Chapitre II Protocoles de la couche transport

Cours Réseaux Master Informatique

Ilyas Bambrik

Table des matières

Protocoles couche application

I

- Les applications interfacent avec la couche transport à travers les sockets ;
- Deux applications ne peuvent pas utiliser le même numéro de port dans la même machine ;
- Il existe des ports réservés pour des protocoles applicatifs prédéfinis [de 0 à 1023] (*FTP=port :20, HTTP=port :80, HTTPS=port :443, DNS=port 53*, ...etc) ;
- Une plage de numéros de ports est disponible pour être allouée dynamiquement aux applications [de 49152 à 65635];
- Si un port est ouvert dans la machine locale par une *application A*, d'autres applications dans la même machine peuvent utiliser un socket pour communiquer avec cette application ;
- Une application serveur est généralement multi-processus (pour chaque connexion un processus est créé afin de traiter la connexion d'un client) ;

Exercice :

II

- Exécutez la commande "*netstat -n*" sur votre CMD (Invité de commande) pour voir les connexions / port ouverts dans votre machine.
- Si vous avez une adresse de machine distante, vous pouvez voir les ports ouverts de celles-ci par la commande *nmap*.

Protocoles couche transport

III

Rappel : Rôle couche transport

- Assure la livraison des données à l'application cible;
- Supposant que *l'application A*, hébergée dans la machine 192.168.1.2, transmet des données vers *l'application B* situés à la machine 192.168.1.3;
- Pour effectuer cette transaction, les données générées par A doivent *être enveloppées avec un entête de la couche transport (cet entête spécifie le numéro de port de l'application source et le numéro de port de l'application destination)* ;

- Pour transmettre / recevoir des données, les protocoles applicatifs (comme FTP ,HTTP) doivent utiliser un protocole de transport (TCP et/ou UDP) ;
- Ce ci est appelé *sous-couchage de protocoles* ;

Exemple : Exemple sou-couchage de protocoles

- Le protocole FTP (File Transfer Protocol) utilise TCP ;
- Le protocole DNS (Domain Name Server) utilise UDP et TCP ;

UDP (User Datagram Protocol)

IV

- UPD est un protocole très simple. Ce protocole n'assure que de faire passer un message à l'application destination (*d'où il est dit non fiable*).
- *Ce protocole n'assure ni la segmentation, ni la réorganisation des messages au niveau du destinataire* (si l'application souhaite assurer la segmentation / classement des segments de l'autre bout, elle doit l'implémenter elle même) ;
- Le seul contrôle fourni par ce protocole est le checksum. Celui-ci permet de vérifier s'il y eu une erreur d transmission. En cas d'erreur, le message envelopper par UDP sera simplement ignoré (la couche transport du destinataire ne livrera pas le contenu à l'application). De même, la source d'un message erroné ne sera pas notifiée de cette échec ;
- *Ce protocole est généralement utilisé dans les applications temps réel* (jeux vidéo online, application de communication audio visuel comme Skype ou TeamSpeak) ;
- En utilisant ce protocole, une application ne peut qu'espérer que le destinataire recevra les données transmise;

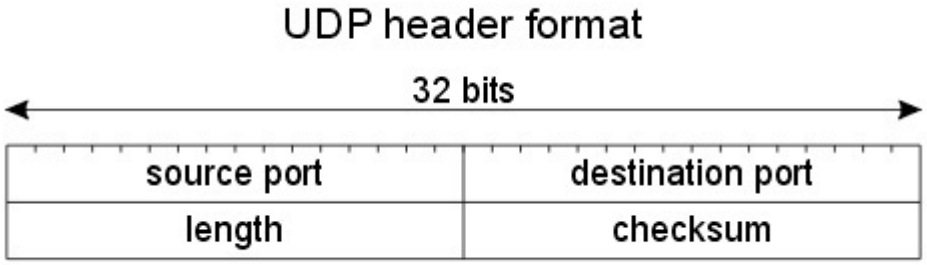


Figure 1. Entête UDP

TCP (Transmission Control Protocol)

V

- TCP est le protocole de transport le plus utilisé par les applications réseau (d'où l'inclusion de son nom dans le modèle TCP/IP) ;
- Ce protocole est *fiable* (dit orienté connexion) car il assure la livraison des données;
- Comme protocole de la couche Transport, le rôle de TCP est d'assurer que la destination appropriée reçoi les données qui lui sont transmises. Pour ce-ci, un segment TCP contient un numéro de port de l'application source (celle qui génère le segment) et le numéro de port du destinataire (celle qui est sensée recevoir le segment) ;
- TCP est un protocole Full-Duplex (c.à.d les deux applications qui établissent une connexion changent des messages simultanément et dans les deux sens avec la même connexion) ;
- Les données sont transmis au protocole TCP sous forme de flux. *Ce protocole ensuite découpe le flux en segments de données avant de les encapsuler et les transmettre à la couche inférieure (Couche Réseau/IP) ;*
- TCP assure aussi le contrôle du flux de transmission (*voir la section sur le Sliding Window*);
- Chaque fois qu'un segment est transmis, un chronomètre (compteur de temps) est lancé. Si le chronomètr expire sans que l'acquittement ne soit reçu, le segment doit être retransmis ;
- Si la retransmission d'un segment se répète plusieurs fois sans succès (pas d'acquittement), le protocole TCP arrête les tentatives ;

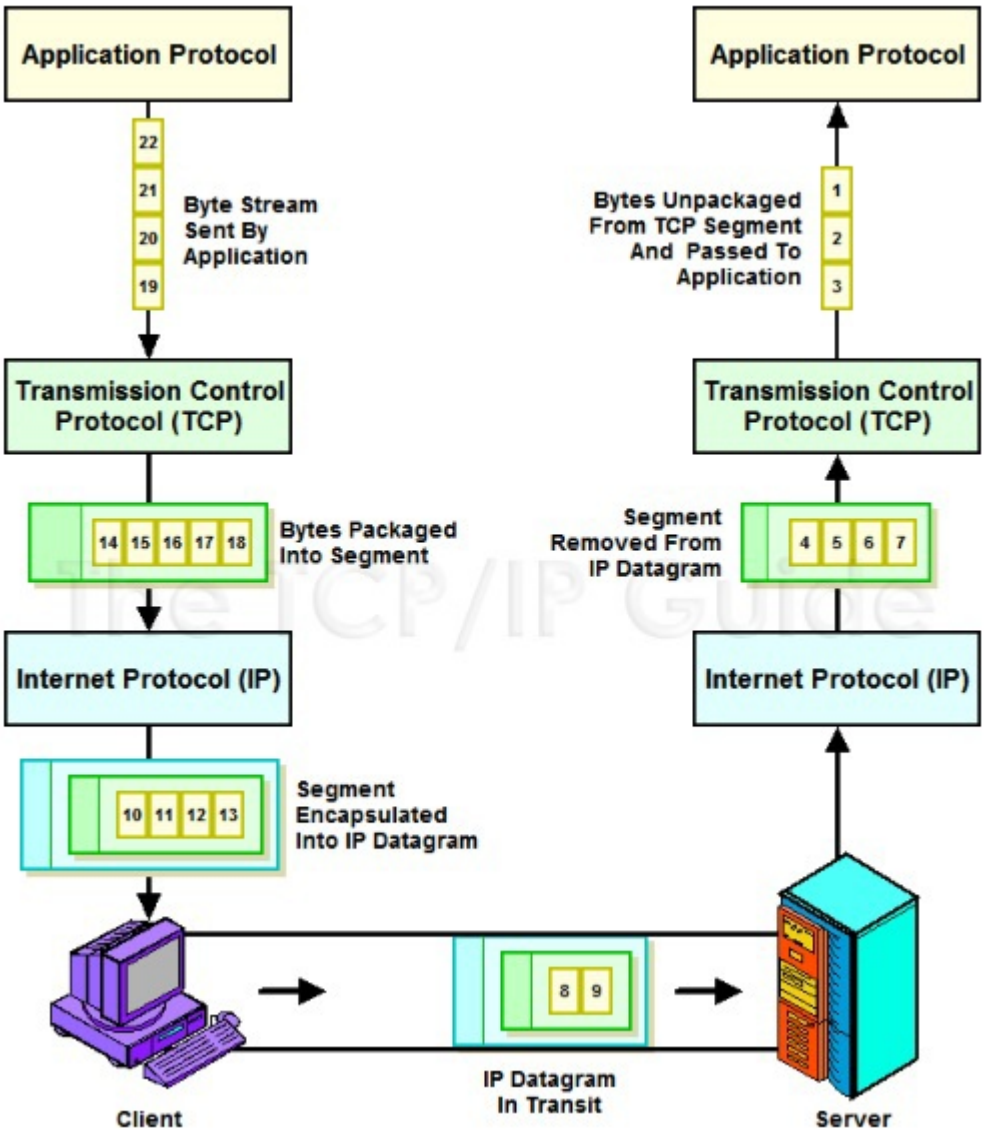


Figure 2. Segmentation TCP [*TCPguide.com*]

Entête TCP

VI

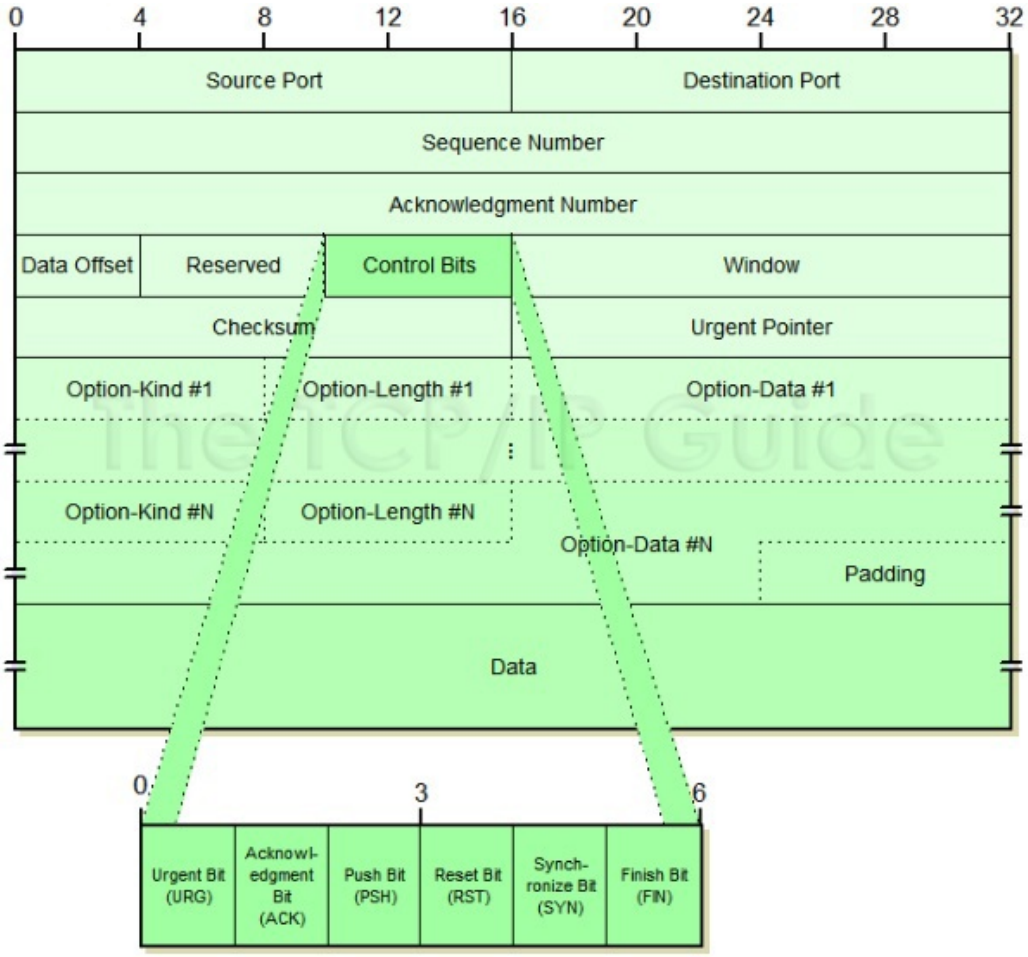


Figure 3. Entête TCP [*TCPguide.com*]

1. Un segment TCP commence par *le numéro de port de l'application source* (2 octets)
2. Ensuite, *le numéro de port de l'application destination* (2 octets) ;
3. *Numéro de séquence du segment* (4 octets) : *numéro de séquence du premier octet* dans ce segment. Cette valeur est utilisée par le récepteur pour réordonner les segments et détecter les segments non reçus ;
4. *Numéro d'acquittement (Acknowledgment)* (4 octets) : Si le bit du flag ACK (*voir champ 7*: *Flags de contrôle*) est allumé, la valeur de ce champ indique l'intervalle des numéros de séquences acquittés (tout les numéros de séquences [*champ 3*] inférieures à ce numéro ACK sont acquittés car ils ont été correctement reçus par la destination). De plus, cette valeur indique au transmetteur le numéro de séquence du prochain segment à transmettre ;

5. *Offset (4 bit) :* La taille total de l'entête TCP (généralement 20 octets). Ainsi, ce champ indique où l' entête TCP s'arrête et les données de l'application commencent. S'il y a des options, l'offset peut prendre jusqu'à 60 octets. Cette valeur est exprimée en *word* (4octets == 1 word).
6. *Champs réservé (3 bit) :* Ce champs est réservé (non utilisé par le protocole) et prend toujours la valeur 0 ;
7. *Flags de contrôle (6 bit) :* Chaque bit dans ce champs possède une signification (*voir la section suivante titrée "Signification des Bits flag"*);
8. *Window (2 octets) :* Le récepteur de données doit indiquer la taille du buffer (mémoire tampon) dédiée à cette connexion. La valeur de ce champ peut être ajustée au cours de la transmission pour contrôler le flux de transmission ;
9. *Cheksum (2 octets) :* Valeur de contrôle calculée pour assurer l'intégrité du segment ;
10. *Pointeur donnée urgente (2 octets) :* Généralement est égale à la valeur 0. En cas où le flag urgent dans le champ Flags de contrôle (*voir champ 7*) est allumé, la valeur dans ce champ indique où les données qui doivent être traitées en priorité se terminent;
11. *Option* (de 0 à 40 bits) : Généralement inexistant sauf au début de l'établissement de connexion TCP où des paramètres sont échangés. Par exemple, le Maximum Segment Size (MSS) est communiqué dans le champ option au début de la communication par le champ Option;

Signification des Bits flag

VII

URG : Le segment contient des données qui doivent être traités d'urgence ;

ACK : Segment d'acquittement ;

PSH : Du coté transmetteur, le flag PSH allumé indique à la couche transport du transmetteur que le segment doit être transmis immédiatement. Du coté récepteur, le flag PSH allumé indique à la couche transport du récepteur que le segment doit faire passer le paquet à la couche application immédiatement ;

RST : Le transmetteur d'un segment avec un flag RST allumé indique que cette connexion a été inattendue. (Par exemple l'application A essaye de connecté au port 5000 d'une machine distante alors que ce port n'est utilisé par aucune application. Ainsi, la machine distante repend avec un segment TCP avec le flag RST allumé) ;

SYN : Indicateur de synchronisation ;

FIN : Terminaison de connexion ;

Établissement de connexion TCP par "Three way handshake"

VIII

- Soit une application A et une application B ;
- Pour commencer, B doit créer un socket et écouter les connexions entrantes sur le numéro de port *NUMPORT* (en pratique ce-ci est fait par l'instanciation d'un socket avec *NUMPORT* comme argument, l'attente de connexion se fait par la méthode *accept*()) ;
- *Etape 1* : La couche transport de l'application A qui souhaite connecter à B, commence par transmettre un segment avec le numéro de port du destinataire ==*NUMPORT* et un *flag avec le bit SYN allumé*. Ce paquet *ne contient aucune donnée* et signifie que A souhaite *SYNCHRONISER* (*SYN*) avec B (ce-ci est fait par la méthode *connect*() du client);
- *Etape 2* : Après la réception de cette demande de synchronisation, la couche transport de B transmet un segment vide (*pas de donnée dedans*) contenant un *flag avec le bit SYN allumé et ACK allumé*. Ce-ci signifie que la tentative de connexion a été reçu par B ;
- *Etape 3* : Ensuite, la couche transport de A transmet un segment contenant un *flag avec le bit ACK allumé* (pour signifier que la connexion a été établie) ;

Trois étapes d'établissement de connexion = Three way Handshake

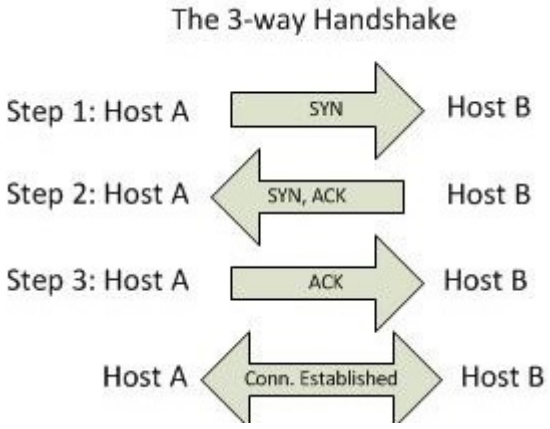


Figure 4. Établissement de connexion avec *"Three way Handshake"* [*paranet.com*]

Clôture de connexion

IX

- Similairement, lors de la clôture de transmission, les couches transports des deux applications s'échangent des segments contenant un flag *avec le bit FIN (FINISHED) et ACK* allumés ;
- La couche transport de l'application A transmet un segment avec le *bit FIN allumé pour dire qu'il n'y a pas d'autres données à transmettre et la connexion entre les deux applications peut être clôturée* ; Pour clôturer la connexion dans les deux dirrections, la couche transport de B effectue la même procédure ;

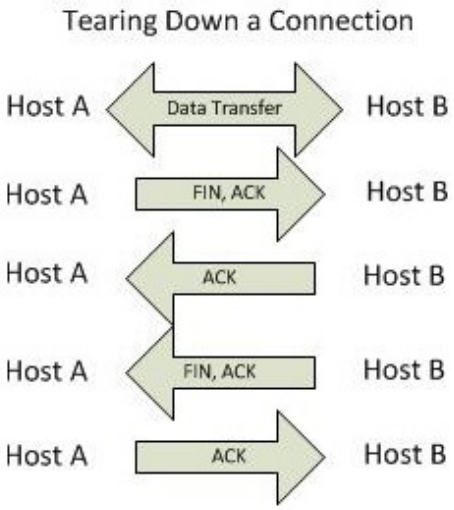


Figure 5. Clôture de connexion avec flag *FIN* [*paranet.com*]

Acquittement des segments

X

- Dans TCP, l'acquittement des segments est cumulative. Cela veut dire que lorsque le récepteur envoie un acquittement avec une valeur (*champ 4*) == 201 (voir la figure en bas), *ce-ci veut dire que tout les BYTES avec un numéro de séquence inférieurs à 201 ont été acquittés*. Si des segments avec des numéros de séquences (*champ 3*) supérieurs ou égales à 201 ont été transmis auparavant, ceux ne sont pas encore acquittés (*même s'ils ont été reçus*) ;
- Ainsi, le récepteur acquitte la réception d'une intervalle d'octets cumulatives ;

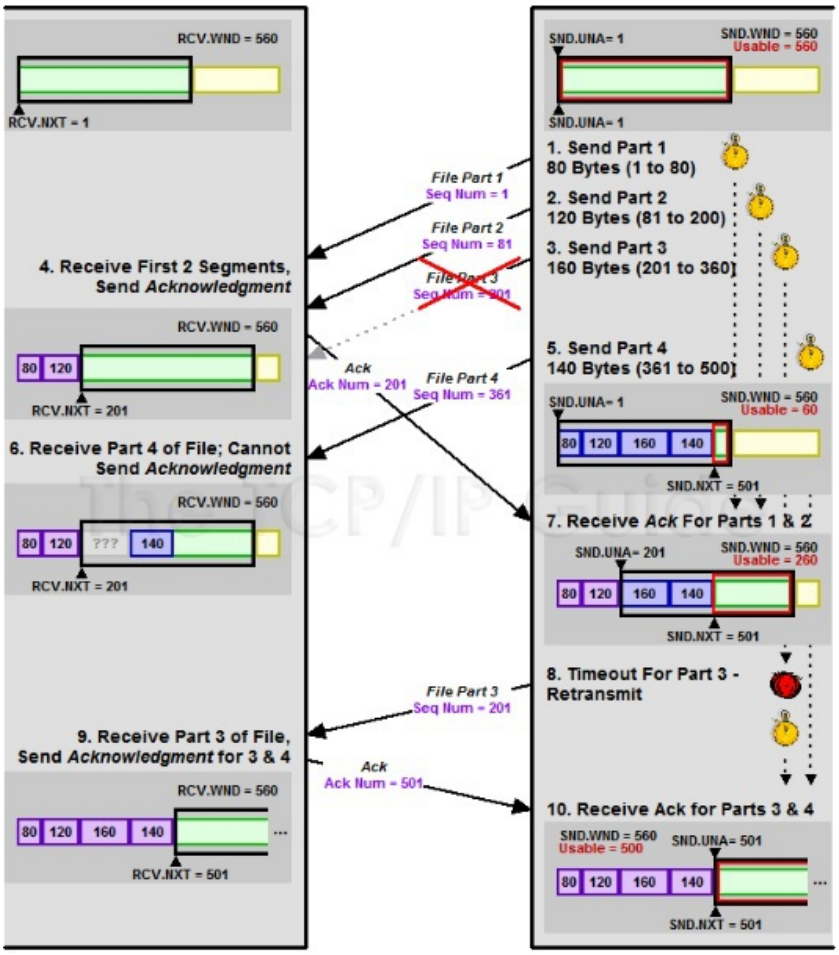


Figure 6. Retransmission et acquittement [*TCPGuide.com*]

Remarque : Numéro de séquence initial

- Le numéro de séquence initial du premier segment des données n'est pas forcément 0 ;

- Le transmetteur et le récepteur doivent s'entendre sur le numéro de séquence du premier segment lors de l'établissement de connexion (*three way handshake*) ;

Mécanisme Sliding Window

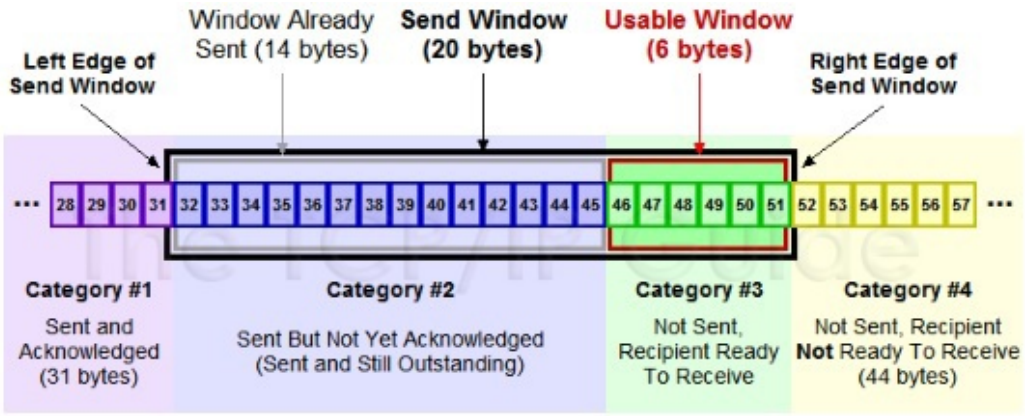
XI

- Chaque segment ACK transmis par le récepteur contient une valeur *Window size* (*champ 8*) pour indiquer au transmetteur la quantité de données qui peut être temporisées (placée dans le buffer) par le récepteur. Ainsi, la somme des tailles des segments envoyés par le transmetteur sans être acquittés ne peut pas dépassée la taille du buffer du récepteur sans que le récepteur acquitte des messages reçus;

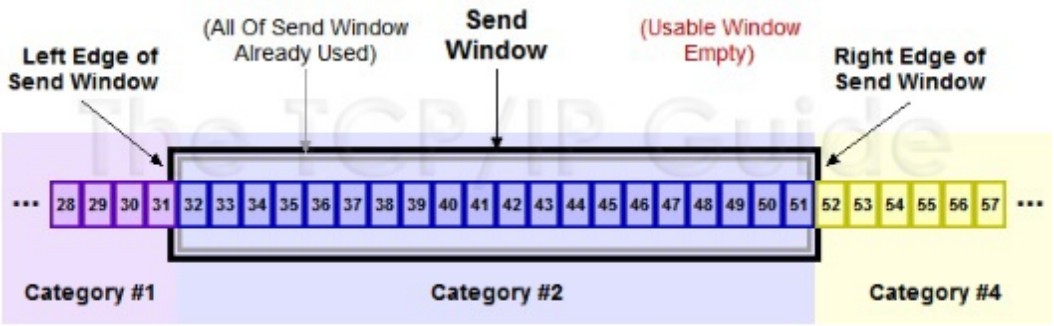
Le transmetteur et le récepteur maintiennent des variables indiquant :

1. L'intervalle de bytes qui ont été transmis et acquittés (*Left Edge* dans la figure en bas, cette valeur= 31). Les bytes inférieurs à cette valeur ont été correctement reçus;
2. L'intervalle de bytes qui ont été transmise et *pas acquittée* (*Window Already Sent* [*encadrée en gris*] dans la figure en bas, cette valeur= 32 jusqu'à 45 dans cet exemple). Les bytes dans cette intervalle ont été transmis mais pas encore acquittés;
3. Intervalle utilisable (*Usable Window* dans la figure en bas, cette valeur= 46 jusqu'à 51). Cette valeur définis combien de données peuvent être transmis sans déborder de la taille du buffer indiqué par le transmetteur ;

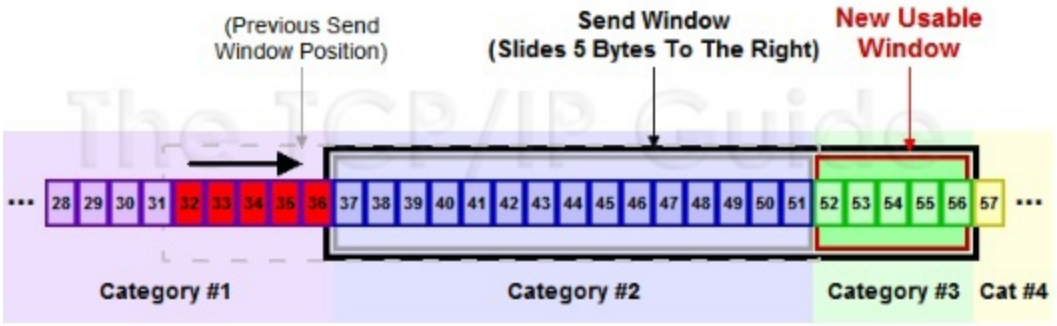
La somme des tailles des trois intervalles == taille du buffer du récepteur ;



Si le transmetteur envoie touts les segments dans l'intervalle utilisable, *la transmission s'arrêtera tant que aucun ACK des bytes transmis mais non acquittés ne soient reçus* (voir la figure suivante) ;



- Après, si le chrono pour un segment transmis mais non acquitté *expire sans qu'un ACK qui l'inclus ne sois reçu*, ce segment est retransmis ;
- Si un ACK est reçu, l'intervalle de transmission est décalée selon le numéro ACK reçu. Par exemple, dans le figure précédente , le récepteur reçois un ACK avec un numéro 37 (et donc les bytes de 32 - 36) ont été acquittés. Par la suite, l'intervalle de transmission glisse vers l'avant ;



- Au cours de la transmission, une option qui permet d'augmenter la taille du buffer est l'option *Window Scale*. La valeur de ce champ peut être ajoutée et inclue dans le champ Options. Celui-ci peut prendre une valeur i entre 0 et 14 qui indique que la valeur du champ *Window Size* doit être multipliée par 2^i.
- Avec le champ *Window Size* et l'option *Window Scale*, le récepteur peut ajuster la vitesse de transmission des données. Ce processus est appelé *Contrôle de Congestion*.
- Le contrôle de la congestion est un mécanisme qui limite le flux de paquets dans le réseau. Ainsi, cela empêche de surchargé le récepteur avec des paquets excessifs. La congestion se produit lorsque le débit de paquets vers un nœud est supérieur à sa capacité de traitement.
  Lorsqu'une congestion se produit, elle ralentit le temps de réponse du réseau. En conséquence, les performances du réseau diminuent. La congestion se produit parce que les routeurs et les commutateurs disposent d'un tampon de file d'attente, contenant les paquets à traiter. Une fois le processus de conservation terminé, ils transmettent le paquet au nœud suivant, ce qui entraîne une congestion du réseau. En cas où un routeur/commutateurs reçoit un paquet mais sont buffer est plain, le paquet reçu est supprimé.
- Pour que le transmetteur et le récepteur décide d'une vitesse de transmission que le réseau peut supporté, le transmetteur commence à augment le nombre de segment à transmettre graduellement avec l'augmentation du *Window Size* par me récepteur. Par la suite, la quantité de donné transmise dans le réseau revue à la baisse lorsque le récepteur reçoit plusieurs paquets avec le même numéro de séquence (ce qui indique les paquets traversent un chemin congestionné) ou bien une large quantité de segments est perdue en transition (les paquets ont été supprimés à cause d'un buffer overflow).

Problème de l'acquittement cumulative

XII

- Supposant que de 100 segments transmis, seulement le troisième n'a pas été correctement reçus. Ainsi, même si la destination a reçu 99 segments, elle ne peut acquitter que les deux premiers segments.

Le transmetteur peut implémenter l'une des deux stratégies :

- Retransmettre touts les segments non acquittés ;
- Attendre l'expiration du chrono de chaque segment individuellement avant de le retransmettre ;

Une nouvelle stratégie est apparue appelée acquittement sélective où le récepteur ajoute des numéros d' acquittements individuels (au lieu d'intervalle) dans le champs option (*champs 11*) d'un segment avec un flag ACK allumé. *Cependant, ce-ci nécessite une négociation préalable lors de la synchronisation*.