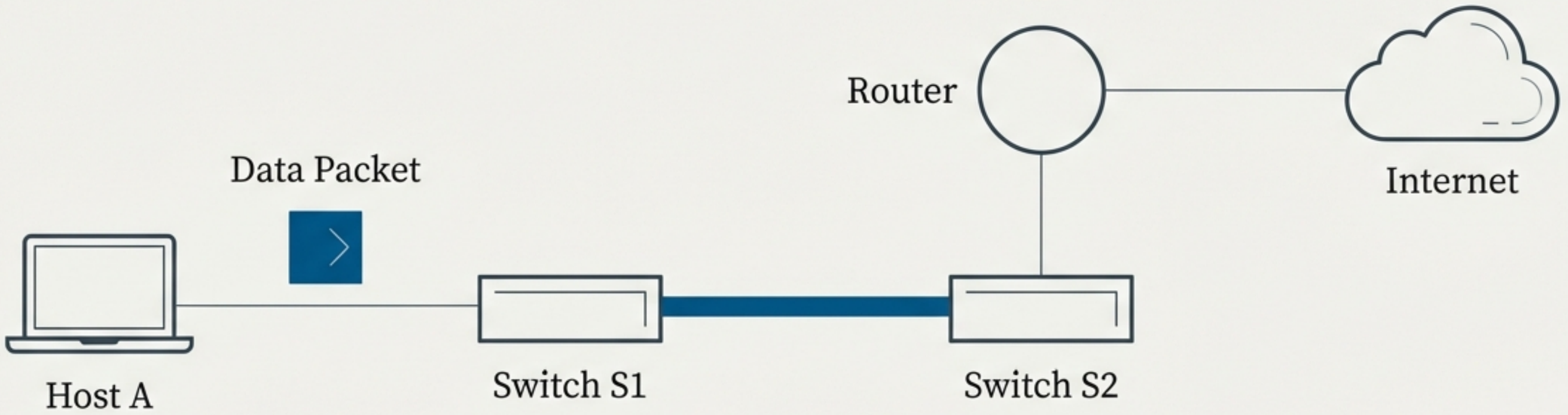
A Packet's Journey Through the Switched Network

From a local host to the wider internet, one frame at a time.



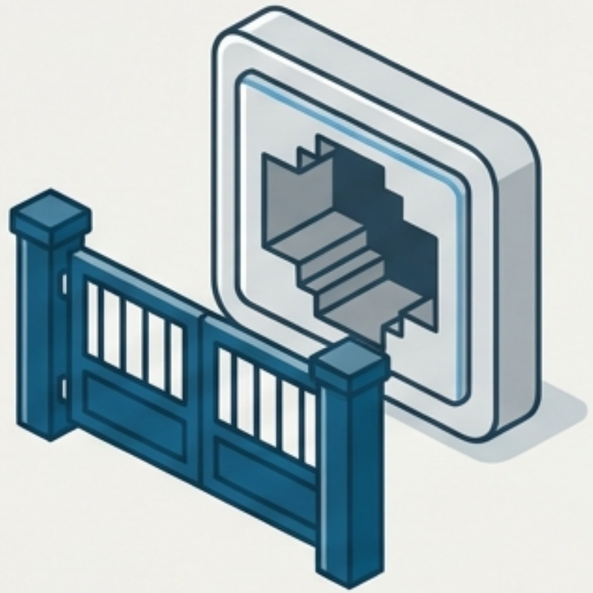
The First Guardian: Securing the Network's Edge

Before any data is sent, the network ensures the device is authorized. Layer 2 Port Security is the first checkpoint, restricting access on a switch port to specific MAC MAC addresses. By default, a switch port is open, but this can be a significant security risk.

Key Concepts

- **Purpose:** Prevents unauthorized users from plugging a device into an open switch port.
- **Mechanism:** Limits the number of MAC addresses that can use a port, or assigns static MAC addresses.

```
// Set port to non-trunking mode
Switch(config-if)# switchport mode access
// Enable port security on the interface
Switch(config-if)# switchport port-security
// Set the maximum number of secure MACs
Switch(config-if)# switchport port-security maximum 1
// Define the action to take on violation
Switch(config-if)# switchport port-security violation shutdown
// Learn the MAC address dynamically and save it
Switch(config-if)# switchport port-security mac-address sticky
```



Violation Modes Explained

Shutdown (Default): Disables the port entirely.

Restrict: Drops violating packets and sends a security notification (SNMP trap).

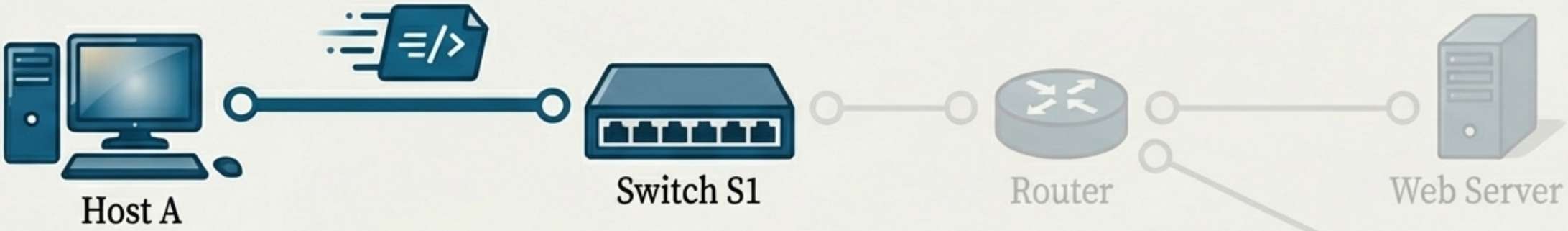
Protect: Silently drops violating packets with no notification.

The Journey Begins: A Request is Made

Scenario

A user on ‘Host A’ wants to access a website. A data packet is created, encapsulated in an Ethernet frame.

Source IP: Host A's IP	**Destination IP:** [Web Server IP]
Source MAC: Host A's MAC (0000.8c01.000A)	**Destination MAC:** Default Gateway's MAC



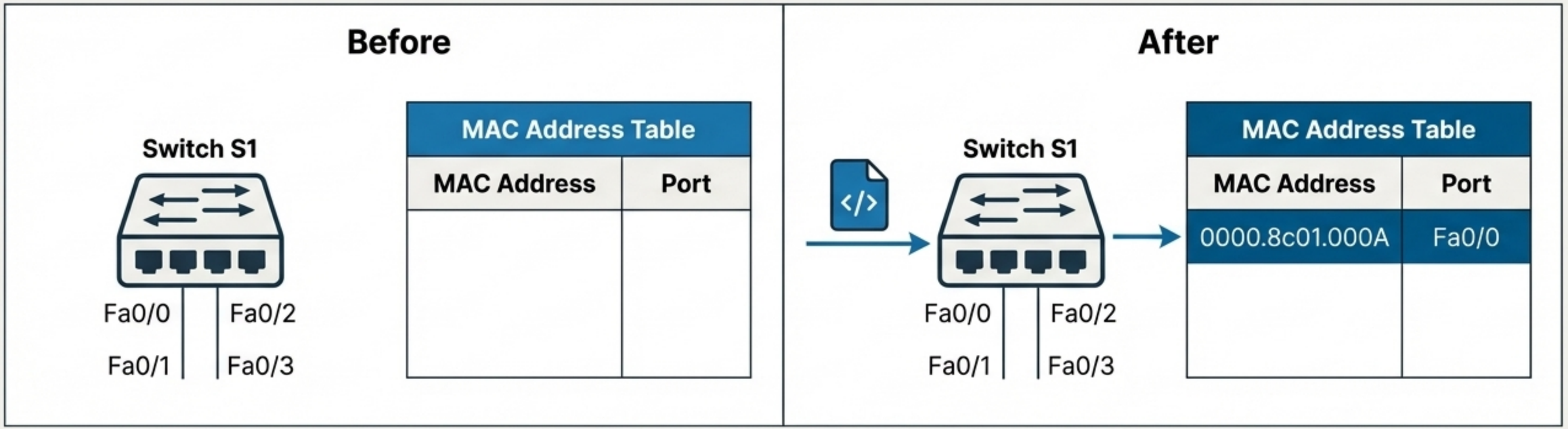
The First Challenge: The frame arrives at Switch S1. The switch has many ports. How does it know where to send it?

Building Intelligence: How a Switch Learns

Core Concept: Layer 2 switches build a MAC address table (also called a forward/filter table or CAM table) to map devices to physical ports. This process has two key steps.

Step 1: Address Learning

- When a frame arrives at a switch port (e.g., Fa0/0), the switch inspects the *source* MAC address.
- It records this source MAC address and the port it came from in its table. This is how the switch learns where devices are located.

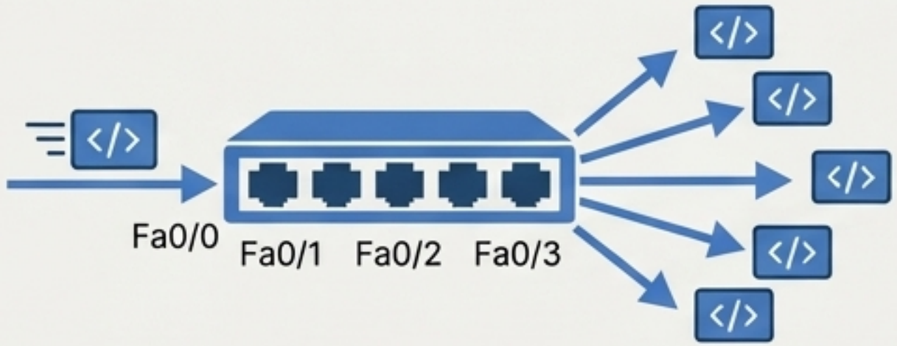


The Crossroads: The Forward or Flood Decision

**Core Concept*: After learning the source, the switch examines the *destination* MAC address to decide the frame's fate.

1. Unknown Destination (Flooding)

If the destination MAC is *not* in the MAC address table, the switch has no choice but to flood the frame. It sends a copy out of *every active port* except the one it arrived on.



2. Known Destination (Filtering/Forwarding)

If the destination MAC *is* in the table, the switch forwards the frame *only* out of the associated port. This preserves bandwidth and is the core of a switch's efficiency.

CLI 'Proof'

```
Switch# sh mac address-table
Vlan  Mac Address          Type       Ports
----  --------------       ----       -----
1     0005.dccb.d74b       DYNAMIC    Fa0/1
1     000a.f467.9e8c       DYNAMIC    Fa0/3
1     0010.7b7f.c2b0       DYNAMIC    Fa0/3
```

A frame destined for `000a.f467.9e8c` will be sent *only* to port Fa0/3.

Crossing Boundaries: Life in a Virtual LAN (VLAN)

Problem/Solution

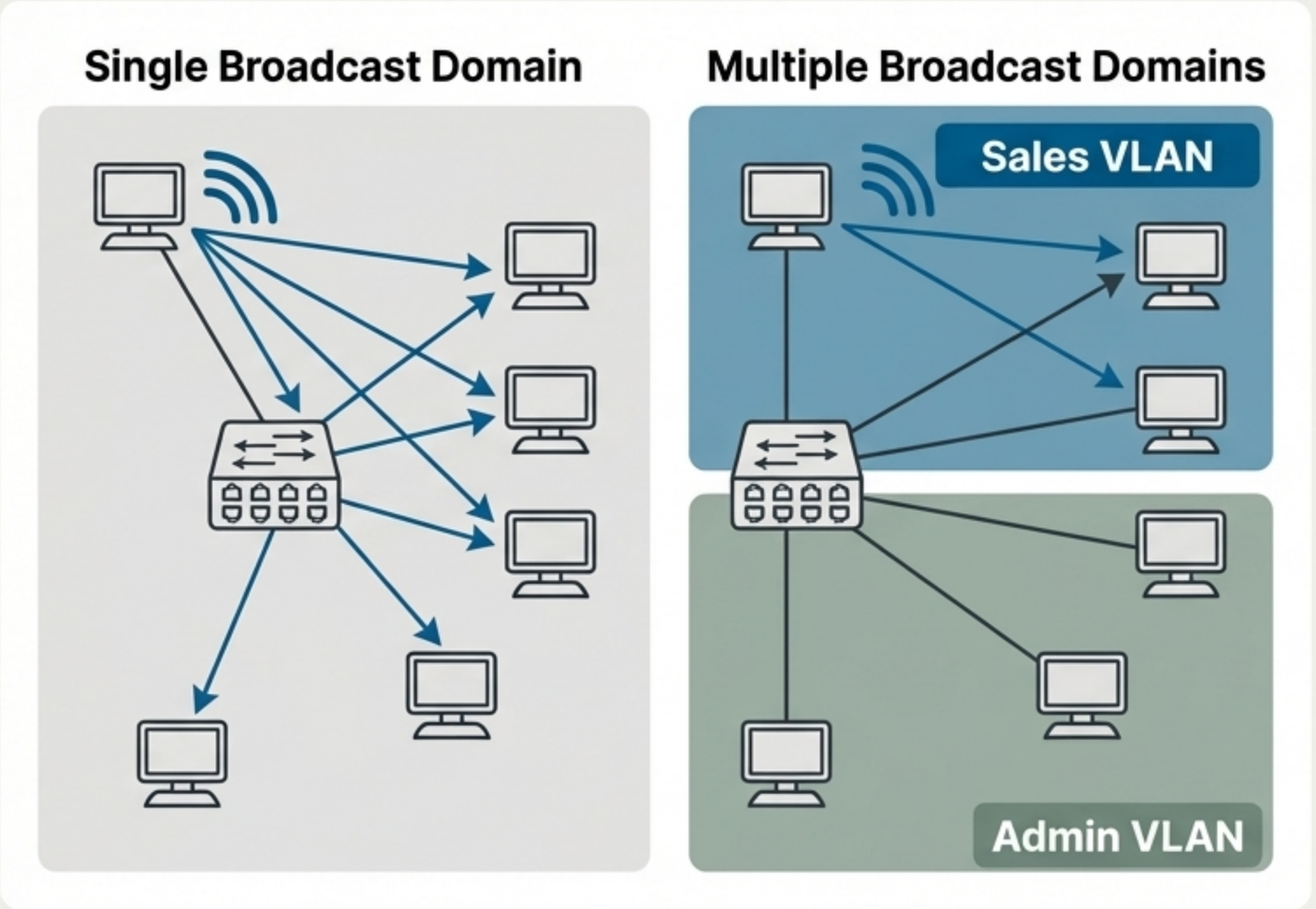
The Problem: In a flat network, every device is in one large broadcast domain. This creates issues with broadcast traffic, security, and management.

The Solution: VLANs: A VLAN is a logical grouping of devices, creating smaller, isolated broadcast domains. Devices in one VLAN cannot directly communicate with devices in another, even if they are connected to the same physical switch.

Benefits:

- **Broadcast Control:** Broadcasts are contained within a VLAN.
- **Security:** Isolates groups of users and resources.
- **Flexibility:** Users can be grouped by function (e.g., Sales, Marketing) regardless of physical location.

Before and After



The Bridge Between Worlds: Trunk Links and Frame Tagging

The Challenge

How do you connect two switches and preserve the VLAN separation for devices on both?

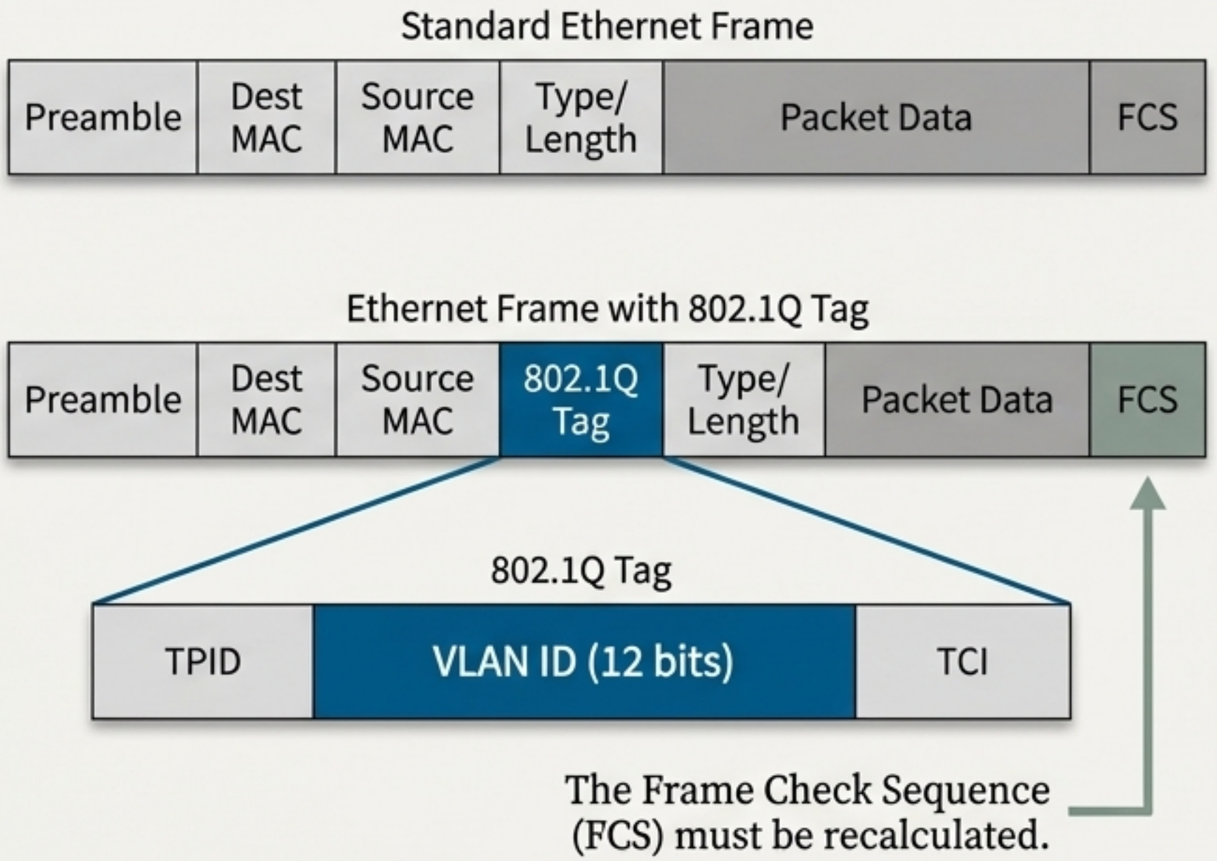
Two Link Types

- **Access Port:** Belongs to a single VLAN. Connects to end devices like PCs and printers.
- **Trunk Port:** Can carry traffic for *multiple* VLANs simultaneously. Connects switches to other switches or routers.

How it Works: Frame Tagging (802.1Q)

As a frame from a specific VLAN prepares to cross a trunk link, the switch inserts a 'tag' into the Ethernet header. This tag contains the VLAN ID (a 12-bit field), allowing the receiving switch to know which VLAN the frame belongs to.

Visual Comparison: Before and After Tagging



Configuring the Trunk

Core Commands

The `switchport mode` command defines the port's behavior.

- **`switchport mode access`**: Sets the port to belong to a single VLAN.
- **`switchport mode trunk`**: Puts the port into permanent trunking mode.
- **`switchport mode dynamic desirable`**: Actively attempts to form a trunk.

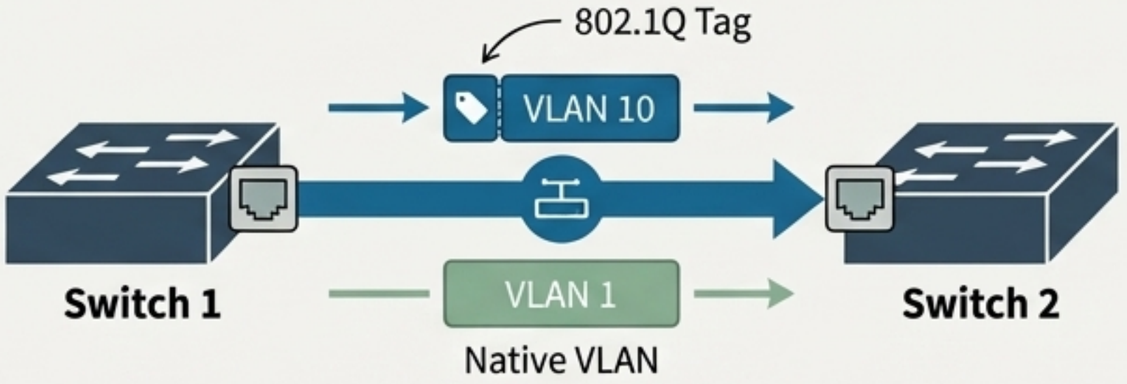
CLI Example

```
S1(config)# int fa0/15
// Specify 802.1Q as the trunking protocol
S1(config-if)# switchport trunk encapsulation dot1q
// Set the interface to trunking mode
S1(config-if)# switchport mode trunk
```

**Note: The `encapsulation` command is only needed on switches that support multiple trunking protocols (like ISL). Modern switches that only use 802.1Q do not require it.*

The Native VLAN

- An 802.1Q trunk port has a "native VLAN" (VLAN 1 by default).
- Traffic belonging to the native VLAN crosses the trunk *untagged*.
- For proper operation, the native VLAN must match on both ends of the trunk link.




```
%CDP-4-NATIVE_VLAN_MISMATCH: Native VLAN mismatch
discovered on FastEthernet0/15 (1), with Switch2
FastEthernet0/15 (10).
```

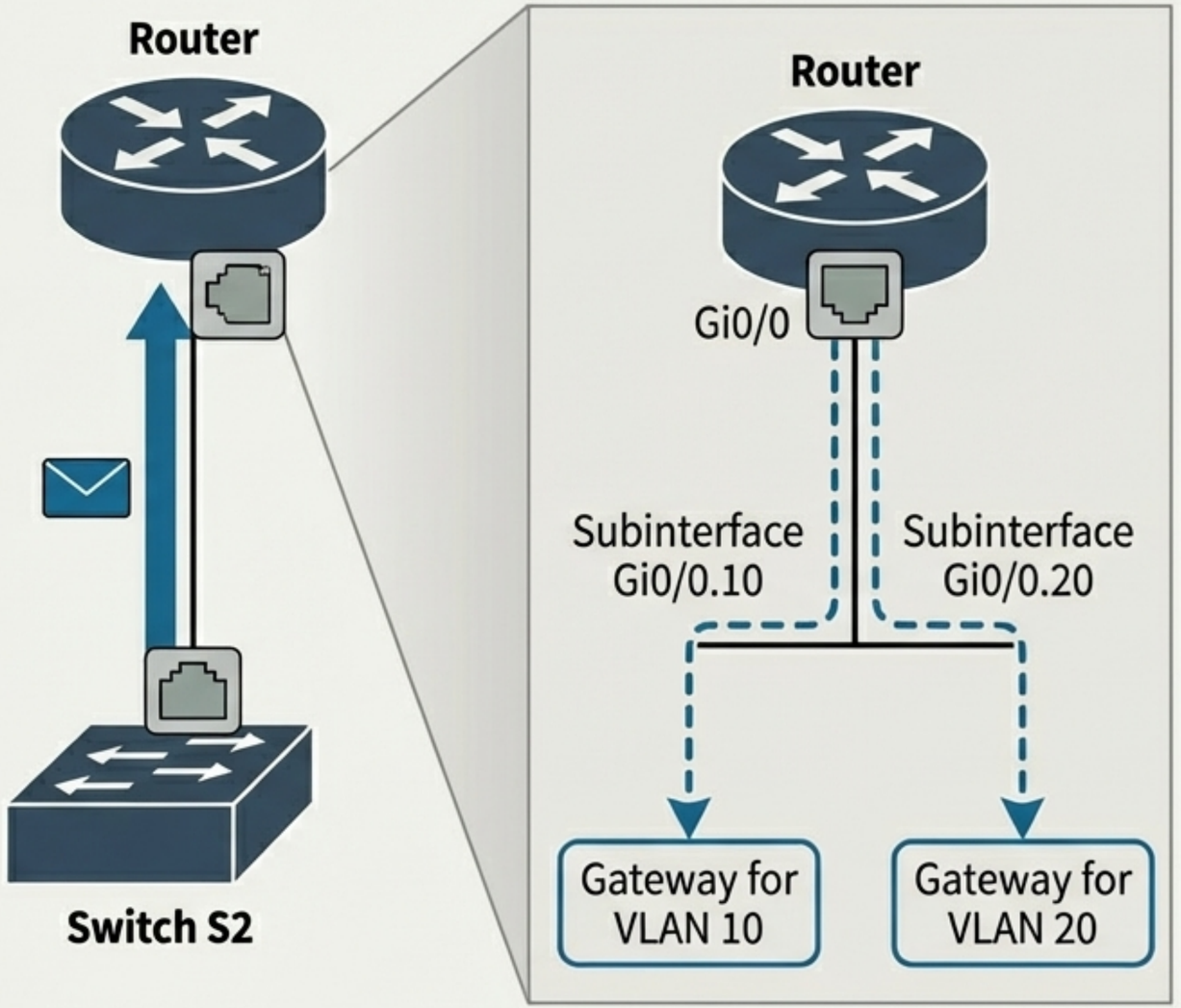
The Great Divide: Routing Between VLANs

The Rule

Devices on different VLANs are in different subnets and broadcast domains. They cannot communicate without a Layer 3 device (a router or multilayer switch).

Solution 1: Router on a Stick (ROAS)

- A single physical router interface is connected to a trunk port on a switch.
- The router interface is configured with logical “subinterfaces”—one for each VLAN it needs to route.
- Each subinterface is assigned an IP address in its VLAN’s subnet and becomes the default gateway for hosts in that VLAN.



Configuring a ‘Router on a Stick’

Two Sides of the Connection

Switch Configuration

The port connected to the router must be a **trunk**.

```
// On the Switch
Switch(config)# interface fa0/8
Switch(config-if)# switchport mode trunk
```

Router Configuration

Create a subinterface for each VLAN, specify the encapsulation, and assign an IP address.

```
// On the Router
Router(config)# interface GigabitEthernet0/0.10
// Tell the subinterface to listen for 802.1Q tags for VLAN 10
Router(config-subif)# encapsulation dot1q 10
// Assign an IP address to act as the gateway for VLAN 10
Router(config-subif)# ip address 192.168.10.1 255.255.255.0

Router(config)# interface GigabitEthernet0/0.20
Router(config-subif)# encapsulation dot1q 20
Router(config-subif)# ip address 192.168.20.1 255.255.255.0
```

Key Detail

The number after `dot1q` (`10` and `20` in this example) must match the VLAN ID. The subinterface number (`.10` and `.20`) is locally significant but is best practice to match to the VLAN ID for clarity.

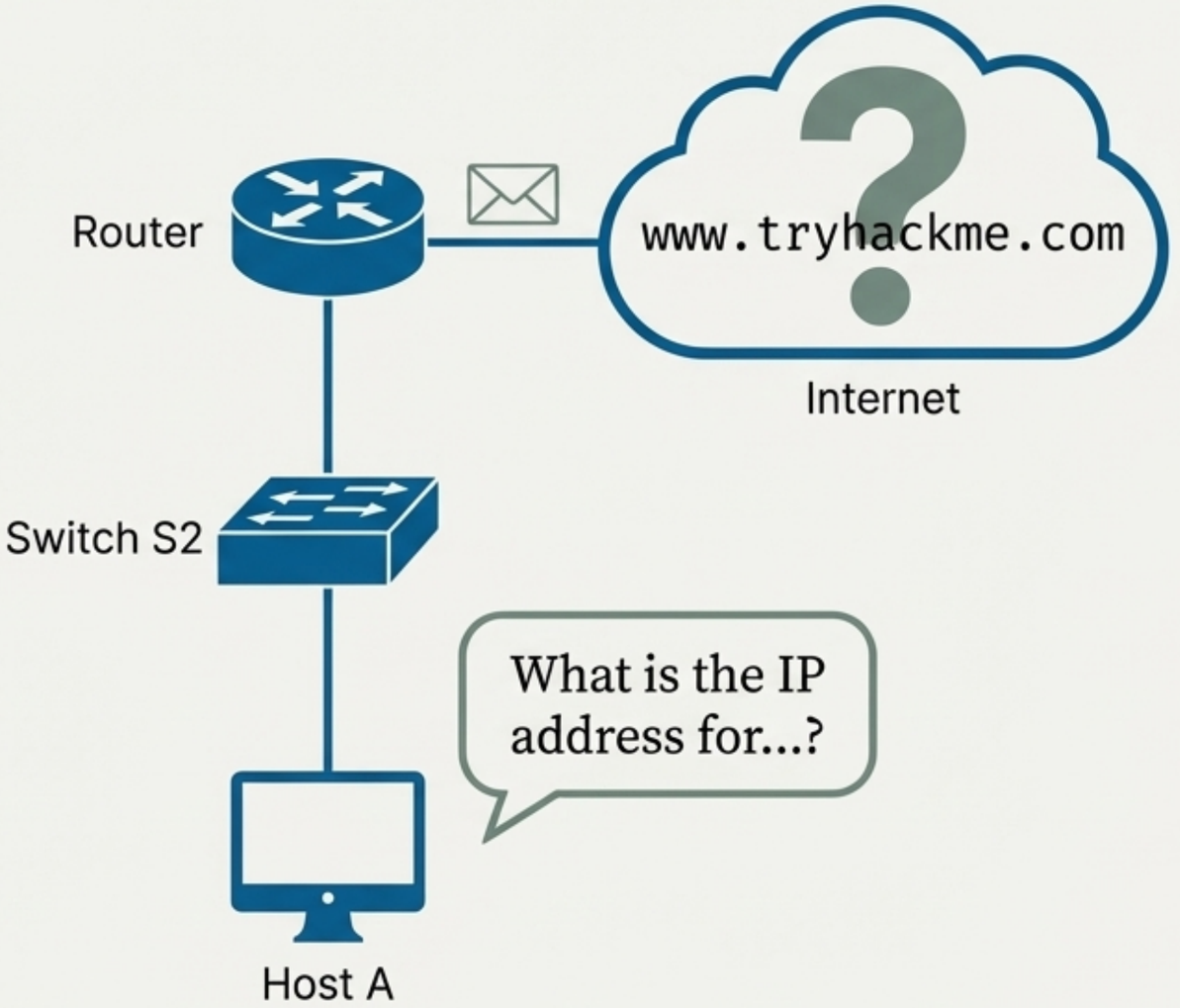
What’s in a Name? Resolving the Resolving the Destination

The Final Hurdle

Host A started with a domain name like `www.tryhackme.com`. Routers and switches work with IP addresses. The host must translate the name to an address before the packet can be sent to the internet.

The Solution: The Domain Name System (DNS)

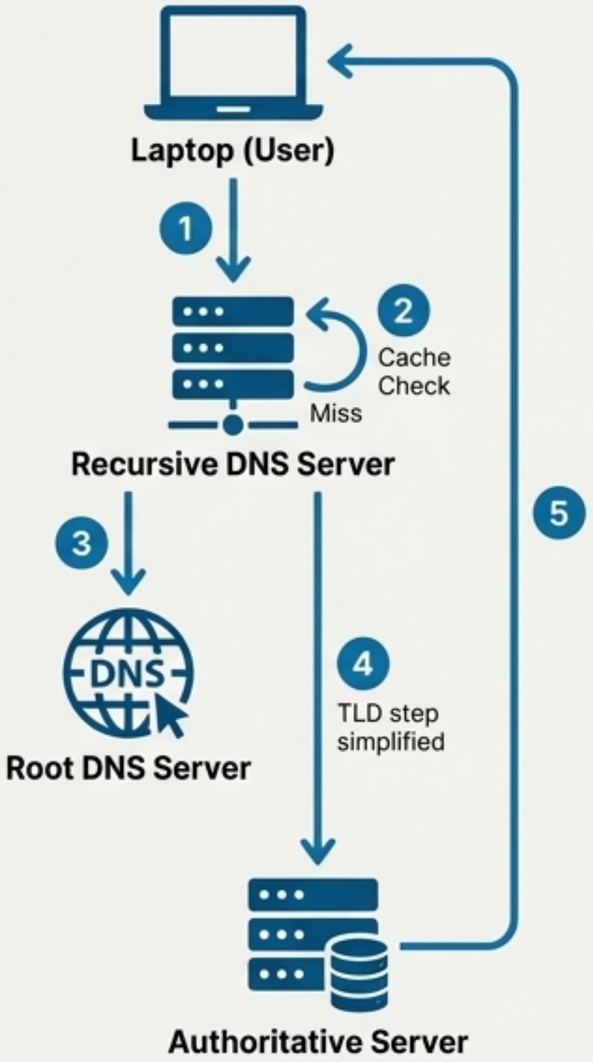
A hierarchical and distributed naming system that translates human-readable domain names into machine-readable IP addresses.



The DNS Query: A Journey of its Own

The 5-Step Process:

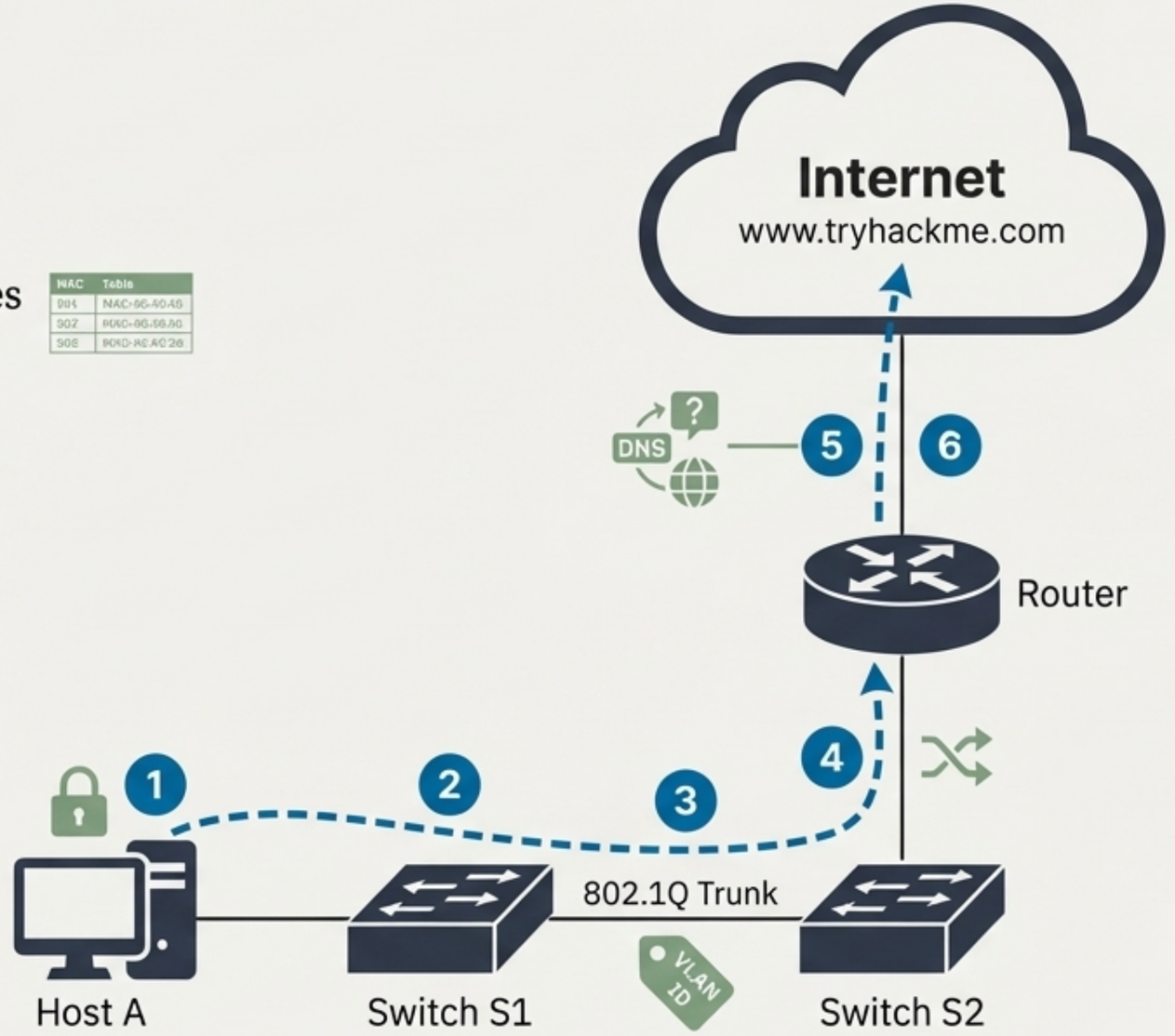
1. **Local Cache & Recursive DNS Server:** Your computer first checks its local cache. If the name isn't there, it asks its configured Recursive DNS Server (usually provided by your ISP).
2. **Recursive Server Cache:** The recursive server checks its own cache for recently looked-up names. If found, the process ends here. If not, the journey continues.
3. **Root DNS Server:** The recursive server asks one of the 13 root server clusters. The root server doesn't know the full answer but knows who handles the Top-Level Domain (TLD). For www.tryhackme.com, it redirects the query to the .com TLD servers.
4. **TLD Server:** The recursive server now asks the .com TLD server. The TLD server knows which Authoritative Nameserver is responsible for the tryhackme.com domain (e.g., `kip.ns.cloudflare.com`).
5. **Authoritative Server & Caching:** The recursive server asks the authoritative server, which holds the actual DNS record and provides the final IP address. This answer is sent back to the recursive server, which caches it (honoring its TTL, or Time To Live) and finally returns it to your computer.



The Full Journey, Visualized

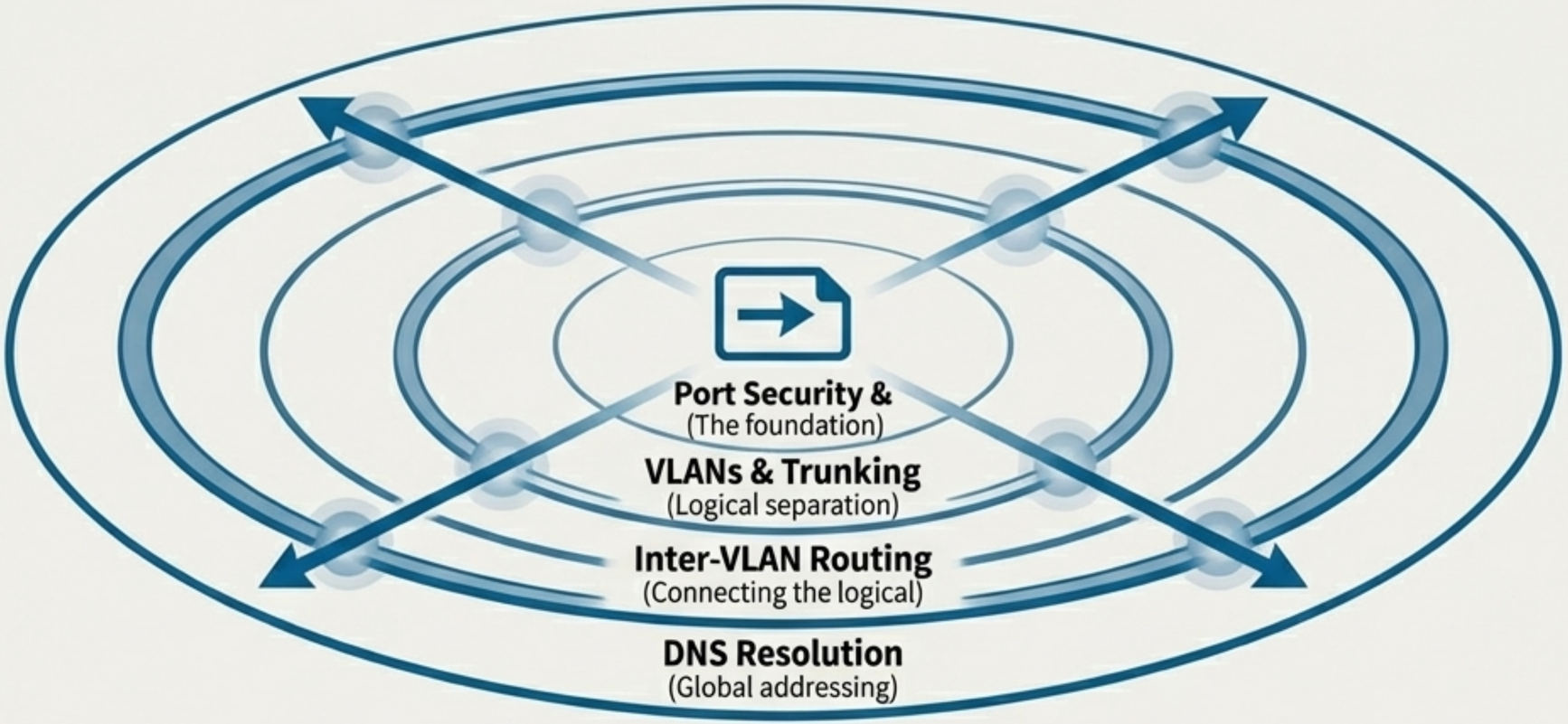
Path Trace

1. **Departure:** Packet leaves Host A, passing a **Port Security** check on S1.
2. **Local Switching:** S1 learns Host A's MAC and uses its **MAC Address Table** to forward the frame.
3. **Crossing VLANs:** The frame is tagged with its **VLAN ID** and sent across the **802.1Q Trunk Link** from S1 to S2.
4. **Routing:** S2 forwards the frame to the router, which performs **Inter-VLAN Routing** to move the packet between subnets.
5. **Name Resolution:** Before final departure, the host performs a **DNS lookup** to get the destination IP address.
6. **To the Internet:** With the correct destination IP, the router forwards the packet to the internet.



The Unsung Choreography of the Network

The journey of a single data packet is not a simple A-to-B trip. It's a complex and elegant sequence, relying on multiple technologies working in perfect concert.



From the physical security of a port to the logical separation of VLANs and the global directory of DNS, these foundational protocols are the unsung heroes of digital communication. They form an interconnected system that transforms a simple request into a seamless connection, all in a fraction of a second.