

# Processes Communication Example

 Uploaded image

give an understandable example of this

Sure! Let's walk through a simple example based on the concepts in your slide about **processes communicating**:

---

## Real-Life Analogy Example:

### Scenario: Watching a Movie Online

- You (the **client process**) want to watch a movie.
  - You use your web browser to visit a movie streaming website (like Netflix).
  - The **Netflix server** (the **server process**) is waiting for users to request movies.
- 

## What Happens Technically:

1. Your web browser is a **client process** — it **initiates communication** by sending a request to the Netflix server asking for a movie.
  2. The Netflix server is a **server process** — it **waits to be contacted** by users like you.
  3. Once contacted, it responds by sending the requested movie data back to your browser.
  4. You receive and watch the movie — this is an example of **processes exchanging messages** across different hosts (your device and Netflix's server).
- 

## Key Concepts in Action:

- **Processes in different hosts** (your computer and Netflix server) communicate by **exchanging messages**.

- Your browser is the client, Netflix server is the server.
  - This is an example of **client-server communication**.
- 

Let me know if you'd like an example using a peer-to-peer (P2P) architecture instead!

no, the process communication happens within a device.  
Give me an example of that

Got it! You're asking for an example of **inter-process communication (IPC)** that happens **within the same device (host)**. Let's look at an easy-to-understand example:

---



## Example: Copy-Pasting Text Between Applications

### Scenario:

You copy text from a **web browser** and paste it into a **word processor** (like Microsoft Word or Google Docs).

---

### What's Happening Behind the Scenes:

- The **web browser** is one **process**.
  - The **word processor** is another **process**.
  - When you copy something, the browser places the text into the **clipboard** — a shared area of memory.
  - When you paste, the word processor reads the data from the clipboard.
- 

### Why This is Inter-Process Communication:

- Both programs (processes) are **running on the same host** (your computer).
- They **don't talk directly**, but use the **Operating System's IPC mechanism** (in this case, the clipboard) to **exchange data**.

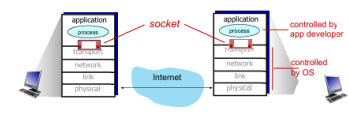
## Other Common IPC Examples on One Device:

- A music player sending track info to a notification system.
- A system monitor app requesting CPU data from the OS.
- A background updater checking with a main app for permission to install updates.

Let me know if you'd like code examples or technical mechanisms like pipes, sockets, or shared memory explained!

### Sockets

- process sends/receives messages to/from its **socket**
- socket analogous to door
  - sending process shoves message out door
  - sending process relies on transport infrastructure on other side of door to deliver message to socket at receiving process
- two sockets involved: one on each side



### Addressing processes

- to receive messages, process must have **identifier**
- host device has unique 32-bit IP address
- **Q:** does IP address of host on which process runs suffice for identifying the process?
  - **A:** no, *many* processes can be running on same host
- **identifier** includes both **IP address** and **port numbers** associated with process on host.
- example port numbers:
  - HTTP server: 80
  - mail server: 25
- to send HTTP message to `gaia.cs.umass.edu` web server:
  - IP address: 128.119.245.12
  - port number: 80
- more shortly...

explain these with details

Great! Let's break this down into two main parts from your slide: **Sockets** and **Addressing Processes** — both are key to enabling communication between processes across a network (like the Internet).

## 1. Sockets

A **socket** is like a **doorway** through which data enters or leaves a process. It is the interface between the application layer and the transport layer.

### Key Concepts:

- Process sends/receives messages through its socket.

- Think of it as a **communication endpoint** — each side of a connection has one.
  - **Two sockets** are involved in communication: one on the sender's side and one on the receiver's side.
- 

## **Analogy: Sending a Letter**

Imagine sending a letter:

- You write the letter (data) and put it in your **mailbox** (socket).
- The **postal system (network infrastructure)** delivers it.
- The recipient receives the letter at their **mailbox** (socket) on their house.

In networking:

- The **sending process** puts the message in the socket.
  - The **OS and network layers** take care of delivering the message.
  - The **receiving process** picks it up from its own socket.
- 

## **Technical Details:**

- **Controlled by App Developer:** What data goes into the socket.
  - **Controlled by OS:** Transport, network, link, and physical layers that move data across networks.
- 

## **2. Addressing Processes**

Now, once you have sockets, how do you know where to send your message? That's where **addressing** comes in.

---

## **Key Concepts:**

To **receive** a message, a **process** needs a unique **identifier**, which includes:

1. **IP address** – identifies the host machine.
  2. **Port number** – identifies the specific process (or service) on that host.
- 

## ? Why Not Just IP Address?

Because:

- One machine (host) can run **many processes** at the same time.
  - So, **just an IP address is not enough** to uniquely identify where to deliver the message.
- 

## 🧩 Example:

To send an HTTP message to a web server like:

`gaia.cs.umass.edu`

You need:

- **IP address:** `128.119.245.12` (host location)
- **Port number:** `80` (HTTP service)

So the full identifier = `128.119.245.12:80`

Other common port numbers:

- HTTP: `80`
  - HTTPS: `443`
  - Mail: `25`
- 

## 💡 Summary:

Term	Meaning
Socket	A communication endpoint in a process

---

Term	Meaning
IP Address	Identifies the device on a network
Port Number	Identifies the specific process/service on that device
Full Address	IP Address + Port Number = Identifies the target process

Would you like a simple Python or C example showing how sockets are used?

### An application-layer protocol defines:

- **types of messages exchanged**,
    - e.g., request, response
  - **message syntax**:
    - what fields in messages & how fields are delineated
  - **message semantics**
    - meaning of information in fields
  - **rules** for when and how processes send & respond to messages
- open protocols**:
    - defined in RFCs, everyone has access to protocol definition
    - allows for interoperability
    - e.g., HTTP, SMTP
  - proprietary protocols**:
    - e.g., Skype, Zoom

Application 1

explain in details

This slide explains what an **application-layer protocol** is and what it defines. The **application layer** is the **top layer of the Internet protocol stack**, where **actual user applications** (like **web browsers, email clients, video calls, etc.**) **interact with the network**.

Let's break it down:

## ◆ What an Application-Layer Protocol Defines:

An **application-layer protocol** governs how two applications (processes) communicate over a network. It defines:

### 1. Types of Messages Exchanged

- What kinds of messages are allowed.
  - Example:
    - HTTP uses:
      - GET (to request a webpage),
      - POST (to submit form data),
      - RESPONSE (sent back by the server).
    - Email protocols use SEND , RECEIVE , FORWARD , etc.
- 

## 2. Message Syntax

- Structure or format of the messages.
- Defines:
  - What fields are in a message.
  - How fields are **separated** (e.g., by colons, new lines, etc.).
- Example in HTTP:

```
vbnet
```

```
GET /index.html HTTP/1.1  
Host: www.example.com
```

## 3. Message Semantics

- Meaning of each field in the message.
  - Tells the receiver what to **do with the data**.
  - Example:
    - In Host: www.example.com , the server knows which domain you are trying to reach.
-

## 4. Rules for Sending/Responding

- Defines when a process should:
    - Send a request.
    - Send a reply.
    - Close the connection.
  - Example:
    - In HTTP, the client sends a request, and the server **must** respond.
- 

## Types of Protocols:

---

### Open Protocols

- **Open to the public:** Defined in **RFCs** (Request for Comments).
  - Anyone can implement them.
  - Promote **interoperability** (different systems can talk to each other).
  - Examples:
    - **HTTP:** Used for web communication.
    - **SMTP:** Used for sending emails.
- 

### Proprietary Protocols

- Owned and **controlled by a company**.
- Often **not publicly documented**.
- Used only in **specific applications**.
- Examples:
  - **Skype** (Microsoft).
  - **Zoom** (Zoom Inc.).



These are optimized for their own apps and may offer unique features, but lack interoperability.

---

## Summary

Term	Meaning
Types of messages	What kinds of interactions are allowed (request, response)
Syntax	Structure of the message (fields, formatting)
Semantics	What each part of the message means
Rules	When and how messages should be sent/received
Open protocol	Publicly available and standardized
Proprietary protocol	Private and controlled by companies

Let me know if you want an example with HTTP message breakdown or comparison with another protocol!