Tps Protocoles TCP et UDP

La suite de protocoles TCP/IP est formée de trois protocoles:

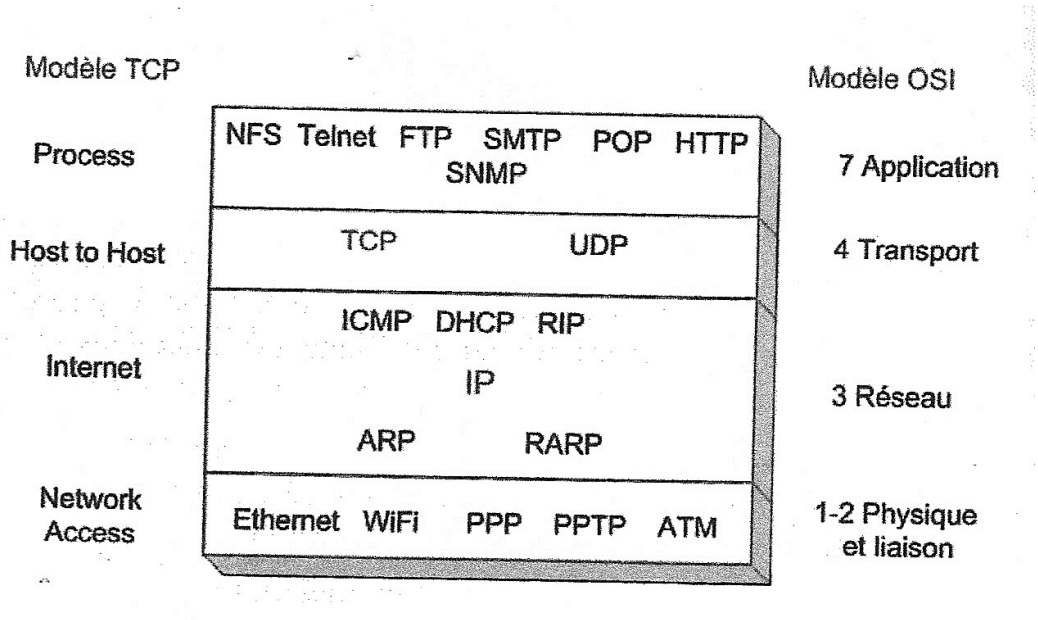
- IP (Internet Protocol) comprend les règles d'adressage des machines (les adresses IP) et les règles de routage des paquets (les données à transmettre sont découpées en blocs appelés paquets et ceux-ci sont transmis sous contrôle du protocole IP et indépendamment les uns des autres.

- TCP (Transmission Control Protocol) s'occupe du réassemblage correct des différents paquets d'un message. En cas de problème (paquet manquant ou en erreur), TCP réclame une réexpédition des paquets. **C'est un mode de fonctionnement dit "connecté"**.

- UDP (User Datagram Protocol) est un protocole plus simple que le précédent et donc plus rapide. UDP s'occupe du réassemblage des paquets mais sans jamais demander de retransmission. Même en cas d'erreur, il ne réclame pas le renvoi des paquets marqués en erreur. UDP convient parfaitement dans le cas de la diffusion audio ou vidéo en temps réel.

  Dans le cas d'une telle diffusion, réclamer la réexpédition d'un paquet perturberait bien plus l'audition ou la visualisation que le paquet en erreur lui-même.

  C'est un mode de fonctionnement dit "non connecté".



Protocole TCP: les classes TCPClient et TcpListener

I) Présentation.

Pour simplifier la communication entre deux applications réseau, l'architecture .net fournit les classes TcpClient et TcpListener.

Ces classes sont plus simples que la classe Socket, et en général elles suffisent pour accéder à une application distante via le protocole TCP/IP.

La classe TcpListener est utilisée coté serveur, la classe TcpClient coté client.

Les données sont envoyées et reçues via un objet NetWorkStream. Cette classe est dérivée de Stream, elle fournit donc les méthodes Read et Write de cette classe.

II) La classe TcpClient

La classe TcpClient fournit des méthodes simples de connexion, d'envoi et de réception de flux de données sur un réseau en mode blocage synchrone.

Afin que TcpClient puisse se connecter et échanger des données, un TcpListener ou un Socket créé à l'aide du Protocole TCP doit être à l'écoute des demandes de connexion entrantes. Il est possible de se connecter à cet "écouteur" des deux manières suivantes :

Créer TcpClient et appeler une des trois méthodes Connect disponibles.

Créer TcpClient à l'aide du nom d'hôte et du numéro de port de l'hôte distant. Ce constructeur tentera automatiquement d'établir une connexion.

Classe `TcpClient`	
`TcpClient ← Object`	
`using System.Net.Sockets;`	
Constructeurs de la classe `TcpClient`	
`TcpClient();`	Constructeur sans argument. Il faudra exécuter `Connect` pour spécifier le programme serveur (adresse IP et numéro de port).
`TcpClient(IPEndPoint);`	Le serveur (adresse IP et numéro de port) est spécifié dans un objet `IPEndPoint`.
`TcpClient(string, int);`	Le serveur est spécifié via son nom (par exemple `www.xyz.com`) et son numéro de port. En interne, les constructeurs avec arguments appellent le constructeur sans argument puis `Connect`.
Méthodes de la classe `TcpClient`	
`void Connect(IPEndPoint);`	Réalise une connexion au serveur. L'exception `SocketException` est levée en cas d'erreur lors de la connexion au serveur.
`void Connect(IPAddress, int);`	
`void Connect(string, int);`	
`NetworkStream GetStream();`	Fournit un objet `NetworkStream` pour envoyer des données au serveur et en recevoir.
`void Close();`	Met fin à la communication avec le serveur.

III) La classe TcpListener

La classe TcpListener fournit des méthodes simples qui écoutent et acceptent les demandes de connexion entrante en mode blocage synchrone.

On peut utiliser TcpClient ou Socket pour se connecter à TcpListener.

On crée un objet TcpListener en utilisant:

- Un objet IPEndPoint,
- une adresse IP locale et un numéro de port,
- ou seulement un numéro de port.

Spécifiez le champ Any pour l'adresse IP locale et 0 pour le numéro de port local si vous souhaitez que le fournisseur de services sous-jacent assigne ces valeurs pour vous. Si vous choisissez de le faire, vous pouvez utiliser la propriété LocalEndpoint pour identifier les informations assignées une fois le socket connecté.

Utilisez la méthode Start pour commencer à écouter les demandes de connexion entrante. La méthode Start met en file d'attente les connexions entrantes jusqu'à ce que vous appeliez la méthode Stop ou que le champ MaxConnections ait été mis en file d'attente.

Utilisez la méthode AcceptSocket ou AcceptTcpClient pour extraire une connexion en provenance de la file d'attente des demandes de connexion entrante. Ces deux méthodes se bloquent. Si vous voulez éviter le blocage, vous pouvez utiliser préalablement la méthode Pending pour déterminer si les demandes de connexion sont disponibles dans la file d'attente.

Appelez la méthode Stop pour fermer TcpListener.

Remarque: MaxConnections est un membre de l'énumération SocketOptionName. L'énumération SocketOptionName définit chaque option de configuration Socket. Les sockets peuvent être configurés à l'aide de la méthode Socket.SetSocketOption.

Classe `TcpListener`	
`TcpListener ← Object`	
`using System.Net.Sockets;`	
Constructeur de la classe `TcpListener`	
`TcpListener(int port);`	Constructeur dans lequel le serveur indique le numéro de port qu'il veut utiliser. Si la valeur zéro est passée en argument, un numéro de port sera choisi au hasard.
Méthodes de la classe `TcpClient`	
`Socket AcceptSocket();`	Se met en attente d'une demande de connexion émanant d'un client. `Start` doit avoir été exécuté au préalable.
`bool Pending();`	Indique s'il y a au moins une requête de connexion en attente.
`void Start();`	Démarre l'écoute de clients.
`void Stop();`	Met fin à l'écoute de clients.

IV) TP protocole TCP

Réaliser une application cliente (Windows form) et une application serveur (console) à l'aide ce ces classes. Le serveur donne l'heure au client lorsque celui-ci se connecte au serveur**.**

V) La classe NetworkStream

La classe NetworkStream fournit les méthodes pour l'envoi et la réception de données via des sockets Stream en mode blocage.

Vous pouvez utiliser la classe NetworkStream à la fois pour les transferts de données synchrones et asynchrones.

Pour créer NetworkStream, vous devez fournir un **Socket connecté**. Vous pouvez également spécifier l'autorisation FileAccess dont dispose NetworkStream sur le Socket fourni. Par défaut, la fermeture de NetworkStream ne ferme pas le Socket fourni. Si vous voulez que NetworkStream ait l'autorisation de fermer le Socket fourni, vous devez spécifier true comme valeur du paramètre ownsSocket.

Utilisez les méthodes Write et Read pour l'E/S bloquante synchrone à thread unique. Si vous voulez traiter votre E/S avec des threads séparés, préférez l'utilisation des méthodes BeginWrite et EndWrite, ou des méthodes BeginRead et EndRead pour la communication.

NetworkStream ne prend pas en charge l'accès aléatoire au flux de données réseau. La valeur de la propriété CanSeek qui indique si le flux prend en charge la recherche est toujours false ; la lecture de la propriété Position, de la propriété Length ou l'appel à la méthode Seek lève NotSupportedException.

Tp Protocole UDP: la classe UDPClient

I) Voir pdf "ExtraitMSDNClasseUDPClient.pdf"

II) TP protocole UDP

On veut espionner (visualiser) les commandes émises par un logiciel client qui émet des trames ascii à intervalles réguliers sur le réseau.

- Réaliser en mode console le serveur qui affiche les commandes émises par le client. Ce serveur utilisera la classe UDPClient.

- Réaliser en mode graphique le client qui émet à intervalles réguliers les trames ascii. Ce client utilisera la classe SOCKET paramétrée pour le protocole UDP.