UDP Client/Server protocol

مايا بدور*

(تاريخ الايداع 2022/6/1.قبل للنشر في 2022/6/14)

□ملخّص□

هدف هذا البحث الى دراسة اهمية فعالية بروتوكول UDP والى توضيح معنى بروتوكول UDP والية عمله وسلبياته و إيجابياته ومن ثم انشاء chatroom باستخدام هذا البروتوكول وكيف تم تطوير هذا البروتوكول ل جعله موثوقا مع كود بايثون يوضح الية عمل هذا البروتوكول حيث تم دمج كود ال server & client في كود واحد.

الكلمات المفتاحية : بروتوكولUDP ، نموذج OSI ، لغة PYTHON.

*قائم بالأعمال ،قسم هندسة الاتصالات والالكترونيات _كلية الهمك ،جامعة تشرين ،اللاذقية، سوريا mayabadour@gmail.com

مقدمة عامة:

اصبحت الاتصالات ضرورة مهمة في حياتنا وحاجة ملحة للجميع للقيام بمعظم الاعمال سواء اكانت تجارية او تعليمية او صناعية .....

مما دفع بعالم التكنولوجيا والاتصالات لابتكار طرق ارسال في غاية الاهمية و السرعة من اجل انجاز جميع المهام التي تحتاج الى زمن كبير لإنجازها يدويا وهذا ما دفع بنا للبحث عن طرق جديدة من الارسال ...

يشهد عالمنا اللاسلكي تطورا كبيرا من حيث ايجاد تصاميم واستخدام لغات برمجية ذات سرعات عالية وقدرات كبيرة على المعالجة ..ذلك اللغات اصبح لديها اصدارات عدة واكثر تقدما ..

أهمية البحث وأهدافه:

وتكمن اهمية البحث فيما يلي:

1-سوف نتعرف على بروتوكول يعد من اهم البروتوكولات الموجودة ضمن النموذج الطبقي المرجعي OSI.

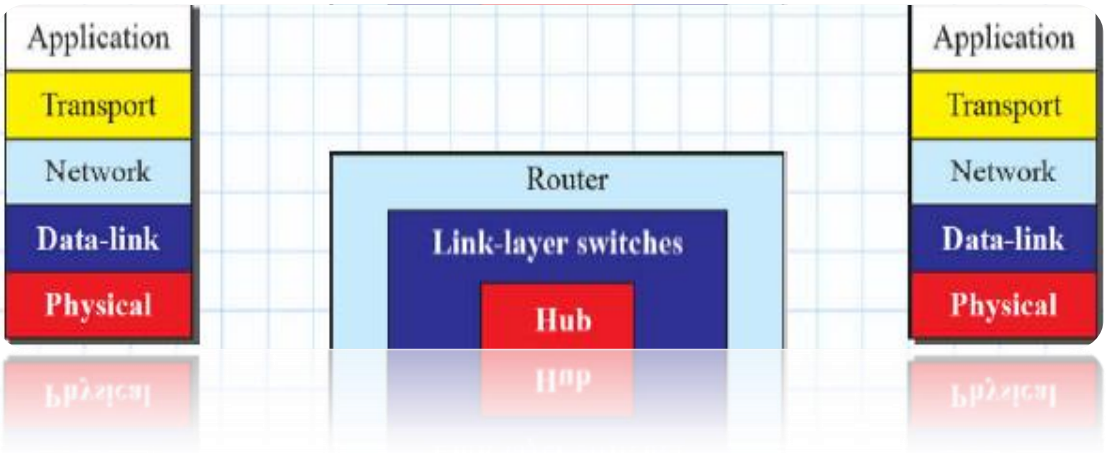
2- ما هو بروتوكول UDP ؟ ماهي اهمية هذا البروتوكول ؟وماهي مساوئه ومحاسنه التي يقدمها خلال عمله ؟وكيف تم جعل هذا البروتوكول موثوقا ؟

3- وسنتعرف على الية كتابة هذا البروتوكول برمجيا باستخدام لغةبايثون.

الإطار النظري:

مفهوم النموذج الطبقي OSI:

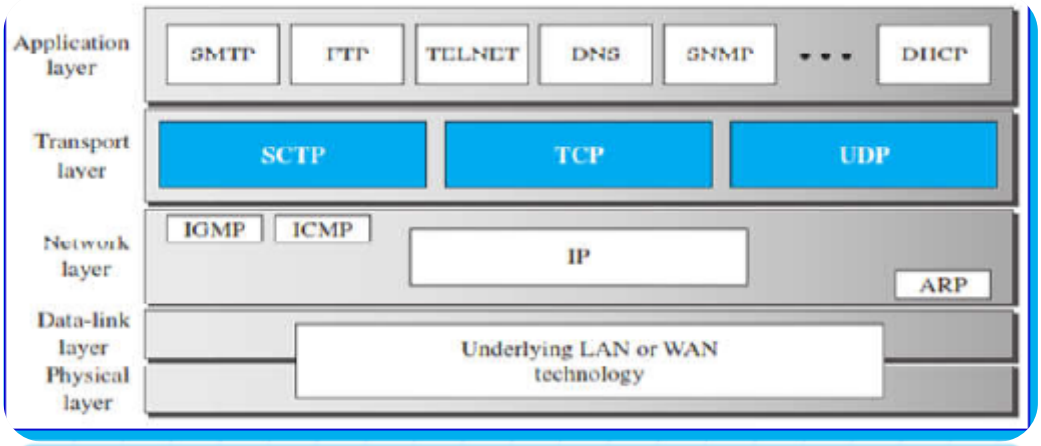
قبل البدء ب توضيح مفهوم ال UDP سنقوم بالبحث عنه ضمن مكدس بروتوكول الشبكة الموضح في الشكل:



الشكل(١)-النموذج الطبقي OSI

يعد TCP/IP هو مكدس بروتوكول الشبكة الذي يهدف الى فرز الشبكة بشكل هرمي.. وهو مكون من خمس طبقات ،يتم تنفيذ الثلاث الطبقات الموجودة اسفل طبقة التطبيق في نواة نظام التشغيل .

يوجد بروتوكول UDP &TCP ضمن طبقة ال Transport كما هو في الشكل (٢)



الشكل(٢)-مكونات كل طبقة من نموذج OSI

طبقة Transport: توفر طبقة الشبكة خدمة غير موثوقة ،فهي ترسل حزما من المنفذ المصدر الى نافذة الوجهة بأسرع ما يمكن ولكنها لا تضمن إمكانية تسليمها في وقت محدد دون اي ضرر كما هو في الشكل (٣).

تعريف البروتوكول UDP:

بروتوكول مخطط بيانات المستخدم (User Datagram Protocol) (UDP): هو بروتوكول طبقة النقل وأيضاً هو جزء من مجموعة بروتوكول الإنترنت، يشار إليها باسم مجموعة (UDP / IP) وعلى عكس(TCP) ، فهو بروتوكول غير موثوق به وغير متصل ولذلك، ليست هناك حاجة لإنشاء اتصال قبل نقل البيانات وأيضاً لعمليات الإرسال الحساسة للوقت بشكل خاص مثل تشغيل الفيديو

أو عمليات البحث عن(DNS) ، يعمل على تسريع الاتصالات من خلال عدم إنشاء اتصال رسميًا قبل نقل البيانات ويسمح ذلك بنقل البيانات بسرعة كبيرة ولكنه قد يتسبب أيضًا في فقد الحزم أثناء النقل ويخلق فرصًا للاستغلال في شكل هجمات (DDoS).

مساوئ ال UDP:

UDP يوفر خدمات نقل معلومات غير موثوقة موجهة نحو Client ، في حال استخدامه للاتصال لا يحتاج الى استقبال ACK كي يتأكد من وصول الاتصال لأنه غير موثوق اي يرسل DATA بشكل دائم .

وهو مفيد ل تطبيقات في الزمن الحقيقي مثل : المهاتفة عبر بروتوكول الانترنت ،ومؤتمرات الفيديو في الزمن الحقيقي .....

محاسن ال UDP :

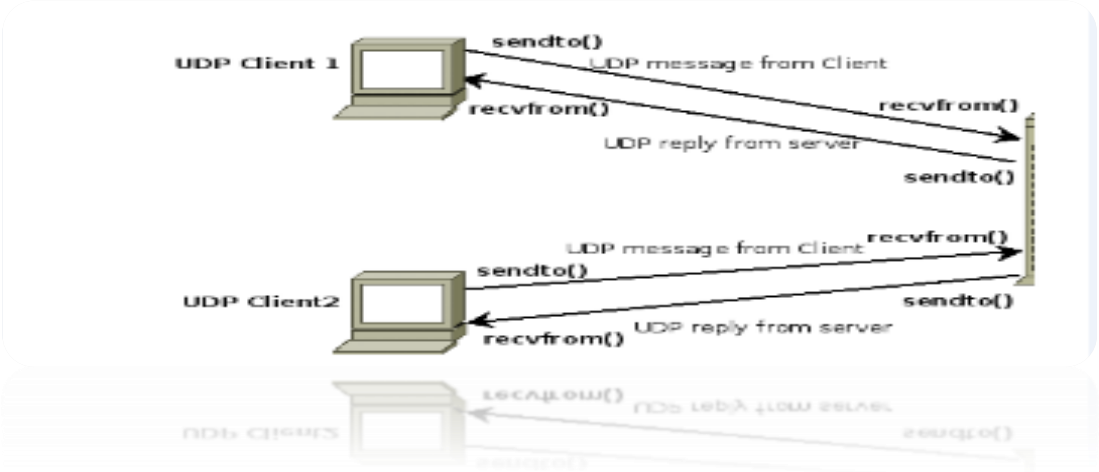
يتميز بروتوكول UDP بأنه يوفر خدمات نقل معلومات بسيطة و يدعم اتصال طرف ل طرف ،او طرف ل عدة اطراف ،او عدة اطراف الى طرف واحد، كما انه يتحكم بالازدحام ل ذلك لن يؤدي ازدحام الشبكة الى تقليل معدل الارسال لمضيف المصدر.

كيف يعمل بروتوكول مخطط بيانات المستخدمUDP_؟

مثل جميع بروتوكولات الشبكات، يعد بروتوكول مخطط بيانات المستخدم طريقة معيارية لنقل البيانات بين جهازي حاسوب في الشبكة ومقارنة بالبروتوكولات الأخرى، ينجز هذا البروتوكول هذه العملية بطريقة بسيطة، حيث يرسل الحزم (وحدات نقل البيانات) مباشرة إلى الحاسوب المستهدف، دون إنشاء اتصال أولاً أو الإشارة إلى ترتيب الحزم المذكورة أو التحقق مما إذا كانت وصلت على النحو المنشود، يُشار إلى حزم UDP باسم "مخططات البيانات". يعتبر بروتوكول (UDP)أسرع ولكنه أقل موثوقية من بروتوكول(TCP) ، وهو بروتوكول نقل شائع آخر وفي اتصال(TCP) ، يبدأ جهازي الحاسوب اتصالاً من خلال عملية آلية تسمى "المصافحة" وبمجرد اكتمال عملية الاتصال هذه، سيقوم أحد أجهزة الحاسوب بالفعل بنقل حزم البيانات إلى الآخر ولا تمر اتصالات (UDP) بهذه العملية وبدلاً من ذلك، يمكن لجهاز حاسوب واحد ببساطة البدء في إرسال البيانات إلى الآخر. بالإضافة إلى ذلك، تشير اتصالات (TCP) إلى الترتيب الذي يجب أن يتم استلام حزم البيانات به وتأكيد وصول الحزم على النحو المنشود، وإذا لم تصل الحزمة فعلى سبيل المثال بسبب الازدحام في الشبكات الوسيطة يتطلب (TCP) إعادة إرسالها ولا تتضمن اتصالات (UDP) أيًا من هذه الوظيفة، وهذه الاختلافات تخلق بعض المزايا ونظرًا لأن بروتوكول مخطط بيانات المستخدم لا يتطلب "مصافحة" أو التحقق مما إذا كانت البيانات تصل بشكل صحيح، فإنه قادر على نقل البيانات بشكل أسرع بكثير من(TCP) .. ومع ذلك، فإن هذه السرعة تخلق مقايضات وفي حالة فقد مخطط بيانات بروتوكول مخطط بيانات المستخدم أثناء النقل، فلن تتم إعادة إرساله،ونتيجة لذلك يجب أن تكون التطبيقات التي تستخدم بروتوكول مخطط بيانات المستخدم قادرة على تحمل الأخطاء والفقدان والازدواجية، من الناحية الفنية، فإن فقدان الحزمة هذا ليس عيبًا في UDP

مثل جميع بروتوكولات الشبكات، يعد بروتوكول مخطط بيانات المستخدم طريقة معيارية لنقل البيانات بين جهازي حاسوب في الشبكة ومقارنة بالبروتوكولات الأخرى، ينجز هذا البروتوكول هذه العملية أكثر من كونه نتيجة لكيفية بناء الإنترنت ولا تقوم معظم أجهزة توجيه الشبكة بتنفيذ طلب الحزم وتأكيد الوصول حسب التصميم، لأن القيام بذلك يتطلب قدرًا غير ممكن من الذاكرة الإضافية وأيضاً (TCP) هو طريقة لسد هذه الفجوة عندما يتطلبها الطلب.

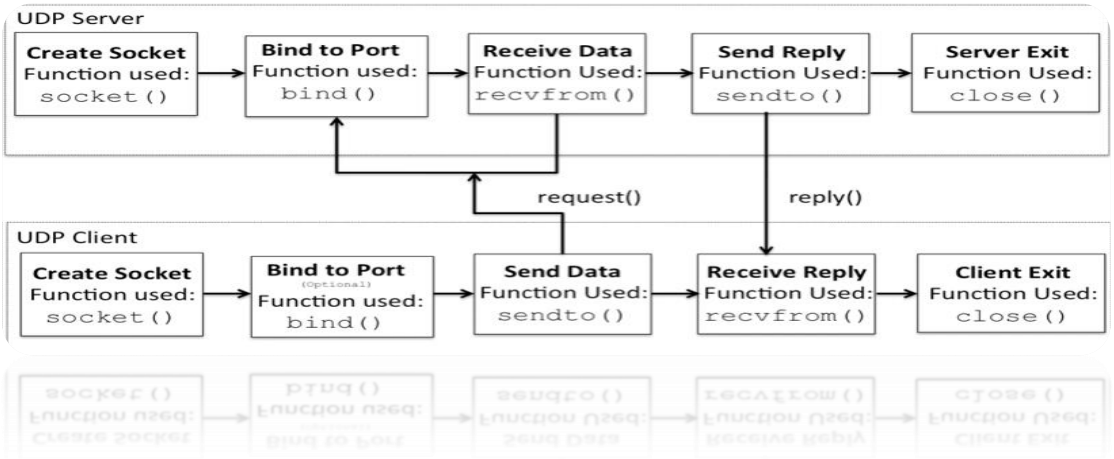
ما هو نوع الخدمات التي تعتمد على بروتوكول مخطط بيانات المستخدمUDP_؟
يستخدم بروتوكول مخطط بيانات المستخدم بشكل شائع في الاتصالات الحساسة للوقت حيث يكون إسقاط الحزم أحيانًا أفضل من الانتظار، ثم يتم إرسال حركة الصوت والفيديو باستخدام هذا البروتوكول لأن كلاهما حساس للوقت ومصمم للتعامل مع مستوى معين من الخسارة، فعلى سبيل المثال) (VOIP الصوت عبر (IP، الذي تستخدمه العديد من خدمات الهاتف القائمة على الإنترنت، يعمل عبر هذا البروتوكول وهذا لأن محادثة هاتفية ثابتة أفضل من تلك التي تكون واضحة تمامًا ولكنها متأخرة بشدة وهذا أيضًا يجعل (UDP) البروتوكول المثالي للألعاب عبر الإنترنت وبالمثل، نظرًا لأن خوادم (DNS) يجب أن تكون سريعة وفعالة ، فإنها تعمل من خلال بروتوكول مخطط بيانات المستخدم أيضًا.



الشكل (٣)

كيف يتم استخدام بروتوكول مخطط بيانات المستخدم UDP_في حجب الخدمة DDoS؟

"مخاطر" بروتوكول مخطط بيانات المستخدم مثل فقدان الحزمة ليست مشكلة خطيرة في معظم حالات الاستخدام ومع ذلك، يمكن استغلال بروتوكول مخطط بيانات المستخدم لأغراض ضارة ونظرًا لأن هذا البروتوكول لا يتطلب مصافحة، يمكن للمهاجمين "إغراق" الخادم المستهدف بحركة مرور (UDP) دون الحصول أولاً على إذن هذا الخادم لبدء الاتصال. يرسل هجوم بروتوكول مخطط بيانات المستخدم النموذجي عددًا كبيرًا من مخططات بيانات (UDP) إلى منافذ عشوائية على جهاز الحاسوب المستهدف ويؤدي هذا إلى إجبار الهدف على الاستجابة بعدد كبير من حزم(ICMP) ، مما يشير إلى أن هذه المنافذ لا يمكن الوصول إليها ويمكن لموارد الحوسبة المطلوبة للرد على كل مخطط بيانات احتيالي استنفاد الهدف، مما يؤدي إلى رفض الخدمة لحركة المرور المشروعة. يمكن للمنظمات الدفاع ضد هجمات الفيضانات بروتوكول مخطط بيانات المستخدم بمجموعة متنوعة من الأساليب وأحدهما هو الحد من معدل استجابة حزم (ICMP)، على الرغم من أن هذا الأسلوب يمكنه أيضًا تصفية الحزم الشرعية وهناك طريقة أخرى تتمثل في تلقي حركة مرور (UDP) والاستجابة لها من خلال شبكة وسيطة من العديد من مراكز البيانات.



الشكل(٤)-الية تصميم UDP برمجيا

محاكاة عمل UDP باستخدام لغة بايثون:

في طرف ال SERVER نستخدم التوابع الاتية كما في الشكل(٦):

اولا نستخدم تابع Socket الذي يتم استدعائه من مكتبة socket الموجودة افتراضيا ضمن ال python ونربطها مع العنوان و رقم المنفذ.

ثم نقوم باستخدام تابع bind الذي يأخذ بارامترات هي ip & portnumber ثم نقوم باستخدام التابع Resvfrom لاستقبال الرسالة من client واذا اردنا ان يقوم ال server بإرسال رد نستخدم التابع sendto اخيرا نستخدم التابع close ..

في طرف ال client نستخدم الية مشابهة لما سبق لكن هنا نستخدم اولا تابع sendto لارسال الرسالة الى server ثم استخدام recvfrom لاستقبال الرد من server..

كما هو موضح في الشكل(٤)..

الان سنضع الكود الذي يوضح الية عمل chatroom بالبايثون.

حيث تم دمج كود ال server & client في كود واحد.

```
import socket
import threading
import pyfiglet
import os
os.system("tput setaf 6")
title = pyfiglet.figlet_format("\t Chat Room ... \n")
print(title)
skt = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
os.system("tput setaf 5")
sender_ip = input("Enter Your System's IP Address: ")
sender_port = int(input("Enter Your System's Port Number: "))
receiver_ip = input("Enter Your Friend's IP Address: ")
receiver_port = int(input("Enter Your Friend's Port Number: "))
skt.bind((sender_ip,sender_port))
def sender():
    while True:
        os.system("tput setaf 1")
        text = input("\nType the Message: ")
        skt.sendto(text.encode(),(receiver_ip,receiver_port))
        print(skt)
def receiver():
    while True:
        os.system("tput setaf 6")
        msgRcv = skt.recvfrom(1024)
        print("\nMessage from Your Friend: " + msgRcv[0].decode())
t1 = threading.Thread(target=sender)
t2 = threading.Thread(target=receiver)
t1.start()
t2.start()
```

```
Enter Your System's IP Address: 192.168.0.108
Enter Your System's Port Number: 3333
Enter Your Friend's IP Address: 192.168.0.107
Enter Your Friend's Port Number: 4444

Type the Message: Hello ! This is OS1
<socket.socket fd=3, family=AddressFamily.AF_INET, type=SocketKind.SOCK_DGRAM, proto=0, laddr=('
192.168.0.108', 3333)>

Type the Message:
Message from Your Friend: Hello! This is OS2

Message from Your Friend: How are you OS1?Hope you are doing good!

<socket.socket fd=3, family=AddressFamily.AF_INET, type=SocketKind.SOCK_DGRAM, proto=0, laddr=('
192.168.0.108', 3333)>

Type the Message: Yes, I'm Absolutely F99! What abt you?
<socket.socket fd=3, family=AddressFamily.AF_INET, type=SocketKind.SOCK_DGRAM, proto=0, laddr=('
192.168.0.108', 3333)>
```

```
Enter Your System's IP Address:192.168.0.107
Enter Your System's Port Number:4444
Enter Your Friend's IP Address:192.168.0.108
Enter Your Friend's Port Number:3333

Type the Message:Hello! This is OS2
Message from Your Friend: Hello ! This is OS1

<socket.socket fd=3, family=AddressFamily.AF_INET, type=SocketKind.SOCK_DGRAM, proto=0, laddr=('
192.168.0.107', 4444)>

Type the Message:How are you OS1?Hope you are doing good!
<socket.socket fd=3, family=AddressFamily.AF_INET, type=SocketKind.SOCK_DGRAM, proto=0, laddr=('
192.168.0.107', 4444)>

Type the Message:
Message from Your Friend:

Message from Your Friend: Yes, I'm Absolutely F99! What abt you?

<socket.socket fd=3, family=AddressFamily.AF_INET, type=SocketKind.SOCK_DGRAM, proto=0, laddr=('
192.168.0.107', 4444)>
```

المراجع :

-*Ref_1 Introduction to Programming using Python*

-UDP protocol specification

-SOCKETS INTERFACE

-PROGRAMMING SOCKET PROGRAMMING.