TCP/IP Fundamentals

Cloud Security

CSE4059

Recap

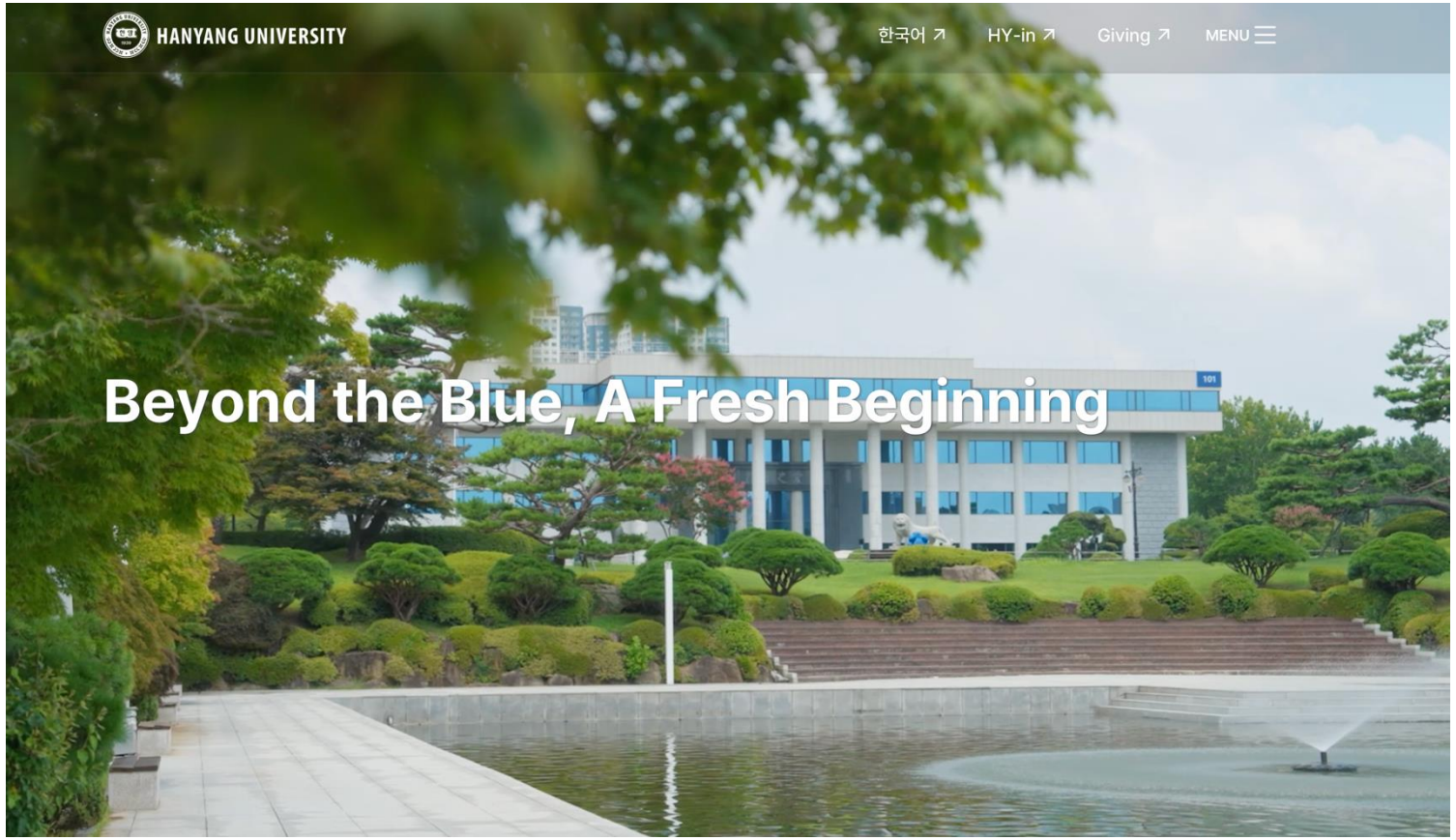
- Five cloud characteristics, each break an old assumption
  - Self-service, broad access, pooling, elasticity, metering
- IaaS / PaaS / SaaS
  - Divide who operates the stack, and separately who is RESPONSIBLE for securing it
- Shared responsibility
  - The provider secures the infrastructure, clients secure what they put on it
  - Neither side substitutes for the other; a provider can hold up its end and an incident still happen

An Example

An Example

- Two students open ***www.hanyang.ac.kr*** in the same minute, on different networks

- Student A
  - The page loads normally



An Example

- Two students open ***www.hanyang.ac.kr*** in the same minute, on different networks

- Student A
  - The page loads normally
- Student B
  - The browser spins.. spins.. Eventually it times out
  - Gets no error message. Nothing says "refused" or "not found"
  - It just… hangs

- A REFUSAL and a HANG are different symptoms, with different causes

A Map: Why Networking Is Built in Layers

- Getting a web page from a server to your screen involves many separate problems
- Networking uses "Layers": each provides a service to the layer above

Application	What the two programs are actually saying to each other (ex. HTTP – "send me this page")
Transport	Getting a stream of data between two programs, reliably or not (TCP, UDP)
Internet	Getting a packet from one machine to another, possibly across the world (IP)
Network Access	Delivering data across the local network (Ethernet, Wi-Fi)

Why Layering Is Worth the Trouble

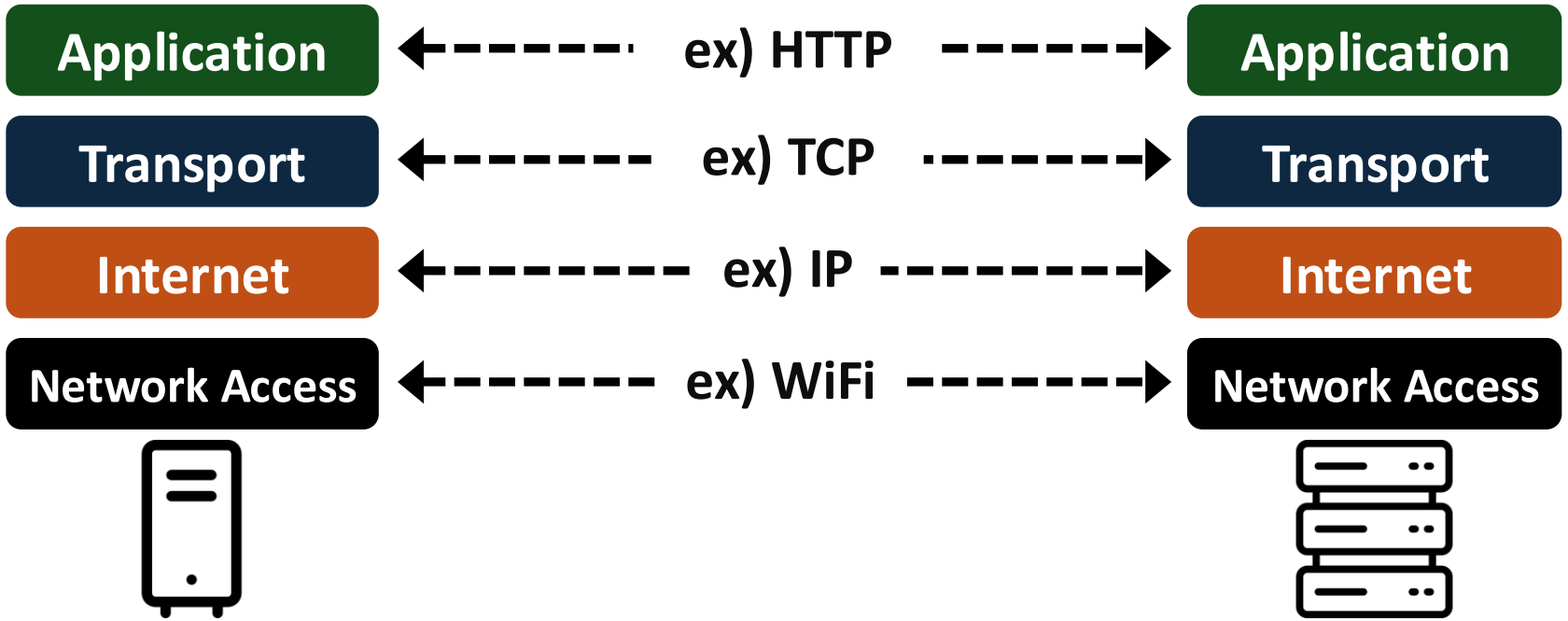
- Each layer only has to solve its own problem, and can assume the layer below works
  - TCP (transport layer) does not know or care whether you are on Wi-Fi, Ethernet, or a mobile network
  - A web browser (application layer) does not know or care whether TCP had to retransmit anything

- We can replace one layer without rewriting the others
  - That is how the internet survived 40 years of change

- The security consequence?
  - A control at one layer may not have access to information from another layer
  - A basic network-layer or transport-layer control can make decisions using packet/connection metadata, but not necessarily application semantics

What is a Protocol?

- **Protocol**
  - A set of rules that two communicating parties agree to follow
  - Define what messages can be sent, how they are formatted, and what they mean
- Different layers use different protocols for different purposes
  - HTTP: rules for requests and responses between web clients and servers



Which Machine? Addressing

IP Address

- A logical address assigned to an interface in an IP network
  - Used by the Internet layer to address and route IP packets across networks
- An address is not a host identity
  - An IP address does not directly identify a person, application, or physical machine
  - A host with multiple interfaces can have multiple IP addresses.
- What does an IP Address tell us?
  - Tell us: where an IP packet is addressed/routed
  - Does not tell us: who is using it, what service is running there, or whether it is authentic

How a Packet Actually Gets There

- A packet (chunks of data) is forwarded from router to router
  - Each forwarding step is a "hop"
  - Each router examines the destination IP address and selects a next hop
- Forwarding is based on local decisions
  - A router does not need the complete end-to-end path
  - Packets may traverse networks operated by different organizations
- Security aspect
  - Packets may traverse networks that we do not control or trust
  - We cannot assume that the network path itself is trusted

An Address Is a Claim

- Every packet carries a source IP address
  - Saying where it came from

- But the IP protocol does not authenticate the source address
- An attacker can forge a source address
  - Unless the network filters spoofed packets

- From a security perspective
  - "Allow traffic from this IP address" does not prove who actually sent the packet
  - This is one reason higher-layer authentication is needed

Which Service? Ports and Protocols

One Machine, Many Services

- A single server can run multiple network services
  - Web, email, and SSH
  - They may share the same IP address

- **Port number**
  - Distinguish transport-layer endpoints
  - TCP and UDP carry source and destination port numbers
  - The receiving machine (Operating System) uses the port number to deliver incoming traffic to the appropriate application

Port Ranges

- Port numbers can be divided into three ranges

Range	Numbers	What lives there
System ports	0 – 1023	Well-known services. On Unix-like systems, listening here usually needs elevated privilege
User ports	1024 – 49151	Port numbers used for specific applications or services (ex. MySQL – 3306)
Dynamic / Ephemeral	49152 – 65535	Never assigned to anyone. Reserved for temporary local use

Well-known Ports

Port	Service	Purpose
22	SSH (Secure Shell)	Remote administration
53	DNS (Domain Name System)	Name-to-IP resolution
80	HTTP (HyperText Transfer Protocol)	Web communication protocol; Unencrypted
443	HTTPS (HTTP over Secure)	Web communication protocol; Encrypted
3389	RDP (Remote Desktop Protocol)	Remote desktop administration

- These port numbers are conventions
  - The protocol does not enforce which application runs on a port

TCP and UDP

Transmission Control Protocol (TCP)

- Establishes a connection before sending data
- Delivers bytes in order
- Detects loss and retransmits
- Both sides know the conversation exists
- Cost? setup time, memory to track it
- Used when correctness matters more than speed

User Datagram Protocol (UDP)

- No connection setup; just send the packet
- No ordering guarantee
- No retransmission; lost is lost, silently
- Neither side tracks a conversation
- Cost? essentially nothing.
- Used when one small exchange is the whole interaction

Who Starts? The Direction of Initiation

- In typical client-server communication, one side INITIATES and the other RESPONDS
  - The client initiates: it knows the server's address and port in advance
  - The server listens: it does not know who will call, or from which port

- This asymmetry is not a detail; it is the basis of security design:
  - "Allow connections OUT, but not new connections IN" is a coherent and common policy

- Ask of any traffic: "Who started this conversation?"
  - The answer usually decides whether it should be allowed

The Five-Tuple

- One flow is identified by 5 values:
  1. **Source IP address**: which interface it came from
  2. **Source port**: which program there
  3. **Destination IP address**: which interface it is going to
  4. **Destination port**: which program there
  5. **Protocol**: TCP or UDP

→ **5-tuple**

The Five-Tuple (cont'd)

- Network filters decide on these 5 values, plus direction
  - Not on who the user is,
  - Not on what the request says,

- For example,
  - A filter can see you connected to port 443
    - But this is a number, not a service
  - Allowing port 443 allows whatever is listening on 443
    - It cannot see whether the request you sent was an ordinary page load or an attack

Where Do These Values Come From?

- Packet header
  - Control information carried along with the data being sent
  - Each protocol adds the information it needs to do its job

- **IP header**
  - Source IP address
  - Destination IP address
  - Protocol
  - TTL
  - …
- **TCP header**
  - Source port
  - Destination port
  - Sequence number
  - Acknowledgment number
  - Flags: SYN, ACK, FIN, RST
  - …
- **UDP header**
  - Source port
  - Destination port
  - …

TCP Connection Setup: The Three-Way Handshake

TCP 3-Way Handshake

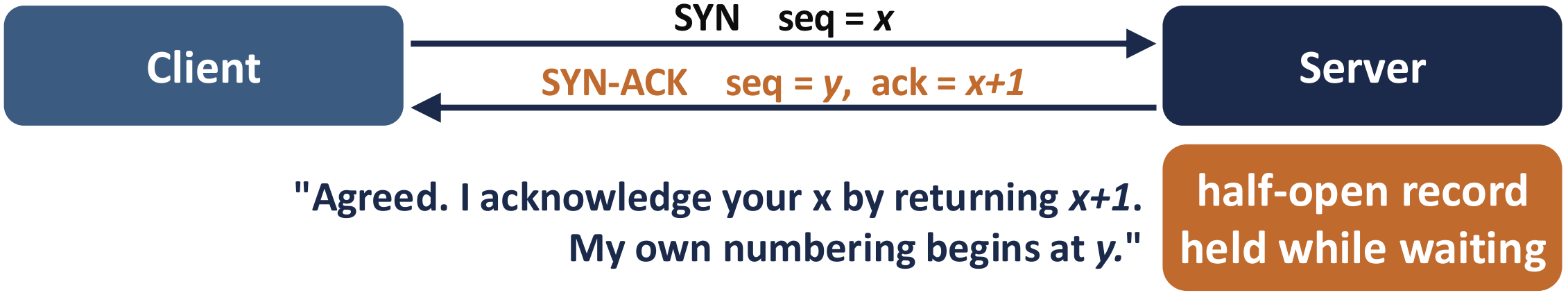
Step 1: SYN (Synchronize)

"I would like to start a conversation. My numbering begins at *x*."

- Nothing is established yet
  - This is a request, not a connection
- The sequence number *x* is chosen by the client
  - What makes this conversation distinguishable

TCP 3-Way Handshake

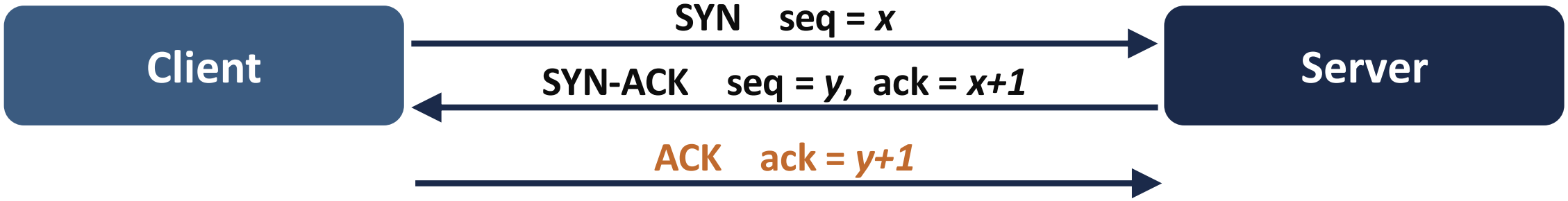
Step 2: SYN-ACK



- Two flags at once
  - SYN: the server starting its own direction
  - ACK: confirming the client's
- The server holds a record and waits for the third message

TCP 3-Way Handshake

Step 3: ACK



"I received your *y*. We can start."

- The TCP connection is now established
- Application data can now be exchanged
- Both sides have exchanged and acknowledged their initial sequence numbers

TCP Flags

- TCP specification defines 8 control bits
- 4 of them

Flag	Meaning	Where you will meet it
SYN	Start a connection	During TCP connection establishment
ACK	Acknowledge what has been received	On almost every TCP packet after the initial SYN
FIN	I am finished sending; close cleanly	Normal connection shutdown
RST	Stop immediately; something is wrong	A refusal or abrupt termination

Closing a Connection

- A graceful close
  - Each side sends FIN and has it acknowledged
- An abrupt close
  - RST – terminate the connection immediately
- But there is a third case: *nothing at all*
  - Ex) a machine crashes, a cable is unplugged, a device silently drops the traffic
  - The other side may continue holding a record of the connection
  - The record remains until the failure is detected or some timeout occurs

Connection State

What "State" Actually Is

- For TCP, state is a record of an ongoing connection
  - Which two addresses and which two ports are communicating
  - What stage the connection has reached (ex. half-open, established, closing)
  - Sequence-number information exchanged during the handshake
- It is essentially a row in a table, held in memory
- With that record, a device sees a new packet and says:
  - "*This belongs to a connection I know*"
- Without it, every packet arrives without that context

Where Is Connection State Kept?

- The client and the server must maintain connection state to run TCP
  - They must track the conversation to function
- The device in the middle may keep its own state for the traffic it observes
- Keeping state costs memory and becomes more complex at scale
  - Keeping no state is simpler and uses less memory, but the device cannot recognize a reply as belonging to a connection it previously allowed

Stateful and Stateless

Stateful

- Remembers connections it has seen
- Recognizes a reply as part of a known connection
- You write one rule; the return traffic follows
- Costs memory, and complexity at scale

Stateless

- Examines each packet entirely on its own
- A reply is just another packet arriving from somewhere
- You must write rules for both directions
- Cheap, simple, predictable
- Cannot recognize a legitimate reply based on previous traffic

Stateful Means Secure?

- **No!**
  - "Stateful" describes memory, not security by itself
  - A stateful device with permissive rules can still allow too much traffic
  - A stateless device with restrictive rules can still be very restrictive
- These are different properties
  - State: does the device remember connections?
  - Strength: how restrictive are its rules?

The Return Path

Ephemeral Ports

- When your browser connects to a web server on port 443
- What is your source port?

- The side that initiates picks a temporary source port for the connection
  - This is an ephemeral port
  - It is used temporarily for the connection and can be reused later

- Why do we need it?
  - A client may have many connections open at the same time
  - The connections must be distinguished from one another

- The server sends the reply back to the client's source port

Ephemeral Ports (cont'd)

- The port registry reserves 49152-65535
  - For dynamic use and does not assign these ports to specific services
- Operating systems may use different ephemeral port ranges in practice
- There is no single ephemeral range that applies to every client
- The ephemeral range is determined by the client system
- With stateless filtering, return traffic must be allowed to the client's ephemeral port
  - This may require allowing a broad port range

Request and Reply Are Two Different Packets

A device that KEEPS state

"I allowed a request to 443 from port 52413. This is its reply."

Lets it through

A device that keeps NO state

"A packet from port 443, heading to port 52413. Who is this?"

Drops it

- The ports and the direction have both swapped
- At the packet level, the reply looks like a completely different packet

Revisit: The Starting Example

- Student B's connection attempt reached the server
  - The server sent traffic back
  - The return traffic hit a device that kept no connection state
  - No rule permitted the return traffic, so it was dropped
- Student B received no response
  - The browser waited and eventually timed out
- Key difference
  - An explicit refusal → quick error
  - A silent drop → no response, then timeout

What an Attacker Can See, and What They Can Forge

Metadata

- A packet passes through many networks between you and the server
- Devices on the path can observe
  - Source & destination addresses: which network endpoints are communicating
  - Source & destination ports: what kind of service
  - The transport protocol
  - Packet sizes and timing

- **Metadata**
  - Information about the communication rather than its contents
  - With encryption techniques, the contents are protected, but much of this metadata remains visible
  - It can still reveal when communication occurs, between which IP addresses, and approximately how much data is exchanged

Forgeable: The Source Address

- IP itself does not authenticate the source address
- An attacker may forge the source IP address of a packet

- "Allow traffic from this address" is not strong authentication

- A forged source address can cause a reply to be sent to someone else
  - This is the basis of reflection attacks

- For logging, a source IP address is useful evidence, but not proof of identity

TCP Authenticates Nothing

- What TCP provides
  - Reliable, ordered delivery between two endpoints
- What TCP does NOT provide
  - Proof of who is at the other end
  - Confidentiality of the transmitted data
  - Cryptographic protection against tampering
- These are not problems TCP was designed to solve
  - They must be provided by higher-layer security protocols (e.g., Transport Layer Security)

End to End: Opening a Page

- You type *www.hanyang.ac.kr* and press Enter.
- In order:
  1. Your machine picks an ephemeral source port
  2. Three-way handshake with the server: SYN, SYN-ACK, ACK
  3. Your browser sends its request; the server sends the response
  4. The TCP connection is eventually closed

Summary

- An address identifies an interface; a port identifies an application endpoint; a protocol defines how communication is handled
- The 5-tuple, together with direction, is what a basic packet filter uses to make decisions
- The 3-way handshake establishes a TCP connection before normal data transfer
- State is the information a device keeps about an existing connection
  - Whether a device keeps state determines how it can recognize return traffic
- An explicit refusal produces a quick error; a silent drop leads to a timeout
- A source IP address is not proof of identity