Berkeley Sockets API in Python

HyperionDev

Lecture – Housekeeping

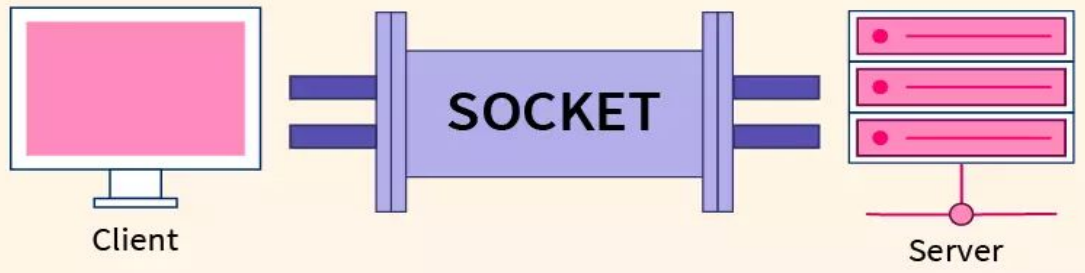
- ❏ The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment - please engage accordingly.
- ❏ No question is daft or silly - **ask them!**
- ❏ There are Q/A sessions midway and at the end of the session, should you wish to ask any follow-up questions.
- ❏ You can also submit questions here: hyperiondev.com/sbc4-cs-questions
- ❏ For all non-academic questions, please submit a query: www.hyperiondev.com/support
- ❏ Report a safeguarding incident: hyperiondev.com/safeguardreporting
- ❏ We would love your feedback on lectures: https://hyperionde.wufoo.com/forms/zsgv4m40ui4i0g/

Previously:

- Recapped on Security Management
- Covered Risk Management

Objectives

- Learn what Berkeley Sockets API is and why it is important



https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.scaler.com%2Ftopics%2Fcomputer-network%2Fsocket-programming%2F&psig=AOvVaw0bp9Irx0sSJD5AzKyTPEI2&ust=1684849926760000&source=images&cd=vfe&ved=0CBEQjRxqFwoTCIjiirSKif8CFQAAAAAdAAAAABAD

What is BSD Socket API?

- The Berkeley Sockets API, also referred to as the BSD Socket API, is an operating system programming interface for network communication. Applications can build, send, receive, and manage network connections and data transfers using the set of functions and data structures it offers. The University of California, Berkeley is where the API was created, thus the name.

- The Berkeley Sockets API is now widely used and accepted as the norm for application-to-application network communication. Many networking protocols and services that are necessary for contemporary computing and internet connectivity are built on its foundation. The Berkeley Sockets API is used by countless programs, including web browsers, email clients, instant messaging services, and online games, to create connections, exchange data, and enable communication over networks.

Sockets in Python

- Python, a flexible and well-liked programming language, offers a potent library called socket that enables programmers to use the Berkeley Sockets API functionality in their Python programs. The socket library acts as a user-friendly and high-level interface for network programming by acting as a Python wrapper around the underlying system-level socket functions.

- Developers can establish connections, send and receive data, and handle network communication in their Python applications by using the socket library to create, configure, and manage network sockets. The difficulties of low-level socket programming are abstracted, and a clear and simple API is provided for typical networking tasks. Developers can take advantage of the Berkeley Sockets API features while avoiding having to deal with the complexities of socket programming at the system level by using the socket library.

...,But wait, there’s more!

Because of the API's ease of use, dependability, and compatibility with a variety of operating systems, it is widely used. Regardless of the underlying platform or programming language, its design enables developers to create network applications that can communicate with one another without interruption. The Berkeley Sockets API enables interoperability and makes it easier to create networked applications that can effectively communicate over a variety of network protocols, including TCP/IP and UDP, by offering a common set of functions and data structures.

Socket function & methods

- socket() Create a new communications endpoint

- .bind() Attach a local Address to a socket
- .listen() Broadcast willingness to accept connections
- .accept() Block caller until a connection request arrives
- .connect() Actively attempt to establish a connection
- .send() Send data over the connection
- .recv() Receive some data over the connection
- .close() Close the connection

IP + Port = Socket

IP addresses and port numbers are the two main elements of ***socket addressing***.

Devices on a network are given IP addresses, which are distinctive identifiers. They enable device-to-device communication by indicating the source and destination of data packets.

IP addresses can be displayed in a variety of ways, including IPv4 and IPv6 formats (for example, 192.168.0.1 and 2001:0db8:85a3:0000:0000:8a2e:0370:7334, respectively). The most popular version is IPv4.

To distinguish between various services running on a device, port numbers are used. They ensure that the data gets to the intended application while allowing multiple applications to coexist on the same device.

IP address: 192.168.0.1

Port number: 8080 (commonly used for web applications)

Socket address: 192.168.0.1:8080 (combining IP address and port number)

Hyperiondev

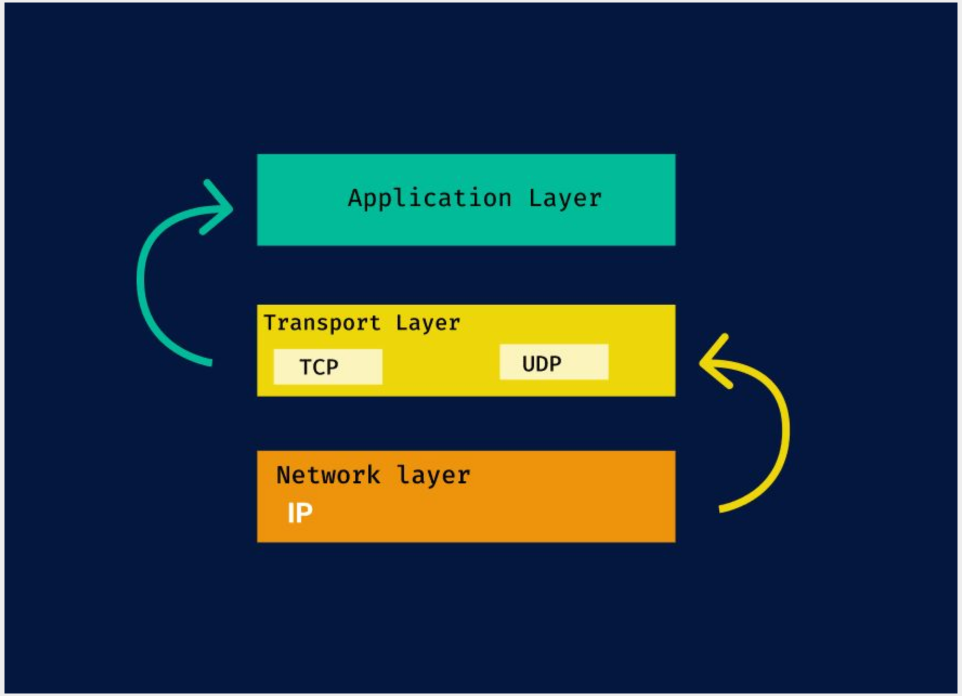
Types of Sockets

- TCP/IP sockets and UDP sockets are two examples of the various communication protocols that can be used with sockets.

- TCP/IP sockets offer dependable, connection-based communication. They create a connection between the client and server, guaranteeing the delivery of accurate and sequential data. For applications like file transfer, email, and web browsing where data integrity and reliability are essential, TCP/IP sockets are frequently used.

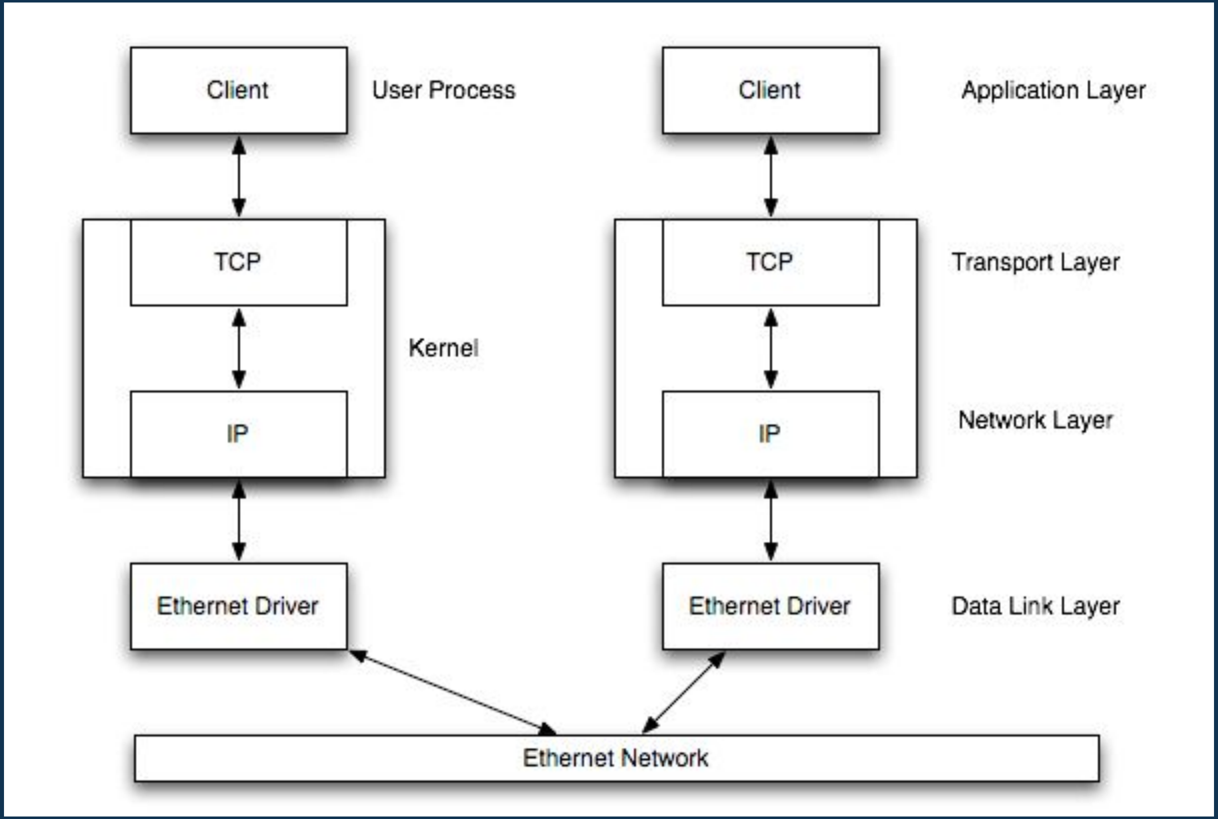
- The UDP(User Datagram Protocol) socket provides connectionless, lightweight communication. They don't create a lasting connection, and they don't ensure data packet delivery or order. Applications like real-time streaming, online gaming, and DNS resolution typically use UDP sockets because speed and efficiency are more important than reliability in these situations.

Sockets (OSI)



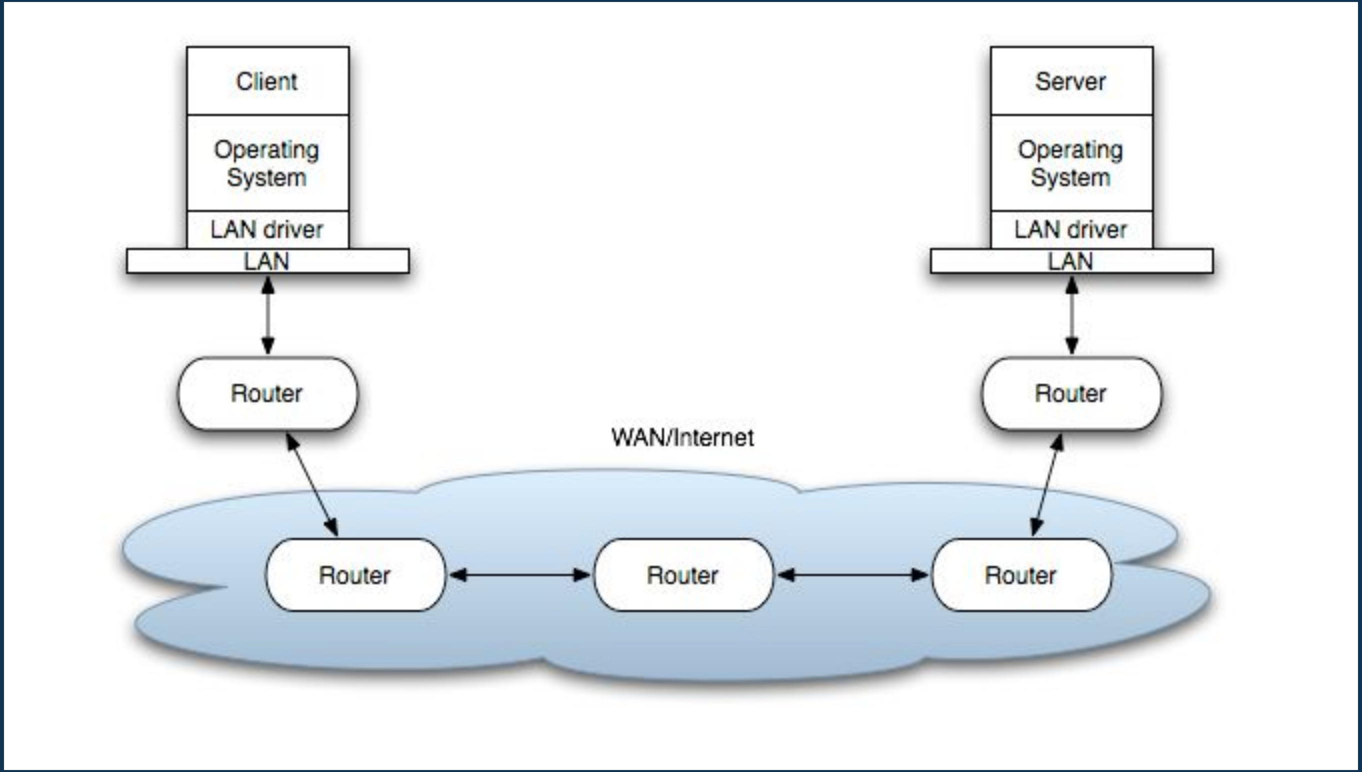
https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Fstart.cybervista.net%2Fhubfs%2FHacker_Chart.png&f=1&nofb=1&ipt=c4a56f8bb2b92f8d78c776f833f92d413fea7123f0d4b8d387a81b4fc844cc50&ipo=images

LAN



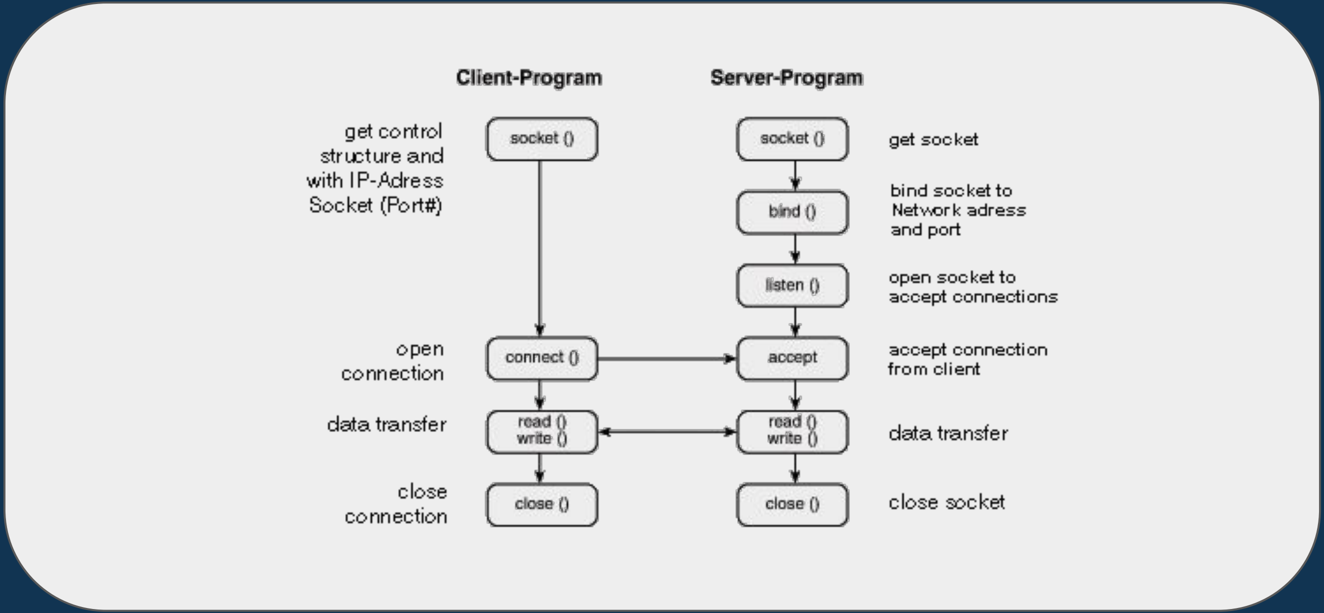
https://www.cs.dartmouth.edu/~campbell/cs60/socketprogramming.html

WAN



https://www.cs.dartmouth.edu/~campbell/cs60/socketprogramming.html

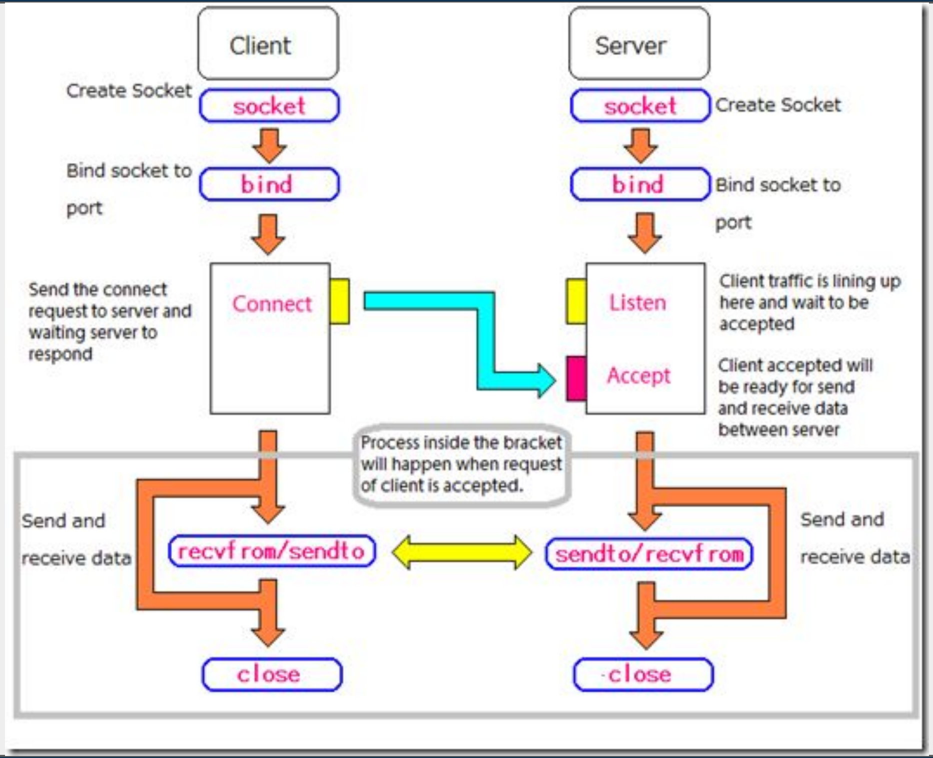
Client -> Server



Hyperiondev

https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Fstart.cybervista.net%2Fhubfs%2FHacker_Chart.pn
g&f=1&nofb=1&ipt=c4a56f8bb2b92f8d78c776f833f92d413fea7123f0d4b8d387a81b4fc844cc50&ipo=images

Sockets (OSI)


Client
Server
Create Socket
socket
socket
Create Socket
Bind socket to port
bind
bind
Bind socket to port
Send the connect request to server and waiting server to respond
Connect
Listen
Client traffic is lining up here and wait to be accepted
Accept
Client accepted will be ready for send and receive data between server
Process inside the bracket will happen when request of client is accepted.
Send and receive data
recvfrom/sendto
sendto/recvfrom
Send and receive data
close
close

Hyperiondev

https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Fstart.cybervista.net%2Fhubfs%2FHacker_Chart.png&f=1&nofb=1&ipt=c4a56f8bb2b92f8d78c776f833f92d413fea7123f0d4b8d387a81b4fc844cc50&ipo=images

Error Handling

- In order to ensure dependable and secure network communication, error handling is essential in socket programming.
- To keep the program stable, socket operations may encounter a variety of errors and exceptions that need to be handled correctly.
- Connection errors, timeouts, address resolution issues, and data transmission issues are all frequent errors and exceptions in socket programming.
- Identification and graceful handling of these errors are necessary for proper error handling in order to avoid crashes, data corruption, or unauthorized access.
- A programming technique called exception handling enables the runtime detection and management of exceptional circumstances.
- Exception handling is a technique used in socket programming to detect and deal with specific exceptions related to socket operations, such as socket timeouts, connection problems, and address resolution issues.

Security

- As network programming with sockets involves sending sensitive data over conceivably untrusted networks, security is of the utmost importance.

- The confidentiality and integrity of data during transmission are guaranteed by encryption, which is a vital security measure. It entails encoding the information so that only permitted parties can decode it.

- Another crucial component of security is authentication, which confirms the identities of parties communicating in order to prevent unauthorized access or impersonation.

- TLS/SSL (Transport Layer Security/Secure Sockets Layer) secure protocols offer a strong framework for secure network communication. These protocols create secure connections, verify the parties taking part, and guarantee data integrity.

Homework: Improve code by including error/exception handling.

Bonus: Improve security

Wrapping Up.

- An application programming interface called Berkeley Sockets API enables network communication between programs. For establishing network connections, sending and receiving data, and managing network resources, it offers a set of features and protocols.

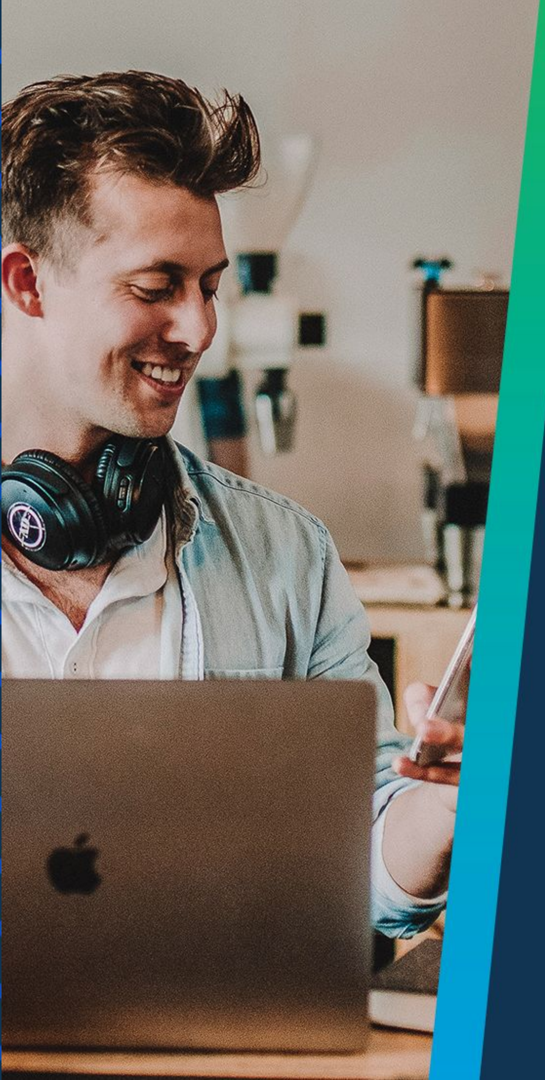
- A strong library that implements the Berkeley Sockets API is the socket module in Python. Using a straightforward and user-friendly interface, it enables developers to create network sockets, establish connections, and carry out various networking operations.

- We explored 2 socket types, including TCP/IP and UDP sockets. Applications that require data integrity can benefit from the dependable, connection-oriented communication offered by TCP/IP sockets. UDP sockets provide simple, connectionless communication that is perfect for programs that place a high priority on efficiency and speed.

Next up

- Linux Tools for Networking and Firewall Management

Questions and Answers



Hyperiondev

Thank you for joining us

1. Take regular breaks
2. Stay hydrated
3. Avoid prolonged screen time
4. Practice good posture
5. Get regular exercise

"With great power comes great responsibility"