Le protocole TCP

(Z:\Polys\Internet RES0\9.TCP.fm- 12 février 2014 13:58)

PLAN

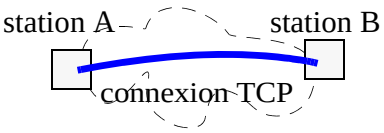
- Présentation
- Les segments TCP
- Le multiplexage
- La fenêtre coulissante
- La connexion
- Les données urgentes
- Les options
- Conclusion
- Implémentations

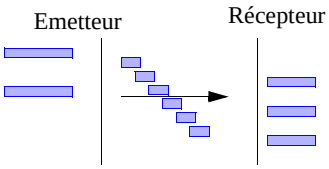
1. Présentation

"Transmission control protocol"
. Rfc 793
. Septembre 1981

Transmission de données numériques :
. Par paquets de tailles variables
. En mode connecté (3 phases)
- Etablissement de la connexion
- Transfert de données
- Libération de la connexion

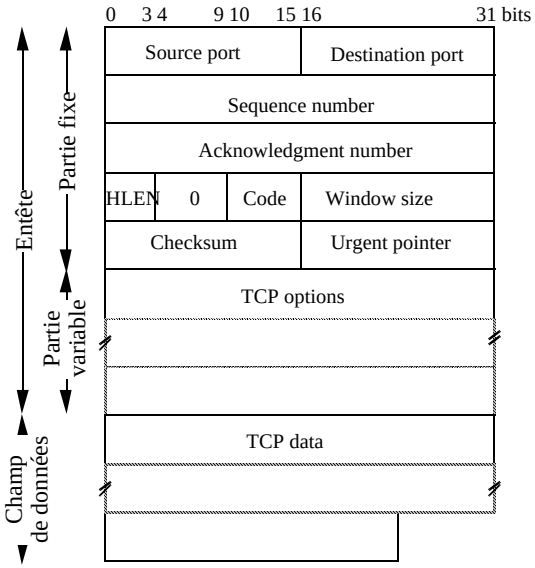
. Bidirectionnelle
. Flux non structuré de données
=> suite d'octets ("Stream")
. Fiable
- contrôle et récupération des erreurs
- contrôle de flux et de congestion
- contrôle de la duplication
- reséquencement





2. Les segments TCP

2.1. Le format général



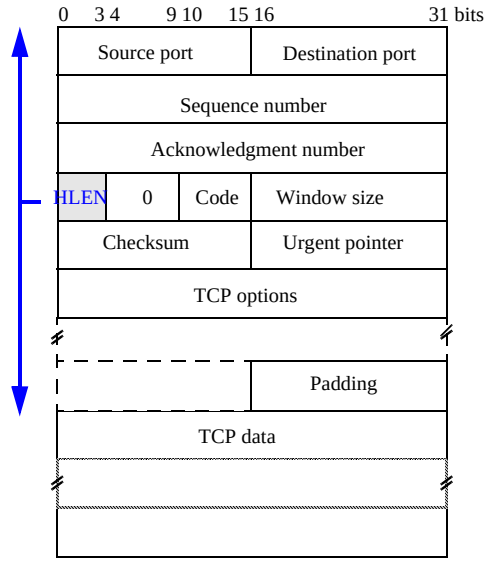
En mots de 32 bits.

Une entête :
. une partie de taille fixe,
. une partie de taille variable (les options).
Un champ de données :
. de longueur variable.

Une connexion <-> double couple :
<adresse IP, numéro de port> du récepteur
+ <adresse IP, numéro de port> de l'émetteur.

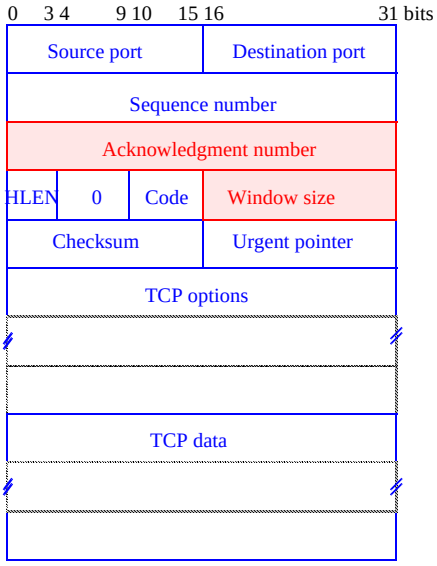
par exemple :
<131.254.31.8, 2345>+<131.254.11.26, 20>

2.2. L'entête



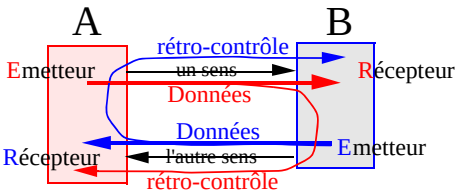
HLEN ("header length") 4 bits :
- Longueur de l'entête en mots de 4 octets (équivalent à IP).
- Déplacement du début du champ de données par rapport au début du segment.

2.3. Le piggy backing

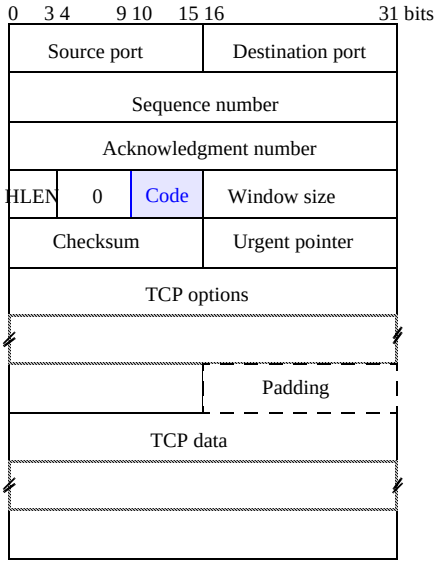


"Piggy backing" :

La connexion étant bidirectionnelle, chaque sens de transmission transmet ses propres données et simultanément les informations de rétro-contrôle relatives à l'autre sens de transmission des données.



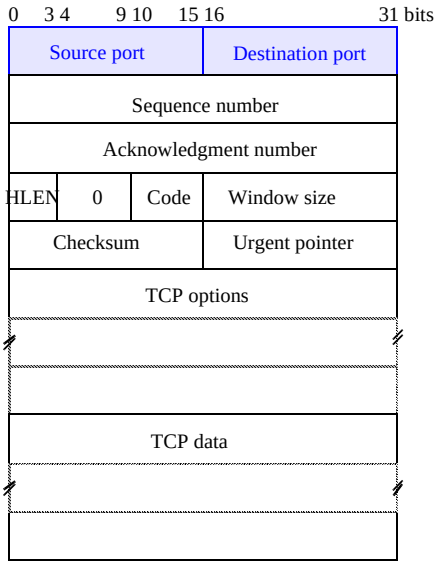
2.4. Les différents rôles des segments



Code (6 bits) :

. Urgent bit
  valide le champ "Urgent pointer"
. Acknowledgment bit
  valide le champ "Acknowledgment number"
. Push bit
  livraison immédiate du segment
. Reset bit
  réinitialisation de la connexion
. Synchronise bit
  demande d'ouverture de la connexion
. Final bit
  demande de libération de la connexion

3. Le multiplexage



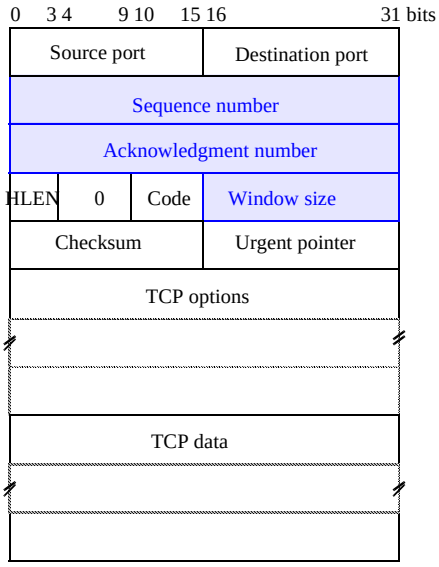
Source port et destination port :
. identique à UDP
. identifie la connexion :
<@IP source + n° port source,
@IP destination + n° port destination>

=> Multiplexage (sur-adressage)

3 types de n° de port :
. n° réservés mondialement (en général n°≤1024),
- attribués à des services généraux spécifiques
- durée de vie infinie
. n° réservés localement
- choisis pour une application
- durée de vie de l'application
. n° attribués dynamiquement,
- choisis par le système parmi les n° libres,
- durée de vie de la connexion

4. La fenêtre coulissante

4.1. Les champs



Sequence number :
. Numéro de premier octet du champ de données dans le flux d'octets transmis (modulo 2^{32}).
. Initialement la valeur du champ est quelconque!

Acknowledgment number :
. Numéro du prochain octet à recevoir.
. Acquitte tous les octets de numéro inférieur.

Window size :
. Nombre d'octets pouvant être envoyés par anticipation.
. Capacité de stockage du récepteur.

. W= 0 => arrêt de l'émission

4.2. Le mécanisme

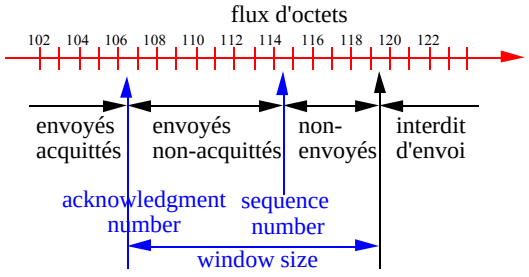
Appelé “Sliding Window”.

Mécanisme permettant à la fois :

- Le contrôle de flux et de congestion.
- Le contrôle des erreurs, pertes, duplication, déséquencialité.
- L'optimisation de l'utilisation de la connexion par l'envoi anticipé de paquets (avant que les octets des paquets précédents soient acquittés).

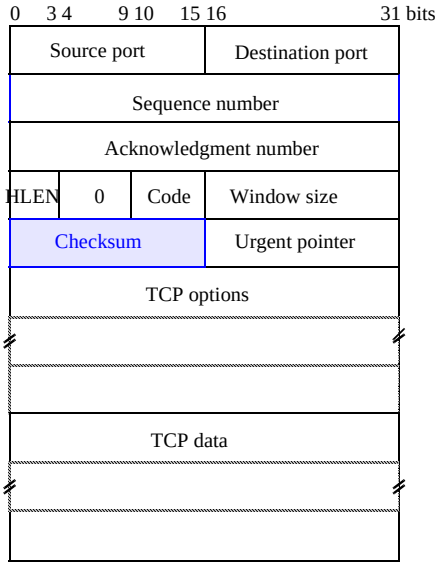
Basé sur

- l'identification des octets (OSI : des paquets !) :
  ⇒ leur numérotation (modulo 2^{32}).
- la détection des erreurs
  ⇒ par “checksum”
- la détection des pertes
  ⇒ par acquittement successifs !
  ⇒ par temporisateur
- la récupération des pertes
  ⇒ par retransmission



#ack <= #seq <= #ack+window_size

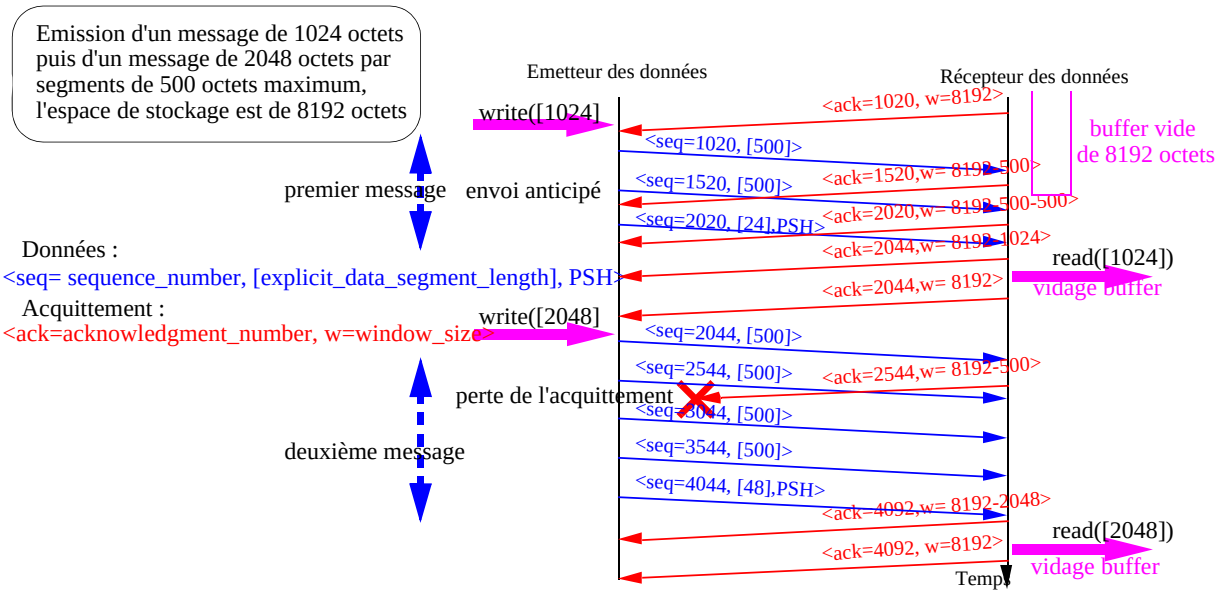
4.3. Protection contre les erreurs



Détection des erreurs :
. Champ de détection d'erreur (Checksum).
. Procédé de calcul strictement identique au même champ de protocole UDP :
  - somme de demi-mots de 16 bits
  - en complément à un
. Peut-être inutilisé (=0).
. Les segments TCP corrompus sont détruits :
  corruption ⇒ perte.

Récupération des pertes (erreurs) s'appuie sur un mécanisme de retransmission des segments :
. un temporisateur est armé lors de l'émission de chaque segment TCP,
. chaque segment est numéroté :
  ⇒ Sliding window.

4.4. Acquittements



Les acquittements sont cumulatifs : ils acquittent tous les octets précédents.

. Nul n'est besoin d'envoyer systématiquement un acquittement

⇒ inconvénient : une vision moins précise de l'état de la connexion

. La perte d'un acquittement ne nécessite pas forcément une réémission.

4.5. Contrôle de flux

Au récepteur :

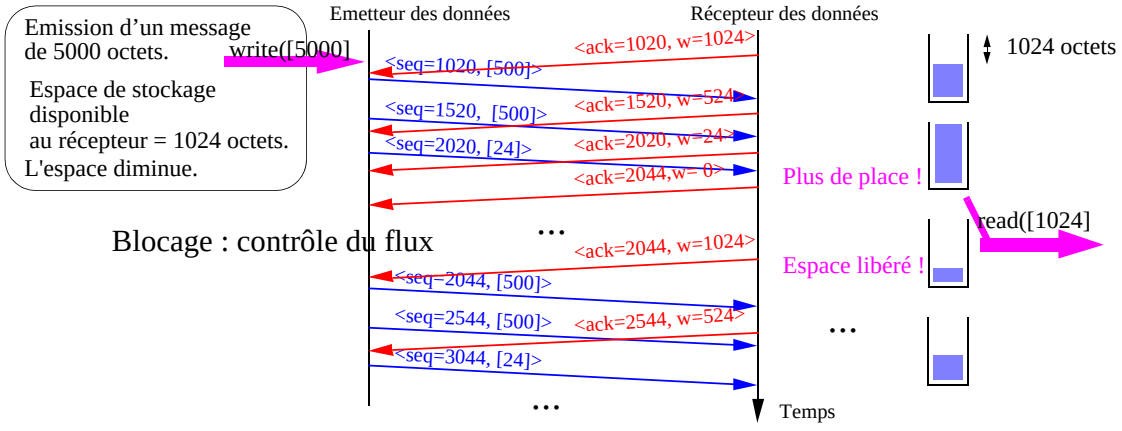
. Nombre d'octets pouvant être stockés par le récepteur.

. Contrôlé par la largeur de la fenêtre (window size) :

- nombre maximum d'octets pouvant être émis sans acquittement par l'émetteur.

Espace de stockage du récepteur presque plein ⇒ diminution de la largeur de la fenêtre (⇒0),

Espace de stockage du récepteur presque vide ⇒ augmentation de la largeur de la fenêtre.



4.6. Gestion des pertes

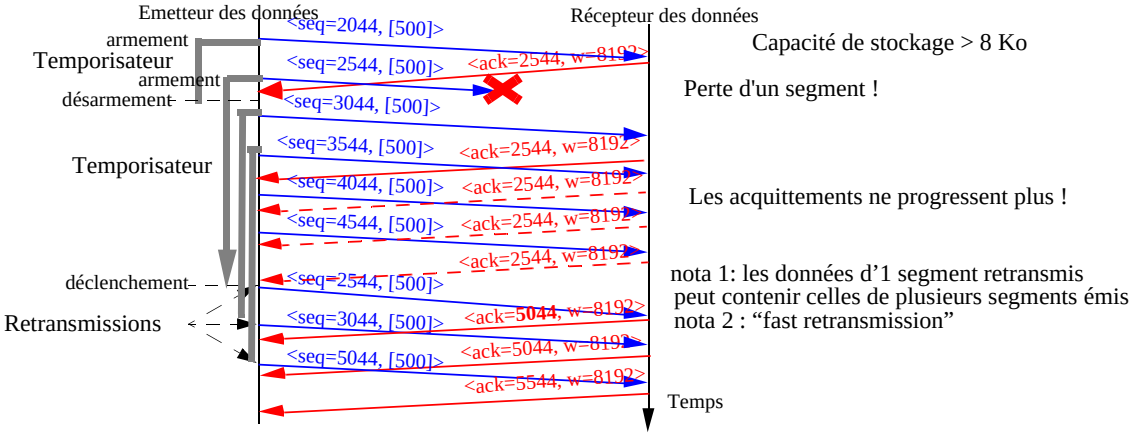
Détection des pertes : à l'émetteur, par temporisateur

⇒ moins efficace (mais plus sûr) que par acquittement explicite (négatif) en provenance du récepteur.

Optimisation de la détection: lorsque l'émetteur reçoit successivement plusieurs acquittements identiques

Récupération des pertes : par retransmission (depuis le segment perdu)

Les segments corrompus sont détruits, cela se traduit par des pertes ! Le mécanisme normal de contrôle des erreurs permet leur retransmission.



4.7. Conclusion

Le mécanisme de la fenêtre coulissante rend de nombreux services :

- protection contre les erreurs et contre les pertes
- contrôle de congestion,
- contrôle de flux,
- maintien de la séquentialité,
- contrôle de la duplication,
- tout en assurant une transmission optimale.

Il existe deux fenêtres indépendantes :

- une pour chaque sens de transmission des données.
- pour chacune des deux fenêtres, chaque partenaire possède une copie de chaque variable permettant de gérer localement cette fenêtre.

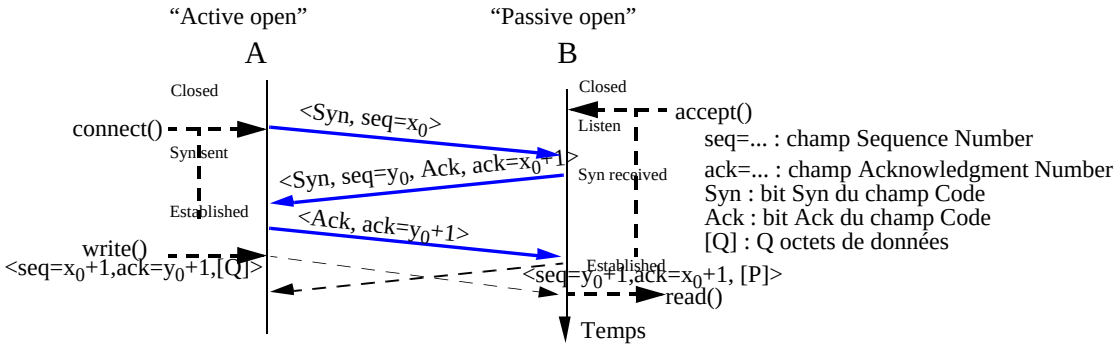
On le trouve dans de très nombreux autres protocoles : HDLC, X25, TP

5. La connexion

5.1. L'établissement de la connexion

Etablissement de la connexion :

. par triple échange ("three-way handshake"),
. initialisation du numéro de séquence initial,
. acceptation/refus de l'établissement,
. double établissement simultané possible,
. on distingue trois types de segments grâce aux bits "Syn", "Ack" du champ Code,
. contrôle par temporisateur.

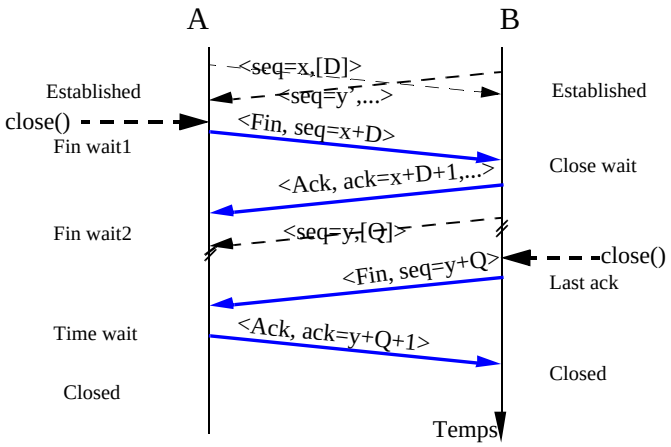


nota : Pour assurer la qualité de la transmission, les segments contenant un bit Syn (resp. Fin) comportent virtuellement un octet de données.
nota : "fast transmission", il est possible de transmettre des données au sein des segments d'établissement, toutefois elles ne seront délivrées que lorsque la connexion sera définitivement établie

5.2. La déconnexion

Libération de la connexion :

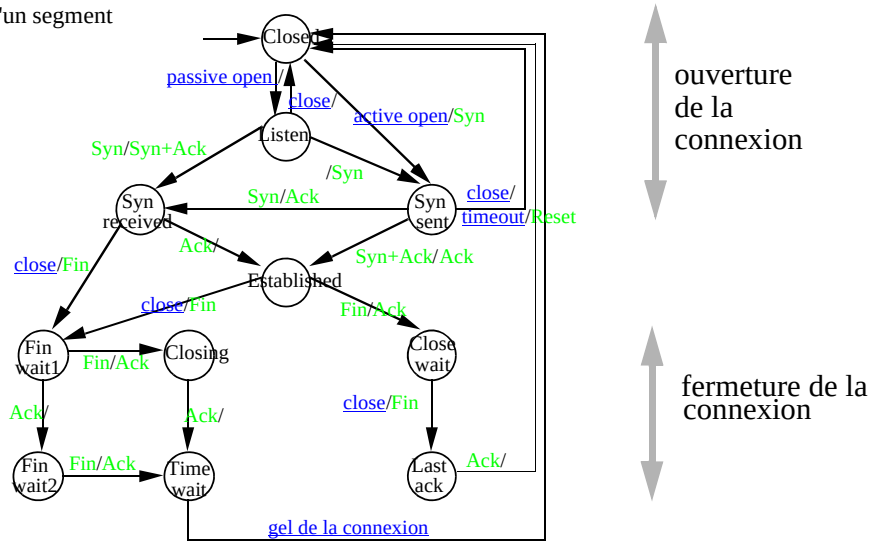
. dans chaque sens indépendamment,
. 2 doubles échanges (2 two-way handshakes),
. utilise deux types de segments identifiés par les bits "Fin" et "Ack" du champ Code.



5.3. Spécification

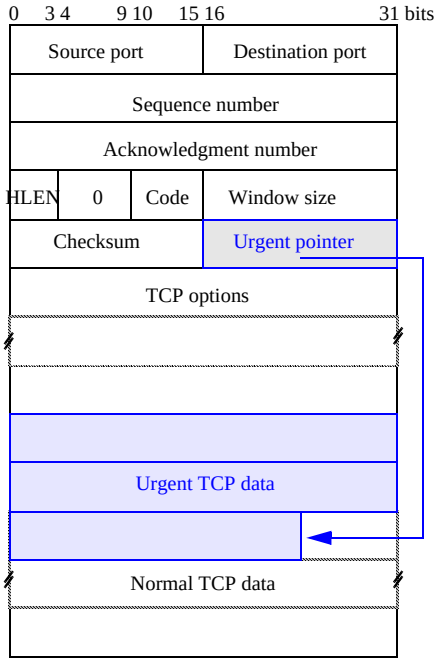
L'automate d'une connexion TCP

Notation :
. condition/action
. en vert réception/émission d'un segment
. en bleu des commandes



Note : Les protocoles IP et UDP n'ont pas besoin d'un automate pour décrire leur comportement.

6. Données urgentes



Appelées "Out of band data"

Urgent pointer : données transmises hors du contrôle de flux. Permet de transmettre des données sans retard, qui doivent être traitées de manière urgente.
Par exemple : commande d'interruption de traitement en cours ou "abort".

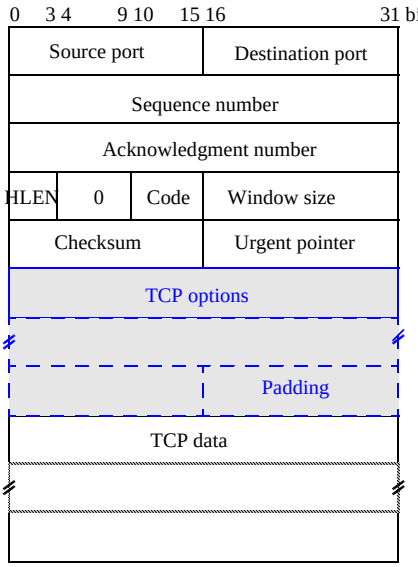
Le champ "urgent pointer" indique la fin de la partie urgente du segment qui commence au début du champ de données du segment.
Le champ "urgent pointer" est validé par le bit "urgent" du champ code.

Limité à un seul segment en attente d'acquittement !

Nota : certaines implémentations limitent arbitrairement la quantité de données urgentes émis dans un segment

7. Les options

7.1. Le format général des options



La partie variable de l'entête :

- Liste d'options
- Sa taille est déterminée grâce au champ HLEN
  en lui soustrayant la taille de la partie fixe
- Format de chaque option de la liste : TLV
  . type of option (1 octet), length of option (1octet), value
- Format similaire à IP.
- Afin de réaliser l'alignement sur les mots de 32 bits,
  un bourrage ("padding") peut être nécessaire

7.2. Quelques options

End of option list [0]: permet à la partie variable de l'entête de se terminer en frontière de mot (idem IP). Un seul octet.

No Operation [1] : alignement de la fin d'une seule option (idem IP). Un seul octet.

MSS option [2] : "Maximum segment size"

. Permet de négocier la taille maximum des segments envoyés.

. Grand segment ⇒ surcoût de traitement (dû à la fragmentation).

. Segment de taille adapté ⇒ minimise le délai d'acheminement

. MSS_par_défaut = 536 octets. C'est à dire MTU_par_défaut - IP_header_length -TCP_header_length = 576 - 20 - 20.

. Exemple : 0x02.04.0400 ⇒ MSS_option =1024 octets.

Window scale factor [3] : <code=3, longueur=3, shift value>

. Augmente la capacité de la fenêtre : par 2^n

. La capacité de la fenêtre limite le débit

. Exemple : 0x03.03.01 ⇒ la largeur de la fenêtre doit être multipliée par 2^1.

. RFC 1323

Timestamp option [8] : <code=8, longueur=10, timestamp, timestamp reply>

. contrôle du délai d'acheminement

8. Conclusion

Protocole offrant un service évolué assurant une transmission de données :

. de bonne qualité,
. en mode connecté,
. d'une suite d'octets,
. avec un service de multiplexage.

Basé sur le mécanisme de la fenêtre coulissante. Permettant à la fois

. une utilisation optimale de la liaison sous-jacente (physique)
et
. la protection contre les erreurs (détection + correction),
. le contrôle des duplications, des pertes, de la séquentialité,
. le contrôle de flux et de congestion.

Protocole aux mécanismes complexes entraînant un certain surcoût tant au niveau des traitements que pour les entêtes des segments. Ses performances sont toutefois très correctes, car optimisées.

Adaptation de TCP : “TCP extensions for high performance - rfc 1323”.

9. Implémentations et optimisations

De nombreux détails d’implémentation permettent d’obtenir un protocole TCP réellement performants.

9.1. Gestion des temporisateurs

Le protocole TCP est très sensible à la durée du temporisateur RTO (“Retransmission Time Out”):

- Trop courte : retransmission inutile
- Trop longue : retransmission tardive

Le protocole doit être efficace quelques soient les conditions de transmission :

- LAN ou réseau étendu, réseau chargé ou non, liens à faible ou haut débit, etc.

  => adaptation du temporisateur en fonction de la durée d’aller-retour : RTT (“Round Trip Time”)

Évaluation du RTT :

- RTT = (α * old_RTT) + ((1 - α) * measured_RTT) , $\alpha \in [0, 1]$
- La durée d’aller-retour est mesurée entre l’émission d’un segment et la réception de son acquittement

- La retransmission et le regroupement de segments rendent difficile cette estimation

  $\Rightarrow$ algorithme de Karn

Lorsque la retransmission d'un segment a lieu, la valeur du RTO est obtenue par :

- RTO = γ * old_RTO, avec $\gamma = 2$

Calcul de la durée du temporisateur RTO :

- Les premières implémentations proposaient :
  . RTO = β * RTT, avec $\beta = 2$
- Les plus récentes proposent un calcul basé sur la variance :
  . Ecart = ((1 - φ) * old_Ecart) + (φ * |old_RTT - measured_RTT|)
  . RTO = RTT + η * Ecart, avec $\eta = 4$, $\varphi = 1/4$, $\alpha = 1\text{-}1/8$

9.2. Fast retransmission

Lors d'une perte, l'attente de l'expiration du temporisateur va réduire les performances du protocole. On peut réagir plus vite.

L'émetteur peut déduire la perte du segment de la réception de plusieurs (3) acquittements identiques successifs :

- Cette déduction n'est pas sûre :
  . cela peut être dû à un retard passager ou à des déséquencements
- Le segment à retransmettre est celui indiqué par le numéro des acquittements répétés
- Ce n'est efficace que pour traiter les pertes fugitives de segments

9.3. Optimisation de la transmission par groupage

Silly window syndrome :

- Une application consommant systématiquement de petits blocs de données (par exemple un seul octet),
- Le protocole envoie un acquittement à chaque libération d'espace au sein du buffer de réception.
- Le producteur génère les données plus vite que le consommateur : le buffer de réception est toujours plein.

=> l'émetteur ne va émettre que des segments de la taille d'un seul petit bloc

. surcharge des processeurs

. surcharge du réseau

Retard des acquittements :

- Avant d'envoyer un acquittement, l'espace disponible du buffer de réception doit être au moins égal, soit à la moitié du buffer, soit à un segment.

=> on retarde les acquittements

- Le retard peut permettre d'utiliser le champ d'acquittement d'un segment envoyé pour transmettre des données en sens inverse ("piggy backing").

=> la durée maximum du retard est conventionnellement de 200 ms

- Un acquittement est immédiatement envoyé

. dès que des données de la taille d'un segment (MSS) sont reçues

. dès que des données hors séquence sont reçues

Groupage des données :

- Les données à émettre sont stockées dans le buffer d'émission tant que :

. leur taille ne dépasse pas celui d'un segment

- Lorsqu'arrive un acquittement :

. on transmet toutes les données accumulées

=> algorithme de Nagle

On remarque que :

- La taille des segments envoyés par le protocole est indépendante de celles des blocs de données remis (ou retirés) par l'application.
- L'envoi d'acquittements n'est ni systématique ni immédiat

9.4. Congestion du réseau

Congestion :

- encombrement des files d'attente dans les routeurs
- délai ! → perte de paquets !!
- cercle vicieux :
  . perte de paquets → retransmission → encombrement → nouvelles pertes
  = écroulement du réseau !

Contrôle de congestion :

⇒ contrôle du débit des émetteurs

. explicitement : notification de congestion (ICMP source quench message)
. implicitement : détection de pertes

TCP utilise la fenêtre coulissante :

- fenêtre_de_la_connexion = min(fenêtre_de_contrôle_flux, fenêtre_de_congestion)

Mécanismes de contrôle :

- "multiplicative decrease"
  . à chaque perte la largeur de la fenêtre est divisée par 2
- "slow start"
  . à chaque segment acquitté, la fenêtre est augmentée de la valeur du segment
  . "additive increase"
- "congestion avoidance"
  . passé un certain seuil (ou dès la 1ère perte), le rythme d'augmentation de la fenêtre est ralenti
  . elle n'est incrémentée d'un segment que si tous les segments de la fenêtre ont été acquittés