**UDP Client/Server Socket and GUI**

**ملخص:**

يتم استخدام المقابس وواجهة برمجة تطبيقات المقبس لإرسال الرسائل عبر الشبكة. وهي توفر شكلا من أشكال الاتصال بين العمليات (IPC). يمكن أن تكون الشبكة شبكة محلية منطقية للكمبيوتر، أو شبكة متصلة فعليا بشبكة خارجية، مع اتصالاتها الخاصة بالشبكات الأخرى. المثال الواضح هو الإنترنت، الذي نتصل به عبر مزود خدمة الإنترنت.

اليوم، على الرغم من أن البروتوكولات الأساسية المستخدمة من قبل واجهة برمجة تطبيقات المقبس قد تطورت على مر السنين، وتطورت بروتوكولات جديدة، إلا أن واجهة برمجة التطبيقات منخفضة المستوى ظلت كما هي.

النوع الأكثر شيوعا من تطبيقات مآخذ التوصيل هي تطبيقات خادم /العميل، حيث يعمل جانب واحد كخادم وينتظر الاتصالات من العملاء. هذا هو نوع التطبيق الذي ستقوم بإنشائه. وبشكل أكثر تحديدا، ستركز على واجهة برمجة تطبيقات المقبس لمقابس الإنترنت، والتي تسمى أحيانا مآخذ Berkeley أو BSD. هناك أيضا مآخذ نطاق يونكس، والتي لا يمكن استخدامها إلا للتواصل بين العمليات على نفس المضيف.

في هذه المقالة، سنتحدث عن كيفية قيام التطبيقات بتسليم البيانات بين جهازين باستخدام بروتوكول التحكم في الإرسال (TCP) أو بروتوكول مخطط بيانات المستخدم (UDP). ونركز على دورsocket .

**.1 مقدمة**

طبقة النقل هي جزء من نموذج شبكات TCP/IP، وتسمى أحيانا بنية الشبكات. يحتوي على مجموعة شاملة من الوظائف التي تصف كل ما هو مطلوب لشبكة الكمبيوتر للعمل. طبقة النقل مسؤولة عن الاتصال المنطقي بين التطبيقات التي تعمل على مضيفين مختلفين، وبالتالي توفير الخدمات لبروتوكولات طبقة التطبيق على طبقة أعلى من نموذج شبكة TCP / IP. على الرغم من وجود العديد من بروتوكولات طبقة النقل، فإن البروتوكولين الأكثر استخداما هما بروتوكول التحكم في الإرسال (TCP) وبروتوكول مخطط بيانات المستخدم (UDP). توفر هذه البروتوكولات وظائف مختلفة لمتطلبات التطبيق المختلفة.[1]

عند استخدام أحد التطبيقات، عادةً ما تظهر البيانات أو الخدمات المقدمة على شكل دفق من البيانات المستمرة، ولكن إرسال البيانات (مثل الفيديو) عبر الشبكة كدفق كامل يمكن أن يستهلك كل النطاق الترددي المتاح للشبكة. هذا يمنع الخدمات الأخرى مثل البريد الإلكتروني من استخدام الوسيط ويجعل استعادة الأخطاء وإعادة إرسال بيانات التلف أكثر صعوبة. تقسم آلية تعدد الإرسال بيانات TCP وUDP إلى أجزاء صغيرة لتمكين الاتصال من مستخدمين مختلفين للتداخل على نفس الشبكة. تعتمد هذه الآلية على مفهوم يعرف باسم المقبس socket.

**2. أهمية البحث وأهدافه:**

تلعب socket دورًا حيويًا في تطبيقات خادم العميل. يمكن للعميل والخادم التواصل مع بعضهما البعض عن طريق الكتابة إلى هذه socket أو القراءة منها. يقدم هذا البحث عناصر من برمجة الشبكات والمفاهيم المتضمنة في إنشاء تطبيقات الشبكة باستخدام socket.

**3. المقبس socket:**

بشكل عام، المقابس هي نقاط نهاية داخلية مصممة لإرسال البيانات واستقبالها. ستحتوي الشبكة الواحدة على مقبسين، واحد لكل جهاز أو برنامج متصل. هذه المقابس هي مزيج من عنوان IP ومنفذ. يمكن أن يحتوي جهاز واحد على عدد "n" من المقابس استنادا إلى رقم المنفذ المستخدم. تتوفر منافذ مختلفة لأنواع مختلفة من البروتوكولات[2]

لقد جاء هذا المصطلح "socket " بعدة طرق.

1. يمكن تعريف المآخذ على أنها نقاط نهاية اتصال بين جهازي كمبيوتر يتم تحديدهما بواسطة عنوان IP ورقم منفذ.
2. يمكن تعريف المقابس أيضًا على أنها تجريد للبرامج يستخدم لتمثيل "محطات" اتصال بين جهازين.
3. يمكن تعريفه أيضًا على أنه فكرة مجردة يتم توفيرها لمبرمج تطبيق لإرسال البيانات أو استقبالها إلى عملية أخرى.
4. المقبس هو الباب بين عملية التطبيق وTCP.
5. المقبس هو واجهة بين التطبيق والشبكة.

**1.3 . عمليات المقبس** **socket:**

يقوم المقبس بأربع عمليات أساسية:[3]

1. الاتصال بجهاز بعيد،

2. إرسال البيانات،

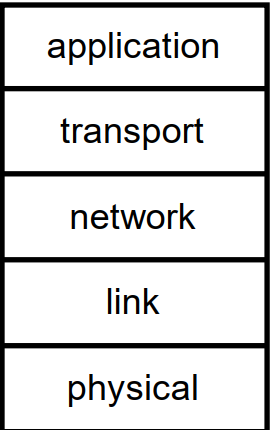
3. تلقي البيانات

4. أغلاق الاتصال.

قد لا يتم توصيل المقبس بأكثر من مضيف واحد في نفس الوقت. ومع ذلك، قد يرسل المقبس البيانات إلى المضيف المتصل به ويستقبلها.

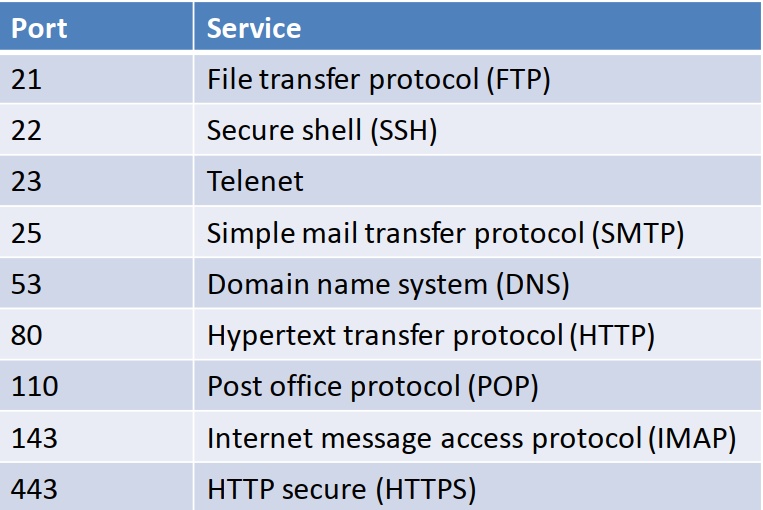
**3.2. المنافذ:**

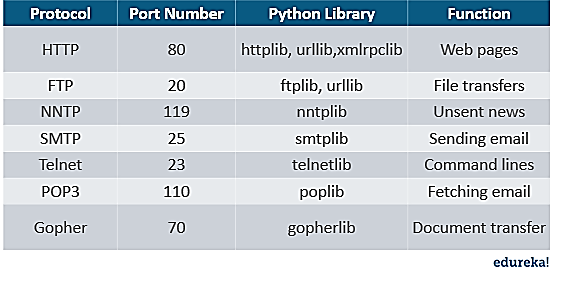
في شبكات الكمبيوتر، يكون المنفذ عبارة عن بناء برنامج خاص بالتطبيق أو خاص بالعملية يعمل كنقطة نهاية للاتصالات في نظام التشغيل المضيف للكمبيوتر. يرتبط المنفذ بعنوان IP الخاص بالمضيف، بالإضافة إلى نوع البروتوكول المستخدم للاتصال، والغرض من المنافذ هو التعرف بشكل فريد على التطبيقات أو العمليات المختلفة التي تعمل على جهاز كمبيوتر واحد، والبروتوكولات التي تستخدم المنافذ بشكل أساسي هي النقل بروتوكولات الطبقة، مثل بروتوكول التحكم في الإرسال (TCP) وبروتوكول مخطط بيانات المستخدم (UDP) الخاص بمكدس برامج الإنترنت، والتي تسمى غالبًا TCP / IP (بروتوكول التحكم في النقل / بروتوكول الإنترنت)، كما هو موضح في الشكل (1). يستخدمون المنافذ لتعيين البيانات الواردة لعملية معينة تعمل على الكمبيوتر. لذا فإن المنفذ سيحدد المقبس على مضيف[4,5].



الشكل (1)

مرة أخرى، لا يكفي عنوان IP لتحديد خادم فريد، لأن العديد من برامج الخادم قد تكون موجودة على جهاز واحد. لذلك نحن بحاجة إلى تعريف فريد لكل خادم. يشار إلى هذا التعريف الفريد باسم المنفذ. عندما نقوم بإعداد عميل أو خادم، يجب علينا اختيار منفذ يوافق عليه كل من العميل والخادم. هذا المنفذ ليس منفذًا فعليًا، ولكنه منفذ منطقي محدد برقم صحيح 16 بت. بعض المنافذ لا. تم حجز من 0 إلى 1024 لدعم الخدمات الشائعة / المعروفة ويجب على المطور أن يكون حريصًا على عدم استخدام أحد أرقام المنافذ المعروفة والمحددة في RFCs أثناء تطوير تطبيق خاص بخادم العميل





أرقام المنافذ الشائعة والبروتوكولات ذات الصلة

**4.** **كيفية تحقيق برمجة المقبس في بايثون:**

لتحقيق برمجة المقبس في بايثون، ستحتاج إلى استيراد وحدة **المقبس** أو [الإطار](https://www.edureka.co/blog/python-frameworks/). تتكون هذه الوحدة من طرق مضمنة مطلوبة لإنشاء مآخذ ومساعدتها على الارتباط ببعضها البعض.

بعض الطرق الهامة هي كما يلي:

|  |  |
| --- | --- |
| أساليب | وصف |
| *socket.socket()* | تستخدم لإنشاء مآخذ التوصيل (مطلوبة على كل من الخادم وكذلك نهايات العميل لإنشاء مآخذ التوصيل) |
| *socket.accept()* | تستخدم لقبول اتصال. يقوم بإرجاع زوج من القيم (conn، العنوان) حيث يكون conn كائن مقبس جديد لإرسال البيانات أو استقبالها والعنوان هو عنوان المقبس الموجود في الطرف الآخر من الاتصال |
| *socket.bind()* | يستخدم للربط بالعنوان المحدد كمعلمة |
| *socket.close()* | تستخدم لوضع علامة على المقبس على أنه مغلق |
| *socket.connect()* | يستخدم للاتصال بعنوان بعيد محدد كمعلمة |
| *socket.listen()* | تمكين الخادم من قبول الاتصالات |

يوفر Python مستويين من الوصول إلى خدمات الشبكة.  على مستوى منخفض، يمكنك الوصول إلى دعم المقبس الأساسي في نظام التشغيل الأساسي، والذي يسمح لك بنشر العملاء والخوادم لكل من البروتوكولات الموجهة إلى الاتصال وغير المتصلة.

 المقابس لها مفردات خاصة بها -

|  |  |
| --- | --- |
| **الرقم.** | **الوصف** |
| 1 | Domain النطاق  مجموعة البروتوكولات المستخدمة كآلية النقل.  هذه القيم هي ثوابت مثل AF\_INET وPF\_INET وPF\_UNIX وPF\_X25. |
| 2 | type النوع  نوع الاتصالات بين نقطتي النهاية، عادةً SOCK\_STREAM للبروتوكولات الموجهة للاتصال وSOCK\_DGRAM للبروتوكولات غير المتصلة. |
| 3 | protocol البرتوكول  عادةً ما تكون صفرية، يمكن استخدام هذا لتحديد متغير بروتوكول داخل مجال ونوع. |
| 4 | hostname اسم المضيف  معرف واجهة الشبكة - سلسلة، يمكن أن تكون اسمًا مضيفًا أو عنوانًا منقطًا رباعيًا أو عنوان IPV6 في علامة النقطتين (وربما نقطية) سلسلة ""، تحدد عنوان INADDR\_BROADCAST.  سلسلة ذات طول صفري تحدد INADDR\_ANY أو عدد صحيح، يتم تفسيره كعنوان ثنائي بترتيب بايت المضيف. |
| 5 | port المنفذ  كل قائمة الخادم للعملاء الذين يتصلون على واحد أو أكثر من المنافذ.  قد يكون المنفذ هو رقم منفذ Fixnum أو سلسلة تحتوي على رقم منفذ أو اسم خدمة. |

**socket Module .1.4برمجة السوكيت**

لإنشاء مأخذ توصيل، يجب عليك استخدام دالة socket.socket () المتاحة في وحدة المقابس، والتي تحتوي على بناء الجملة العام

s = socket.socket (socket\_family, socket\_type, protocol=0)

Here is the description of the parameters − هنا وصف الوسيطات المتغيرة

* socket\_family - هذا إما AF\_UNIX أو AF\_INET ، كما هو موضح سابقًا.  
    
   socket\_type - هذا إما SOCK\_STREAM أو SOCK\_DGRAM.

**Server Socket Methods دوال سيرفر المقابس .2.4**

|  |  |
| --- | --- |
| **الرقم.** | **الدالة والوصف** |
| 1 | s.bind()  هذه الطريقة تربط العنوان (اسم المضيف، زوج رقم المنفذ) بالمقبس. |
| 2 | s.listen()  تقوم هذه الطريقة بإعداد وتشغيل مستمع TCP. |
| 3 | s.accept()  يقبل هذا بشكل سلبي اتصال عميل TCP، في انتظار حتى وصول الاتصال (حظر). |

**. General Socket Methods دوال عامة بالسوكيت.3.4**

|  |  |
| --- | --- |
| **الرقم.** | **الدالة والوصف** |
| 1 | s.recv()  هذه الطريقة تستقبل رسالة TCP |
| 2 | s.send()  هذه الطريقة تنقل رسالة TCP |
| 3 | s.recvfrom()  هذه الطريقة تستقبل رسالة UDP |
| 4 | s.sendto()  هذه الطريقة تنقل رسالة UDP |
| 5 | s.close()  هذه الطريقة تغلق المقبس |
| 6 | socket.gethostname()  إرجاع اسم المضيف. |

**6. القسم العملي:**

التطبيق عبارة عن UDP Client/Server

سنقوم ببناء Server يستفيد من خدمة على الانترنت لمعرفة أسعار الصرف بحيث يجلب أسعار الصرف لعملات محددة التي نرغب بها فقط. ويقوم باستلام رسائل من الزبائن لطلب سعر عملة ما والرد عليها بالسعر المناسب.

وسنقوم ببناء Client يعرض للمستخدم قائمة بالخدمات التي يقدمها وهي: 1- معرفة أسعار الصرف لجميع العملات. 2- تحويل مبلغ محدد من الدولار لعملة أخرى. ثم يقوم بمطالبته بإدخال رقم الخدمة المطلوبة. وطلبها من السيرفر وعرض النتائج.

وسنقوم ببناء GUI Client هو نسخة مطورة من برنامج العميل يقدم نفس الخدمات ولكن عن طريق واجهة رسومية مناسبة للمستخدم.

**Server code:**

import socket

import requests

s = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

address = ("127.0.0.1", 8888)

s.bind(address)

currencies\_rates = {}

# load\_currencies

url = "https://api.apilayer.com/exchangerates\_data/latest?base=USD&symbols=USD,EUR,SYP,AED,CAD"

res = requests.get(url,headers={ 'Accept':'application/json', 'apikey':'hNyyYLiFXgHQqYMPWa5fou4ac1ptoB9P' })

if res.status\_code != 200:

    print('error while loading currencies')

else:

    data = res.json()

    currencies\_rates = data['rates']

    print('currencies loaded successfully')

    print('waiting a message on port 8888')

    while True:

        message, address = s.recvfrom(1024)

        print('Message from address: ', address)

        currency = message.decode()

        if currency in currencies\_rates:

            rate = str(currencies\_rates[currency])

            s.sendto(rate.encode(), address)

        else:

            s.sendto("NOT\_FOUND".encode(), address)

**Client code:**

import socket

s = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

server\_address = ("127.0.0.1", 8888)

print('Welcome to currencies exchange')

print('1- List all currencies rates')

print('2- Convert USD to another currency')

select = int(input('select 1 or 2: '))

if select == 1:

    for currency in ['EUR', 'SYP', 'AED', 'CAD']:

        s.sendto(currency.encode(), server\_address)

        message, address = s.recvfrom(1024)

        rate = float(message.decode())

        print('1 USD =', round(rate,2), currency)

elif select == 2:

    currency = input('Enter currency (EUR, SYP, AED, CAD): ')

    if currency not in ['EUR', 'SYP', 'AED', 'CAD']:

        print('Invalid currency')

    else:

        s.sendto(currency.encode(), server\_address)

        message, address = s.recvfrom(1024)

        rate = float(message.decode())

        usd\_amount = float(input('Enter the amount (in USD):'))

        amount = usd\_amount \* rate

        print(round(amount,2),currency)

**GUI Client Code:**

import socket

from tkinter import \*

# build GUI

w = Tk()

w.title("Currencies Exchange")

frame = Frame(width=400, height=400)

frame.pack()

label1 = Label(master=frame, text="Currencies rate:", font=('Arial', 12, 'bold'))

label1.place(x=25, y=25)

all\_currencies\_label = Label(master=frame, bg="#DDDDDD", width=50, height=5, anchor='w')

all\_currencies\_label.place(x=25, y=60)

label2 = Label(master=frame, text="Convert USD to other currencies:", font=('Arial', 12, 'bold'))

label2.place(x=25, y=180)

usd\_amount\_entry = Entry(master=frame, width=20)

usd\_amount\_entry.place(x=25, y=217)

def calculate(event):

    usd\_amount = float(usd\_amount\_entry.get())

    t = ""

    for currency, rate in currencies.items():

        amount = usd\_amount \* rate

        t += "{} USD = {} {}\n".format(usd\_amount, round(amount,2), currency)

    result\_label.config(text = t)

button = Button(master=frame, width=15, text="Calculate")

button.place(x=160, y=215)

button.bind("<Button-1>", calculate)

result\_label = Label(master=frame, bg="#DDDDDD", width=50, height=5, anchor='w')

result\_label.place(x=25, y=250)

# load currencies

currencies ={}

s = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

server\_address = ("127.0.0.1", 8888)

t = ""

for currency in ['EUR', 'SYP', 'AED', 'CAD']:

    s.sendto(currency.encode(), server\_address)

    message, address = s.recvfrom(1024)

    rate = float(message.decode())

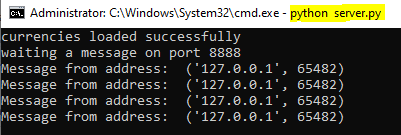
    currencies[currency] = rate

    t += "1 USD = {} {}\n".format(round(rate,2), currency)

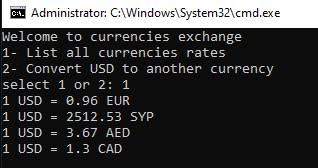
all\_currencies\_label.config(text = t)

w.mainloop()

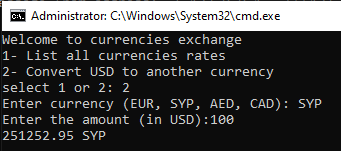
**تنفيذ الكود وعرض النتائج:**



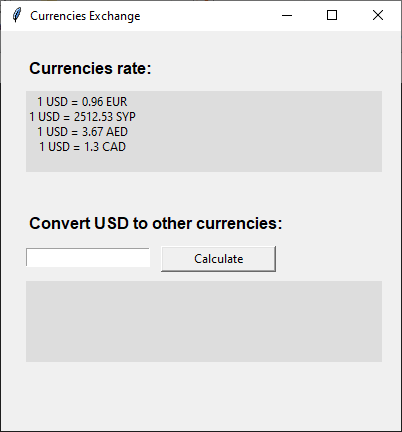
اختبار الخدمة الأولى:

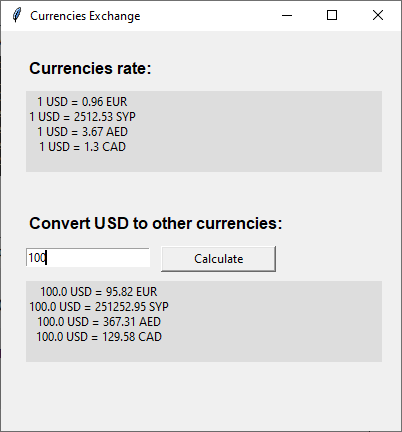


اختبار الخدمة الثانية:



اختبار الواجهة الرسومية:





**.5المراجع:**

|  |  |
| --- | --- |
| <https://www.studytonight.com/network-programming-in-python/working-with-tcp-sockets>. | [1] |
| <https://codingshiksha.com/python/python-3-script-to-transfer-large-images-pdf-documents-from-remote-server-using-socket-full-project-for-beginners/> | [2] |
| <https://cisco.box.com/v/devnet1040>. | [3] |
| A tutorial on Networking Programming using Sockets  <http://beej.us/guide/bgnet/output/print/bgnet_USLetter> | [4] |
| <http://en.wikipedia.org/wiki/Berkeley_sockets> | [5] |