**TRƯỜNG ĐẠI HỌC HÀNG HẢI VIỆT NAM  
KHOA CÔNG NGHỆ THÔNG TIN**

**-----\*\*\*-----**

****

**BÁO CÁO BÀI TẬP LỚN**

**HỌC PHẦN “LẬP TRÌNH MẠNG”**

***Đề tài:***

**Xây dựng ứng dụng mô phỏng hoạt động**

**chiếc nón kỳ diệu thông qua mạng**

***GVHD: Nguyễn Trung Quân***

***Sinh viên thực hiện: Trần Thế Anh – Msv: 90739***

***Hải Phòng, tháng 04 năm 2024***

**TRƯỜNG ĐẠI HỌC HÀNG HẢI**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BỘ MÔN KHOA HỌC MÁY TÍNH**

**-----\*\*\*-----**

**BÀI TẬP LỚN**

**HỌC PHẦN: LẬP TRÌNH MẠNG**

**Mã đề tài: < 007 >**

1. **Tên đề tài**

Xây dựng ứng dụng mô phỏng hoạt động của chiếc nón kỳ diệu thông qua mạng

1. **Mục đích**

Tìm hiểu về lập trình Socket dựa trên giao thức TCP

1. **Công việc cần thực hiện**

* Tìm hiểu về giao thức TCP/IP, lập trình mạng với Python Socket
* Thực hiện viết chương trình
* Làm báo cáo bài tập lớn
* Bảo vệ bài tập lớn

1. **Yêu cầu**

* Kết quả làm bài tập lớn: Báo cáo bài tập lớn
* Hạn nộp báo cáo bài tập lớn

1. **Tài liệu tham khảo**

-https://vi.wikipedia.org/wiki/TCP/IP

-https://hocvienagile.com/tat-tan-tat-ve-lap-trinh-socket-python/

-https://kb.pavietnam.vn/tcp-ip-la-gi-toan-tap-ve-giao-thuc-tcp-ip.html#:~:text=TCP%2FIP%20(Transmission%20Control%20Protocol,n%C4%83ng%20ph%E1%BB%A5c%20h%E1%BB%93i%20t%E1%BB%B1%20%C4%91%E1%BB%99ng.

- https://youtu.be/3QiPPX-KeSc?si=PTzex0nSllRU\_rsG

***Hải Phòng, tháng 04 năm 2024***

**NGƯỜI HƯỚNG DẪN**

**MỤC LỤC**

[Chương 1: Cơ sở lý thuyết 1](#_Toc164785192)

[1. Giới thiệu về mô hình TCP/IP 1](#_Toc164785193)

[1.1. Giới thiệu 1](#_Toc164785194)

[1.2. Cách thức hoạt động của mô hình TCP/IP 2](#_Toc164785195)

[1.3. Chức năng của các tầng trong mô hình TCP/IP 2](#_Toc164785196)

[1.4 Ưu - nhược điểm của mô hình TCP/IP 4](#_Toc164785197)

[2. Giới thiệu về Python Socket 6](#_Toc164785198)

[2.1. Giới thiệu 6](#_Toc164785199)

[2.2. Một số thuộc tính của Python Socket 7](#_Toc164785200)

[2.3. Mô hình lập trình Python Socket 7](#_Toc164785201)

[2.4. Các phương thức trong Python Socket 8](#_Toc164785202)

[Chương 2: Xây dựng ứng dụng mô phỏng hoạt động của chiếc nón kỳ diệu thông qua mạng 10](#_Toc164785203)

[1. Chương trình 10](#_Toc164785204)

[1.1. Server 10](#_Toc164785205)

[1.2. Client 11](#_Toc164785206)

[2. Giao diện 12](#_Toc164785207)

[Chương 3: Kết quả thử nghiệm 13](#_Toc164785208)

# Chương 1: Cơ sở lý thuyết

## 1. Giới thiệu về mô hình TCP/IP

### 1.1. Giới thiệu

Bộ giao thức TCP/IP (Transmission Control Protocol/ Internet Protocol) là một mô hình khái niệm và một tập hợp các giao thức truyền thông dùng trong mạng Internet và các hệ thống mạng máy tính tương tự. Tên gọi TCP/IP đến từ hai giao thức nền tảng của bộ giao thức là TCP (Tranmision Control Protocol) và IP (Internet Protocol). TCP và IP cũng là hai giao thức đầu tiên được định nghĩa.

Như nhiều bộ giao thức khác, bộ giao thức TCP/IP có thể được coi là một tập hợp các tầng, mỗi tầng giải quyết một tập các vấn đề có liên quan đến việc truyền dữ liệu, và cung cấp cho các giao thức tầng cấp trên một dịch vụ được định nghĩa rõ ràng dựa trên việc sử dụng các dịch vụ của các tầng thấp hơn. Về mặt logic, các tầng trên gần với người dùng hơn và làm việc với dữ liệu trừu tượng hơn, chúng dựa vào các giao thức tầng cấp dưới để biến đổi dữ liệu thành các dạng mà cuối cùng có thể được truyền đi một cách vật lý.

Ý tưởng hình thành mô hình TCP/IP được bắt nguồn từ Bộ giao thức liên mạng trong công trình DARPA vào năm 1970. Trải qua vô số năm nghiên cứu và phát triển của hai kỹ sư Robert E. Kahn và Vinton Cerf cùng sự hỗ trợ của không ít các nhóm nghiên cứu. Đầu năm 1978, giao thức TCP/IP được ổn định hóa với giao thức tiêu chuẩn được dùng hiện nay của Internet đó là mô hình TCP/IP Version 4.

Vào năm 1975, cuộc thử nghiệm thông nối giữa hai mô hình TCP/IP được diễn ra thành công. Cũng bắt đầu từ đây, cuộc thử nghiệm thông nối giữa các mô hình TCP/IP được diễn ra nhiều hơn và đều đạt được kết quả tốt. Cũng chính vì điều này, một cuộc hội thảo được Internet Architecture Broad mở ra, với sự tham dự của hơn 250 đại biểu của các công ty thương mại, từ đây giao thức và mô hình TCP/IP được phổ biến rộng rãi trên khắp thế giới.

### 1.2. Cách thức hoạt động của mô hình TCP/IP

TCP/IP bao gồm hai phần chính:

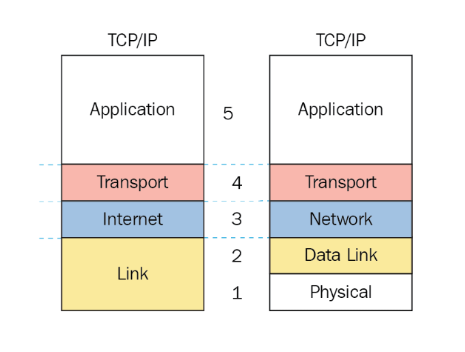
* Transmission Control Protocol (TCP): Là một giao thức truyền tải đáng tin cậy được sử dụng để chia nhỏ và lắp ráp các gói dữ liệu trước khi chúng được gửi qua mạng. TCP đảm bảo rằng các gói dữ liệu được gửi đi sẽ đến đích một cách chính xác và đúng thứ tự. Nó xác định các kết nối, đồng bộ hóa truyền tải và quản lý lưu lượng dữ liệu.
* Internet Protocol (IP): Là một giao thức định tuyến và chuyển tiếp dữ liệu trong mạng. IP định danh và địa chỉ hóa các thiết bị trong mạng, đồng thời quản lý việc định tuyến (routing) các gói dữ liệu qua mạng. Giao thức này đảm bảo gói dữ liệu được gửi đến đúng đích và định vị các thiết bị trong mạng.

Trong quá trình truyền nhận thông tin, nếu giao thức TCP phát hiện ra thông tin bị lỗi, giao thức này sẽ yêu cầu hệ thống gửi thêm các gói tin khác. Để hiểu rõ hơn về cách thức hoạt động và chức năng của TCP/IP, hãy cùng tìm hiểu chức năng của từng tầng trong mô hình này.

### 1.3. Chức năng của các tầng trong mô hình TCP/IP

Một mô hình TCP/IP tiêu chuẩn bao gồm 4 lớp được chồng lên nhau:

* Tầng vật lý (Physical)
* Tầng mạng (Network)
* Tầng giao vận (Transport)
* Tầng ứng dụng (Application)



Tuy nhiên, một số ý kiến lại cho rằng mô hình TCP/IP là 5 tầng, tức các tầng 4 đến 2 đều được giữ nguyên, nhưng tầng Datalink sẽ được tách riêng và là tầng nằm trên so với tầng vật lý.

Cụ thể chức năng của 4 tầng như sau:

**1.3.1. Tầng Ứng dụng (Application)**

Đây là lớp giao tiếp trên cùng của mô hình. Đúng với tên gọi, tầng Ứng dụng đảm nhận vai trò giao tiếp dữ liệu giữa 2 máy khác nhau thông qua các dịch vụ mạng khác nhau (duyệt web, chat, gửi email, một số giao thức trao đổi dữ liệu: SMTP, SSH, FTP,…). Dữ liệu khi đến đây sẽ được định dạng theo kiểu Byte nối Byte, cùng với đó là các thông tin định tuyến giúp xác định đường đi đúng của một gói tin.

Chức năng chính của tầng 3 là xử lý vấn đề giao tiếp giữa các máy chủ trong cùng một mạng hoặc khác mạng được kết nối với nhau thông qua bộ định tuyến. Tại đây dữ liệu sẽ được phân đoạn, mỗi đoạn sẽ không bằng nhau nhưng kích thước phải nhỏ hơn 64KB. Cấu trúc đầy đủ của một Segment lúc này là Header chứa thông tin điều khiển và sau đó là dữ liệu.

Trong tầng này còn bao gồm 2 giao thức cốt lõi là TCP và UDP. Trong đó, TCP đảm bảo chất lượng gói tin nhưng tiêu tốn thời gian khá lâu để kiểm tra đầy đủ thông tin từ thứ tự dữ liệu cho đến việc kiểm soát vấn đề tắc nghẽn lưu lượng dữ liệu. Trái với điều đó, UDP cho thấy tốc độ truyền tải nhanh hơn nhưng lại không đảm bảo được chất lượng dữ liệu được gửi đi.

**1.3.2. Tầng Giao vận (Transport)**

Chức năng chính của tầng 3 là xử lý vấn đề giao tiếp giữa các máy chủ trong cùng một mạng hoặc khác mạng được kết nối với nhau thông qua bộ định tuyến. Tại đây dữ liệu sẽ được phân đoạn, mỗi đoạn sẽ không bằng nhau nhưng kích thước phải nhỏ hơn 64KB. Cấu trúc đầy đủ của một Segment lúc này là Header chứa thông tin điều khiển và sau đó là dữ liệu.

Trong tầng này còn bao gồm 2 giao thức cốt lõi là TCP và UDP. Trong đó, TCP đảm bảo chất lượng gói tin nhưng tiêu tốn thời gian khá lâu để kiểm tra đầy đủ thông tin từ thứ tự dữ liệu cho đến việc kiểm soát vấn đề tắc nghẽn lưu lượng dữ liệu. Trái với điều đó, UDP cho thấy tốc độ truyền tải nhanh hơn nhưng lại không đảm bảo được chất lượng dữ liệu được gửi đi.

**1.3.3. Tầng mạng (Internet)**

Gần giống như tầng mạng của mô hình OSI. Trên TCP/IP, nó cũng được định nghĩa là một giao thức chịu trách nhiệm truyền tải dữ liệu một cách logic trong mạng. Các phân đoạn dữ liệu sẽ được đóng gói (Packets) với kích thước mỗi gói phù hợp với mạng chuyển mạch mà nó dùng để truyền dữ liệu. Lúc này, các gói tin được chèn thêm phần Header chứa thông tin của tầng mạng và tiếp tục được chuyển đến tầng tiếp theo. Các giao thức chính trong tầng là IP, ICMP và ARP.

**1.3.4. Tầng vật lý (Physical)**

Là sự kết hợp giữa tầng vật lý (physical) và tầng liên kết dữ liệu (Datalink) của mô hình OSI. Chịu trách nhiệm truyền dữ liệu giữa hai thiết bị trong cùng một mạng. Tại đây, các gói dữ liệu được đóng vào khung (gọi là Frame) và được định tuyến đi đến đích đã được chỉ định ban đầu.

### 1.4 Ưu - nhược điểm của mô hình TCP/IP

**1.4.1. Ưu điểm**

TCP/IP là giao thức có tính thực tế và tính ứng dụng cao. Đây là giao thức có rất nhiều ưu điểm nổi bật. Chẳng hạn như:

* Có khả năng tạo lập kết nối giữa các máy tính khác nhau
* TCP/IP hoạt động độc lập với hệ điều hành. Do đó ít bị ảnh hưởng khi lỗi win.
* TCP/IP hỗ trợ tương tác với nhiều giao thức trực tuyến khác nhau.
* Có khả năng mở rộng cao nhờ việc kết nối giữa máy client và máy server.
* Giao thức có thể hoạt động hoàn toàn độc lập.
* Hỗ trợ nhiều giao thức khác nhau
* Không ảnh hưởng tới internet hay gây áp lực trên máy tính nhờ có dung lượng nhẹ.

**1.4.2. Nhược điểm**

Tất nhiên, không có một chương trình nào là hoàn hảo 100% và TCP/IP cũng còn một số nhược điểm như:

* Khó quản lý, cài đặt khá cầu kỳ, phức tạp.
* Tầng giao vận đôi khi không đảm bảo được việc phân phối các gói tin, dữ liệu.
* TCP/IP gồm các giao thức cố định, rất khó để thay thế.
* Bên cạnh đó, TCP/IP cũng không có sự tách biệt giữa khái niệm với giao diện. Với những trang web mới trong mạng mới, TCP/IP là giao thức chưa thực sự hiệu quả.
* TCP/IP dễ bị tấn công và từ chối dịch vụ.

Hy vọng qua bài viết các bạn đã hiểu rõ hơn và có câu trả lời cho câu hỏi giao thức TCP/IP là gì. Đây là giao thức truyền dữ liệu trên internet có nhiều ưu điểm nổi bật. Việc hiểu rõ về TCP/IP sẽ giúp bạn có thêm các thông tin hữu ích về việc truyền tải thông tin trên internet.

## 2. Giới thiệu về Python Socket

### 2.1. Giới thiệu

Python Socket là một thư viện thuộc ngôn ngữ lập trình Python cho phép chúng ta làm việc ở hai cấp độ truy cập của dịch vụ mạng.

Giữa hai chương trình chạy trên mạng cần có liên kết hai chiều để kết nối 2 ứng dụng với nhau. Điểm cuối, điểm đầu nút của liên kết song hướng (endpoint) được gọi là socket. Cụ thể, khi bạn viết một ứng dụng và cần tương tác với ứng dụng khác, chúng ta sẽ dựa vào mô hình client/server:

* Server: ứng dụng có khả năng phục vụ, cung cấp cho bạn thông tin.
* Client: ứng dụng gửi yêu cầu đến server.

Để yêu cầu được server thực hiện điều gì đó, client phải có khả năng kết nối đến server. Cơ chế kết nối này được trừu tượng hóa gọi là socket, tương tự như việc cắm dây điện vào ổ cắm. Server được coi là ổ cắm, Client là phích cắm. Một ổ cắm có thể cắm nhiều phích điện, tương tự việc một Server có thể kết nối và phục vụ cho nhiều Client.

* Steam Socket (dựa trên giao thức TCP): Việc truyền dữ liệu chỉ được thực hiện giữa 2 quá trình đã thiết lập kết nối. Steam socket đảm bảo dữ liệu truyền đi đáng tin cậy nhờ có cơ chế chống tắc nghẽn và cơ chế quản lý luồng lưu thông trên mạng.
* Datagram Socket (dựa trên giao thức UDP): Việc truyền dữ liệu không cần có thiết lập kết nối giữa 2 quá trình. Trái ngược với TCP, truyền dữ liệu theo giao thức UDP kém tin cậy, có thể sai trình tự và bị lặp lại. Tuy nhiên cơ chế của Datagram đơn giản hơn nên có tốc độ nhanh, thường được ứng dụng trong các ứng dụng chat hoặc game online.

### 2.2. Một số thuộc tính của Python Socket

|  |  |
| --- | --- |
| **Khái niệm** | **Miêu tả** |
| Domain  (Vùng) | Vùng dùng xác định hạ tầng mạng, nơi diễn ra giao tiếp của socket. Vùng giao tiếp thông dụng nhất hiện nay là AF\_INET. AF\_UNIX chỉ dùng trong giao tiếp cục bộ nhưng trong thực tế ít được sử dụng. Ngoài ra chúng ta có AF\_IPX, AF\_ISO và AF\_NS |
| Type  (kiểu) | Với mỗi vùng của domain, bạn có nhiều cách để giao tiếp: SOCK\_DGRAM (giao tiếp một chiều) và SOCK\_STREAM (giao tiếp 2 chiều) |
| Protocol  (giao thức) | Là cách quy ước gửi nhận dữ liệu giữa hai hay nhiều máy tính trong mạng. Mặc định là 0, sử dụng để nhận diện một biến thể của giao thức bên trong domain |

### 2.3. Mô hình lập trình Python Socket

Mô tả mô hình:

1. Chúng ta mở một socket – socket() để tạo ổ cắm socket cho Server. Đây là quá trình Hệ điều hành phân bổ tài nguyên, chuẩn bị kết nối. Ta cần chỉ định tên hoặc số hiệu port cho socket để Client biết đến ổ cắm của Server.
2. Chúng ta liên kết máy chủ với host hoặc một máy và một port – bind().
3. Server sẽ bắt đầu lắng nghe các kết nối từ Client đưa đến trên port – listen().
4. Một yêu cầu kết nối được gửi từ client tới server – connect().Server chấp nhận yêu cầu của client, kết nối từ đó được thiết lập – accept().
5. Đã có thể gửi và nhận tin – read() / write() tương tự dùng lệnh read/write để đọc ghi trên tập tin. Socket dựa vào số mô tả (socket descriptor) để xác định cần đọc ghi cho hàm read/write.
6. Đóng kết nối – close()

### 2.4. Các phương thức trong Python Socket

|  |  |
| --- | --- |
| socket\_family (Address Family, hay còn gọi là kiểu thiết lập kết nối) | * AF\_UNIX * AF\_INET (Ipv4) hoặc AF\_INET6 (Ipv6) * AF\_IPX (vùng giao thức IPX mạng Novell) * AF\_ISO (chuẩn giao thức ISO) * AF\_NS (giao thức Xerox Network System) |
| socket\_type (cách thiết lập giao thức) | * SOCK\_STREAM (giao thức TCP) * SOCK\_DGRAM (giao thức UDP) |

Ngoài ra chúng ta còn thêm một số phương thức:

|  |  |
| --- | --- |
| **Phương thức** | **Ý nghĩa** |
| s.bind((HOST, PORT)) | Đăng ký tên, gán địa chỉ vào socket |
| s.close() | Đóng kết nối |
| s.listen(5) | Socket lắng nghe tới >= 5 kết nối |
| data = client.recv(1024) | Nhận dữ liệu |
| client, addr = s.accept() | Client gõ cửa, server chấp nhận và tạo ra một socket mới. Client và server đã có thể nhận và truyền dữ liệu. |
| str\_data = data.decode(“utf8”) | Phân tích dữ liệu đã nhận |
| s.sendall(bytes(msg, “utf8”)) | Gửi dữ liệu trên Steam Socket (thông qua giao thức TCP) |

# Chương 2: Xây dựng ứng dụng mô phỏng hoạt động của chiếc nón kỳ diệu thông qua mạng

## 1. Chương trình

### 1.1. Server

import socket

import random

def generate\_random\_item():

items = ['5 triệu', '10 triệu', '20 triệu', '30 triệu']

return random.choice(items)

def handle\_client(client\_socket):

request = client\_socket.recv(1024).decode()

if request == 'generate\_item':

item = generate\_random\_item()

client\_socket.send(item.encode())

else:

client\_socket.send('Invalid request'.encode())

client\_socket.close()

def main():

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server\_socket.bind(('localhost', 8888))

server\_socket.listen(5)

print("Server is listening...")

while True:

client\_socket, address = server\_socket.accept()

print(f"Connection from {address} has been established!")

handle\_client(client\_socket)

if \_\_name\_\_ == "\_\_main\_\_":

main()

### 1.2. Client

import socket

import tkinter as tk

from tkinter import messagebox

from PIL import Image, ImageTk

def send\_request(request):

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

client\_socket.connect(('localhost', 8888))

client\_socket.send(request.encode())

response = client\_socket.recv(1024).decode()

client\_socket.close()

return response

def generate\_item():

item = send\_request('generate\_item')

messagebox.showinfo("Chiếc nón kỳ diệu", f"Bạn đã quay được {item}!")

def main():

root = tk.Tk()

root.title("Chiếc nón kỳ diệu")

root.geometry("900x900")

image = Image.open("chiecNon.jpg")

photo = ImageTk.PhotoImage(image)

label = tk.Label(root, image=photo)

label.pack()

item\_button = tk.Button(root, text="Quay chiếc nón !!!", command=generate\_item)

item\_button.pack(pady=30)

root.mainloop()

if \_\_name\_\_ == "\_\_main\_\_":

main()

## 2. Giao diện

# Chương 3: Kết quả thử nghiệm