

إعداد: مرام خالد ونوس , مرّح محمود وسوف , حلا أحمد دنيا

## تطبيق خادم اعميل لإرسال الملفات باستخدام البروتوكول TCP

ملخص: أصبحت الشبكات أداة ضرورية لجميع مجالات الحياة، وأدى التطور في هذا المجال والتكامل مع لغات البرمجة إلى تطوير أدوات كثيرة وبسيطة من أجل تسهيل واختصار أمور روتينية. من أهم لغات البرمجة التي تتعامل مع الشبكات هي لغة بايثون التي تقدم تجريد في التعامل مع النموذج المرجعي للشبكة وتتيح التعامل مع مختلف البروتوكولات في مختلف الطبقات. في هذا المشروع سنقوم بإنشاء خدمة إرسال الملفات عبر البروتوكول TCP، حيث سننشئ خادم مهمته تقديم محاضرات وملفات مقرر برمجة الشبكات لأي زبون يتصل بهذا المخدم بالاعتماد على لغة بايثون والبروتوكول TCP الذي يعمل في طبقة النقل وأيضاً سننشئ زبون ونهيؤه للاتصال بالخادم لاستقبال الملفات الموجودة ضمن الخادم بحيث سيكون الإرسال هو بث عام لجميع الملفات من أجل جميع المتصلين.

## **Server/Client Application for sending file using TCP**

**Abstract:** Networks have become a necessary tool for all areas of life, and the development in this field and integration with programming languages has led to the development of many simple tools in order to facilitate and shorten routine matters. One of the most important programming languages that deal with networks is Python, which provides an abstraction in dealing with the network reference model and allows dealing with different protocols in different layers. In this project, we will create a TCP file transmission service, where we will create a server whose mission is to provide lectures and files for network programming to any client that connects to this server based on the Python language and the TCP protocol that works in the transport layer. We will also create a client and prepare it to connect to the server to receive the files within the server so The transmission will be a public broadcast of all files for all callers.

## مقدمة:

برمجة الشبكات هو مصطلح جديد يتم استخدامه كثيراً مؤخراً، لأنه أثبت أن له فوائد وأهمية كبيرة للغاية في المجال الصناعي والاقتصادي وحتى الأمني. ففكرة برمجة الشبكات تقوم على ربط مجموعة من الأجهزة سوياً عن طريق جهاز اتصال خاص. وهذا الربط يساعد على عدة أشياء منها سهولة نقل المعلومات والبيانات من جهاز إلى آخر، وسهولة السيطرة على الجهاز من على بعد وذلك بهدف الإطلاع على ما يحتويه من بيانات. فنجد أنه بعد أن أصبحت الأجهزة متصلة ببعضها البعض أصبح من السهل إرسال واستقبال المعلومات والبيانات والأرقام عن طريق الرسائل. والسبب الأساسي لانتشار هذه الخدمة في العالم كله هو انتشار أجهزة الحاسوب والكمبيوتر الشخصي في كل البيوت تقريباً، وظهور حاجة شديدة لوجود رابط بين هذه الأجهزة. فهناك العديد من الخدمات المختلفة التي ظهرت واستفاد منها العالم نتيجة لظهور خدمة برمجة الشبكات، ومن هذه الخدمات خدمات البريد الإلكتروني التي أصبحت اليوم مسؤولة عن أغلب الأعمال ويعتمد عليها بشكل كبير في التواصل. كما كان لها دور كبير في خدمة الإدارة المركزية في العديد من الجهات الكبرى حول العالم، وكان لها الفضل في تحقيق قفزة اقتصادية كبيرة في المجال الصناعي والتجاري. كما تستخدم خدمة برمجة الشركات في التأمين والحفاظ على إستقرار وحفظ أمان المنشآت، وكل هذه الخدمات وغيرها تطورت بشكل ملحوظ بعد أن تطورت نظم البرمجة في العالم كله، مما جعلها ماثراً للجدل والبحث والتطوير دائماً. من المؤكد أن لبرمجة الشركات العديد من الفوائد في أغلب مجالات الحياة، فالبرمجة بشكل عام أصبحت هي اللغة الحديثة في هذا الزمن والتي يعتمد عليها في كل شيء، فهي أساس كل الأعمال والنشاطات الاقتصادية والتجارية، وبسبب تأثيرها القوي لفتت انتباه الرأي العام لأهميتها ولضرورة تسليط الضوء عليها، فمن فوائد برمجة الشبكات:

- خلق رابط بين مجموعة من الأجهزة يساعد على سرعة نقل المعلومات والبيانات، ويتكون هذا الرابط عن طريق أحد وسائل الاتصال.
- سهولة وسرعة مشاركة المعلومات من مكان إلى مكان آخر، مع توفير في الوقت والجهد وبأفضل شكل ممكن، مما يساعد على تفادي الكثير من المشكلات التي كانت تحدث بسبب الروتين والتباطؤ في الإنجاز.
- اختفت الآن بشكل كبير وسائل التخزين التقليدية القديمة المتحركة، وحل محلها وسائل التخزين الحديثة، وأصبح من الإمكان الاحتفاظ وتبادل وتخزين البرامج والملفات والبيانات الهامة والمعلومات المهمة بشكل سهل ومن دون تعقيدات.
- خلق بيئة جديدة للعمل والدراسة، فمع انتشار البرمجة أصبح هناك بيئة عمل جديدة توفر العديد من المزايا للعاملين وللإداريين، كما وفرت بيئة عمل مميزة للدارسين عن طريق تسهيل الوصول للمعلومات، أو تسهيل الحصول على دورات تدريبية بأقل تكلفة وغيرها من المزايا الأخرى التي غيرت بشكل كبير طرق البحث والتواصل.
- اعتمدت الإدارات المركزية في العديد من المؤسسات المختلفة على هذه الخدمة بشكل كبير، فقد كان لها دور مهم في جعل أعمالهم أكثر سهولة وأكثر دقة بأقل تكلفة، فتعمل هذه الخدمة على توفير الكثير من الوقت والجهد، فكل ما عليهم

فعله هو أن يقوموا بإنشاء شبكة برمجيات متكاملة خاصة بهم، وسيكون من السهل حينها السيطرة مع كل أطراف المؤسسة.

## أدوات وطرائق البحث:

- 1- لغة بايثون
- 2- مكتبة socket
- 3- مكتبة threading
- 4- مكتبة os
- 5- بيئة PyCharm

## المنهجيات العلمية:

### 1- أنواع الشبكات البرمجية

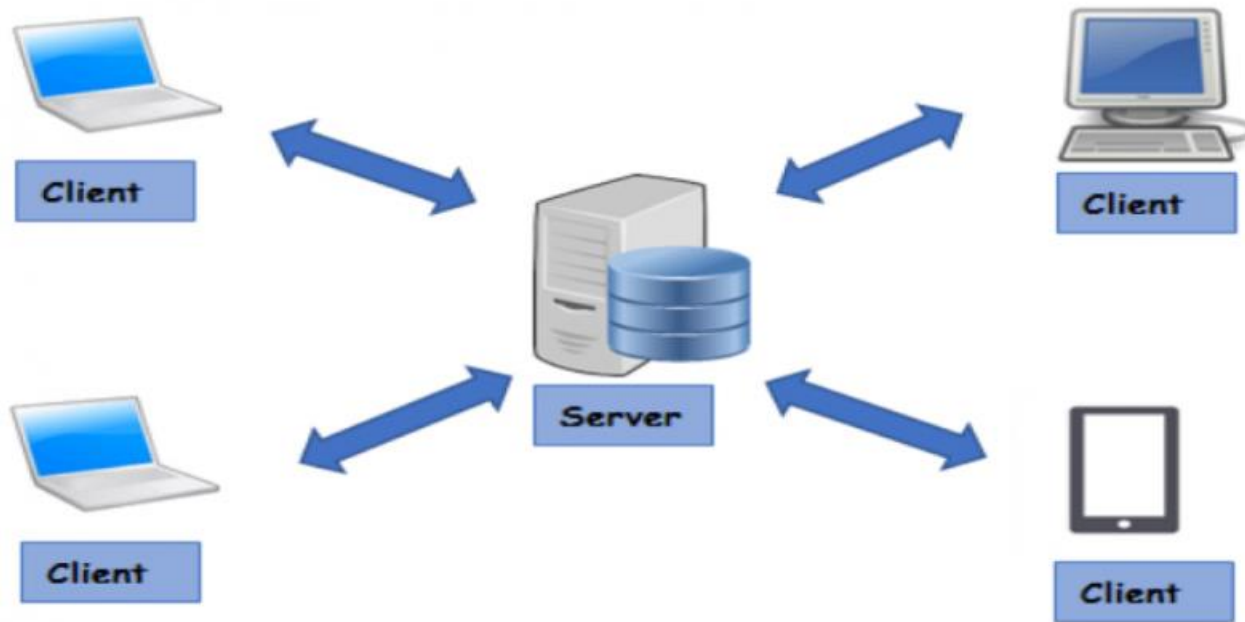
تختلف الشبكات البرمجية حسب حاجة كل منشأة، كما تختلف اللغة البرمجية المستخدمة لكل شبكة، فلكل نوع شكل معين يتميز بها، ومن أشهر الأنواع المستخدمة في المنشآت والمؤسسات المختلفة:

- الشبكة الواسعة (Wide Area Network (WAN) ويتم اختصارها وتسمى شبكة WAN، فهي من أكثر أنواع الشبكات إستخداماً، فكما يشير إسمها فهي شبكة واسعة، أي تستطيع أن تخدم منطقة جغرافية كبيرة للغاية نسبياً. ومن أشهر الأمثلة عليها هي شبكة الإنترنت، فهي تقوم في الأساس على الإعتماد على أن تقوم بتوصيل الأجهزة ببعض رغم بعد المسافات، وهي الشبكة التي تربط العالم كله ببعض وتسهل كثيراً من وسائل التواصل والتعلم.
- الشبكات المحلية (Local Area Network (LAN) ويتم اختصارها وتسمى شبكة LAN، ومن أشهر أمثلة هذه الشبكة هي شبكة الكلية مثل شبكة كلية الهمك أو شبكة كلية العلوم.... فهي تقوم من الأساس على ربط مجموعة من الأجهزة معاً، على أن يكون بين هذه الأجهزة رابط مشترك، ولابد أن يكون هذا الإتصال عن طريق وجود وسيط مشترك، ويتم إستخدام حينها بروتوكولات مشتركة تخص هذا النوع من الشبكات.
- شبكة الند للند (Workgroup) وشبكة الند بالنند تسمى أيضاً بشبكة Peer to Peer أو شبكة Workgroup، وتقوم هذه الشبكة بربط عدد محدد من أجهزة الحاسوب، يمكن أن يصل عددها إلى 10 أجهزة حاسوب. وذلك دون الاعتماد

في الأساس على وجود خادم أو وسيط، وتستخدم هذه التقنية في الأغلبية في المنشآت صغيرة الحجم والتي تحتاج في أعمالها إلى ربط أجهزتها ببعضهم البعض.

- شبكة العاصمة: شبكة العاصمة تسمى أيضاً بشبكة Local Metropolitan Network، وهي شبكة متوسطة الحجم، فهي ليست كبيرة للغاية وتخدم عدد كبير مثل الشبكة العالمية. ليست صغيرة وتخدم عدد محدود مثل الشبكات المحلية، بل هي تأخذ مكان في المنتصف بينهم، فهي تخدم عدد لا بأس به، ولكن ليس بالطبع في نفس قوة وإنتشار الشبكات العالمية.
- شبكة الخادم والعميل: شبكة الخادم والعميل تسمى أيضاً بشبكة Server / Client، وتكون هذه الشبكة أكثر قوة وإنتشاراً قليلاً من الشبكات المحلية. فيمكن أن تقوم بربط عدد لا بأس به من الأجهزة، التي يمكن أن يصل عددها إلى 1024 جهاز، ولكن مع وجود وسيط إتصالي أو خادم ومع استخدام اللغة البرمجية المناسبة.

## 2- نموذج الخادم والعميل:



الشكل 1 نموذج الخادم والعميل

نموذج طلب الخدمة أو نموذج العميل/الخادم أو نموذج المُستخدم/المُخدّم Client/Server Model هو نموذج بُنيوي لتطبيق مُوزّع حيث يجري توزيع المُهام أو الأعمال بين الطرف الذي يُقدّم الخدمات أو الموارد ويُسمّى المُخدّم والطرف الذي يطلب الخدمة

ويُسمَّى العميل أو مُستخدم الخدمة. غالباً ما يتَّصل المُخدّم مع العميل عبر شبكة حواسِب، حيث يعمل كل منهما على منصّة مُنفصلة، ولكن يُمكن أن يتواجد المُخدّم والعميل ضمن نفس النظام.

يُمكن للمُخدّم أن يُشغّل برنامجاً واحداً أو أكثر من البرامج الخاصة بطرف المُخدّم لتقديم خدمة واحدة أو أكثر أو مُشاركة الموارد مع عميلٍ واحدٍ أو أكثر، أمّا العميل فلا يُشارك موارده مع أحد، ولكنّه يطلب الخدمة أو الموارد من المُخدّم. لذلك يبدأ العملاء بإنشاء قنوات اتصال مع المُخدّمات التي تنتظر مبادرتهم وتعمل على تقديم الخدمة المُناسبة لهم. يوجد العديد من البروتوكولات التي يتم التعامل معها حسب الطبقة التي يتم التعامل معها ولكن في الأساس يتم الاعتماد على طبقة النقل أي على البروتوكول TCP أو البروتوكول UDP.

### 3- بروتوكول التحكم بالنقل TCP:

بروتوكول التحكم بالنقل أو بروتوكول التحكم بالإرسال (بالإنجليزية: Transmission Control Protocol TCP) هو أحد البروتوكولات الأساسيّة في حزمة بروتوكولات الإنترنت، موصوف بالوثيقة (RFC 793)، ويُؤمن نقلًا موثوقاً خالياً من الأخطاء لدفق من البايتات بين مُضيفين يتصلان مع بعضهما البعض عبر شبكة تدعم بروتوكول الإنترنت. تعتمد معظم تطبيقات الإنترنت الرئيسيّة مثل الويب والبريد الإلكتروني ونقل الملفات على بروتوكول التحكم بالنقل.

يشكل هذا البروتوكول وسيطاً بين التطبيقات التي تريد إرسال بيانات على شبكة الإنترنت وبين بروتوكول IP الذي يتحكم بعمليات العنونة وتوجيه الرسالة إلى الوجهة المراد الإرسال إليها، وذلك حتى نخفف العبء على هذه التطبيقات التي تريد الإرسال باستخدام بروتوكول IP ، لأن بروتوكول IP في الحقيقة يقوم بتقسيم المعلومات المراد إرسالها على الشبكة إلى طرود مما يزيد من مهمات التطبيقات التي تريد الإرسال باستخدامه، لذلك يقوم TCP بهذه العمليات كلها ليجرد التطبيقات العليا من جميع عمليات التحكم بالنقل (سواء الإرسال أو الاستقبال).

ولا تقتصر مهمة TCP على عملية تقسيم البيانات على شكل طرود يمكن لـ IP أن يرسلها، في الحقيقة يقوم أيضاً بعمليات تصحيح أخطاء النقل كلها أيضاً، حيث من الممكن أن تضيع الطرود أو أن تصل تالفة إلى المستقبل، وبالتالي يتوجب على TCP في هذه الحالة أن يقوم بإعادة إرسالها من المرسل مرة أخرى، وذلك بأن يقوم TCP الذي يوجد عند المستقبل بطلب إعادة إرسال من المرسل والذي يقوم بفهم هذا الطلب هو TCP الموجود عند المرسل، كما ويقوم هذا البروتوكول بعملية إعادة ترتيب الطرود التي وصلته عند المستقبل ليقوم بإيصال النسخة الكاملة من البيانات المرسلّة إلى الطبقة العليا التي تحوي التطبيقات (البرامج) التي ستقوم بمعالجة هذه البيانات وفهمها.

يتم استخدام بروتوكول TCP من قبل العديد من تطبيقات الإنترنت المشهورة مثل البريد الإلكتروني (E-Mail) والشبكة العنكبوتية العالمية (World Wide Web) وبروتوكول نقل الملفات (File Transfer Protocol) وتطبيقات نقل الوسائط (Streaming Media Applications) والعديد من التطبيقات أخرى.

### 3-1- ترويسة TCP:

جدول 1 ترويسة TCP

31 - 24	23 — 16						15 — 10	9 — 4	البتات 3 — 0	+
عنوان بوابة المصدر							عنوان بوابة الوجهة			0
الرقم التسلسلي										32
رقم التأكيد (الإقرار)										64
مقدار الإزاحة	محجوز	1ع	2ع	3ع	4ع	5ع	6ع	حجم النافذة		96
مجموع الاختبار							بيانات مستعجلة			128
خيارات وتذييل										160
المعلومات المرسلة										+

تحتوي ترويسة TCP على الحقول التالية:

- عنوان بوابة الوجهة: وتحتوي رقم التطبيق في طبقة التطبيق (Application layer) عند المستقبل، أي التطبيق الذي سيقوم بالتعامل مع البيانات في هذا الطرد.
- عنوان بوابة المصدر: وتحتوي رقم التطبيق الذي قام بإرسال هذه البيانات باستخدام بروتوكول TCP.
- الرقم التسلسلي: ويدل على رقم أول ثمانية في البيانات المرسلة في هذا الطرد.
- رقم التأكيد: ويدل على رقم الثمانية التي يقوم مرسل هذا الطرد بانتظار وصولها وجميع الثمانيات التي سبقتها قد وصلت (كون نوع إشارات التأكيد التي يستخدمها TCP إيجابية).
- حجم النافذة: وهو رقم يخبر به مرسل هذا الطرد الطرف الآخر بعدد الثمانيات التي يمكن لمرسل هذا الطرد أن يستقبلها ابتداءً من رقم التأكيد الذي يوجد في هذا الطرد (هذه تحدد حجم النافذة المترحلة عند مرسل هذا الطرد).
- محجوز: هو حقل غير مستخدم ولكنه سيستخدم في المستقبل.

- مقدار الإزاحة: ويحوي هذا الحقل على عدد البايتات التي توجد في ترويسة TCP، وتتراوح قيمته بين 5 إلى 15 كلمة لأنه يقدر بوحدة الكلمة وهي 4 بايتات أي 32 بت.
- أعلام مميزة:
  - ع1: (URG) وهو بت واحد يدل على أن البيانات التي توجد في هذا الطرد مستعجلة، ويجب على المستقبل أن يقرأ الرقم الذي يوجد في حقل «بيانات مستعجلة»، وإذا كان يحوي هذا البت القيمة صفر فهذا يدل على أن البيانات التي في هذا الطرد ليست مستعجلة وبالتالي لا يقوم المستقبل بقراءة الرقم الموجود في الحقل (بيانات مستعجلة).
  - ع2: (ACK) وهو بت واحد يدل على أن البيانات الموجودة في هذا الطرد تحتاج إلى تأكيد من قبل مستقبلها.
  - ع3: (PSH) ويدل هذا الحقل في حالة وضعه على القيمة 1 على أن البيانات التي توجد في هذا الطرد يجب أن يتم رفعها إلى الطبقات العليا بأسرع ما يمكن.
  - ع4: (RST) إذا كان على القيمة 1 يدل على أنه يجب إعادة تأسيس الاتصال. (Reset the connection)
  - ع5: (SYN) يحوي القيمة 1 عند فتح الرابطة لمساعد بعملية التزامن بين المرسل والمستقبل.
  - ع6: (FIN) لتدل على إغلاق الرابطة بشكل نظامي بين المرسل والمستقبل.
- بيانات مستعجلة: لا يقوم المستقبل بقراءة هذا الحقل إلا إذا كان العلم ع1 (URG) فعالاً (أي على القيمة 1)، ويدل على رقم آخر ثمانية في المعطيات المستعجلة مما يسمح للتطبيق بمعرفة حجم البيانات المستعجلة القادمة (أي يحوي على عدد الثمانية المستعجلة).
- مجموع الاختبار (Checksum) : وهو مجموع قيم البتات في الطرد وترويسة TCP، ويستخدم من أجل كشف الأخطاء، حيث يقوم المرسل بجمع هذه البتات وتخزين نتيجة الجمع في هذا الحقل، والمستقبل عند استقبال الطرد يقوم أيضاً بعملية الجمع نفسها لنفس العدد من البتات ويقارن بين القيمتين، وفي حال الاختلاف يكتشف وجود خطأ في الطرد المنقول ليطلب من المرسل بعدها أن يعيد إرساله إليه.

### 3-2- خواص بروتوكول TCP:

- موثوق : يهتم TCP بعمليات النقل الموثوق للبيانات بدون أخطاء، وفي حالة حصول أخطاء في النقل يقوم بإعادة إرسال البيانات الضائعة أو الخاطئة من المرسل مرة أخرى وهذا ما جعل استخدامه بالنقل ليس مناسباً لجميع التطبيقات لأنه يقوم بتضييع الكثير من الوقت في الانتظار وتصحيح أخطاء النقل بإعادة الإرسال، لذلك يتم استخدام بروتوكولات أخرى في التطبيقات التي تحتاج لنقل لحظي ومباشر للبيانات، مثل نقل الصوت عبر الآي بي (VoIP)، ولكنه جيد جداً ومفيد في التطبيقات التي تحتاج لبيانات دقيقة وصحيحة مثل نقل الملفات باستخدام بروتوكول نقل الملفات FTP
- يقيم رابطة (اتصال) بين المرسل والمستقبل قبل الشروع بالإرسال: يستخدم TCP تقنية المصافحة من أجل إقامة رابطة أو اتصال مع المستقبل، حيث يقوم بإرسال 3 طرود (3 إشارات مصافحة) بين كل من المرسل والمستقبل:



- 1 -يقوم المرسل بإرسال طرد طلب فتح رابطة إلى المستقبل.
  - 2 -يقوم المستقبل بتأكيد هذا الطلب والموافقة عليه (إن كان يستطيع فتح رابطة وليس مشغولاً بعمليات نقل أخرى مع غيره من العقد على الشبكة).
  - 3 -يرد المرسل على المستقبل بإشارة تأكيد أخيرة ليدل على أنه جاهز للإرسال. والفائدة من إقامة رابطة بين الطرفين هي أن يعلم المستقبل ترتيب الطرود الحقيقي الذي أرسل وفقه المرسل من خلال وضع ذاكرة عند المستقبل تحتفظ بالطرود التي استقبلها وفقاً لترقيم معين في كل طرد منها يدل على ترتيب الطرد بين العدد الكلي للطرود التي يرسلها المرسل، مما يساعد في كشف ضياع أو فقدان أحد الطرود من قبل المستقبل خلال النقل ليطلب الأخير من المرسل إعادة إرساله، هذا هو السبب الرئيس الذي جعل هذا البروتوكول موثوقاً.
- يقوم بترقيم ثمانيات البيانات بدلاً من ترقيم الطرود: يقوم TCP بترقيم تسلسلي للثمانيات المرسلة بدلاً من أن يقوم بترقيم الطرود المرسلة، فمثلاً إن كان يريد المرسل إرسال 10 طرود، وكل طرد يتكون من 100 ثمانية(Byte) ، فيقوم TCP بترقيم الثمانيات من الرقم 1 إلى الرقم 1000 ويوزع الثمانيات بالتساوي على الطرود العشرة، في حين بعض البروتوكولات الأخرى تقوم بترقيم الطرود العشرة من 1 إلى 10 بدلاً من ترقيم الثمانيات، ويقوم بإرفاق عدد الثمانيات وأرقامها في ترويسة من البتات تضاف مع البيانات المرسلة.
  - يستخدم إشارات التأكيد الإيجابية.
  - يستخدم طريقة إعادة الإرسال التراكمي.(Cumulative Retransmission or Go-Back-N Retransmission).
  - يعتمد وجود مؤقت من أجل إعادة الإرسال: ويوجد عند مرسل الطرد، ويفيد في حالة عدم وصول إشارة التأكيد إليه من الطرف المستقبل، ويعمل هذا المؤقت بإحدى الطريقتين التاليتين:
- 1 -يوجد لكل طرد مؤقت خاص فيه ويبدأ بالعمل فور إرسال الطرد ويتوقف عند استقبال إشارة التأكيد الخاصة بالطرد التالي لهذا الطرد ومشكلتها أنها طريقة مكلفة.
  - 2 -يتم تشغيل المؤقت فور استلام إشارة التأكيد الخاصة بأحد الطرود في أحد الأطراف وذلك لأن هذه الإشارة تدل على رقم الطرد المنتظر من الطرف المرسل لها.
- يعتمد طريقة النافذة المتزحقة من أجل التحكم بالدفق.
  - يعتمد على نافذة الاختناق لتجنب اختناق الشبكة. هناك عدة طرق لضبط نافذة الاختناق تعتمد على خوارزم التحكم في الاختناق المستخدم مثل الزيادة الخطية و النقصان الضريبي.

## القسم العملي:

سنقوم بإنشاء برنامج سيرفر مهمته إرسال جميع الملفات الموجودة ضمن مجاد معين إلى أي زبون يقوم بالاتصال به لذلك في البداية نستورد المكتبات اللازمة:

```
import socket
import threading
import os
import time
```

مكتبة socket من أجل إنشاء سوكيت السيرفر ومكتبة threading من أجل إنشاء thread خاصة بكل زبون يقوم بالاتصال لضمان تخدم أكثر من زبون في نفس الوقت ومكتبة os من أجل التعامل مع المجلدات والملفات الموجودة ضمن النظام ومكتبة time من أجل إضافة تأخير زمني بين عمليات إرسال الملفات لضمان عدم حدوث أخطاء.

```
host = "127.0.0.1"
port = 1234
path="E:\\programming network"
```

نعرف المتغيرات إنشاء السوكيت الخاصة بالسيرفر واستقبال الاتصالات:

نعرف التابع main الذي يحوي التعليمات التالية:

```
server_socket=socket.socket()
server_socket.setsockopt(socket.SOL_SOCKET,socket.SO_REUSEADDR,1)
server_socket.bind((host,port))
server_socket.listen(3)
print(f"Programming Network Course Files Server is Working at {host}:{port}")
```

في البداية تعليمة إنشاء السوكيت وهي عبارة عن اشتقاق غرض من الكلاس socket الموجود ضمن المكتبة socket والتعليمة التالية هي من أجل تحرير عنوان السوكيت في حال توقف البرنامج بشكل غير طبيعي وبعدها نقوم بربط السوكيت إلى العنوان ورقم المنفذ وذلك باستخدام التابع bind وسنضع السيرفر في حالة انتظار للاتصال باستخدام التابع listen.

```
while True:
    try:
        csock,caddr=server_socket.accept()
        print(f"[*] NEW CLIENT CONNECTED at{caddr}")
        files = os.listdir(path)
        csock.send("*SEP*".join(files).encode())

    th=threading.Thread(target=handle_file_send,args=(csock,caddr,files))
    th.start()
    except socket.error as err:
        print(err)
        break
server_socket.close()
```

نقوم بإنشاء حلقة لا نهائية من أجل استقبال أكثر من اتصال حيث أول خطوة هي قبول الاتصال باستخدام التابع Accept الذي يعيد غرض هو السوكيت الخاص بالزبون وعنوان هذا الزبون وبعدها نقوم بجلب جميع محتويات المجلد الخاص بالملفات ووضعهم ضمن list اسمها files باستخدام التابع listdir من المكتبة os وبعدها نقوم بإرسال أسماء الملفات إلى الزبون.

الخطوة التالية هي إنشاء thread خاصة بهذا الزبون وهذه ال thread تتعامل مع التابع handle\_file\_send الذي يأخذ بارامترات هي السوكيت الخاص بالزبون وعنوانه وقائمة الملفات. وبعدها نقوم بتشغيل هذه ال thread باستخدام التابع start ليتم نقل التنفيذ إلى thread جديدة يتم فيها تنفيذ هذا التابع.

قمنا بمعالجة الاستثناءات الخاصة بقبول الاتصال والارسال وفي حال الفشل يتم طباعة الخطأ لإنهاء تنفيذ حلقة while وإغلاق السوكيت الخاصة بالسيرفر.

التابع handle\_file\_send معرف كالتالي:

```
def handle_file_send(csock,caddr,files):
    for file in files:
        print(f"Sending to {caddr}>>>",file)
        filename = f"{path}\\{file}"
        filesize=os.path.getsize(filename)
        try:
            csock.send(str(filesize).encode())
            with open(filename,"rb") as f:
                bytes_read=f.read(filesize)
                time.sleep(1)
                csock.sendall(bytes_read)
        except socket.error as err:
            print(err)
    csock.close()
```

ضمن هذا التابع توجد حلقة for من أجل المرور على جميع الملفات حيث في كل مرة يتم أخذ اسم ملف موجود ضمن المجلد وبعدها إنشاء مسار الملف ووضعه ضمن متغير filename ونقوم بعدها بالحصول على حجم الملف باستخدام التابع getsize ووضعه ضمن المتغير filesize وقمنا بهذه العملية من أجل ضبط قيمة ال buffer الخاصة بتعليمة الاستقبال على حجم الملف من أجل ضمان استقبال كامل الملف وسنقوم بإرسال هذا الحجم إلى الزبون باستخدام التابع send وبعدها نقوم بفتح ملف بوضع القراءة كبايتات لنقوم بقراءة الملف وبعدها إرساله باستخدام التابع sendall وهذه العملية ستكرر بفضل حلقة for من أجل جميع ملفات المجلد.

قمنا بمعالجة الاستثناءات أيضاً من أجل عملية الإرسال.

بالنسبة للزبون:

```
import socket
import os
from colorama import Fore
```

نقوم في البداية باستيراد المكتبات وهنا لدينا المكتبة colorama من أجل إضافة ألوان إلى تعليمة الطباعة.

نعرف التابع main :

```
def main():
    c_socket=socket.socket()
    socket.timeout(5)
    c_socket.connect(("127.0.0.1",1234))
    files_names=c_socket.recv(1024).decode()
    files=files_names.split("*SEP*")
    print(files)
    try:
        os.mkdir("Received")
    except:
        pass
    for file in files:
        buffer_size=int(c_socket.recv(1024).decode())
        print("Recieving",Fore.RED,file,Fore.BLUE,
buffer_size/10**6,"MB",Fore.RESET)
        data=c_socket.recv(buffer_size)
        path=os.path.join("Received", file)
        with open(path,"wb") as f:
            f.write(data)
            f.close()
    print(Fore.GREEN,"All Files Recived")
    c_socket.close()
```

في البداية أنشأنا السوكيت الخاصة بالزبون وبعدها نضبط المؤقت الخاص بانتظار الاتصال على 5 ثواني ونقوم بالاتصال بالسيرفر.

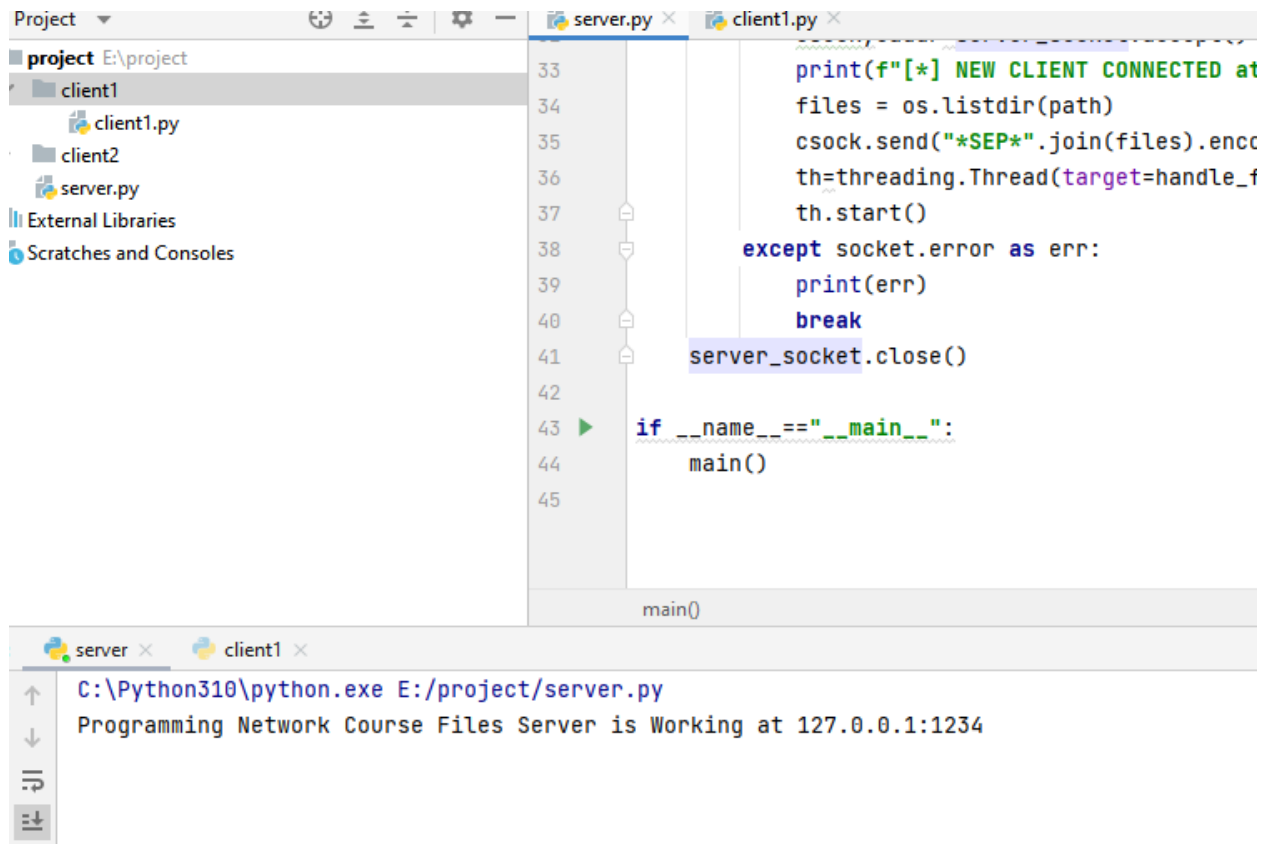
في البداية نقوم باستقبال أسماء الملفات وطباعتها.

ونقوم بإنشاء مجلد Received الذي سنضع فيه الملفات المستلمة عن طريق التابع mkdir من المكتبة os وقمنا بمعالجة الاستثناءات هنا لأن التابع سيعيد خطأ في حال كان المجلد موجود مسبقاً وبعدها نقوم بتعريف حلقة for من أجل استقبال جميع الملفات.

أول تعليمة هي لاستقبال حجم الملف من أجل ضبط buffer تابع الاستقبال recv وبعدها نقوم بفتح ملف في وضع الكتابة من أجل كتابة البايتات المستلمة الخاصة بالملف.

## النتائج والمناقشة:

نقوم في البداية بتشغيل السيرفر فيتنصت على العنوان 127.0.0.1 والمنفذ 1234.



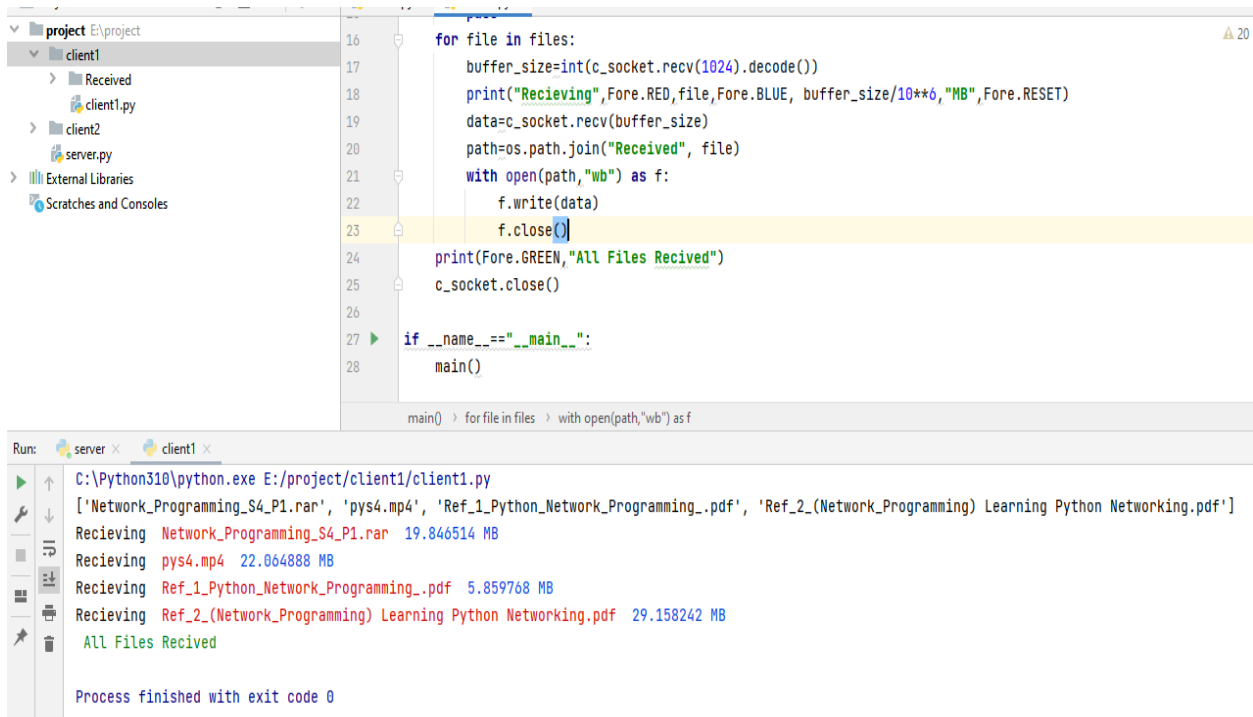
الشكل 2 تشغيل السيرفر

نرى محتويات المجلد الخاص بملفات المقرر التي سيرسلها السيرفر:

Organize		New	Open
is PC > Local Disk (E:) > programming network			
Name	Type	Size	
Ref_2_(Network_Programming) Learning ...	Adobe Acrobat D...	28,475 KB	
Ref_1_Python_Network_Programming_.pdf	Adobe Acrobat D...	5,723 KB	
Network_Programming_S4_P1.rar	WinRAR archive	19,382 KB	
pys4.mp4	MP4 Video File	21,548 KB	

الشكل 3 ملفات سيتم ارسالها

نقوم بتشغيل الزبون الأول فنلاحظ إنشاء مجلد ضمن مجلد الزبون يخوي على الملفات المستلمة:



The screenshot shows a code editor with a file explorer on the left and a console window at the bottom. The file explorer shows a project structure with a 'client1' folder containing 'Received' and 'client1.py'. The code editor shows the following Python code:

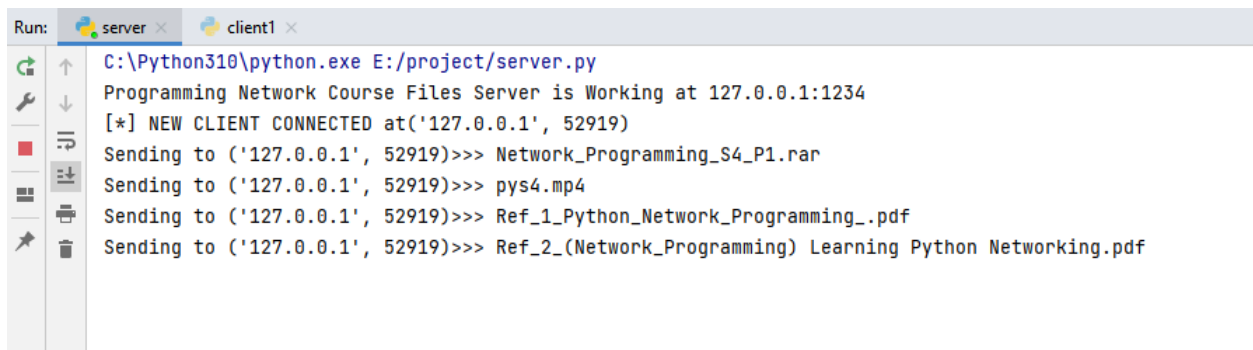
```
16 for file in files:
17     buffer_size=int(c_socket.recv(1024).decode())
18     print("Receiving",Fore.RED,file,Fore.BLUE, buffer_size/10**6,"MB",Fore.RESET)
19     data=c_socket.recv(buffer_size)
20     path=os.path.join("Received", file)
21     with open(path,"wb") as f:
22         f.write(data)
23     f.close()
24     print(Fore.GREEN,"All Files Recieved")
25     c_socket.close()
26
27 if __name__=="__main__":
28     main()
```

The console window shows the output of the client1.py script:

```
Run: server x client1 x
C:\Python310\python.exe E:/project/client1/client1.py
['Network_Programming_S4_P1.rar', 'pys4.mp4', 'Ref_1_Python_Network_Programming_.pdf', 'Ref_2_(Network_Programming) Learning Python Networking.pdf']
Receiving Network_Programming_S4_P1.rar 19.846514 MB
Receiving pys4.mp4 22.064888 MB
Receiving Ref_1_Python_Network_Programming_.pdf 5.859768 MB
Receiving Ref_2_(Network_Programming) Learning Python Networking.pdf 29.158242 MB
All Files Recieved
Process finished with exit code 0
```

الشكل 4 تشغيل أول زبون

يكون خرج السيرفر:



The screenshot shows a code editor with a file explorer on the left and a console window at the bottom. The file explorer shows a project structure with a 'server' folder containing 'server.py'. The code editor shows the following Python code:

```
16 for file in files:
17     buffer_size=int(c_socket.recv(1024).decode())
18     print("Receiving",Fore.RED,file,Fore.BLUE, buffer_size/10**6,"MB",Fore.RESET)
19     data=c_socket.recv(buffer_size)
20     path=os.path.join("Received", file)
21     with open(path,"wb") as f:
22         f.write(data)
23     f.close()
24     print(Fore.GREEN,"All Files Recieved")
25     c_socket.close()
26
27 if __name__=="__main__":
28     main()
```

The console window shows the output of the server.py script:

```
Run: server x client1 x
C:\Python310\python.exe E:/project/server.py
Programming Network Course Files Server is Working at 127.0.0.1:1234
[*] NEW CLIENT CONNECTED at('127.0.0.1', 52919)
Sending to ('127.0.0.1', 52919)>>> Network_Programming_S4_P1.rar
Sending to ('127.0.0.1', 52919)>>> pys4.mp4
Sending to ('127.0.0.1', 52919)>>> Ref_1_Python_Network_Programming_.pdf
Sending to ('127.0.0.1', 52919)>>> Ref_2_(Network_Programming) Learning Python Networking.pdf
```

الشكل 5 خرج السيرفر بعد اتصال زبون

في حال اتصال زبونين في نفس اللحظة يكون خرج السيرفر:

```
server x client1 x client2 x
[*] NEW CLIENT CONNECTED at('127.0.0.1', 52937)
Sending to ('127.0.0.1', 52937)>>> Network_Programming_S4_P1.rar
Sending to ('127.0.0.1', 52937)>>> pys4.mp4
Sending to ('127.0.0.1', 52937)>>> Ref_1_Python_Network_Programming_.pdf
[*] NEW CLIENT CONNECTED at('127.0.0.1', 52938)
Sending to ('127.0.0.1', 52938)>>> Network_Programming_S4_P1.rar
Sending to ('127.0.0.1', 52937)>>> Ref_2_(Network_Programming) Learning Python Networking.pdf
Sending to ('127.0.0.1', 52938)>>> pys4.mp4
Sending to ('127.0.0.1', 52938)>>> Ref_1_Python_Network_Programming_.pdf
Sending to ('127.0.0.1', 52938)>>> Ref_2_(Network_Programming) Learning Python Networking.pdf
```

الشكل 6 اتصال زبونين في نفس الوقت

## الاستنتاجات والتوصيات:

نسنتج أنه يمكن تحقيق برنامج ارسال ملفات عبر الشبكة دون استخدام بروتوكول نقل الملفات FTP أو غير وذلك فقط عن طريق مكتبة السوكيت باستخدام البروتوكول TCP كما يمكن في المستقبل تحويل هذا البرنامج من برنامج بث عام لملفات المقرر إلى برنامج يستطيع من خلاله الكلاينت طلب ملف محدد وتنزيله.

## المراجع:

- 1- <https://www.cisco.com/c/en/us/solutions/enterprise-networks/what-is-network-programming.html#~q-a>
- 2- <https://docs.python.org/3/library/socket.html>
- 3- <https://docs.python.org/3/howto/sockets.html>
- 4- <https://realpython.com/python-sockets/>
- 5- مهند عيسى، برمجة الشبكات اتصالات سنة خامسة، جامعة تشرين، 2022