جَـامعة
المَـنارة
MANARA UNIVERSITY

جامعة المنارة

كلية: الهندسة

قسم: المعلوماتية

اسم المقرر: نظم تشغيل ٢

رقم الجلسة (٨)

عنوان الجلسة

(برمجة المقابس باستخدام البروتوكول TCP)

الفصل الدراسي

الأول

العام الدراسي

٢٠٢٣_٢٠٢٤

جَـامعة
المَـنارة
MANARA UNIVERSITY

جدول المحتويات

Contents

الغاية من الجلسة:

تعريف الطلاب ب

- ✓ الربط بين العقد بواسطة المقابس باستخدام البروتوكول TCP
- ✓ انشاء مقبس من جهة المخدم يعمل كمستمع للطلبات القادمة من الزبائن
- ✓ انشاء مقبس من جهة طرقية يعمل كزبون يرسل الطلبات إلى المقيس من جهة الخادم

بروتوكول TCP :

هو معيار اتصال عالمي تستخدمه الأجهزة لنقل البيانات بشكل موثوق. يتم تعريف TCP على أنه موجه للاتصال، مما يعني أنه يجب إنشاء اتصال بين العميل والخادم قبل إرسال البيانات. وهذا يعني أن البيانات موثوقة ومرتبة ويتم التحقق من الأخطاء أثناء نقلها. إنه أحد البروتوكولات الرئيسية لمجموعة بروتوكولات الإنترنت - وغالبًا ما يشار إلى المجموعة بأكملها باسم TCP/IP.

إن إنشاء اتصال بين العميل والخادم باستخدام TCP يعتبر استخدام مناسب للتطبيقات التي تتطلب موثوقية عالية، يتم استخدام البروتوكول TCP مع بروتوكولات أخرى مثل HTTP، HTTPs، FTP، SMTP، Telnet. يقوم TCP بإعادة ترتيب حزم البيانات بالترتيب المحدد. هناك ضمان مطلق بأن البيانات المنقولة تظل سليمة وتصل بنفس الترتيب الذي تم إرسالها به.

المخطط المبين أدناه يظهر طبقات البروتوكول TCP/IP الأربعة مقارنة مع طبقات النظام المعياري OSI السبعة

7	**Application**	e.g. HTTP, SMTP, SNMP, FTP, Telnet, SSH and Scp, NFS, RTSP etc.
6	**Presentation**	e.g. XDR, ASN.1, SMB, AFP etc.
5	**Session**	e.g. TLS, SSH, ISO 8327 / CCITT X.225, RPC, NetBIOS, ASP etc.
4	**Transport**	e.g. TCP, UDP, RTP, SCTP, SPX, ATP etc.
3	**Network**	e.g. IP/IPv6, ICMP, IGMP, X.25, CLNP, ARP, RARP, BGP, OSPF, RIP, IPX, DDP etc.
2	**Data Link**	e.g. Ethernet, Token ring, PPP, HDLC, Frame relay, ISDN, ATM, 802.11 Wi-Fi, FDDI etc.
1	**Physical**	e.g. wire, radio, fiber optic etc.

بشكل عام، الطبقات الثلاث العليا لنموذج OSI (التطبيق والعرض التقديمي والجلسة) هي تعتبر طبقة تطبيق واحدة في مجموعة TCP/IP والطبقتين السفليتين أيضًا تعتبر طبقة وصول واحدة للشبكة. يوضح الشكل التالي بروتوكول TCP/IP

4	**Application layer**	BGP, FTP, HTTP, HTTPS, IMAP, IRC, NNTP, POP3, RTP, SIP, SMTP, SNMP, SSH, SSL, Telnet, UUCP, Finger, Gopher, DNS, RIP, Traceroute, Whois, IMAP/IMAP4, Ping, RADIUS, BGP etc.
3	**Transport layer**	DCCP, OSPF, SCTP, TCP, UDP, ICMP etc.
2	**Network/Internet layer**	IPv4, IPv6, ICMP, ARP, IGMP etc
1	**Physical/ Data Link layer**	Ethernet, Wireless (WAP, CDPD, 802.11, Wi-Fi), Token ring, FDDI, PPP, ISDN, Frame Relay, ATM, SONET/SDH, xDSL, SLIP etc. RS-232, EIA-422, RS-449, EIA-485 etc.

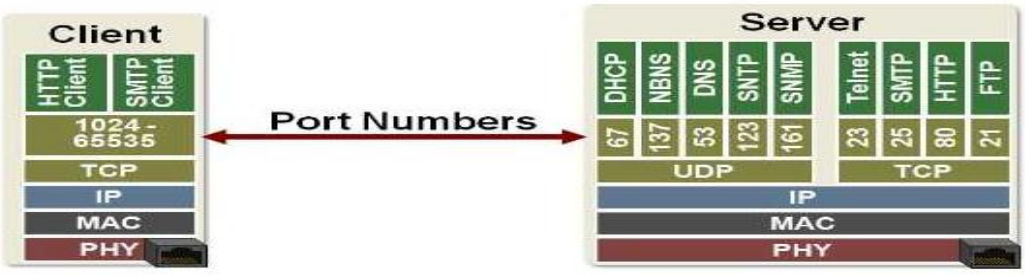
يقوم TCP بالتحكم في التدفق ويتطلب ثلاث حزم لإعداد اتصال مأخذ توصيل قبل إرسال أي بيانات

يعالج TCP الموثوقية والتحكم في الازدحام. كما يقوم أيضًا بفحص الأخطاء واسترداد الأخطاء. يتم إعادة إرسال الحزم الخاطئة من المصدر إلى الوجهة.

المنافذ ports :

تُستخدم المنافذ لتحديد العمليات التي يتم تشغيلها في التطبيقات الموجودة على المضيف.

لنفترض أن لدينا تطبيقين يعملان على جهاز كمبيوتر واحد ويتطلبان اتصالات TCP/IP. افترض أن أحدهما هو متصفح ويب والآخر هو عميل بريد إلكتروني. يرسل كلا التطبيقين ويستقبلان حزمًا بنفس عنوان IP، فكيف تفرق طبقة النقل بين حزمة متصفح الويب وحزمة البريد الإلكتروني؟ الجواب هو أرقام المنافذ.

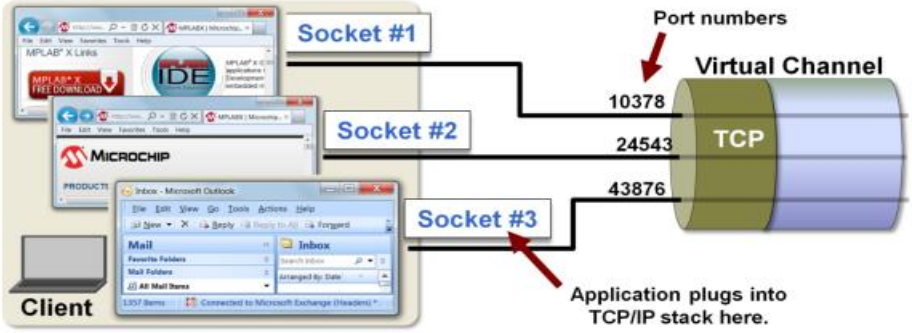


المقابس sockets :

هي واجهة برمجة تطبيقات قياسية (API) تسمح بتطبيقين أو أكثر (أو العمليات) للتواصل إما على نفس الجهاز أو عبر الشبكة وهي في المقام الأول مصممة لتعزيز اتصالات البيانات عبر الشبكة.

عندما يبدأ تطبيق على مضيف، يتم تعيين رقم منفذ لعملية أو وظيفة تعمل فيه. عندما يريد هذا التطبيق الاتصال بمضيف آخر، (انتقل إلى موقع ويب على سبيل المثال)، يتم إنشاء مأخذ توصيل.

يوضح هذا المثال ثلاثة تطبيقات تتطلب ثلاث قنوات اتصال TCP: قناتان لكل من متصفحي الويب اللذين يعملان كعملاء HTTP، وواحدة لتطبيق البريد الإلكتروني الذي يعمل كعميل SMTP.



Winsock هو واجهة برمجة شبكة يوفر Winsock واجهة البرمجة للتطبيقات للتواصل باستخدام بروتوكولات الشبكة الشائعة مثل التحكم في الإرسال البروتوكول/بروتوكول الإنترنت (TCP/IP) winsock ورثت قدرًا كبيرًا من تطبيق BSD على منصات UNIX. في بيئات Windows، تطورت الواجهة إلى واجهة مستقلة عن البروتوكول ، خاصة مع إصدار Winsock 2

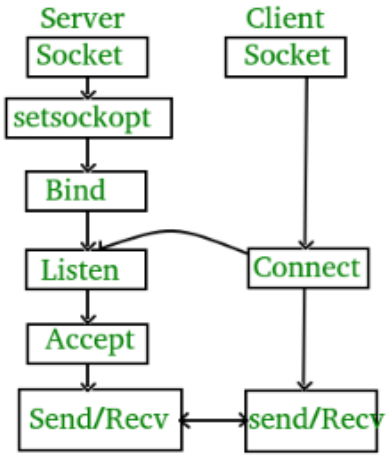
عملية الاتصال بين المخدم و الزبون :

تكون عملية الاتصال بين المخدم و الزبون كما هو مبين بالشكل التالي

- ✓ حيث يقوم المخدم بإنشاء المقبس TCP Socket و تحديد المنفذ
- ✓ Bind() للربط مع عنوان المخدم
- ✓ Listen() لوضع المقبس في حالة انتظار و الاستماع لطلبات الاتصال من الزبون
- ✓ Accept() اتمام الاتصال مع الزبون و اصبح الاتصال جاهز لتبادل البيانات
- ✓ العودة إلى Listen() بالنظار طلب اتصال جديد من زبون آخر

بالنسبة للزبون :

- ✓ إنشاء المقبس TCP Socket
- ✓ التوصيل بالخادم



كود التطبيق

و في ما يلي تطبيق باستخدام لغة ال C++ لبرمجة الاتصال بواسطة المقابس بين المخدم و الزبون

```
/*
    Bind socket to port 8888 on localhost
*/
#include<io.h>
#include<stdio.h>
#include<winsock2.h>

#pragma comment(lib,"ws2_32.lib") //Winsock Library

int main(int argc , char *argv[])
{
    WSADATA wsa;
    SOCKET s , new_socket;
    struct sockaddr_in server , client;
```

```
int c;
char *message;

printf("\nInitialising Winsock...");
if (WSAStartup(MAKEWORD(2,2),&wsa) != 0)
{
    printf("Failed. Error Code : %d",WSAGetLastError());
    return 1;
}

printf("Initialised.\n");

//Create a socket
if((s = socket(AF_INET , SOCK_STREAM , 0 )) == INVALID_SOCKET)
{
    printf("Could not create socket : %d" , WSAGetLastError());
}

printf("Socket created.\n");

//Prepare the sockaddr_in structure
server.sin_family = AF_INET;
server.sin_addr.s_addr = INADDR_ANY;
server.sin_port = htons( 8888 );

//Bind
if( bind(s ,(struct sockaddr *)&server , sizeof(server)) == SOCKET_ERROR)
{
    printf("Bind failed with error code : %d" , WSAGetLastError());
}

puts("Bind done");

//Listen to incoming connections
listen(s , 3);
```

```
    //Accept and incoming connection
    puts("Waiting for incoming connections...");

    c = sizeof(struct sockaddr_in);
    new_socket = accept(s , (struct sockaddr *)&client, &c);
    if (new_socket == INVALID_SOCKET)
    {
        printf("accept failed with error code : %d" , WSAGetLastError());
    }

    puts("Connection accepted");

    //Reply to client
    message = "Hello Client , I have received your connection. But I have to go now, bye\n";
    send(new_socket , message , strlen(message) , 0);

    getchar();

    closesocket(s);
    WSACleanup();

    return 0;
}
```

شرح الكود

تطبيق بسيطًا لخادم TCP باستخدام مكتبة Winsock على نظام Windows. يرتبط بالمنفذ 8888 على المضيف المحلي ويستمع للاتصالات الواردة. عند إنشاء الاتصال، يرسل رسالة إلى العميل ثم يغلق الاتصال.

دعنا نتصفح الكود خطوة بخطوة:

١. تضمين الرأس

```
#include<io.h>
 #include<stdio.h>
#include<winsock2.h>
```

ملفات االتضمين الضرورية لعمليات الإدخال/الإخراج، ووظائف الإدخال/الإخراج القياسية، ومكتبة Winsock (Windows Connectors).

٢. الربط مع مكتبة winsock

```
#pragma comment(lib,"ws2_32.lib")
```

يتم استخدام pragma هذا لربط البرنامج بمكتبة Winsock. يقوم بإعلام الرابط بإضافة مكتبة "ws2_32.lib" إلى قائمة المكتبات التي سيتم الارتباط بها.

٣. التابع الرئيسي

```
int main(int argc , char *argv[])
```

نقطة دخول البرنامج، الوظيفة الرئيسية، تأخذ وسيطات سطر الأوامر.

٤. تشغيل ال winsock

```
WSADATA wsa; if (WSAStartup(MAKEWORD(2,2),&wsa) != 0) { printf("Failed. Error Code : %d",WSAGetLastError()); return 1; }
```

يقوم هذا الجزء من الكزد بتهيئة مكتبة .Winsock إذا فشل، فإنه يطبع رسالة خطأ ويخرج من البرنامج .

٥. بناء المقبس socket

```
SOCKET s; if((s = socket(AF_INET , SOCK_STREAM , 0 )) == INVALID_SOCKET) { printf("Could not create socket : %d" , WSAGetLastError()); }
```

يقوم بإنشاء مقيس بالمعلمات المحددة (IPv4، TCP). إذا فشل إنشاء مأخذ التوصيل، فإنه يطبع رسالة خطأ.

٦. ربط المقبس مع عنوان الخادم

```
server.sin_family = AF_INET; server.sin_addr.s_addr = INADDR_ANY; server.sin_port = htons( 8888 ); if( bind(s ,(struct sockaddr *)&server , sizeof(server)) == SOCKET_ERROR) { printf("Bind failed with error code : %d" , WSAGetLastError()); }
```

يقوم بإعداد بنية عنوان الخادم وربط المقبس بالمنفذ المحدد (8888). إذا فشل الربط، فإنه يطبع رسالة خطأ.

٧. الاستماع للاتصالات القادمة

```
listen(s , 3);
```

تسمح وظيفة الاستماع للمقبس بالاستماع للاتصالات الواردة، وتحدد المعلمة ٣ الحد الأقصى لعدد الاتصالات التي يمكن وضعها في قائمة الانتظار.

٨. قبول الاتصالات من الزبائن

```
new_socket = accept(s , (struct sockaddr *)&client, &c); if (new_socket == INVALID_SOCKET) { printf("accept failed with error code : %d" , WSAGetLastError()); }
```

البرنامج ينتظر اتصالا واردا. عند إنشاء اتصال، فإنه يقبل الاتصال ويحصل على مقبس جديد للاتصال. إذا فشل القبول، فإنه يطبع رسالة خطأ.

٩. ارسال رسالة إلى الزبون

```
message = "Hello Client , I have received your connection. But I have to go now, bye\n"; send(new_socket , message , strlen(message) , 0);
```

يرسل رسالة بسيطة إلى العميل المتصل باستخدام وظيفة الإرسال.

١٠. التنظيف و الخروج

```
getchar(); closesocket(s); WSACleanup(); return 0;
```

ينتظر البرنامج الضغط على المفتاح، ويغلق المقبس، ثم يقوم بتنظيف مكتبة Winsock قبل الخروج.

يعد هذا الرمز مثالًا أساسيًا لخادم TCP، وهو يوضح الخطوات الأساسية لتهيئة Winsock، وإنشاء مأخذ توصيل، وربطه بعنوان، والاستماع للاتصالات، وقبول الاتصالات، وأخيرًا، إرسال البيانات.

آلية العمل

عند التنفيذ الخرج عند المخدم سيكون

```
Initialising Winsock...Initialised.

Socket created.

Bind done

Waiting for incoming connections...
```

الآن أصبح خادم المقبس جاهزًا وينتظر الاتصال الوارد. في هذه المرحلة نحتاج إلى الاتصال به باستخدام عميل مثل telnet.

```
C:\>ncat localhost 8888
```

بمجرد اتصال برنامج العميل بالخادم، حاول إرسال بعض الرسائل عن طريق الكتابة أولاً ثم الضغط على زر الإدخال.
سوف يقوم الخادم بالرد بنفس الرسالة