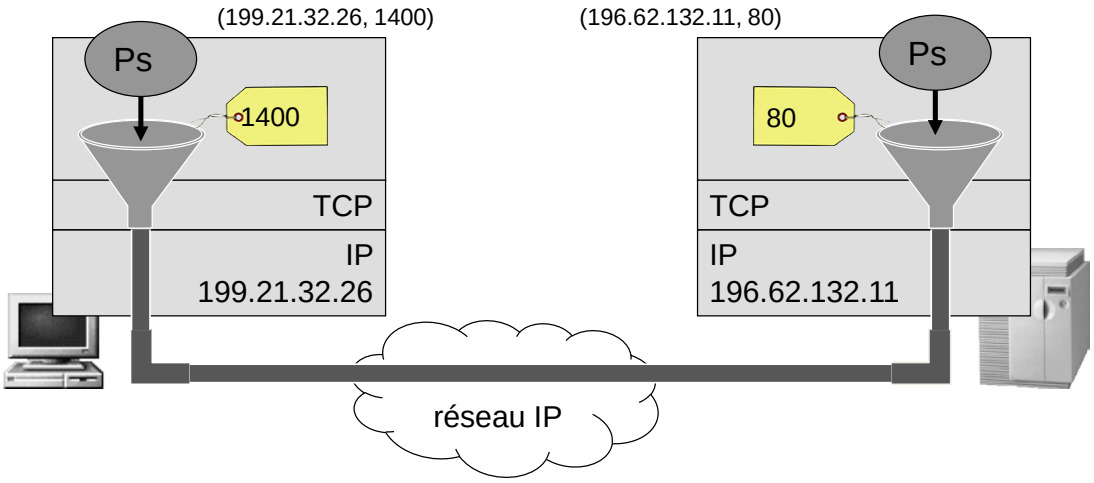
# 3 Les protocoles UDP et TCP

TCP comme UDP s'exécute au-dessus de IP et se fonde sur les services fournis par ce dernier.

- TCP (*Transport Control Protocol*) assure un service de transmission de données **fiable** avec une **détection et** une **correction d'erreurs** de bout en bout.

- UDP (*User Datagram Protocol*) offre un service de transmission de **datagrammes sans connection**.

Avec TCP ou UDP, il est possible de remettre des données à des processus d'application s'exécutant sur une machine distante. Ces processus d'application sont identifiés par numéros de port. Une *socket* (historiquement développé dans Unix BSD) est un point de communication par lequel un processus peut émettre et recevoir des informations. C'est la combinaison d'une adresse IP et d'un numéro de port. La combinaison de 2 sockets définit complètement une connexion TCP ou un échange UDP.
La RFC 1060 indique les ports prédéfinis pour les services :
- 20 : FTP
- 23 : Telnet
- 25 : SMTP
- 53 : DNS
- 69 : TFTP
- 80 : HTTP
- …




## 1. Protocole UDP : User Datagram Protocol

Le protocole UDP permet aux applications d'accéder directement à un service de transmission de datagrammes, tel que le service de transmission qu'offre IP.

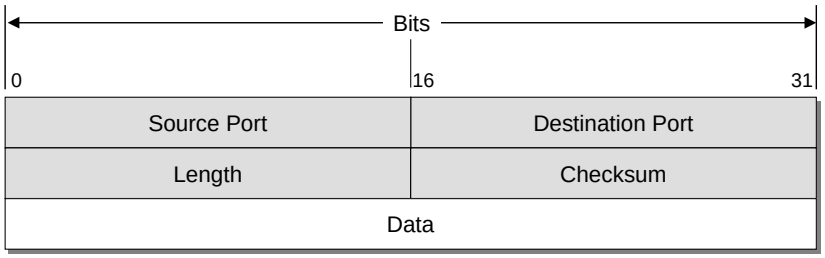
Caractéristiques de UDP :

- UDP possède un mécanisme permettant d'identifier les processus d'application à l'aide de numéros de port UDP.
- UDP est orienté datagrammes (sans connexion), ce qui évite les problèmes liés à l'ouverture, au maintien et à la fermeture des connexions.
- UDP est efficace pour les applications en diffusion/multidiffusion. Les applications satisfaisant à un modèle du type « interrogation-réponse » peuvent également utiliser UDP. La réponse peut être utilisée comme étant un accusé de réception positif à l'interrogation. Si une réponse n'est pas reçue dans un certain intervalle de temps, l'application envoie simplement une autre interrogation.
- UDP ne séquence pas les données. La remise conforme des données n'est pas garantie.
- UDP peut éventuellement vérifier l'intégrité des données (et des données seulement) avec un total de contrôle.
- UDP est plus rapide, plus simple et plus efficace que TCP mais il est moins robuste.

Le protocole UDP permet une transmission sans connexion, mais aussi sans sécurité. Pourtant de nombreuses applications reposent sur UDP :

- TFTP
- DNS
- NFS
- SNMP
- RIP

L'en-tête a une taille fixe de 8 octets.


Le champ *Source Port* occupe 16 bits. Il indique :

- le numéro de port du processus émetteur,
- le numéro de port où on peut adresser les réponses lorsque l'on ne dispose d'aucun autre renseignement.
- si sa valeur est 0, cela signifie qu'aucun numéro de port n'est attribué.

Le champ *Destination Port* identifie le processus correspondant à l'adresse IP de destination auquel on envoie les données UDP. UDP effectue le démultiplexage des données à l'aide de numéros de port. Lorsque UDP reçoit un datagramme sans numéro de port, il génère un message d'erreur ICMP indiquant qu'il est impossible de contacter le port et il rejette le datagramme.

Le champ *Length* contient la longueur du paquet UDP en octets (en-tête + données). La valeur minimale est 8 et correspond à un paquet où le champ de données est vide.

Le pseudo en-tête de préfixe de l'en-tête UDP contient l'adresse d'origine, l'adresse de destination, le protocole (UDP = 17) et la longueur UDP. Ces informations sont destinées à prévenir les erreurs de routage.

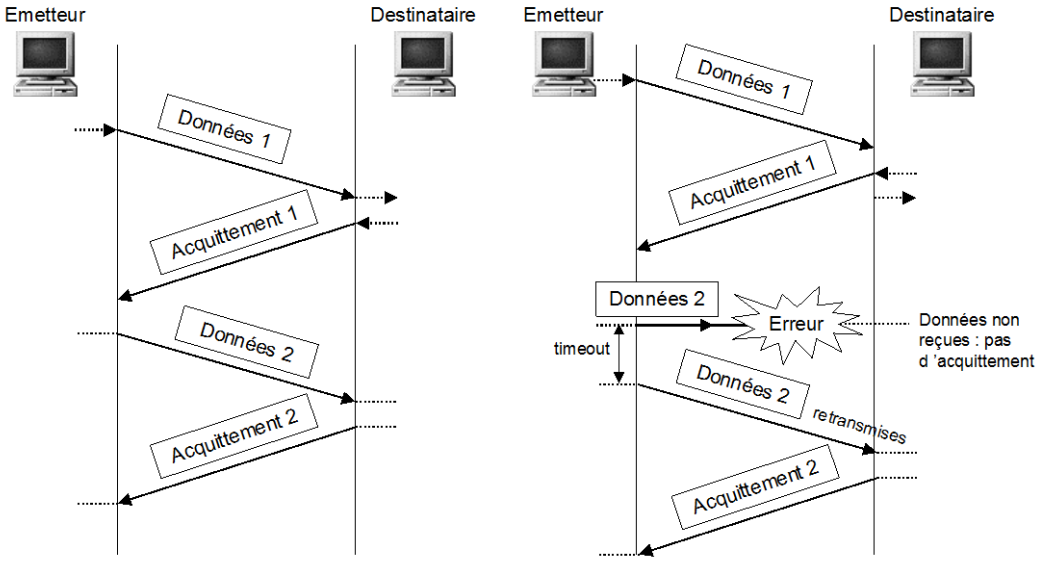
# 2. Protocole TCP : Transport Control Protocol

Les applications utilisent TCP pour garantir une transmission des données fiable. TCP est un protocole **fiable**, **orienté connexion**, à **flot d'octets**..

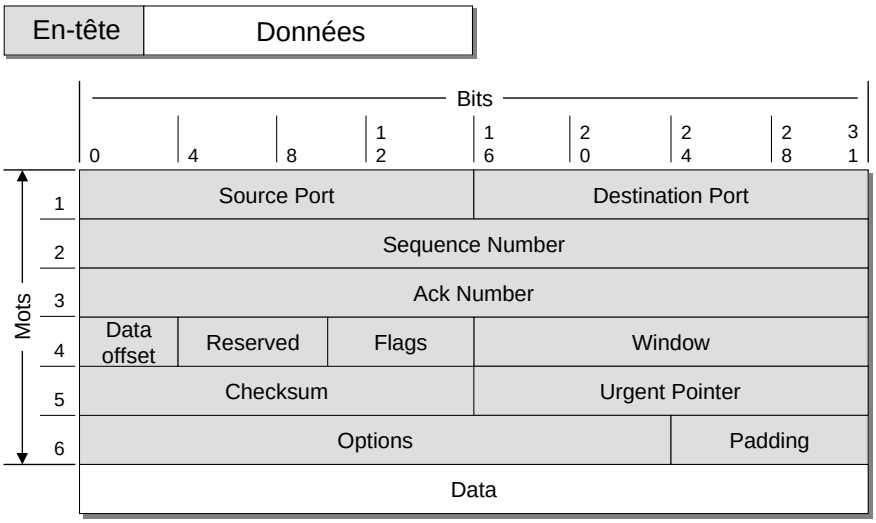
## 2.1. Fiabilité

L'utilisation d'un mécanisme appelé PAR (*Positive Acknowlegment with Retransmission,* Accusé de réception positif avec la retransmission) permet à TCP de garantir des transmissions fiables.

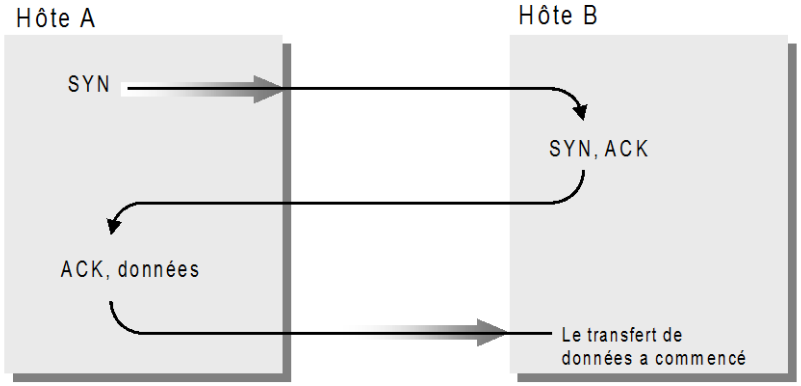
Un système utilisant PAR envoie à nouveau les données, à moins que le système à distance ne lui renvoie un message précisant que les données sont arrivées correctement.




L'unité utilisée pour l'échange de données entre modules TCP coopérants est appelée *segment*. Chaque segment contient un total de contrôle que le destinataire utilise pour vérifier que les données n'ont pas été endommagées pendant leur transmission. Si le segment de données est reçu en parfait état, le récepteur renvoie *un accusé de réception positif* à l'émetteur. Dans la négative, le récepteur élimine ce segment de données. Après un délais d'attente déterminé, le module TCP d'envoi retransmet les segments pour lesquels aucun accusé de réception positif n'a été reçu.




TCP est un protocole orienté connexion. Il établit une **connexion logique** de bout en bout entre deux machines-hôtes communicantes. Une information de contrôle, appelée *handshake* (« poignée de main »), est échangée entre les deux extrémités de manière à établir un dialogue avant de procéder à la transmission des données. TCP active la fonction de contrôle d'un segment en mettant à 1 le bit approprié du champ *Flags* du 4$^{e}$ mot de l'en-tête du segment.




TCP utilise un mécanisme *de poignée de main* mettant en œuvre trois échanges de segment : « *three-way handshake* ».

La machine-hôte A établit la connexion en envoyant d'abord à la machine-hôte B un segment contenant la séquence de bits « numéro de la séquence de synchronisation » (SYN). Ce segment indique à la machine-hôte B que A souhaite établir une connexion et lui précise le numéro de séquence que A va utiliser comme numéro d'initialisation de la transmission de ses segments. (les numéros de séquence sont utilisés pour conserver l'ordre des données). La machine-hôte répond à A en lui renvoyant un segment contenant l'accusé de réception (ACK) et les bits SYN. Le segment de B accuse réception de la bonne transmission du segment de B et transmet alors les premières données.

Une fois cet échange terminé, le TCP de la machine-hôte A constate que le TCP à distance est actif et est prêt à recevoir les données. Après avoir établi la connexion, les données sont transmises. Dès que les transferts de données entre modules coopératifs sont terminés, ils

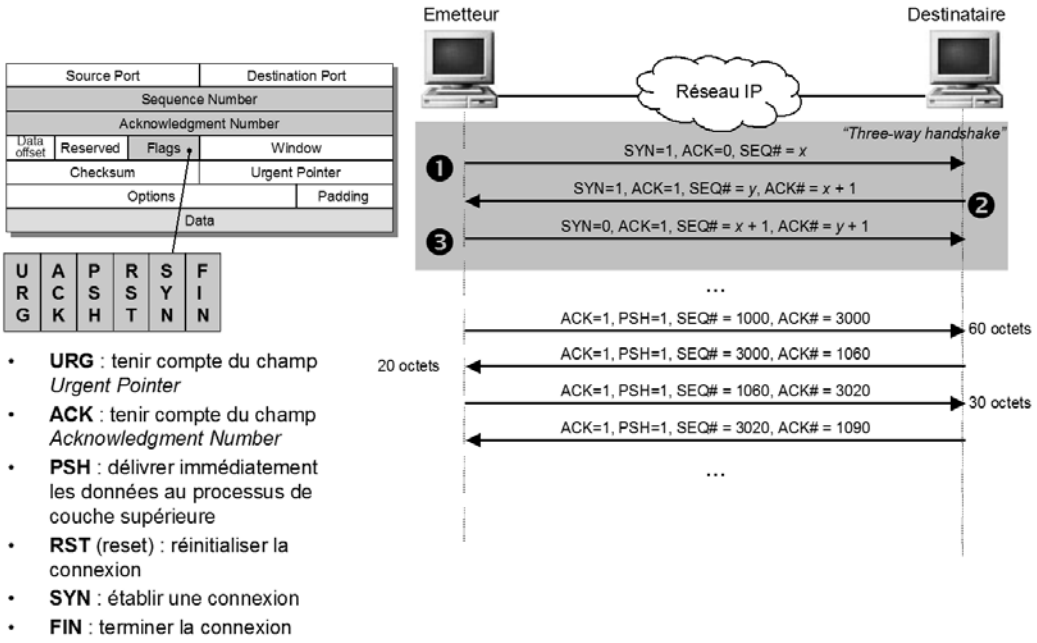
échangent alors une poignée de main de type Three-way handshake dont les segments contiennent le bit « plus aucune donnée à transmettre » (appelé le bit FIN) pour indiquer la fin de la connexion. L'échange de données bout en bout assure la connexion logique entre les deux systèmes.

## 2.2. Transfert de données de base

TCP se représente les données qu'il envoie sous la forme d'un **flot continu d'octets** et non sous la forme de paquets indépendants. Par conséquent, TCP veille à maintenir l'ordre dans lequel les octets ont été transmis et reçus.

Les champs *Sequence Number* et *Ack Number* de l'en-tête du segment TCP prennent en charge le suivi des octets.

Le standard TCP n'exige pas que les octets de numérotation de départ de chaque système correspondent à un nombre déterminé ; chaque système choisit le numéro qu'il va utiliser comme élément d'initialisation. Pour garantir un suivi correct du flot de données, il convient que chaque extrémité de la connexion connaisse le numéro d'initialisation de l'autre extrémité. Les deux extrémités de la connexion synchronisent les systèmes de numérotation au niveau de l'octet en échangeant les segments SYN pendant « la poignée de main ». Le champ *Sequence Number* du segment SYN contient le *numéro de séquence initiale* (ISN), qui correspond au numéro d'initialisation du système de numérotation des octets (en standard, ISN est égal à 0).




Chaque octet de données est numéroté séquentiellement à partir du numéro ISN, de sorte que le numéro de séquence du premier octet de données envoyé est le suivant : ISN + 1 (généralement 1).

Le numéro de séquence contenu dans l'en-tête d'un segment de données permet d'identifier la position séquentielle du premier octet de données d'un segment appartenant à un flot de données déterminé.
Exemple : si le premier octet du flot de données portait le numéro de séquence 1 (ISN=0) et que 4000 octets de données ont déjà été transférés, la longueur du premier octet de données du segment courant est alors 4001 octets et le numéro de séquence correspond à 4001.

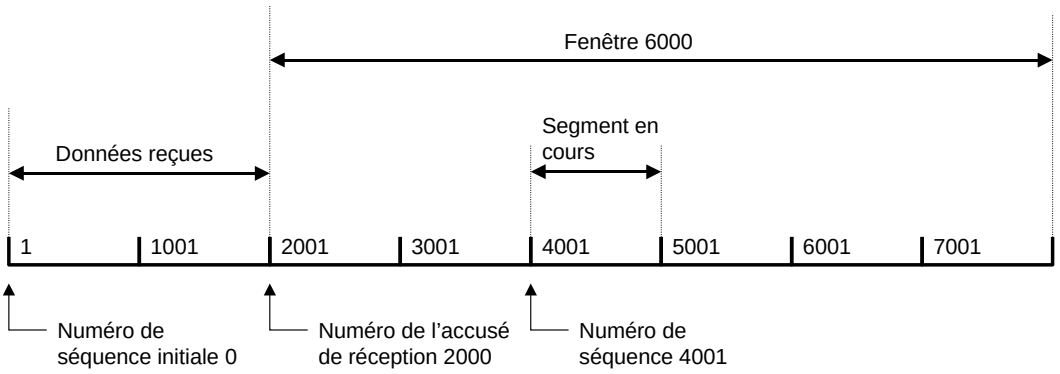
Le segment d'accusé de réception (ACK) exécute deux fonctions - l'accusé de réception positif et le contrôle de flux.

L'accusé de réception indique à l'émetteur le nombre d'octets de données déjà reçus ainsi que le nombre d'octets de données que peut encore recevoir le récepteur. Le numéro de l'accusé de réception correspond au numéro de séquence du dernier octet reçu à l'extrémité à distance. Le protocole standard ne requiert pas l'envoi d'un accusé de réception pour chaque paquet. Le numéro de l'accusé de réception correspond à un accusé de réception positif dont le numéro est égal au nombre d'octets transmis. Exemple : si le numéro du premier octet transmis est 1 et que 2000 octets ont été reçus correctement, le numéro de l'accusé de réception est : 2000.

## 2.3. Contrôle de flux

Les machines qui émettent et reçoivent des segments de données TCP ne le font pas toutes au même rythme. Il peut donc arriver que l'émetteur envoie ses données beaucoup plus rapidement que le récepteur ne peut les gérer. C'est pourquoi TCP implémente un mécanisme de contrôle de flux de données.

Le champ *Window* contient le nombre d'octets que l'extrémité à distance peut recevoir. Si le récepteur peut recevoir encore 6000 octets, le champ *Window* est alors égal à 6000 octets. Ce champ indique à l'émetteur qu'il peut poursuivre l'envoi de segments tant que le nombre total d'octets envoyés est inférieur au nombre d'octets que peut accepter le récepteur. Celui-ci procède au contrôle du flux d'octets provenant de l'émetteur en modifiant la taille de la fenêtre. Une fenêtre de taille nulle indique à l'émetteur d'interrompre la transmission jusqu'à ce qu'il reçoive une valeur de fenêtre non nulle.




L'émetteur envoie un flot de données TCP commençant par un numéro de séquence d'initialisation égal à 0. Le récepteur a reçu 2000 octets et en a accusé réception, de sorte que le numéro effectif de l'accusé de réception est 2000.

Le récepteur dispose également d'un espace en mémoire tampon suffisant pour stocker 6000 octets de plus ; il a dès lors inséré une fenêtre correspondant à 6000 octets. L'émetteur transmet alors un segment de 1000 octets commençant par le numéro de séquence 4001. L'émetteur n'a reçu aucun accusé de réception pour les octets situés au-delà de 2001. Toutefois, il continue d'envoyer des données tant que la fenêtre peut les accepter. Si l'émetteur remplit la fenêtre et ne reçoit aucun accusé de réception concernant les données précédemment envoyées, il renvoie, après un certain délai, les données à partir d'un premier octet sans accusé de réception.

La retransmission devrait commencer à partir de l'octet 2001, pour autant qu'aucun accusé de réception n'ait été reçu entre-temps. Cette procédure garantit que l'extrémité à distance du réseau a réceptionné les données transmises.

## 2.4. Muliplexage

TCP peut servir simultanément à plusieurs processus de la même machine par multiplexage. Ces processus communiquent par la même interface réseau et partagent donc la même adresse IP que l'interface réseau. TCP associe un numéro de port à chaque application qui utilise ses services.

Une connexion s'établit entre le numéro de port de l'émetteur et celui du récepteur. On appelle cela les extrémités (end points) de la connexion : une extrémité est définie par une paire de valeurs : l'adresse IP et le numéro de port. Les numéros des *ports source* et de *destination* sont contenus dans le premier mot de l'en-tête du segment.

TCP peut de plus transmettre les données sur une connexion dans un sens ou dans l'autre (connexion full duplex).

Périodiquement une RFC décrit les numéros de ports usuels. Les numéros compris entre 0 et 1023 sont réservés et sont les numéros de port usuels.