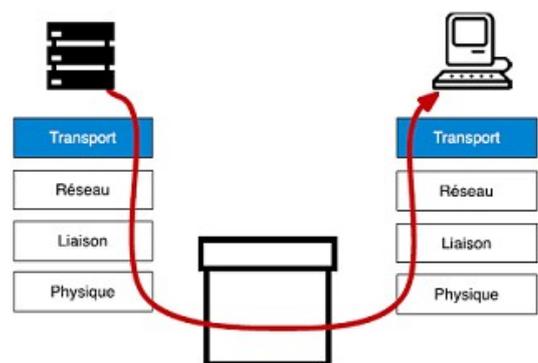


La couche transport

Table des matières

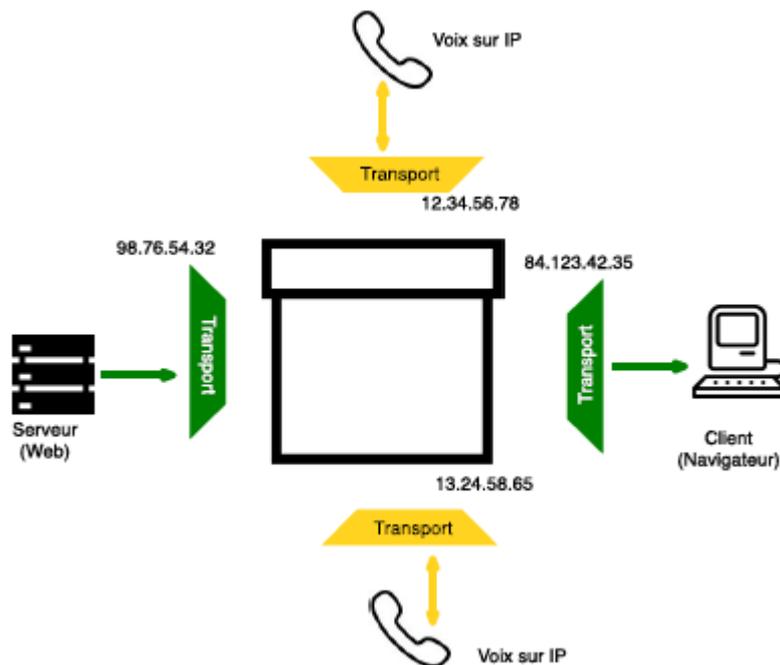
1. Le transport de bout-en bout.....	2
2. Le transport face aux erreurs de transmission.....	5
3. Session TCP : démarrer et terminer un dialogue.....	9

La couche transport se positionne au dessus de la couche réseau pour fournir aux applications une interface, un canal de communication, qui leur permettra de communiquer entre elles, sans se préoccuper du fonctionnement exact du réseau.



1. Le transport de bout-en bout

Une application désirant communiquer avec une autre application distante va demander au système d'exploitation qui l'héberge d'ouvrir ce canal de communication à destination de la machine qui accueille l'application réceptrice. Le système d'exploitation va alors se charger de contacter cette machine, identifiée par son adresse IP et d'ouvrir ce canal de communication sur lequel les applications pourront alors échanger des données.

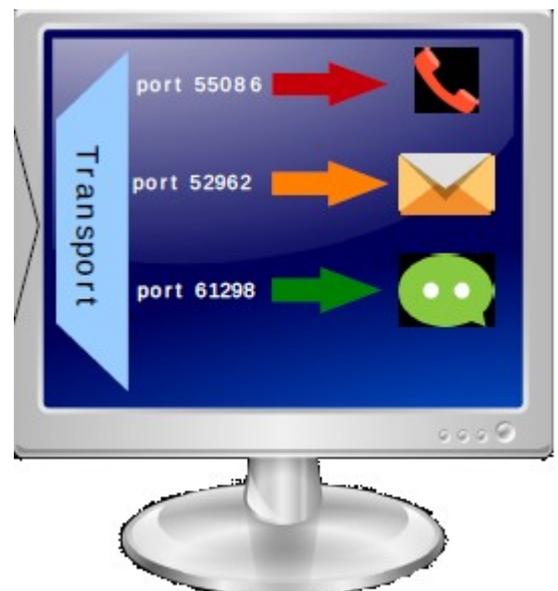


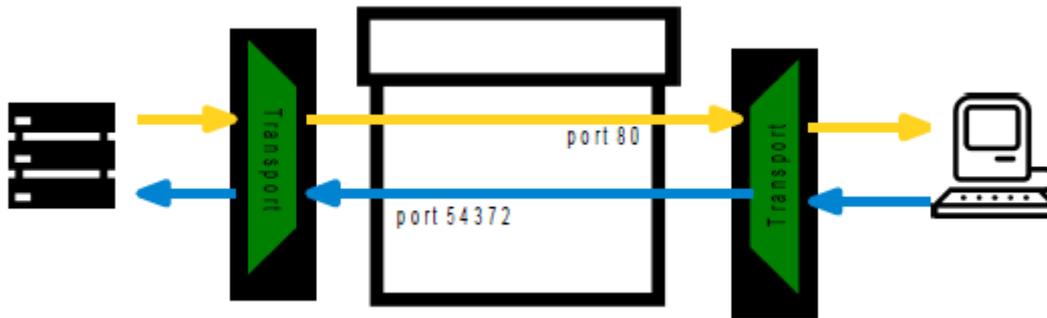
Pour permettre à plusieurs applications de coexister sur la même machine, la couche transport introduit un niveau d'adressage supplémentaire qu'on appelle le **numéro de port**. Le numéro de port est un simple numéro, sur 16 bits, qui vient s'ajouter à l'adresse IP, permettant ainsi d'identifier une application fonctionnant sur une machine.

Le système d'exploitation va allouer à une application un ou plusieurs numéros de port de façon exclusive. Aucune autre application ne pourra utiliser le même numéro de port au même moment.

Une application serveur, dont le fonctionnement classique est d'attendre les connexions des applications clientes, va réserver son ou ses numéros de port au moment de son lancement. Une application cliente, dont le fonctionnement est plus dynamique, va réserver son numéro de port au moment où elle fera la demande de connexion vers le serveur distant qu'elle cherche à joindre.

Pour contacter son correspondant, une application va donc envoyer une requête à destination d'un numéro de port prédéterminé. Sa requête contiendra par ailleurs un numéro de port source qui lui servira à recevoir les réponses.

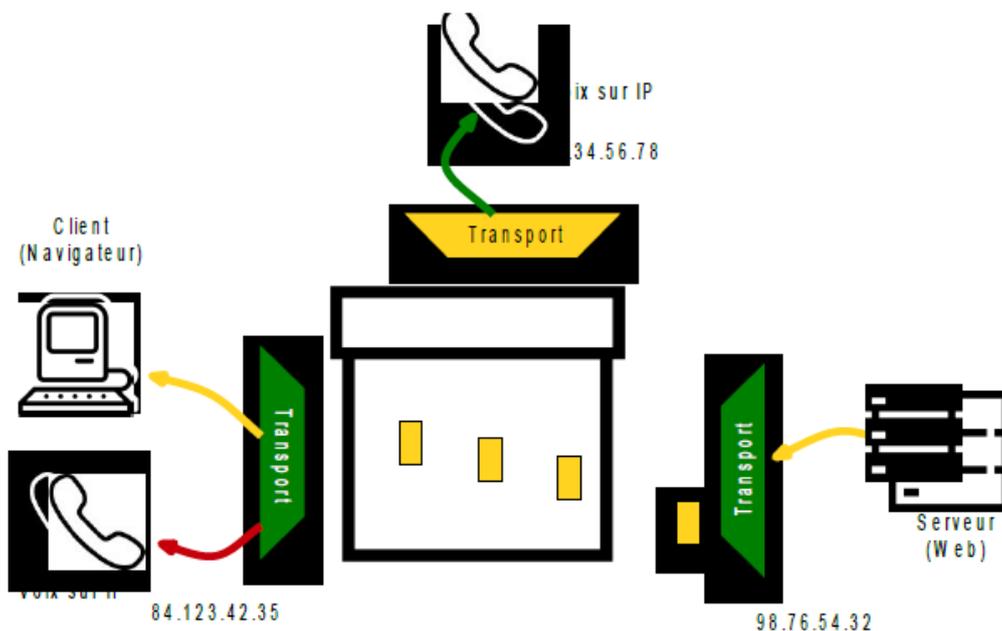




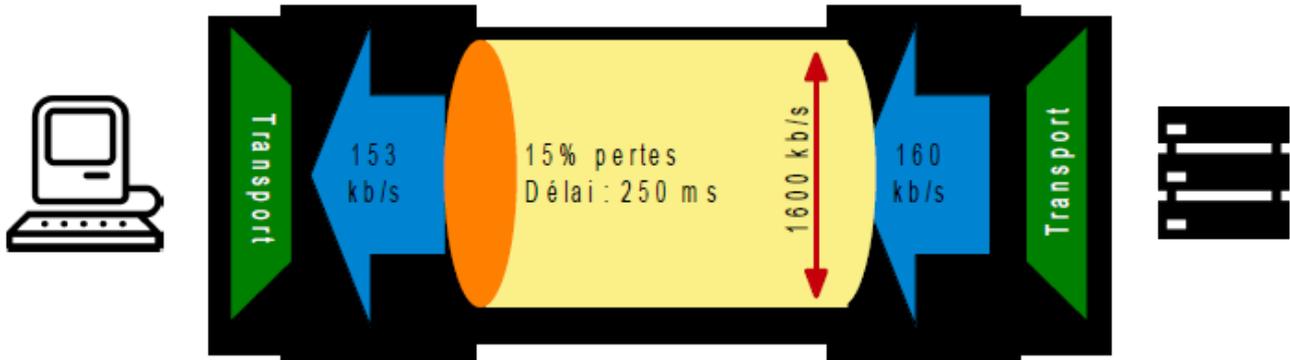
Les numéros de port de 0 à 1023 sont généralement utilisés pour les applications très largement répandues telles que les serveurs web, les serveurs FTP (File Transfert Protocol), les serveurs e-mail, etc. Les numéros de port de 1024 à 49151 sont destinés à des applications qui possèdent une, deux ou assez peu d'implémentations et sont déclarées en général auprès d'un organisme central qui s'appelle l'IANA (Internet Assigned Numbers Authority). Les numéros de port de 49152 à 65535 sont réservés à l'usage privé et sont en général utilisés pour recevoir les réponses aux requêtes que l'on envoie.

Numéro de port	Usage	Exemples
Well-known (1—1023)	Services et protocoles standard	Web (HTTP) : 80 e-mail (SMTP) : 25 session (SSH) : 22 DNS : 53
Enregistrés (1024—49151)	Applications déclarées à l'IANA	Bittorrent : 6881 HTTP proxy : 8080 Quake server : 26000
Dynamiques, privés (49152—65535)	Aucune réservation possible	

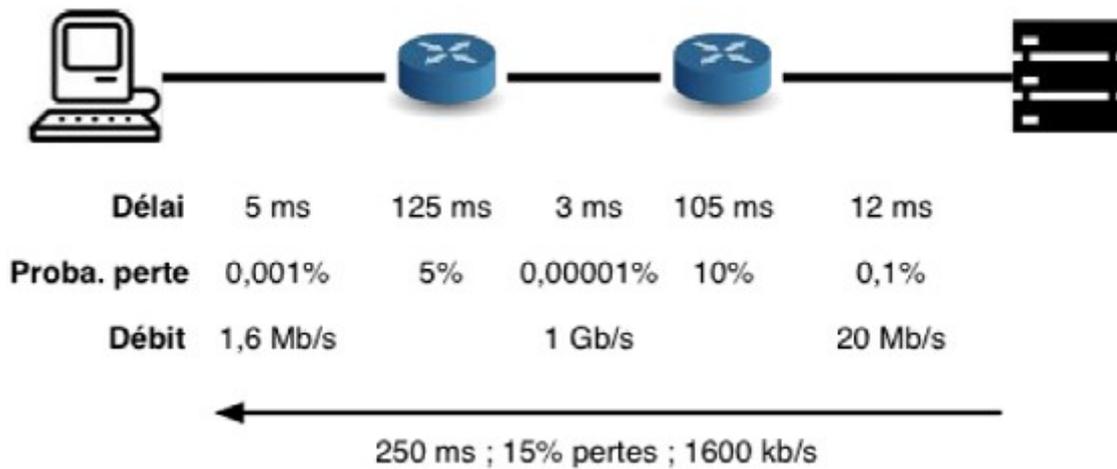
La couche transport va recevoir de la part des applications des données de taille arbitraire. Elle va se charger de découper ces données en blocs d'information de taille conforme à ce que peut acheminer le réseau. Ces blocs d'information se nomment **segments ou datagrammes** selon le mode utilisé. La couche transport réceptrice va recevoir ces segments ou ces datagrammes et va les réassembler afin de fournir à l'application réceptrice un flux d'informations cohérent.



La couche transport crée donc entre deux applications un canal de communication que l'on peut voir comme un tuyau possédant certaines propriétés en termes de débit, de délai, de taux de pertes, etc. Ces propriétés dépendent du fonctionnement exact du réseau.

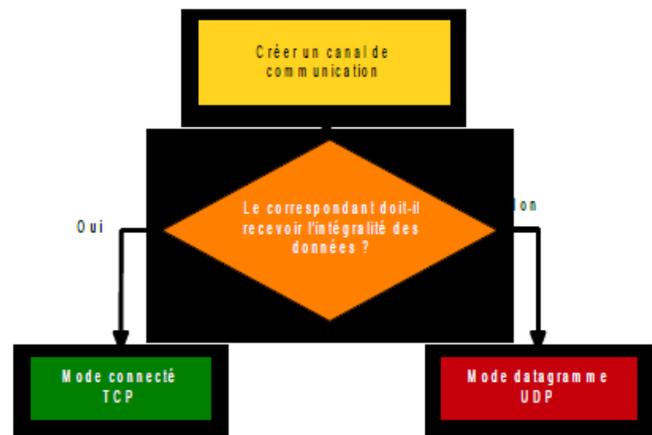


La longueur des chemins empruntés, par exemple, va influencer le délai de bout en bout. Les différentes couches liaison successives traversées vont influencer la probabilité de perdre un segment. Une application ne peut pas contrôler ou exiger de la part du réseau un certain niveau de performance, en termes de débit ou de délai en tout cas. En revanche, en sélectionnant le mode de fonctionnement de la couche transport, elle peut demander à la couche transport d'assurer un certain niveau de fiabilité sur la communication.



Certaines applications ont besoin de s'assurer que l'intégralité des données qu'elles envoient soit bien reçue par leur correspondant. Ces applications vont utiliser un mode de communication que l'on qualifie de mode connecté et un protocole de transport qui s'appelle TCP (Transmission Control Protocol). Ce faisant, elles vont demander au système d'exploitation qui les accueille de s'assurer de la bonne réception de l'intégralité des segments envoyés. Le système d'exploitation va, si nécessaire, provoquer des retransmissions des segments qui seraient perdus en cours de route.

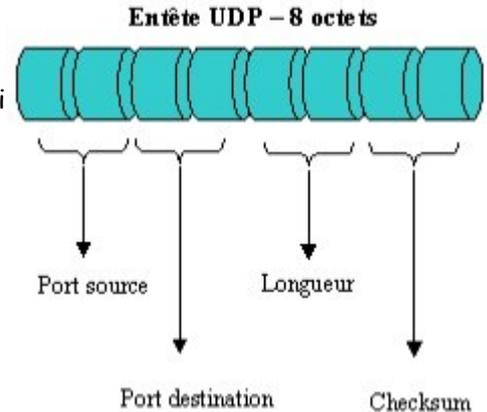
Cependant, ce n'est pas le cas de toutes les applications. En effet, pour certaines applications, notamment pour les applications multimédia telles que la vidéoconférence, la voix sur IP ou encore la diffusion audio ou vidéo, une perte d'information se traduit généralement par une baisse momentanée



de la qualité. Ces applications préféreront souvent tolérer une telle baisse de qualité plutôt que d'attendre une retransmission d'un segment perdu. En effet, retransmettre un segment perdu prend du temps et ce délai supplémentaire introduit engendre généralement une pause dans la diffusion du flux.

Ces applications n'ont pas besoin que le système d'exploitation se charge d'assurer une transmission fiable à 100%. Elles utiliseront un mode de communication que l'on nomme **datagramme** et sélectionneront un protocole de transport qui s'appelle **UDP** (User Datagram Protocol).

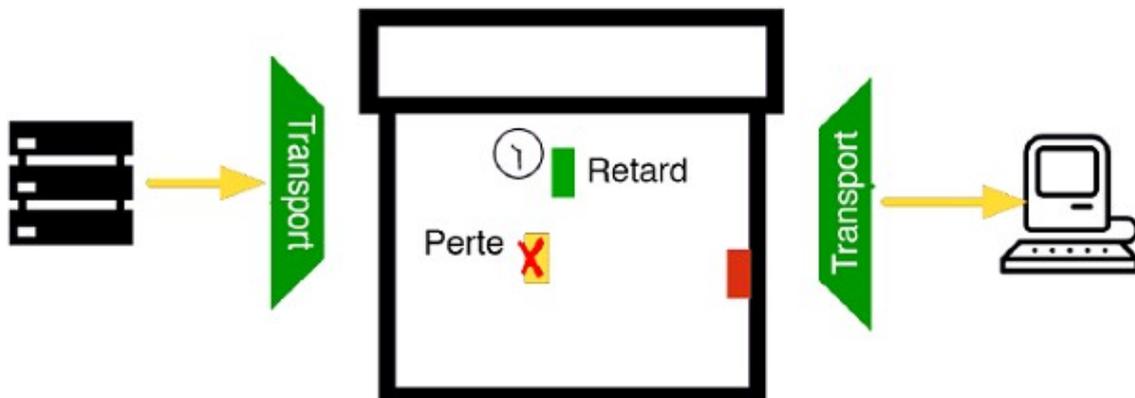
UDP est un protocole de transport minimaliste. Il n'ajoute aux données transmises par les applications que 4 champs. Les deux premiers champs sont les numéros de port source et destination qui identifient l'application émettrice et l'application réceptrice. UDP ajoute en outre la longueur des données transmises ainsi qu'une somme de contrôle qui permet de vérifier l'intégrité des données. En effet, si **UDP ne provoque aucune retransmission en cas d'erreur**, il ne transmet pas aux applications non plus de données erronées.



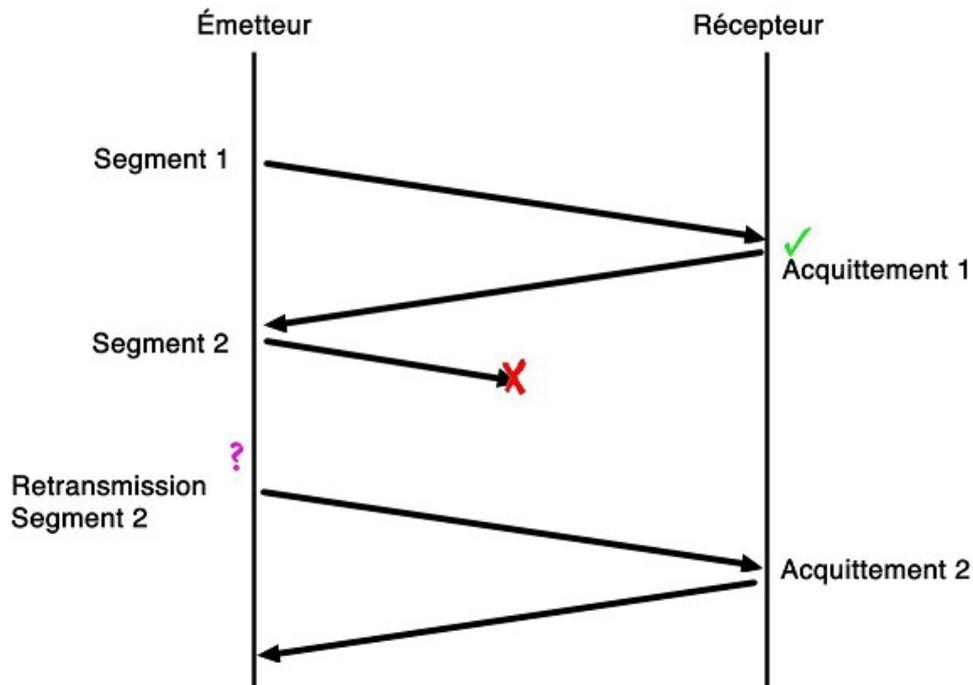
À l'inverse, TCP fournit aux applications une transmission fiable en utilisant une des techniques les plus classiques qui soit : la destination acquittera chacun des segments reçus correctement, et la source, en l'absence de réception d'un acquittement au bout d'un certain temps, provoquera une retransmission.

2. Le transport face aux erreurs de transmission

TCP assure aux applications une transmission fiable et ordonnée, ce qui veut dire que TCP va mettre en œuvre les mécanismes nécessaires à compenser les pertes et les retards introduits par le passage des segments dans le réseau. Les segments peuvent être perdus pour diverses raisons : des erreurs de transmission peuvent survenir sur les liens de communication et, le plus souvent, des paquets vont être perdus dans les files d'attente des routeurs pour des questions de surcharge. TCP va alors s'assurer que l'application émettrice détecte ces pertes de paquets et assure la retransmission des paquets puisqu'elle est la seule à disposer encore et avec certitude de la donnée.



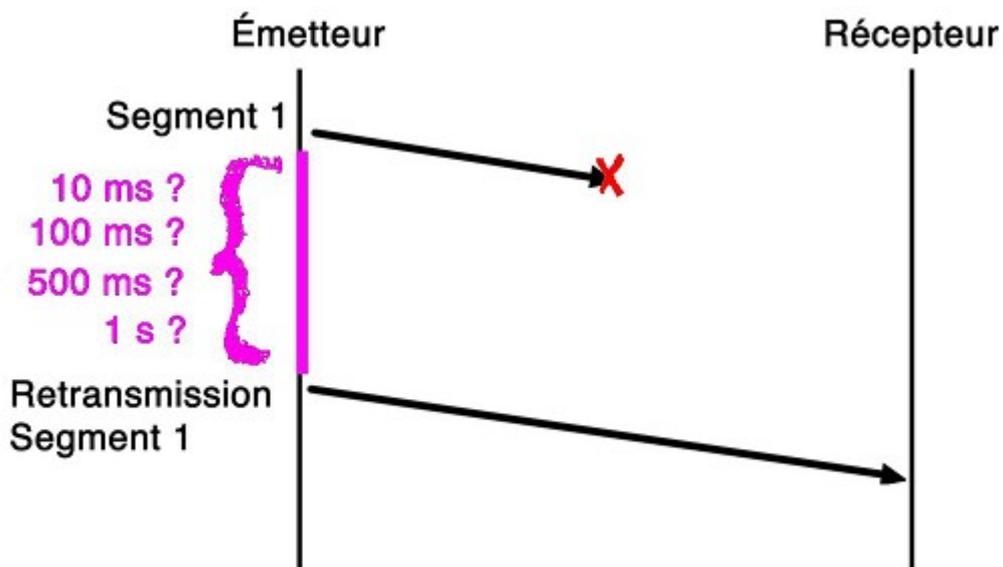
Le récepteur, quant-à-lui, a pour tâche d'acquitter chaque segment de donnée qui lui parvient correctement, c'est-à-dire sans erreur. Il va, à cet effet, envoyer à l'émetteur des acquittements qui sont des paquets IP comme les autres, qui vont transiter par le même réseau que le segment initial. L'émetteur, en l'absence de réception d'un tel acquittement, va conclure que le segment initial est perdu et provoquer une retransmission au bout d'un certain temps.



Le récepteur acquitte donc chaque segment de données reçu correctement. Cette stratégie peut sembler inefficace car il peut sembler plus économe de ne demander explicitement la retransmission que des segments manquants ou erronés. Cependant, ces demandes de retransmission sont des paquets comme les autres et transitent par un réseau imparfait et vont donc être soumis, eux aussi, à des pertes et des retards. Par conséquent, l'utilisation d'acquittements positifs est la seule façon de s'assurer d'une fiabilité à 100% de la réception de l'intégralité des données, même si cette stratégie est moins économe.

Pour pouvoir conclure au succès ou à l'échec d'une transmission, l'émetteur devra estimer le temps au bout duquel il devrait recevoir l'acquittement.

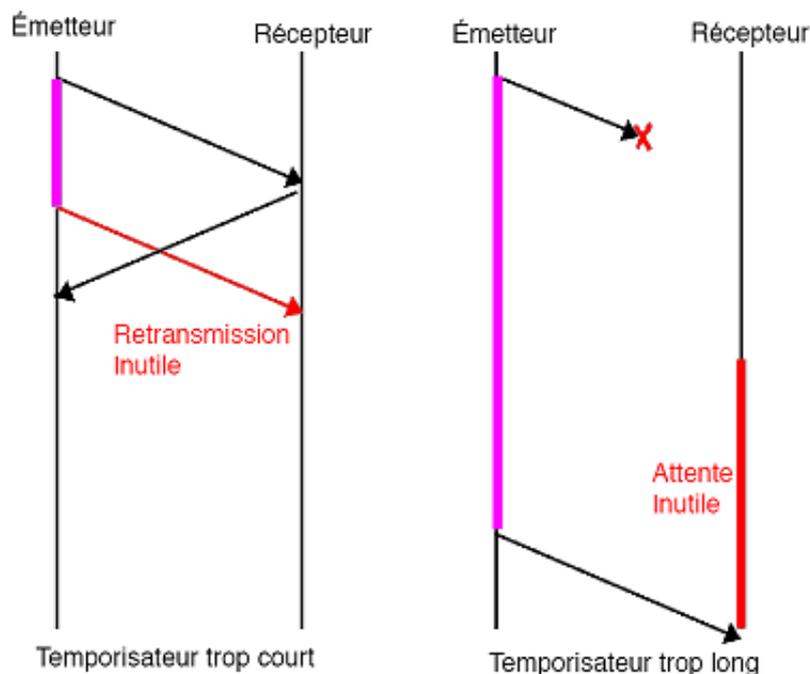
Cependant, ceci n'est pas chose facile dans l'Internet, car contrairement à la couche liaison, il n'est pas possible d'utiliser un temporisateur fixe car le récepteur peut aussi bien être très proche de l'émetteur que très éloigné.



Exemples de délais (depuis Paris)

- Institut Mines-Télécom : 25 ms (0 km)
- MIT (Boston, MA) : 20 ms (5600 km)
- Stanford (Californie) : 180 ms (9000 km)
- IIT Delhi (Inde) : 250 ms (6600 km)
- Tokyo Institute of Technology : 350 ms (9700 km)
- UFRGS (Porto Alegre, Brésil) : 310 ms (10200 km)

Un bon dimensionnement du temporisateur qui définit au bout de combien de temps l'émetteur doit recevoir l'acquittement est primordial pour assurer une bonne performance du réseau. En effet, un dimensionnement trop court de ce temporisateur, comme sur l'exemple de la figure de gauche, conduit à provoquer des retransmissions inutiles. En effet, ici, l'acquittement arrive après qu'a été prise la décision de retransmettre un paquet. À l'inverse, si le temporisateur est trop long, comme sur la figure de droite, l'émetteur patientera très longtemps avant de prendre la décision de retransmettre un paquet qu'il aurait du retransmettre. C'est donc le récepteur qui patientera jusqu'à la bonne transmission de ce paquet.

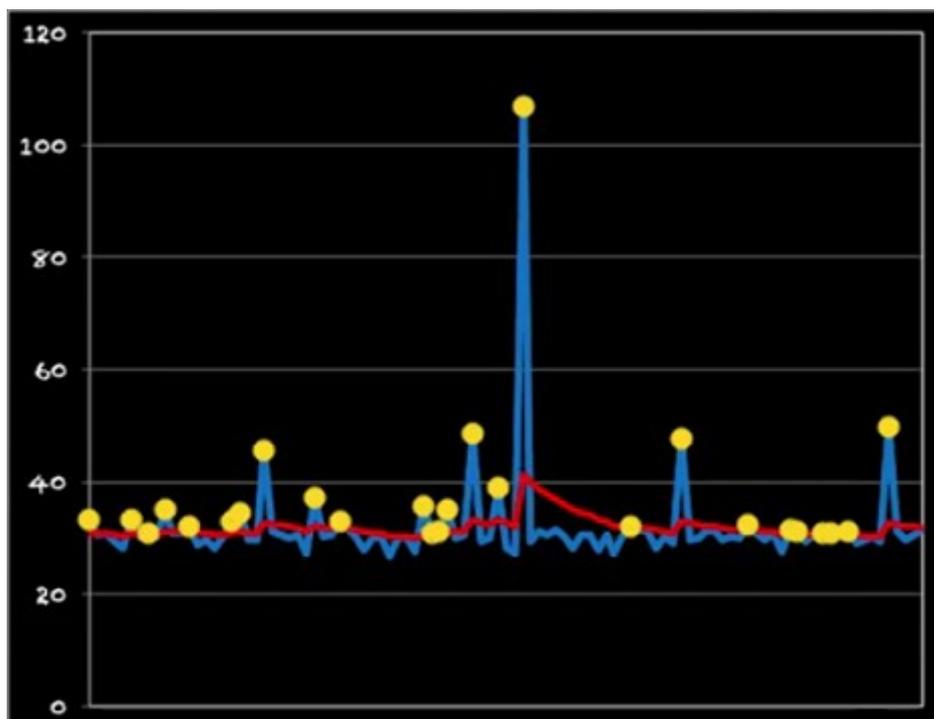


Dans un réseau, une source et une destination peuvent être très proches comme très lointaines. Par conséquent, le temps nécessaire à acheminer un paquet de la source à la destination et à acheminer l'acquittement correspondant de la destination à la source peut être de l'ordre de quelques millisecondes, comme de l'ordre de quelques secondes. Dans ces conditions, il est impossible de dimensionner un temporisateur de manière statique. C'est pourquoi TCP, qui a été conçu pour l'Internet au départ, adopte un **dimensionnement dynamique** de ce temporisateur, basé sur des statistiques réalisées sur les paquets.

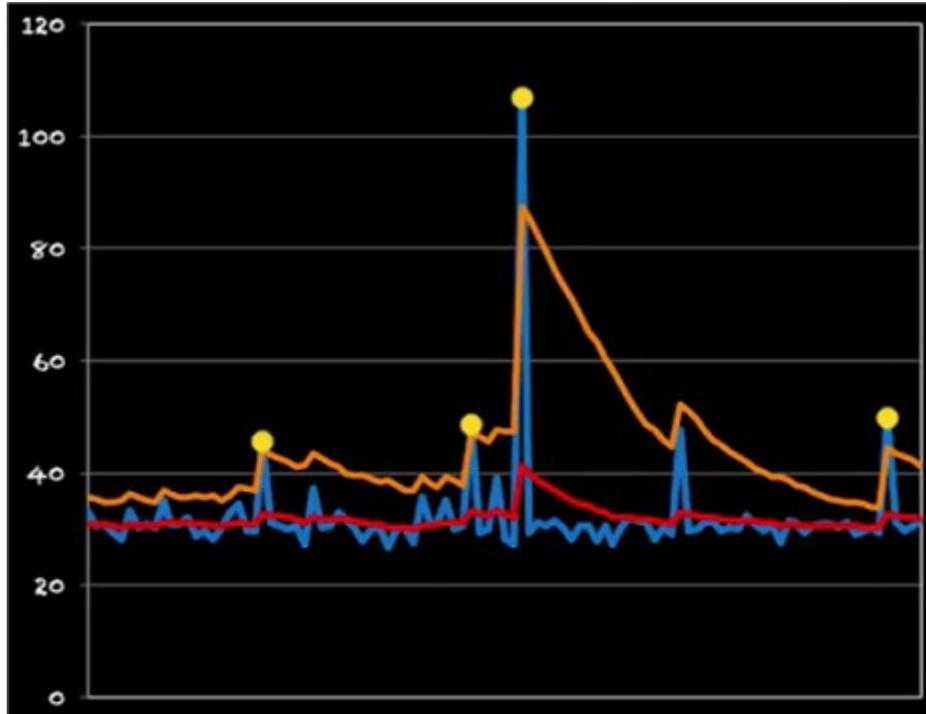


Pour bien dimensionner son temporisateur, TCP va se baser sur une estimation du délai d'aller-retour séparant l'émetteur et le récepteur. Le délai d'aller retour est défini comme le temps séparant l'envoi d'un segment de la réception de l'acquittement correspondant. TCP va essayer d'estimer une moyenne de ce délai, mais ne va pas réaliser cette moyenne sur l'intégralité des segments depuis le début de la transmission, mais plutôt sur les quelques derniers segments de la transmission.

Sur ce graphique, la courbe bleue représente le délai réel d'aller-retour entre un émetteur et un récepteur. La courbe rouge représente la moyenne calculée par TCP sur un historique récent. On peut remarquer que le temporisateur d'acquittement ne peut pas être égal à la moyenne. En effet, de nombreux acquittements mettent un temps supérieur à cette moyenne pour revenir à l'émetteur. Les segments correspondants seraient alors considérés comme perdus (point orange). C'est pourquoi TCP ajoute à cette moyenne une estimation de la variation du délai pour calculer son temporisateur.



Le temporisateur est en pratique égal à la moyenne plus quatre fois l'écart moyen à cette moyenne. La courbe orange, sur ce graphique, représente l'évolution du temporisateur au cours du temps. On peut voir que beaucoup moins de segments sont considérés comme perdus. On peut aussi constater que ne conserver qu'un historique récent permet de s'adapter rapidement aux variations du délai.

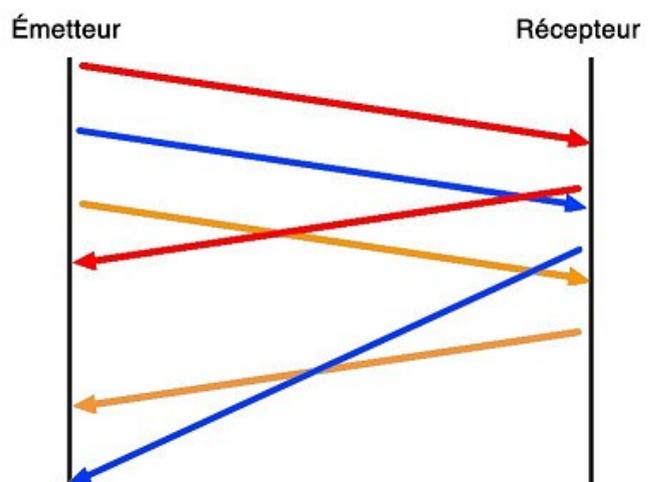


TCP utilise des acquittements pour s'assurer de la bonne réception des segments de données, mais aussi pour s'adapter aux variations des conditions du réseau. TCP est capable de détecter si un segment a été bien reçu ou perdu. Cependant, cela ne suffit pas pour s'assurer de la bonne réception de l'intégralité de l'information.

3. Session TCP : démarrer et terminer un dialogue

TCP utilise les acquittements pour s'assurer de la bonne réception des segments. Un émetteur envoie dans le réseau une séquence de segments et reçoit en retour une séquence d'acquittements. Cependant les segments comme les acquittements sont des paquets IP qui transitent dans le réseau. Ils peuvent donc être retardés et rien ne garantit que l'ordre dans lequel les acquittements seront reçus corresponde bien à l'ordre dans lequel les segments ont été envoyés.

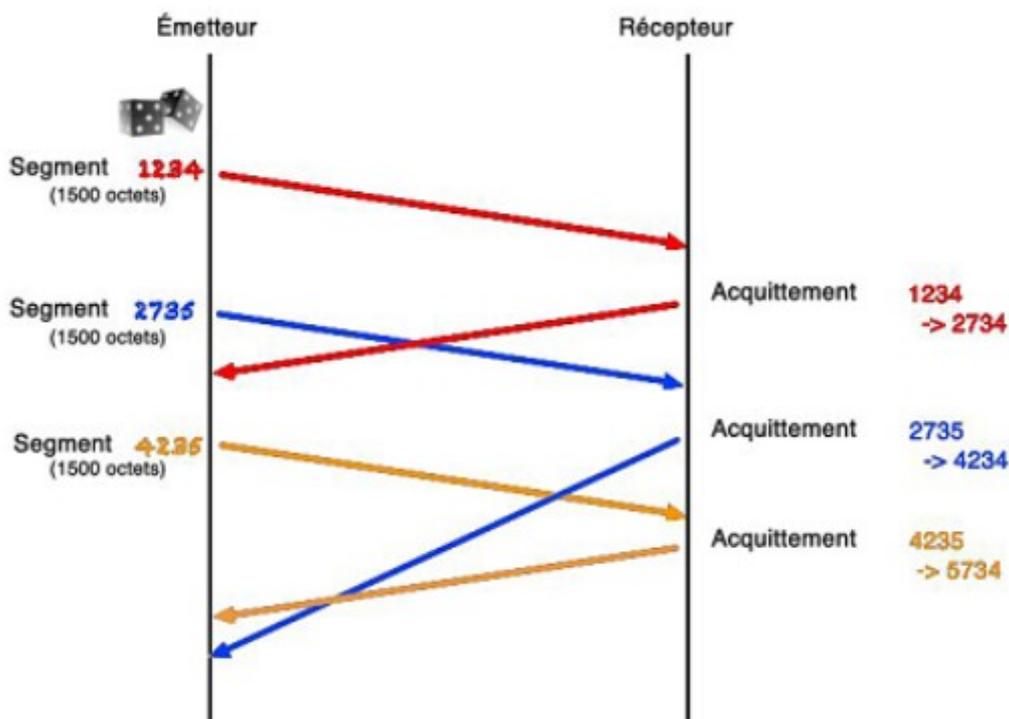
Pour faire correspondre les acquittements aux segments, TCP associe à chaque segment un numéro de séquence auquel l'acquittement correspondant fera référence. En pratique, ce numéro de séquence est l'index du premier octet contenu dans le segment. Ces index sont calculés à partir du premier octet du premier segment du flux de données. L'index initial, quant-à-lui, est tiré aléatoirement. Il pourrait sembler plus logique de démarrer cette numérotation à 1, cependant cette stratégie serait source



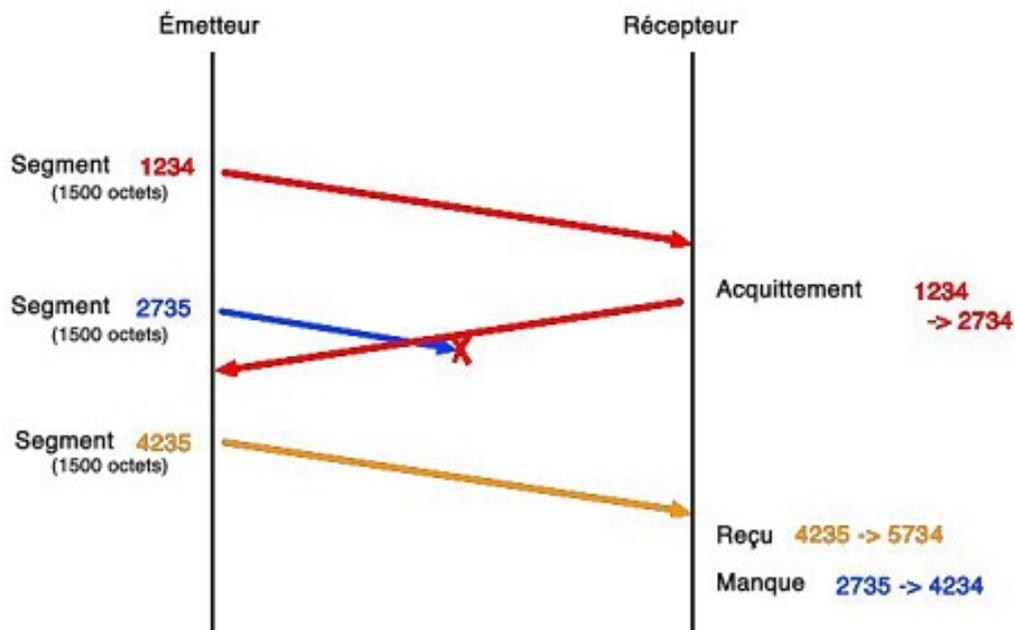
d'ambiguïté dans certains cas. Notamment lorsque plusieurs essais successifs d'un flux très court sont réalisés.

Par exemple :

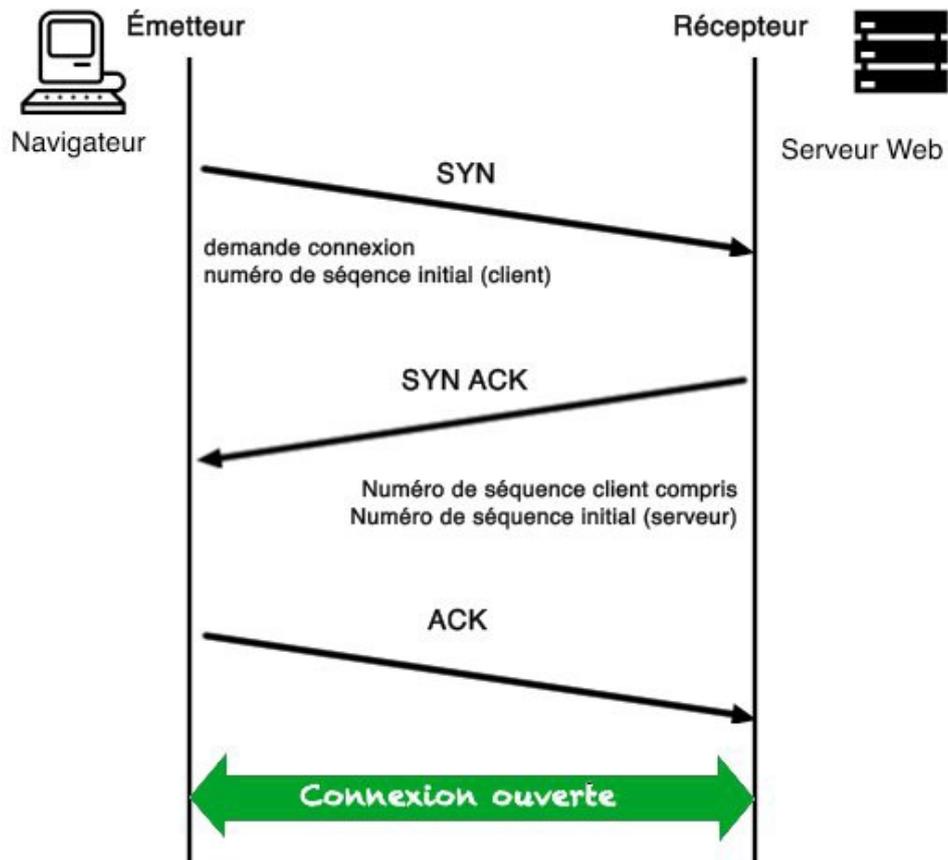
1. L'émetteur envoie un segment au récepteur contenant 1500 octets de données, un numéro de séquence égal à 1234 (Seq = 1234) et un numéro d'acquittement égal à 5678 (Ack = 5678).
2. Le récepteur envoie un segment ACK à l'émetteur. Le numéro de séquence de ce segment correspond au numéro d'acquittement de l'émetteur (Seq = 5678), et le numéro d'acquittement au numéro de séquence de l'émetteur tel que reçu par le récepteur, augmenté de la quantité de données en octets reçue (Ack = 1234 + 1500 = 2734).
3. L'émetteur envoie un nouveau segment de séquence Seq = Ack + 1 (2734 + 1 = 2735) et un nouveau numéro d'acquittement Ack = 2734 + 1500 = 4234.
4. Et ainsi de suite : le segment suivant aura un numéro de séquence Seq = 4234 + 1 = 4235 et un numéro d'acquittement Ack = 4234 + 1500 = 5734.



Avec ces numéros de séquence, le récepteur d'un flux TCP sait détecter la présence de trous dans la transmission. En effet, s'il a reçu l'intégralité du flux jusqu'à l'octet numéro n, le segment suivant doit démarrer à l'octet n+1. Cependant, il doit encore s'assurer d'avoir reçu correctement le début et la fin du flux de données. En effet, **les numéros de séquence initiaux et finaux sont arbitraires** et l'émetteur doit donc les communiquer explicitement au récepteur. C'est pourquoi TCP impose une phase explicite d'ouverture de connexion et une phase explicite de fermeture de connexion.

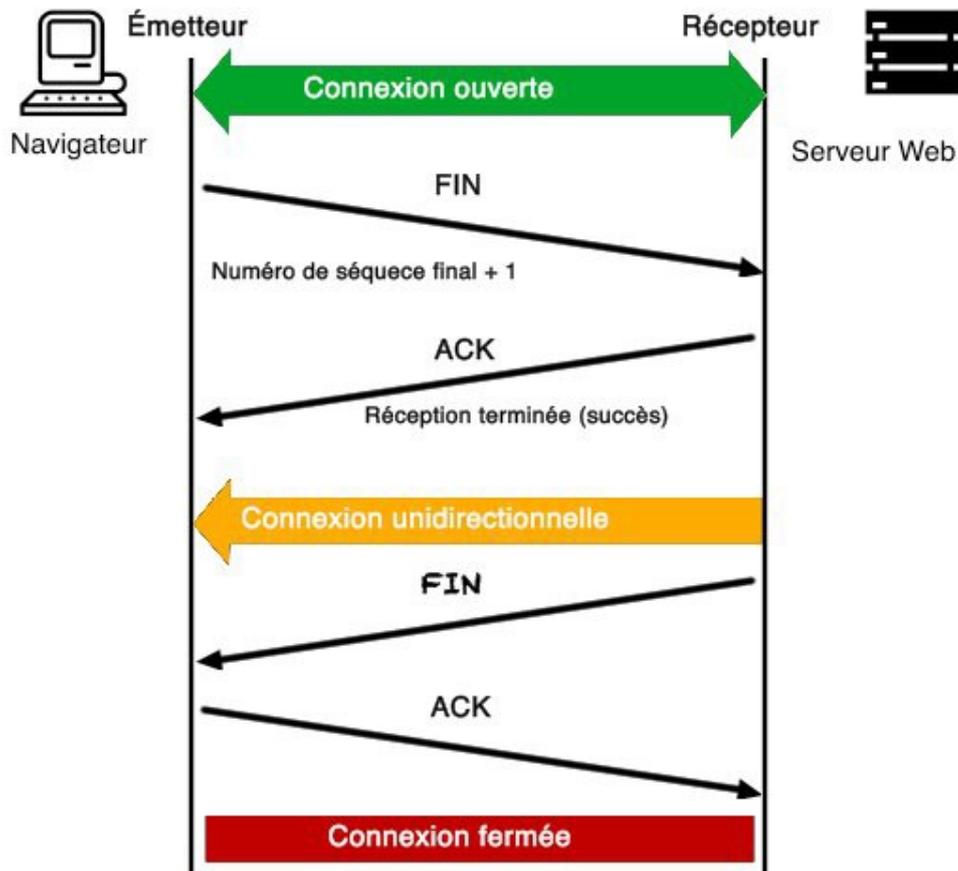


La phase d'ouverture de connexion est composée de 3 messages. Le premier message, appelé synchronisation, est émis par l'initiateur de la connexion. Prenons l'exemple du dialogue entre un navigateur web et un serveur. Le navigateur démarre cet échange en émettant ce message de synchronisation dans lequel il inclura son numéro de séquence initial. Le serveur lui répondra en acquittant la réception de ce numéro de séquence et en incluant dans ce message d'acquiescement son propre numéro de séquence initial. En effet, **une connexion TCP est toujours ouverte dans les 2 sens** et les deux flux de données sont, du point de vue de TCP, indépendants. Leurs numéros de séquence ne sont donc pas alignés. À la réception de ce 2ème message, le navigateur renverra au serveur un dernier acquiescement et l'échange pourra alors commencer.



protocole d'ouverture d'une session TCP

Une fois que le navigateur a terminé la transmission de ses données, il envoie au serveur un message de fin. Ce message contient le dernier numéro de séquence des données plus un. À la réception de ce message, le serveur peut vérifier qu'il a bien reçu l'intégralité des données et, le cas échéant, enverra au navigateur un message d'acquiescement. La connexion est alors fermée dans le sens navigateur vers serveur. Elle reste cependant ouverte dans le sens serveur vers navigateur et devra être fermée de manière analogue à l'initiative du serveur.



protocole de fermeture d'une session TCP

Nous avons vu comment un émetteur TCP s'assurait de la bonne réception de l'intégralité des données qu'il envoie. En pratique, ce jeu d'envoi de données indexées et d'acquittements est réalisée grâce à l'en-tête TCP dont voici un extrait. Vous pouvez noter la place et la taille du numéro de séquence et du numéro d'acquittement. Les phases d'ouverture de connexion et de fin de connexion sont elles aussi réalisées au moyen de cet en-tête. En pratique, on envoie un segment de donnée vide dans lequel des drapeaux sont positionnés pour indiquer s'il s'agit d'un message de synchronisation, d'acquittement ou de fin.

