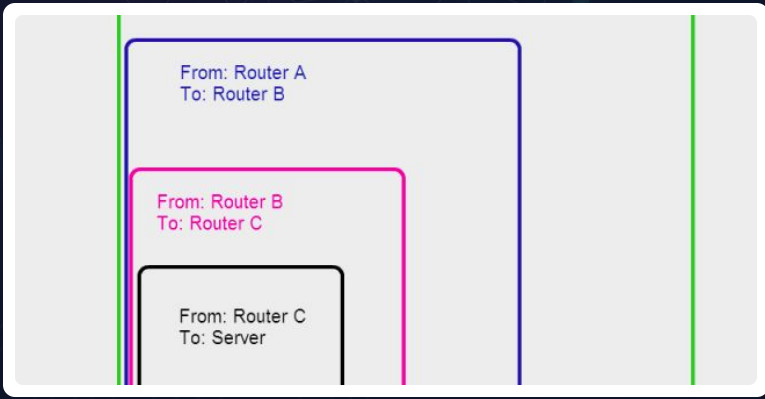
Hidden Services & Exit Nodes

Provides access to hidden onion services internally, or connects safely to normal web servers via designated exit nodes.

Circuit Path & Node Routing



Layered Encryption Structure



Deep Web & Tor Network

Connecting to normal web servers via Exit Nodes

Normal Servers via Exit Nodes

When accessing the "Clear Web", your client constructs a random 3-node circuit through volunteer nodes before exiting to the destination.

1. Entry Node (Guard)

The first hop. Sees your IP address, but cannot see the final destination or content.

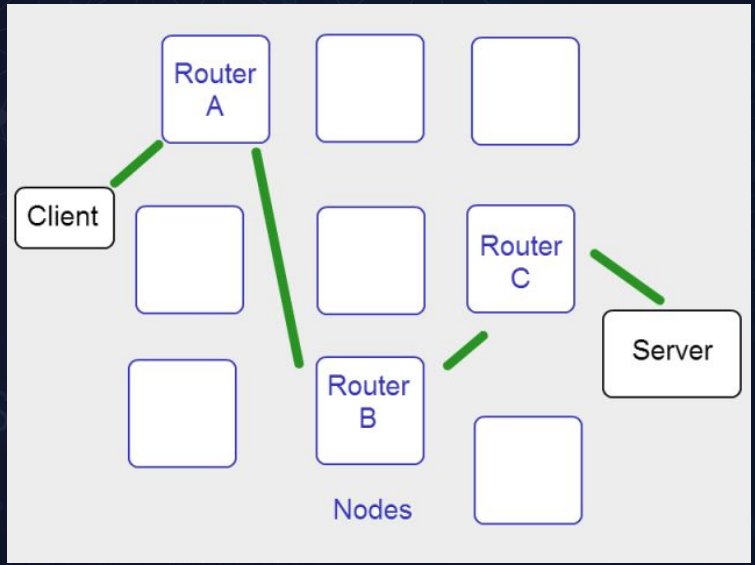
2. Middle Node

The second hop. Acts as an isolated buffer between entry and exit nodes.

3. Exit Node

The final hop. Leaves the Tor network and connects directly to the destination server for you.

Circuit Path & Node Routing



Deep Web & Tor Network

Layered Encryption: Encrypting data three times (like an onion)

1. Layer 1 (Outer) — Entry Node

Encrypted for the Entry Node. Stripped off first by the entry relay.

2. Layer 2 (Middle) — Middle Node

Encrypted for the Middle Node. Protects routing privacy between relays.

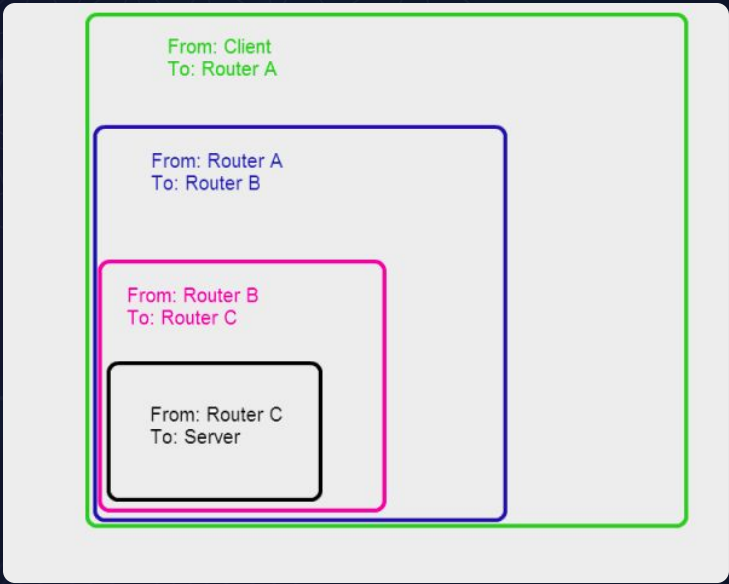
3. Layer 3 (Inner) — Exit Node

Encrypted for the Exit Node. Decrypted at the final hop before reaching the destination.

Data Packet Traversal

The data packet travels through the circuit, getting "peeled" at each step, ensuring no single node knows both source origin and destination.

Layered Encryption Structure



Deep Web & Tor Network

The Server's View & Node Visibility

The Server's View

The website (Google) sees a request coming from the Exit Node's IP address. It has no way of knowing your real IP address.

1. Entry Node

Sees the Client (You) and the Middle Node. → **It sees the Client!**

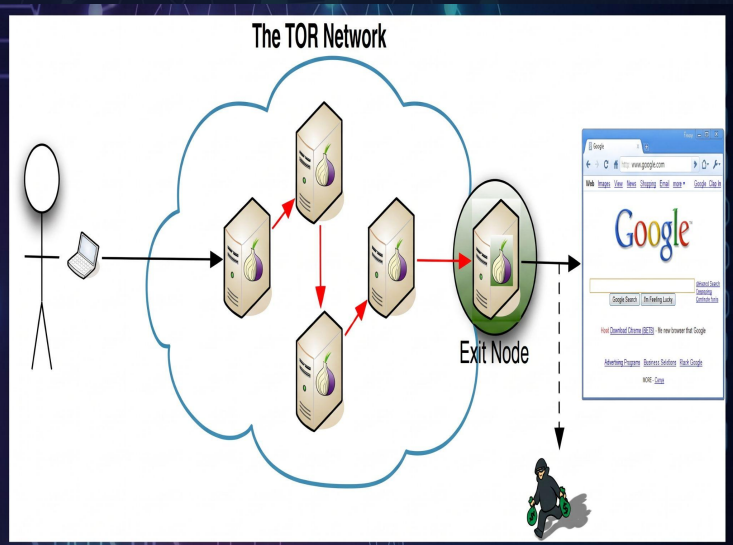
2. Middle Node

Sees the Entry Node and the Exit Node. → **It sees neither Client nor Server.**

3. Exit Node

Sees the Middle Node and the Server. → **It sees the Server!**

The TOR Network Routing



Tor Hidden Services (.onion)

How Onion Servers remain anonymous without publishing their IP address

Anonymity & .onion Addresses

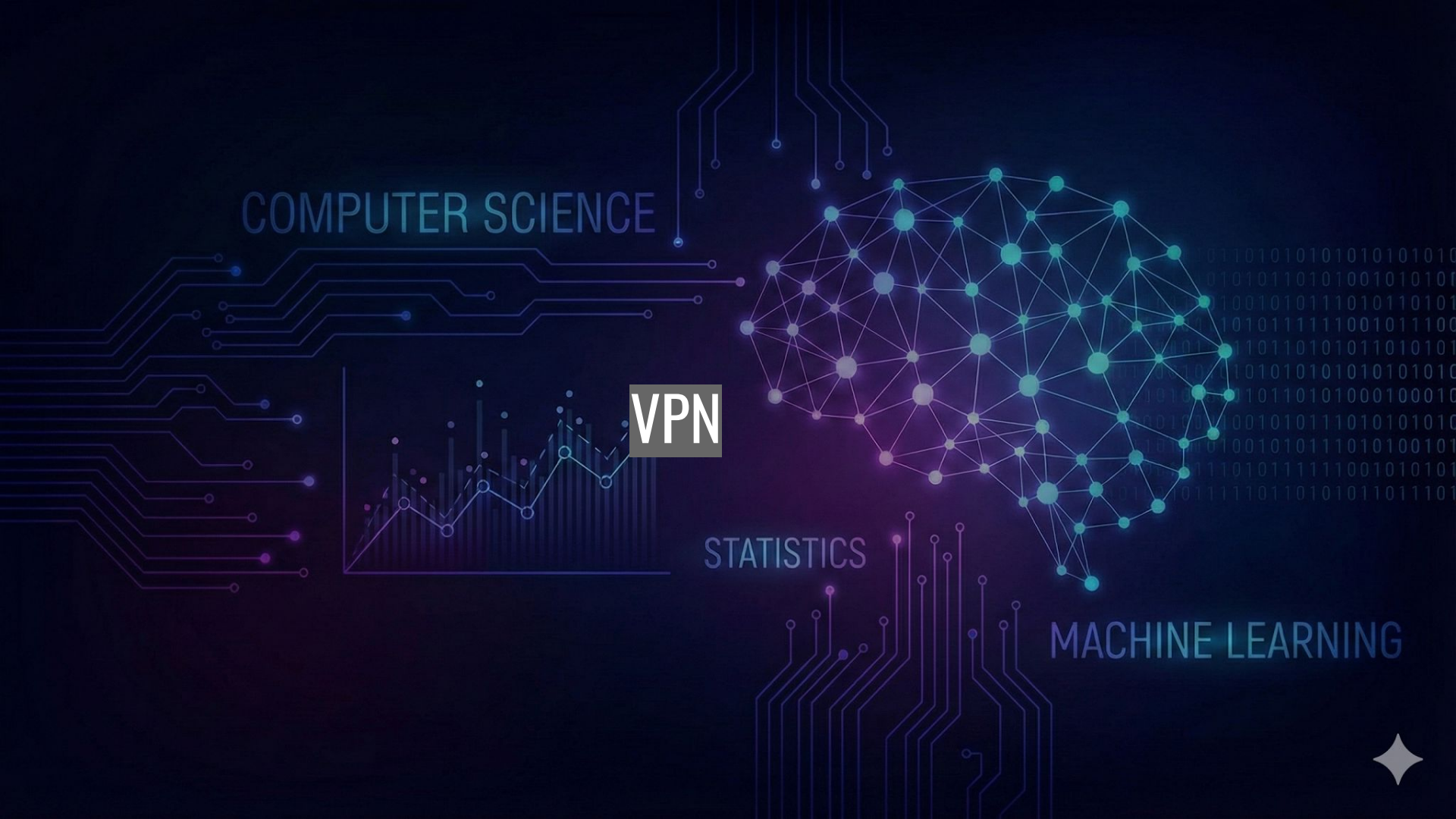
Tor Hidden Services use addresses ending in .onion. The Onion Server operates anonymously on the network and intentionally does not publish its IP address.

1. Selecting Introduction Points

The server picks 3 random relays in the network to act as its "Introduction Points." These relays serve as contact intermediaries for clients wanting to establish a connection.

2. Establishing Encrypted Circuits

It builds encrypted circuits to these Introduction Points and instructs them: "If anyone wants to talk to me, forward their message here." This allows full communication without revealing the server's real identity.


COMPUTER SCIENCE
VPN
STATISTICS
MACHINE LEARNING

VPN: Virtual Private Network

Securing internet traffic through encryption, encapsulation, and IP masking

1. The Secure Tunnel

Think of it as driving a private armored car through a public highway. The VPN creates an encrypted connection over the public Internet, isolating your data from other traffic.

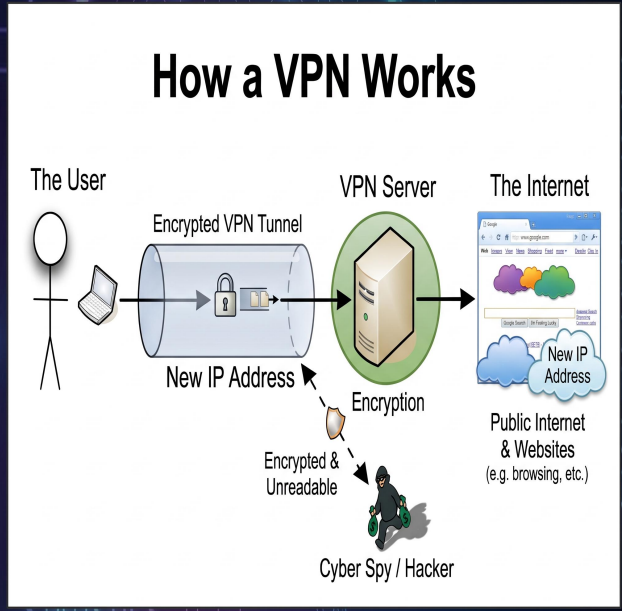
2. Encapsulation

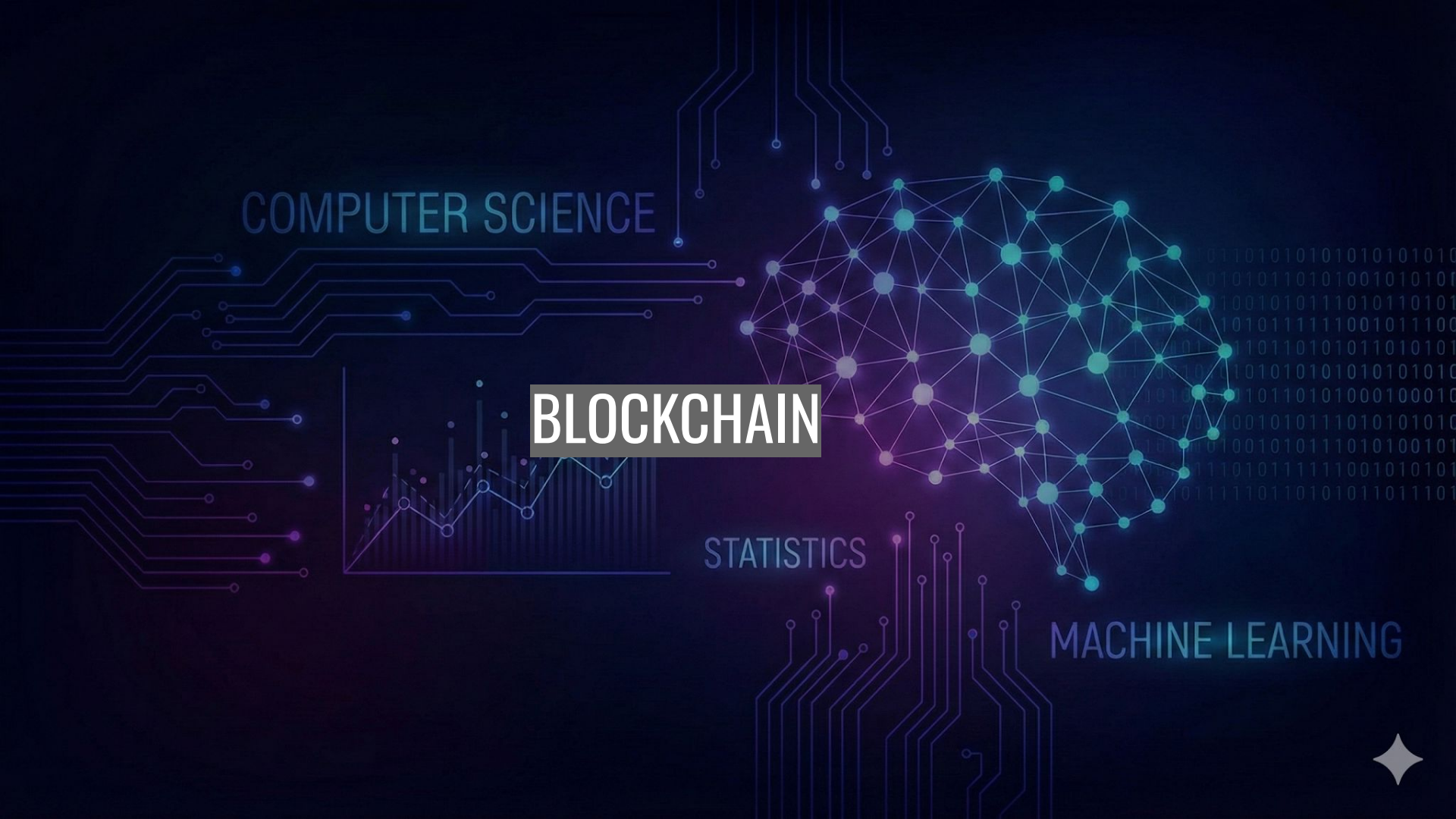
Your original data packets (Letter) are wrapped inside new VPN packets (Envelope). The ISP only sees the outer envelope, not the letter inside.

3. IP Masking

You connect to a VPN Server, and the Server connects to the website. The website sees the Server's IP address, not yours, effectively hiding your location.

How VPN Tunneling Works




COMPUTER SCIENCE
BLOCKCHAIN
STATISTICS
MACHINE LEARNING

Bitcoin: Decentralized Architecture

Peer-to-peer digital currency powered by cryptography and distributed consensus

1. Decentralized Network & Distributed Ledger

Unlike most traditional currencies, Bitcoin operates without a central authority. It utilizes a distributed database shared across network nodes to maintain transaction history, using cryptography to govern money generation and ownership attribution.

2. Cryptographic Consensus & SHA-256 Hashing

Based on cryptography and hashing algorithms, Bitcoin uses SHA-256 to generate verifiable random numbers requiring a predictable amount of computational effort. Generating a hash value lower than the target difficulty solves a block.

Blockchain: Distributed Ledger

Imagine a notebook (Ledger / "libro mastro") that records transactions.

Centralized (Traditional)

The bank keeps the single authority notebook.

You must trust the bank not to alter, edit, or modify the transaction records.

Blockchain (Decentralized)

Everyone in the network holds an exact copy of the notebook.

To write a new page, the majority of the network must agree through distributed consensus.

Blockchain: The Block (The Container)

Each block contains three main things:

1. Data

The transactions (e.g., "Alice sends 5 BTC to Bob").

2. The Hash

The block's unique digital fingerprint (SHA-256).

3. The Previous Hash

This is the magic glue.

Each block contains the Hash of the previous block.

Blockchain: The Chain (The Link)

Because each block contains the Hash of the previous block, they are mathematically linked.

Block 1 (Genesis)

Data: [Data]

Hash: **A123**

Prev Hash: 0000

Block 2

Data: [Data]

Hash: **B456**

Prev Hash: **A123**

Block 3

Data: [Data]

Hash: **C789**

Prev Hash: **B456**

Blockchain: Security & Immutability

What happens if a hacker tries to alter a transaction in Block 1?

1. Data Altered

The data inside Block 1 is modified by the attacker.

2. Hash Changes

The Hash of Block 1 changes immediately (e.g., from **A123** to **X999**).

3. Chain Breaks

Block 2 becomes invalid instantly because it expects **"Prev: A123"**, but the real previous block is now **"X999"**.

4. Cascade Task

To fix this, the hacker would have to recalculate hashes for Block 2, Block 3, and every subsequent block.

Why Blockchain is Practically Unhackable

Three core pillars that protect the network from malicious attacks

1. 51% Attack Cost

To rewrite history, an attacker must control **more than 51%** of the entire network's computing power.

For major networks, acquiring this immense hardware and energy is **financially impossible**.

2. Distributed Consensus

Every node holds a full copy of the ledger.

If one node alters data locally, the thousands of other honest nodes **instantly reject** the fraudulent copy during validation.

3. Cryptographic Domino Effect

Changing a single historical transaction breaks every hash down the line.

The attacker would have to **re-mine every block faster** than the rest of the world combined.