Équipe d'ingénierie de l'Internet (IETF)
Request for Comments : 9293
STD 7
RFC rendues obsolètes : 793, 879, 2873, 6093, 6429, 6528, 6691
RFC mises à jour : 1011, 1122, 5961
Catégorie : Sur la voie de la normalisation
ISSN: 2070-1721

Wesley M. Eddy, MTI Systems

août 2022

Traduction Claude Brière de L'Isle

Protocole de contrôle de transmission (TCP)

Résumé
Ce document spécifie le protocole de contrôle de transmission (TCP, *Transmission Control Protocol*). TCP est un protocole de couche de transport important dans la pile de protocoles Internet, et il n'a cessé d'évoluer au cours de décennies d'utilisation et de croissance de l'Internet. Au cours de cette période, un certain nombre de changements ont été apportés à TCP tel qu'il était spécifié dans la RFC 793, bien que ceux-ci n'aient été documentés que de manière fragmentaire. Le présent document collecte et rassemble ces modifications avec la spécification de protocole de la RFC 793. Le présent document rend obsolète la RFC 793, ainsi que les RFC 879, 2873, 6093, 6429, 6528 et 6691 qui ont mis à jour certaines parties de la RFC 793. Il met à jour les RFC 1011 et 1122, et il devrait être considéré comme un remplacement pour les parties de ces documents qui traitent des exigences de TCP. Il met également à jour la RFC 5961 en ajoutant une petite précision sur le traitement de la réinitialisation dans l'état SYN-RECEIVED. Les bits de contrôle de l'en-tête TCP de la RFC 793 ont également été mis à jour sur la base de la RFC 3168.

Table des matières

1. Objet et portée

En 1981,la [RFC0793] a été publiée, documentant le protocole de contrôle de transmission (TCP, *Transmission Control Protocol*) et remplaçant les spécifications publiées précédemment pour TCP.

Depuis lors, TCP a été largement mis en œuvre et a été utilisé comme protocole de transport pour de nombreuses applications sur l’Internet.

Pendant plusieurs décennies, la RFC 793 et un certain nombre d’autres documents se sont combinés pour servir de spécification de base pour TCP [RFC7414]. Au fil du temps, un certain nombre d’errata ont été déposés sur la RFC 793. Des lacunes ont également été constatées et résolues en matière de sécurité, de performances et de nombreux autres aspects. Le nombre d’améliorations a augmenté au fil du temps dans de nombreux documents distincts. Ceux-ci n’ont jamais été regroupés dans une mise à jour complète de la spécification de base.

L’objectif du présent document est de rassembler tous les changements sur la voie de la normalisation de l’IETF et autres précisions qui ont été apportées à la spécification fonctionnelle TCP de base (RFC0793) et de les unifier dans une version à jour de la spécification.

Certains documents d’accompagnement sont référencés pour des algorithmes importants qui sont utilisés par TCP (par exemple, pour le contrôle de l’encombrement) mais n’ont pas été complètement inclus dans ce document. Il s’agit d’un choix conscient, car cette spécification de base peut être utilisée avec plusieurs algorithmes supplémentaires développés et incorporés séparément. Ce document se concentre sur la base commune que toutes les mises en œuvre de TCP doivent prendre en charge pour pouvoir interagir. Étant donné que certaines fonctionnalités supplémentaires de TCP sont devenues assez compliquées (par exemple, la récupération évoluée des pertes et le contrôle de l'encombrement) de futurs documents d’accompagnement pourraient tenter de les rassembler de la même manière.

En plus de la spécification de protocole qui décrit le format, la génération et les règles de traitement des segments TCP qui doivent être mis en œuvre dans le code, la RFC 793 et d’autres mises à jour contiennent également du texte informatif et descriptif permettant aux lecteurs de comprendre les aspects de la conception et du fonctionnement du protocole. Le présent document n’a pas pour but de modifier ou de mettre à jour ce texte informatif et se concentre uniquement sur la mise à jour de la spécification du protocole normatif. Le présent document conserve les références à la documentation contenant les explications et justifications importantes, lorsque approprié.

Ce document est destiné à être utile à la fois pour vérifier la conformité des mises en œuvre TCP existantes et pour écrire de nouvelles mises en œuvre.

2. Introduction

La RFC 793 contient une discussion sur les objectifs de conception du TCP et fournit des exemples de son fonctionnement, y compris des exemples d'établissement de connexion, de terminaison de connexion et de retransmission de paquets pour réparer les pertes.

Le présent document décrit les fonctionnalités de base attendues dans les mises en œuvre modernes de TCP et remplace la spécification du protocole de la RFC 793. Il ne reproduit pas ou ne tente pas de mettre à jour l'introduction et le contenu philosophique des Sections 1 et 2 de la RFC 793. D'autres documents sont référencés pour fournir des explications sur la théorie de fonctionnement, les raisons, et une discussion détaillée des décisions de conception. Ce document se concentre uniquement sur le comportement normatif du protocole.

La "feuille de route de TCP" [RFC7414] fournit un guide plus complet des RFC qui définissent TCP et décrivent divers algorithmes importants. La feuille de route de TCP contient des sections sur les améliorations fortement encouragées qui améliorent les performances et d'autres aspects de TCP au-delà du fonctionnement de base spécifié dans ce document. À titre d'exemple, la mise en œuvre d'un système de contrôle de l'encombrement (par exemple, de la [RFC5681]) est une exigence de TCP, mais il s'agit d'un sujet complexe en soi qui n'est pas décrit en détail dans le présent document, car il existe de nombreuses options et possibilités qui n'ont pas d'impact sur l'interopérabilité de base. De même, la plupart des mises en œuvre de TCP incluent aujourd'hui des extensions de hautes performances [RFC7323], mais celles-ci ne sont pas strictement requises ou discutées dans le présent document. Les considérations relatives aux chemins d'accès multiples pour TCP sont également spécifiées séparément dans la [RFC8684].

Une liste des modifications par rapport à la RFC 793 est contenue dans la Section 5.

2.1 Langage des exigences

Les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDE", "PEUT", et "FACULTATIF" en majuscules dans ce document sont à interpréter comme décrit dans le BCP 14, [RFC2119], [RFC8174] quand, et seulement quand ils apparaissent tout en majuscules, comme montré ci-dessus.

Chaque utilisation des mots-clés de la RFC 2119 dans le document est étiquetée individuellement et mentionnée à l'Annexe B, qui résume les exigences de mise en œuvre.

Les phrases utilisant "DOIT" sont étiquetées comme "DOIT-X" avec un identifiant numérique X permettant de localiser facilement le besoin référencé à partir de l'Annexe B.

De même, les phrases utilisant "DEVRAIT" sont étiquetées avec "DVRT-X", "PEUT" avec "PEUT-X", et "RECOMMANDÉ" avec "REC-X".

Pour les besoins de cet étiquetage, "NE DEVRAIT PAS" et "NE DOIT PAS" sont étiquetés de la même manière que les cas de "DEVRAIT" et "DOIT".

2.2 Concepts clés de TCP

TCP fournit un service de flux d'octets fiable, dans l'ordre, aux applications.

Le flux d'octets de l'application est acheminé sur le réseau via des segments TCP, chaque segment TCP étant envoyé sous la forme d'un datagramme du protocole Internet (IP, *Internet Protocol*).

La fiabilité TCP consiste à détecter les pertes de paquets (via les numéros de séquence) et les erreurs (via des sommes de contrôle par segment) ainsi que la correction par retransmission.

TCP prend en charge la livraison de données en envoi individuel. Il existe des applications d'envoi à la cantonade qui peuvent réussir à utiliser TCP sans modification, bien qu'il y ait un certain risque d'instabilité dû à des modifications du

comportement de transfert de la couche inférieure [RFC7094].

TCP est en mode connexion, bien qu'il n'inclue pas intrinsèquement de capacité de détection de vie.

Le flux de données est pris en charge de manière bidirectionnelle sur les connexions TCP, bien que les applications soient libres d'envoyer des données en unidirectionnel, si tel est leur choix.

TCP utilise les numéros d'accès pour identifier les services d'application et pour multiplexer des flux distincts entre les hôtes.

Une description plus détaillée des fonctionnalités de TCP par rapport aux autres protocoles de transports se trouve au paragraphe 3.1 de la [RFC8095]. Plus de description des motivations du développement de TCP et de son rôle dans la pile de protocoles Internet se trouve à la Section 2 de la [RFC0793] et dans les versions antérieures de la spécification TCP.

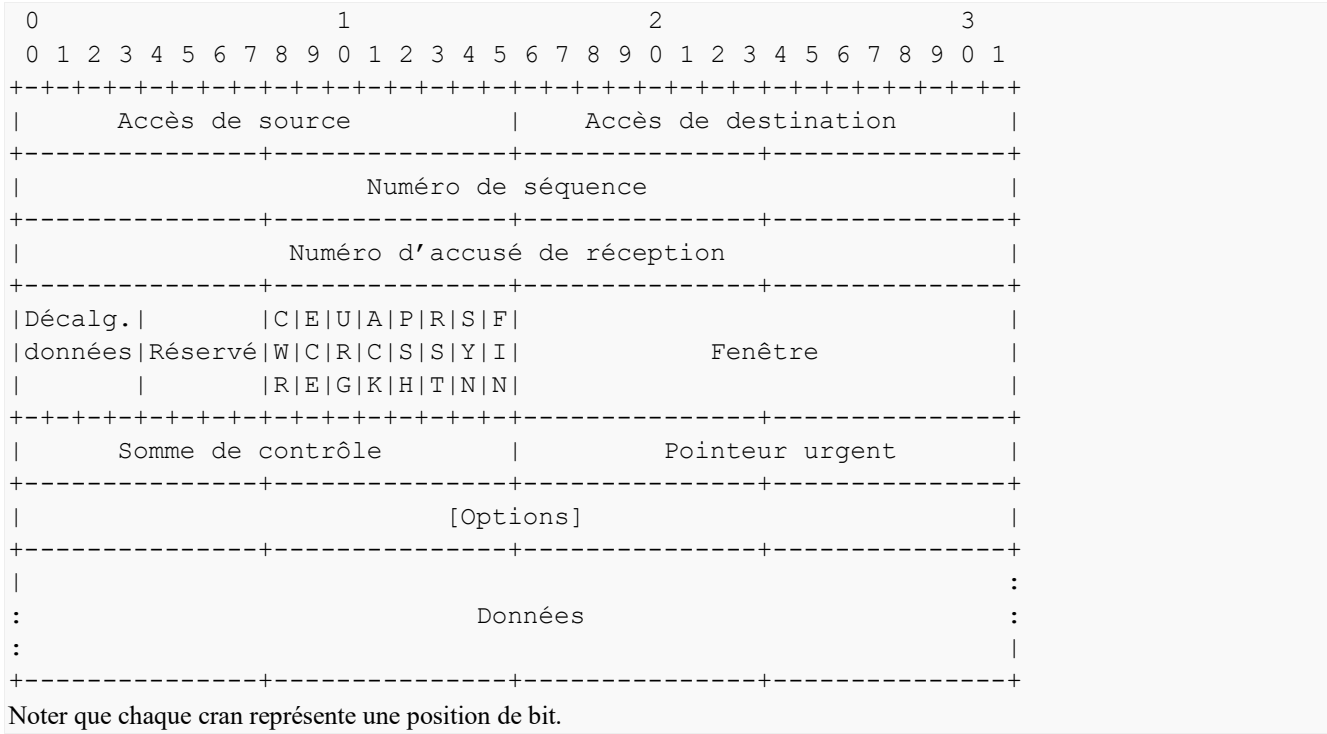
3. Spécification fonctionnelle

3.1 Format de l'en-tête

Les segments TCP sont envoyés sous forme de datagrammes Internet. L'en-tête de protocole Internet (IP) porte plusieurs champs d'information, y compris les adresses de source et de destination d'hôte [RFC0791], [RFC8200]. Un en-tête TCP suit les en-têtes IP, fournissant des informations spécifiques de TCP. Cette division permet l'existence de protocoles au niveau de l'hôte autres que TCP. Au début du développement de la suite de protocoles Internet, les champs d'en-tête IP faisaient partie de TCP.

Ce document décrit TCP, qui utilise des en-têtes TCP.

Un en-tête TCP, suivi de toutes les données d'utilisateur du segment, est formaté comme suit, en utilisant le style de [66] :



Noter que chaque cran représente une position de bit.

Figure 1 : Format de l'en-tête TCP

où :

Accès de source : 16 bits. Numéro de l'accès de source.

Accès de destination : 16 bits. Numéro de l'accès de destination.

Numéro de séquence : 32 bits. Numéro de séquence du premier octet de données de ce segment (sauf lorsque le fanion SYN est activé). Si SYN est établi, le numéro de séquence est le numéro de séquence initial (ISN, *initial sequence number*) et le premier octet de données est ISN+1.

Numéro d'accusé de réception : 32 bits. Si le bit de contrôle ACK est établi, ce champ contient la valeur du prochain numéro de séquence que l'envoyeur du segment s'attend à recevoir. Une fois une connexion établie, ceci est toujours envoyé.

Décalage de données (DOffset) : 4 bits. Nombre de mots de 32 bits dans l'en-tête TCP. Cela indique où les données commencent. L'en-tête TCP (même s'il inclut des options) est un multiple entier de 32 bits.

Réservé (rsrvd) : 4 bits. Un ensemble de bits de contrôle réservés pour une utilisation future. Doit être à zéro dans les segments générés et doit être ignoré dans les segments reçus si les fonctionnalités futures correspondantes ne sont pas mises en œuvre par l'hôte envoyeur ou receveur.

Bits de contrôle : les bits de contrôle sont aussi appelés "fanions". L'allocation est gérée par l'IANA à partir du registre"Fanions d'en-tête TCP" [IANA]. Les bits de contrôle actuellement attribués sont CWR, ECE, URG, ACK, PSH, RST, SYN et FIN.
CWR : 1 bit. Réduction de la fenêtre d'encombrement (voir [RFC3168]).
ECE : 1 bit. ECN-Echo (voir [RFC3168]).
URG : 1 bit. Le champ de pointeur urgent est important.
ACK : 1 bit. Le champ Accusé de réception est important.
PSH : 1 bit. Fonction Push (voir la description de l'envoi d'un appel au paragraphe 3.9.1).
DT : 1 bit. Réinitialiser la connexion.
SYN : 1 bit. Synchroniser les numéros de séquence.
FIN : 1 bit. Pas d'autres données de l'envoyeur.

Fenêtre : 16 bits. Le nombre d'octets de données en commençant par celui indiqué dans le champ d'accusé de réception que l'envoyeur de ce segment est disposé à accepter. La valeur est décalée lorsque l'extension de mise à l'échelle de fenêtre est utilisée [RFC7323]. La taille de la fenêtre DOIT être traitée comme un nombre non signé, ou bien les grandes tailles de fenêtre vont apparaître comme des fenêtres négatives et TCP ne va pas fonctionner (DOIT-1). Il est RECOMMANDÉ que les mises en œuvre réservent des champs de 32 bits pour les tailles de fenêtre d'envoi et de réception dans l'enregistrement de connexion et effectuent tous les calculs de fenêtre avec 32 bits (REC-1).

Somme de contrôle: 16 bits. Le champ de somme de contrôle est le complément à un de 16 bits de la somme des compléments à un de tous les mots de 16 bits dans l'en-tête et le texte. Le calcul de la somme de contrôle doit garantir l'alignement sur 16 bits des données additionnées. Si un segment contient un nombre impair d'octets d'en-tête et de texte, l'alignement peut être réalisé en bourrant le dernier octet avec des zéros sur sa droite pour former un mot de 16 bits pour les besoins de la somme de contrôle. Le bourrage n'est pas transmis au titre du segment. Lors du calcul de la somme de contrôle, le champ "Checksum" lui-même est remplacé par des zéros.

La somme de contrôle couvre également un pseudo-en-tête (Figure 2) préfixé conceptuellement à l'en-tête TCP. Le pseudo-en-tête est de 96 bits pour IPv4 et de 320 bits pour IPv6. L'inclusion du pseudo-en-tête dans la somme de contrôle donne la protection de la connexion TCP contre les segments mal acheminés. Ces informations sont transportées dans les en-têtes IP et sont transférées via l'interface TCP/réseau dans les arguments ou résultats des appels par la mise en œuvre TCP sur la couche IP.

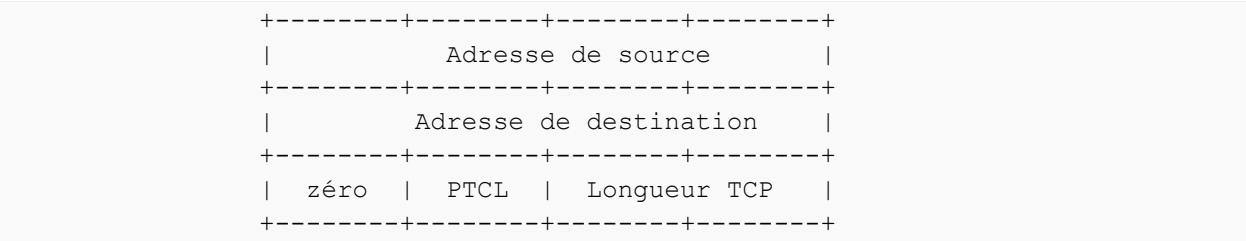


Figure 2 : Pseudo-en-tête IPv4

Composants de pseudo-en-tête pour IPv4 :
Adresse de source : l'adresse de source IPv4 dans l'ordre des octets du réseau
Adresse de destination : l'adresse de destination IPv4 dans l'ordre des octets du réseau

zéro : bits mis à zéro
PTCL : numéro de protocole de l'en-tête IP
Longueur TCP : longueur de l'en-tête TCP plus longueur des données en octets (cette quantité n'est pas transmise explicitement, mais est calculée) et ne compte pas les 12 octets du pseudo-en-tête. Pour IPv6, le pseudo-en-tête est défini au paragraphe 8.1 de la [RFC8200] et contient l'adresse de source et l'adresse de destination IPv6, une longueur de paquet de couche supérieure (valeur de 32 bits autrement équivalente à la longueur TCP dans le pseudo-en-tête IPv4) trois octets de bourrage de zéros et une valeur de prochain en-tête, qui diffère de la valeur d'en-tête IPv6 s'il existe des en-têtes d'extension entre IPv6 et TCP.

La somme de contrôle TCP n'est jamais facultative. L'envoyeur DOIT la générer (DOIT-2) et le receveur DOIT la vérifier (DOIT-3).

Pointeur urgent : 16 bits. Ce champ communique la valeur actuelle du pointeur urgent sous la forme d'un décalage positif par rapport au numéro de séquence dans ce segment. Le pointeur urgent pointe sur le numéro de séquence de l'octet qui suit les données urgentes. Ce champ ne doit être interprété que dans les segments qui ont le bit de contôle URG établi.

Options : [Option TCP] ; taille(Options) == (DOffset-5)*32 ; présent seulement quand DOffset > 5. Noter que cette expression de taille inclut aussi tout bourrage à la fin des options réelles présentes. Les options peuvent occuper de l'espace à la fin de l'en-tête TCP et sont des multiples de 8 bits. Toutes les options sont incluses dans la somme de contrôle. Une option peut commencer à n'importe quelle limite d'octet. Il y a deux cas pour le format d'une option :
Cas 1 : Un seul octet de type option.
Cas 2 : Un octet de type d'option (Type), un octet de longueur d'option, et les octets de données d'option réels.
La longueur d'option compte les deux octets de type d'option et de longueur d'option ainsi que les octets de données d'option.

Noter que la liste des options peut être plus courte que ce que le champ Décalage de données pourrait impliquer. Le contenu de l'en-tête au-delà de la Fin de l'option Liste d'options DOIT être le bourrage de zéros de l'en-tête (DOIT-69).
La liste de toutes les options actuellement définies est gérée par l'IANA [IANA], et chaque option est définie dans d'autres RFC, comme indiqué ici. Cet ensemble comprend des options expérimentales qui peuvent être étendues pour prendre en charge plusieurs utilisations simultanées [RFC6994].

Une mise en œuvre donnée de TCP peut prendre en charge toute option actuellement définie, mais les options suivantes DOIVENT être prises en charge (DOIT-4 -- noter que la prise en charge de l'option Taille de segment maximale fait également partie de DOIT-14 au paragraphe 3.7.1) :

Tableau 1 : Ensemble d'options obligatoires

Type	Longueur	Signification
0	-	Fin de l'option Liste d'options.
1	-	Non fonctionnement.
2	4	Taille maximale de segment.

Ces options sont spécifiées en détail au paragraphe 3.2.

Une mise en œuvre de TCP DOIT être capable de recevoir une option TCP dans n'importe quel segment (DOIT-5).

Une mise en œuvre de TCP DOIT (DOIT-6) ignorer sans erreur toute option TCP qu'elle ne met pas en œuvre, en supposant que l'option comporte un champ Longueur. Toutes les options TCP, à l'exception de l'option Fin de liste d'options (EOL) et de l'option Non fonctionnement (NOP) DOIT AVoir des champs de longueur, y compris toutes les options futures (DOIT-68). Les mises en œuvre de TCP DOIVENT être prêtes à gérer une longueur d'option illégale (par exemple, zéro) ; une procédure suggérée consiste à réinitialiser la connexion et enregistrer la cause de l'erreur (DOIT-7).

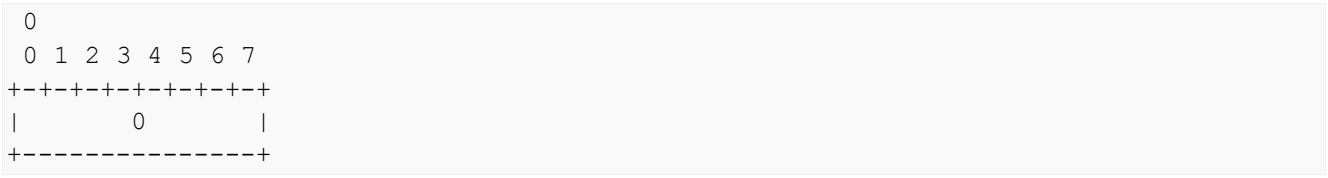
Note : Des travaux sont en cours pour étendre l'espace disponible pour les options TCP, tels que [65].

Données : Longueur variable. Données d'utilisateur portées par le segment TCP.

3.2 Définitions des options spécifiques

Une option TCP, dans l'ensemble d'options obligatoires, est une des options Fin de liste d'options, Non fonctionnement ou Taille maximum de segment.

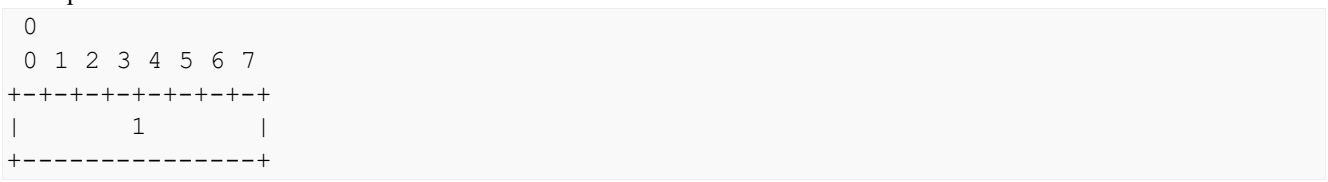
Une option Fin de liste d'options est formatée comme suit :



où :

Type : 1 octet ; type == 0. Ce code d'option indique la fin de la liste des options. Cela peut ne pas coïncider avec la fin de l'en-tête TCP selon le champ Décalage des données. C'est utilisé à la fin de toutes les options, pas à la fin de chaque option, et ne doit être utilisé que si autrement la fin des options ne coïncideraient pas avec la fin de l'en-tête TCP.

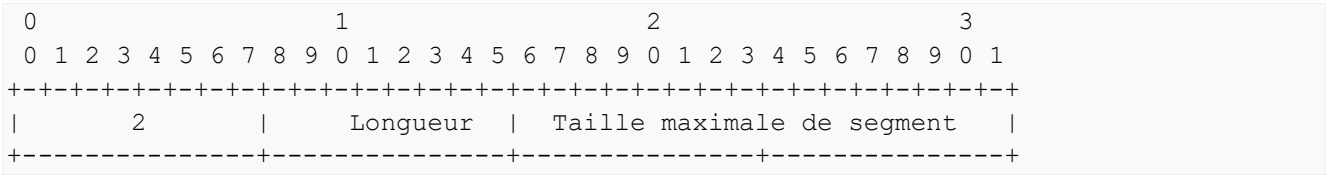
Une option Non fonctionnement est formatée comme suit :



où :

Type : 1 octet ; type == 1. Ce code d'option peut être utilisé entre les options, par exemple, pour aligner le début d'une option suivante sur une limite de mot. Il n'y a aucune garantie que les envoyeurs utiliseront cette option, donc les receveurs DOIVENT être prêts à traiter les options même si elles ne commencent pas sur une limite de mot (DOIT-64).

Une option Taille maximale de segment est formatée comme suit :



où:

Type : 1 octet ; type == 2. Si cette option est présente, elle communique la taille maximum du segment en réception au point d'extrémité TCP qui envoie ce segment. Cette valeur est limitée par la limite de réassemblage IP. Ce champ peut être envoyé dans la demande de connexion initiale (c'est-à-dire dans des segments avec le bit de contrôle SYN établi) et NE DOIT PAS être envoyé dans d'autres segments (DOIT-65). Si cette option n'est pas utilisée, toute taille de segment est permise. Une description plus complète de cette option est fournie au paragraphe 3.7.1.

Longueur : 1 octet ; Longueur == 4. Longueur de l'option en octets.

Taille maximale du segment (MSS) : 2 octets. Taille maximale du segment de réception au point d'extrémité TCP qui envoie ce segment.

3.2.1 Autres options courantes

D'autres RFC définissent d'autres options couramment utilisées qui sont de mise en œuvre recommandée pour des performances élevées, mais qui ne sont pas nécessaires pour l'interopérabilité TCP de base. Il s'agit de l'option TCP d'accusé de réception sélectif (SACK) [RFC2018], [RFC2883], de l'option TCP d'horodatage (TS) [RFC7323] et de l'option TCP Échelle de fenêtre (WS) [RFC7323].

3.2.2 Options TCP expérimentales

Les valeurs d'options TCP expérimentales sont définies dans la [RFC4727] et la [RFC6994] décrit l'utilisation actuellement recommandée pour ces valeurs expérimentales.

3.3 Aperçu de la terminologie TCP

Cette section comprend une vue d'ensemble des termes clés nécessaires pour comprendre le fonctionnement détaillé du protocole dans le reste du document. La Section 4 contient un glossaire des termes.

3.3.1 Variables clés de l'état de la connexion

Avant de pouvoir discuter en détail du fonctionnement de la mise en œuvre de TCP, on doit introduire une terminologie détaillée. La maintenance d'une connexion TCP nécessite de conserver l'état de plusieurs variables. On conçoit la mémorisation de ces variables dans un enregistrement de connexion appelé Bloc de contrôle de transmission (TCB, *Transmission Control Block*). Parmi les variables mémorisées dans le TCB sont les adresses IP locales et distantes et les numéros d'accès, le niveau de sécurité IP, et le compartiment de la connexion (voir à l'Annexe A.1) des pointeurs sur les mémoires tampon d'envoi et de réception de l'utilisateur, des pointeurs sur la file d'attente de retransmission et sur le segment actuel. De plus, plusieurs variables relatives aux numéros de séquence d'envoi et de réception sont mémorisées dans le TCB.

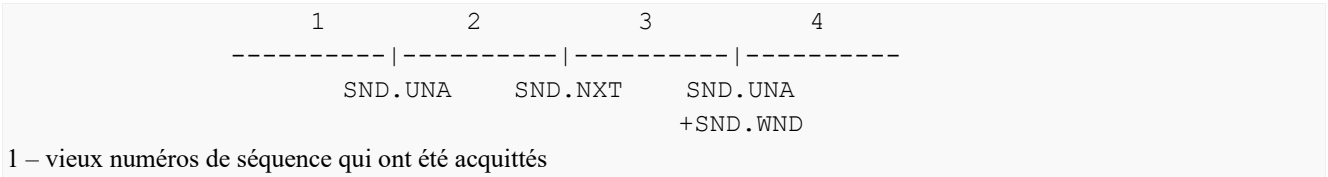
Tableau 2 : Variables de séquence d'envoi

Variable	Description
SND.UNA	Envoi sans accusé de réception
SND.NXT	Envoi suivant
SND.WND	Fenêtre d'envoi
SND.UP	Pointeur d'envoi urgent
SND.WL1	Numéro de séquence de segment utilisé pour la dernière mise à jour de la fenêtre
SND.WL2	Numéro d'accusé de réception du segment utilisé pour la dernière mise à jour de la fenêtre
ISS	Numéro de séquence d'envoi initial

Tableau 3 : Variables de séquence de réception

Variable	Description
RCV.NXT	Réception suivante
RCV.WND	Fenêtre de réception
RCV.UP	Pointeur de réception urgente
IRS	Numéro de séquence de réception initial

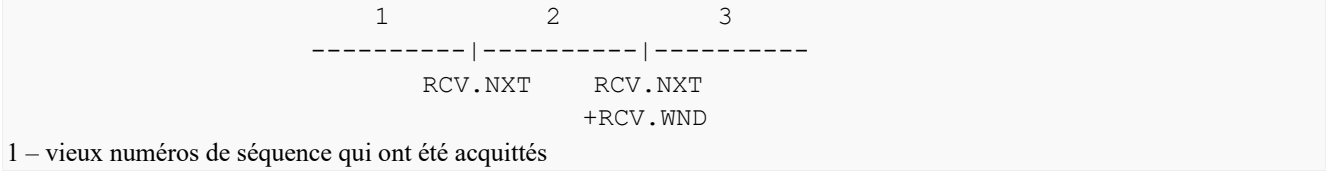
Les diagrammes suivants peuvent aider à relier certaines de ces variables à l'espace de séquence.



1 – vieux numéros de séquence qui ont été acquittés
2 – numéros de séquence de données non acquittées
3 – numéros de séquence permis pour de nouvelles transmissions de données
4 – futurs numéros de séquence qui ne sont pas encore permis

Figure 3 : Espace de séquence d'envoi

La fenêtre d'envoi est la partie de l'espace de séquence marquée 3 dans la Figure 3.



1 – vieux numéros de séquence qui ont été acquittés

2 – numéros de séquence permis pour de nouvelles réceptions
3 – futurs numéros de séquence qui ne sont pas encore permis

Figure 4 : Espace de séquence de réception

La fenêtre de réception est la partie de l'espace de séquence marqué 2 dans la Figure 4.

Il y a aussi des variables fréquemment utilisées dans la discussion qui prennent leurs valeurs des champs du segment actuel.

Tableau 4 : Variables du segment actuel

Variable	Description
SEG.SEQ	Numéro de séquence du segment
SEG.ACK	Numéro d'accusé de réception de segment
SEG.LEN	Longueur du segment
SEG.WND	Fenêtre de segment
SEG.UP	Pointeur de segment urgent

3.3.2 Vue d'ensemble de l'automate d'état

Une connexion progresse à travers une série d'états au cours de sa vie. Les états sont : LISTEN, SYN-SENT, SYN-RECEIVED, ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT, et l'état fictif CLOSED. CLOSED est fictif parce qu'il représente l'état lorsqu'il n'y a pas de TCB, et donc, pas de connexion. En bref, les significations des états sont les suivantes :

LISTEN - représente l'attente d'une demande de connexion à partir de n'importe quel homologue et accès TCP.

SYN-SENT - représente l'attente d'une demande de connexion correspondante après avoir envoyé une demande de connexion.

SYN- RECEIVED - représente l'attente d'un accusé de réception de confirmation de demande de connexion après avoir reçu et envoyé une demande de connexion.

ESTABLISHED - représente une connexion ouverte, les données reçues peuvent être livrées à l'utilisateur. L'état normal de la phase de transfert de données de la connexion.

FIN-WAIT-1 - représente l'attente d'une demande de terminaison de connexion de la part de l'homologue TCP distant ou d'un accusé de réception de la demande de terminaison de connexion précédemment envoyée.

FIN-WAIT-2 - représente l'attente d'une demande de terminaison de connexion de la part de l'homologue TCP distant.

CLOSE-WAIT - représente l'attente d'une demande de terminaison de connexion de l'utilisateur local.

CLOSING - représente l'attente d'un accusé de réception de demande de terminaison de connexion de l'homologue TCP distant.

LAST-ACK - représente l'attente d'un accusé de réception de la demande de terminaison de connexion précédemment envoyée à l'homologue TCP distant (cette demande de terminaison envoyée à l'homologue TCP distant comprenait déjà un accusé de réception de la demande de terminaison envoyée de l'homologue TCP distant).

TIME-WAIT - représente l'attente d'un temps suffisant pour être sûr que l'homologue TCP distant a reçu l'accusé de réception de sa demande de terminaison de connexion et pour éviter que les nouvelles connexions ne soient affectées par des retards de segments des connexions précédentes.

CLOSED - ne représente aucun état de connexion.

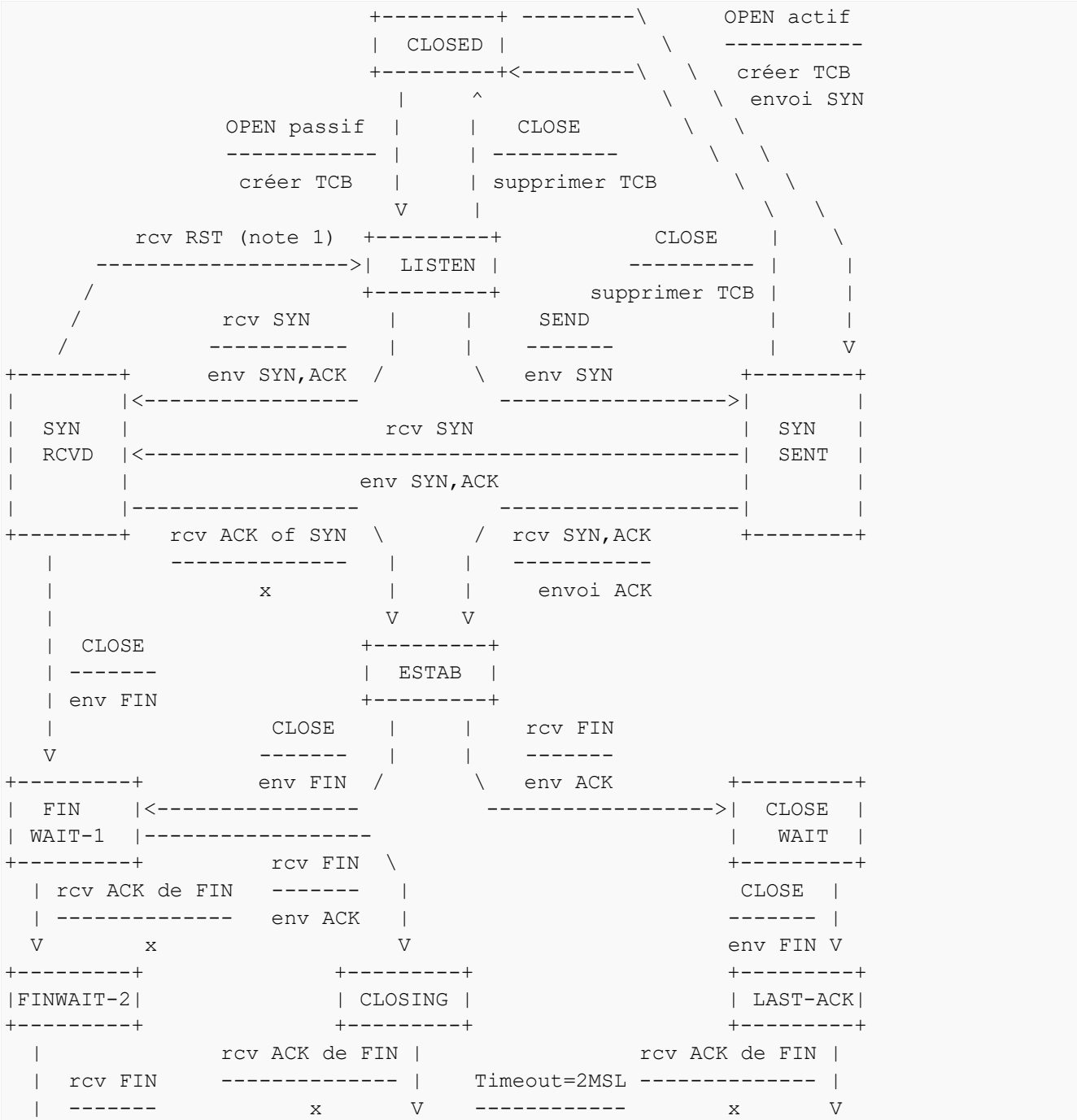
Une connexion TCP passe d'un état à un autre en réponse à des événements. Les événements sont les appels de l'utilisateur, OPEN, SEND, RECEIVE, CLOSE, ABORT et STATUS ; les segments entrants, en particulier ceux contenant les fanions

SYN, ACK, RST et FIN ; et les fins de temporisation.

L'appel OPEN spécifie si l'établissement de connexion doit être activement recherché ou être attendu passivement.
Une demande OPEN passive signifie que le processus veut accepter les demandes de connexion entrantes, contrairement à un OPEN actif qui tente d'initier une connexion.

Le diagramme d'état de la Figure 5 illustre uniquement les changements d'état, avec les événements générateurs et les actions qui en résultent, mais n'aborde ni les conditions d'erreur ni les actions résultantes qui ne sont pas liées à des changements d'état. Dans une section ultérieure, on donne plus de détails sur la réaction de la mise en œuvre de TCP aux événements. Certains noms d'état sont abrégés ou coupés différemment dans le diagramme par rapport à la façon dont ils apparaissent ailleurs dans le document.

Note : Ce schéma n'est qu'un résumé et ne doit pas être considéré comme la spécification totale. De nombreux détails ne sont pas inclus.



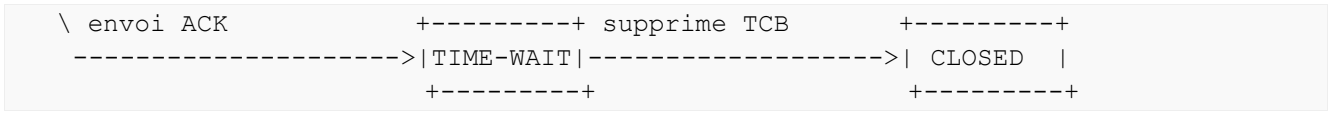


Figure 5 : Diagramme d'état de connexion TCP

Les notes suivantes s'appliquent à la Figure 5 :

Note 1 : La transisition de SYN-RECEIVED à LISTEN à réception d'un RST est à condition d'avoir atteint SYN-RECEIVED après un OPEN passif.

Note 2 : La figure omet une transition de FIN-WAIT-1 à TIME-WAIT si un FIN est reçu et si le FIN local est aussi acquitté.

Note 3 : Un RST peut être envoyé à partir de n'importe quel état avec une transition correspondante à TIME-WAIT (voir [70] pour les raisons. Ces transitions ne sont pas explicitement montrées ; autrement, le schéma deviendrait très difficile à lire. De même, la réception d'un RST de n'importe quel état entraîne une transition vers LISTEN ou CLOSED, bien que cela soit également omis du diagramme pour des raisons de lisibilité.

3.4 Numéros de séquence

Une notion fondamentale dans la conception est que chaque octet de données envoyé sur une connexion TCP a un numéro de séquence. Comme chaque octet est séquencé, chacun d'eux peut être acquitté. Le mécanisme d'acquittement employé est cumulatif de sorte qu'un accusé de réception de numéro de séquence X indique que tous les octets jusqu'à X non inclus ont été reçus. Ce mécanisme permet une détection de duplication simple en présence de retransmissions. Le schéma de numérotation des octets à l'intérieur d'un segment est le suivant : le premier octet de données qui suit immédiatement l'en-tête a le plus petit numéro et les octets suivants sont numérotés consécutivement.

Il est essentiel de se rappeler que l'espace réel des numéros de séquence est fini, bien que grand. Cet espace va de 0 à 2^{32} - 1. Comme l'espace est fini, toute l'arithmétique qui traite des numéros de séquence doit être effectuée modulo 2^{32}. Cette arithmétique non signée préserve la relation des numéros de séquence lorsqu'ils repassent de 2^{32} - 1 à 0. Il y a quelques subtilités à l'arithmétique modulo par ordinateur, il faut donc faire très attention pour la programmation de la comparaison de ces valeurs. Le symbole "≤" signifie "inférieur ou égal" (modulo 2^{32}).

Les types normaux de comparaisons de numéros de séquence que la mise en œuvre de TCP doit effectuer comprennent :

a) Déterminer qu'un accusé de réception fait référence à un numéro de séquence envoyé mais pas encore acquitté.
b) Déterminer que tous les numéros de séquence occupés par un segment ont fait l'objet d'un accusé de réception (par exemple, pour retirer le segment d'une file d'attente de retransmission).
c) Déterminer qu'un segment entrant contient des numéros de séquence qui sont attendus (c'est-à-dire, que le segment "chevauche" la fenêtre de réception).

En réponse à l'envoi de données, le point d'extrémité TCP reçoit des accusés de réception. Les comparaisons suivantes sont nécessaires pour traiter les accusés de réception :

SND.UNA = plus ancien numéro de séquence non acquitté
SND.NXT = numéro de séquence suivant à envoyer
SEG.ACK = accusé de réception de l'homologue TCP receveur (numéro de séquence suivant attendu par l'homologue TCP receveur)
SEG.SEQ = premier numéro de séquence d'un segment
SEG.LEN = nombre d'octets occupés par les données du segment (en comptant SYN et FIN)
SEG.SEQ+SEG.LEN-1 = dernier numéro de séquence d'un segment

Un nouvel accusé de réception (appelé "accusé de réception acceptable") est un accusé de réception pour lequel l'inégalité ci-dessous est vérifiée : SND.UNA < SEG.ACK ≤ SND.NXT

Un segment de la file d'attente de retransmission fait l'objet d'un accusé de réception complet si la somme de son numéro de séquence et de sa longueur est inférieure ou égale à la valeur de l'accusé de réception dans le segment entrant.

Lorsque des données sont reçues, les comparaisons suivantes sont nécessaires :

RCV.NXT = numéro de séquence suivant attendu sur un segment entrant, et est le bord gauche ou inférieur de la fenêtre de réception
RCV.NXT+RCV.WND-1 = dernier numéro de séquence attendu sur un segment, et constitue le bord droit ou supérieur de la fenêtre de réception
SEG. SEQ = premier numéro de séquence occupé par le segment entrant

SEG.SEQ+SEG.LEN-1 = dernier numéro de séquence occupé par le segment entrant

Un segment est considéré comme occupant une partie de l'espace de séquence de réception valide si
RCV.NXT ≤ SEG.SEQ < RCV.NXT+RCV.WND
ou
RCV.NXT ≤ SEG.SEQ+SEG.LEN-1 < RCV. XT+RCV.WND

La première partie de cet essai vérifie si le début du segment tombe dans la fenêtre, la deuxième partie de l'essai vérifie si la fin du segment tombe dans la fenêtre ; si le segment réussit à l'une ou l'autre partie de l'essai, il contient des données dans la fenêtre.

En fait, c'est un peu plus compliqué que cela. En raison de fenêtres zéro et de segments de longueur nulle, on a quatre cas pour l'acceptabilité d'un segment entrant :

Tableau 5 : Essais d'acceptabilité des segments

Longueur du segment	Fenêtre de réception	Essai
0	0	SEG.SEQ = RCV.NXT
0	>0	RCV.NXT ≤ SEG.SEQ < RCV.NXT+RCV.WND
>0	0	Inacceptable
>0	>0	RCV.NXT ≤ SEG.SEQ < RCV.NXT+RCV.WND ou RCV.NXT ≤ SEG.SEQ+SEG.LEN-1 < RCV.NXT+RCV. ND

Noter que lorsque la fenêtre de réception est nulle, aucun segment ne doit être acceptable, à l'exception des segments ACK. Donc, il est possible à la mise en œuvre TCP de tenir une fenêtre de réception nulle tout en transmettant des données et en recevant des ACK. Un receveur TCP DOIT traiter les champs RST et URG de tous les segments entrants, même lorsque la fenêtre de réception est nulle (DOIT-66).

On a tiré parti du schéma de numérotation pour protéger également certaines informations de contrôle. Ceci est réalisé en incluant implicitement des fanions de contrôle dans l'espace de séquence afin qu'ils puissent être retransmis et reconnus sans confusion (c'est-à-dire, une seule et unique copie du contrôle sera mise en œuvre). Les informations de contrôle ne sont pas physiquement portées dans l'espace de données du segment. Par conséquent, on doit adopter des règles pour allouer implicitement des numéros de séquence au contrôle. Le SYN et le FIN sont les seuls contrôles nécessitant cette protection, et ces contrôles ne sont utilisés qu'à l'ouverture et à la fermeture de la connexion. Pour les besoins du numéro de séquence, le SYN est considéré comme se produisant avant les premières données réelles du segment dans lequel il se produit, tandis que le FIN est considéré se produire après le dernier octet de données réel dans le segment dans lequel il survient. La longueur du segment (SEG.LEN) comporte à la fois les données et la séquence des commandes occupant l'espace. Lorsqu'un SYN est présent, alors SEG.SEQ est le numéro de séquence du SYN.

3.4.1 Sélection initiale du numéro de séquence

Une connexion est définie par une paire de prises. Les connexions peuvent être réutilisées. Les nouvelles instances d'une connexion sont appelées des incarnations de la connexion. Le problème qui en découle est le suivant -- "comment la mise en œuvre de TCP identifie-t-elle les segments en double provenant des précédentes incarnations de la connexion ? Ce problème devient apparent si la connexion s'ouvre et se ferme en successions rapides, ou si la connexion est interrompue avec perte de mémoire et est ensuite rétablie. Pour prendre cela en charge, l'état TIME-WAIT limite le taux de réutilisation de la connexion, tandis que le choix initial du numéro de séquence décrit ci-dessous protège davantage contre l'ambiguïté sur à quelle incarnation d'une connexion correspond un paquet entrant.

Pour éviter toute confusion, on doit empêcher les segments d'une incarnation d'une connexion d'être utilisés alors que les mêmes numéros de séquence peuvent toujours être présents dans le réseau à partir d'une incarnation antérieure. On veut assurer cela même si un point d'extrémité TCP perd toute connaissance des numéros de séquence qu'il a utilisés. Lorsque de nouvelles connexions sont créées, un générateur de numéro de séquence initial (ISN, *initial sequence number*) est utilisé qui choisit un nouvel ISN de 32 bits. Il y a des problèmes de sécurité qui se produisent si un l'attaquant est capable de prédire ou de deviner les valeurs de l'ISN [RFC6528].

Les numéros de séquence initiaux TCP sont générés à partir d'une séquence de numéros qui augmente de manière

monotone jusqu'à ce qu'ils reviennent à zéro, appelé de façon vague une "horloge". Cette horloge est un compteur de 32 bits qui s'incrémente normalement au moins une fois toutes les environ 4 microsecondes, bien qu'il ne soit pas supposé qu'il s'agisse d'un temps précis ni réel, et n'a pas besoin de persister d'un redémarrage à l'autre. Le composant "horloge" est destiné à s'assurer qu'avec une durée de vie maximale de segment (MSL, *Maximum Segment Lifetime*) les ISN générés seront uniques puisqu'ils effectuent un cycle d'environ 4,55 heures, ce qui est beaucoup plus long que la MSL. Noter que pour les réseaux modernes qui prennent en charge des débits de données élevés où la connexion peut commencer et faire avancer rapidement les numéros de séquence au sein de la MSL, il est recommandé de mettre en œuvre l'option d'horodatage mentionnée plus loin au paragraphe 3.4.3.

Une mise en œuvre de TCP DOIT utiliser le type "d'horloge" ci-dessus pour le choix des numéros de séquence initiaux à l'aide d'une horloge (DOIT-8), et DEVRAIT générer ses numéros de séquence initiaux avec l'expression :

ISN = M + F(localip, localport, remoteip, remoteport, secretkey)

où M est le temporisateur de 4 microsecondes, et F() est une fonction pseudo-aléatoire (PRF, *pseudorandom function*) des paramètres d'identification de la connexion ("localip, localport, remoteip, remoteport") et une clé secrète ("secretkey") (DVRT-1). F() NE DOIT PAS être calculable de l'extérieur (DOIT-9) ou un attaquant pourrait alors deviner les numéros de séquence à partir de l'ISN utilisé pour une autre connexion. La PRF pourrait être mise en œuvre sous la forme d'un hachage cryptographique de l'enchaînement des paramètres de connexion TCP et de certaines données secrètes. Pour plus d'informations sur le choix d'un algorithme de hachage spécifique et la gestion des données de clé secrète, voir la Section 3 de la [RFC6528].

Pour chaque connexion, il y a un numéro de séquence d'envoi et un numéro de séquence de réception. Le numéro de séquence d'envoi initial (ISS, *Initial Send Sequence number*) est choisi par l'homologue TCP envoyeur des données, et le numéro de séquence de réception initial (IRS, *Initial Receive Sequence number*) est appris lors de la procédure d'établissement de la connexion.

Pour qu'une connexion soit établie ou initialisée, les deux homologues TCP doivent synchroniser leurs numéros de séquence initiaux. Cela se fait dans un échange de segments d'établissement de connexion portant un bit de contrôle appelé "SYN" (pour synchroniser) et les numéros de séquence initiaux. En abrégé, les segments portant le bit SYN sont également appelés des "SYN". Par conséquent, la solution nécessite un mécanisme approprié pour prendre un numéro de séquence initial et une prise de contact légèrement compliquée pour l'échange les ISN.

La synchronisation exige que chaque partie envoie son propre numéro de séquence initial et reçoive une confirmation par un accusé de réception de l'homologue TCP distant. Chaque côté doit aussi recevoir le numéro de séquence initial de l'homologue distant et envoyer un accusé de réception le confirmant.

```
1) A --> B  SYN mon numéro de séquence est X
2) A <-- B  ACK ton numéro de séquence est X
3) A <-- B  SYN mon numéro de séquence est Y
4) A --> B  ACK ton numéro de séquence est Y
```

Comme les étapes 2 et 3 peuvent être combinées dans un seul message, c'est appelé la prise de contact à trois phases (ou à trois messages) (3WHS, *3 Ways HandShake*).

Une 3WHS est nécessaire car les numéros de séquence ne sont pas liés à une horloge globale dans le réseau, et les mises en œuvre de TCP peuvent avoir des mécanismes différents de choix des ISN. Le receveur du premier SYN n'a aucun moyen de savoir si le segment était ancien ou non, à moins qu'il se souvienne du dernier numéro de séquence utilisé sur la connexion (ce qui n'est pas toujours possible) et doit donc demander à l'envoyeur de vérifier ce SYN. La prise de contact à trois phases et les avantages d'un schéma fondé sur l'horloge pour le choix de l'ISN sont abordés dans [69].

3.4.2 Savoir quand se taire

Il existe un problème théorique où les données pourraient être corrompues en raison de la confusion entre les anciens segments du réseau et les nouveaux dans le réseau après le réamorçage d'un hôte si les mêmes numéros d'accès et le même espace de numéros de séquence sont réutilisés. Le concept de "temps de silence" discuté ci-dessous aborde ce sujet, et la discussion inclut des situations où cela pourrait être pertinent, même si l'on ne pense pas qu'il soit nécessaire dans la plupart des mises en œuvre actuelles. Le problème était plus pertinent plus tôt dans l'histoire de TCP. Dans la pratique de l'Internet d'aujourd'hui, les conditions enclines à l'erreur sont suffisamment improbables pour qu'il soit sûr de l'ignorer.

Les raisons pour lesquelles il est aujourd'hui négligeable sont les suivantes : a) l'ISS et l'accès éphémère aléatoire ont réduit la probabilité de réutilisation des numéros d'accès et des numéros de séquence après les redémarrages, (b) la MSL effective de l'Internet a diminué lorsque les liaisons sont devenues plus rapides, et (c) de toutes façons, les redémarrages prennent souvent plus de temps qu'une MSL.

Pour être sûr qu'une mise en œuvre de TCP ne crée pas un segment portant un numéro de séquence qui pourrait être dupliqué d'un ancien segment restant dans le réseau, le point d'extrémité TCP doit rester silencieux pendant une MSL avant d'allouer des numéros de séquence au démarrage ou à la récupération d'une situation où la mémoire des numéros de séquence utilisés a été perdue. Pour la présente spécification, la MSL est prise comme étant de 2 minutes. Il s'agit d'un choix d'ingénierie, qui peut être modifié si l'expérience montre qu'il est souhaitable de le faire. Noter que si un point d'extrémité de TCP est réinitialisé dans un certain sens, mais conserve sa mémoire des numéros de séquence utilisés, alors il n'a pas besoin d'attendre du tout ; il doit seulement s'assurer d'utiliser des numéros de séquence plus grands que ceux récemment utilisés.

3.4.3 Le concept de temps de silence TCP

Des hôtes qui, pour une raison quelconque, perdent la connaissance des derniers numéros de séquence transmis sur chaque connexion active (c'est-à-dire non fermée) doivent retarder l'émission de tout segment TCP pour au moins la MSL convenue dans le système Internet dont l'hôte fait partie. Dans les paragraphes ci-dessous, une explication de cette spécification est donnée. Les mises en œuvre de TCP peuvent enfreindre la restriction du "temps de silence", mais seulement au risque de faire accepter certaines données anciennes comme nouvelles ou d'avoir de nouvelles données rejetées comme anciennes données dupliquées par certains récepteurs sur le système Internet.

Les points d'extrémité TCP consomment de l'espace de numéro de séquence chaque fois qu'un segment est formé et entré dans la file d'attente de sortie réseau sur un hôte de source. L'algorithme de détection et de séquençage des dupliqués dans TCP repose sur le lien unique des données de segment à l'espace des numéros de séquence dans la mesure où les numéros de séquence ne parcourent pas toutes les 2^{32} valeurs avant que les données de segment liées à ces numéros de séquence aient été livrées et aient fait l'objet d'un accusé de réception par le destinataire et que toutes les copies dupliquées des segments aient été "drainées" de l'Internet. Sans une telle hypothèse, deux segments TCP distincts pourraient avoir une attribution de numéros de séquence identiques ou chevauchants, créant de la confusion au niveau du receveur pour savoir quelles données sont nouvelles et quelles sont anciennes. Se souvenir que chaque segment est lié à autant de numéros de séquence consécutifs qu'il y a d'octets de données et de fanions SYN ou FIN dans le segment.

Dans des conditions normales, les mises en œuvre de TCP gardent une trace du numéro de séquence suivant à émettre et des plus anciens en attente d'accusé de réception afin d'éviter la réutilisation par erreur d'un numéro de séquence avant que sa première utilisation soit acquittée. Cela ne garantit pas à lui seul que les anciennes données dupliquées sont évacuées du réseau, de sorte que l'espace de numéros de séquence a été fait grand pour réduire la probabilité qu'un doublon errant cause des problèmes à l'arrivée. À 2 Mbit/s, il faut 4,5 heures pour utiliser jusqu'à 2^{32} octets d'espace de numéros de séquence. Étant donné que la durée de vie sur le réseau du segment maximum n'est pas susceptible de dépasser quelques dizaines de secondes, c'est réputé une protection suffisante pour les réseaux prévisibles, même si les débits des données grimpent à 10 Mbit/s. À 100 Mbit/s, le temps de cycle est de 5,4 minutes, ce qui peut être un peu court mais quand même dans la limite du raisonnable. Des débits de données beaucoup plus élevés sont possibles aujourd'hui, avec des implications décrites dans le dernier paragraphe de la présente sous-section.

L'algorithme de base de détection et de séquençage des dupliqués dans TCP peut être cependant mis en échec, si un point d'extrémité de source TCP ne dispose pas de la mémoire des numéros de séquence qu'il a utilisés pour la dernière fois sur une connexion donnée. Par exemple, si la mise en œuvre de TCP devait commencer toutes les connexions avec le numéro de séquence 0, alors au redémarrage de l'hôte, un homologue TCP peut reformer une connexion (éventuellement après une résolution de connexion semi-ouverte) et émettre des paquets dont les numéros de séquence sont identiques ou qui se chevauchent avec des paquets encore dans le réseau, qui ont été émis sur une incarnation de la même connexion. En l'absence de connaissances sur les numéros de séquence utilisés sur une connexion particulière, la spécification TCP recommande que la source retarde de MSL secondes avant d'émettre des segments sur la connexion, pour laisser le temps aux segments de l'incarnation précédente de la connexion de s'évacuer du système.

Même les hôtes qui peuvent se souvenir de l'heure et l'utiliser pour choisir les valeurs initiales de numéro de séquence ne sont pas à l'abri de ce problème (c'est-à-dire même si l'heure est utilisée pour choisir un numéro de séquence initial pour chaque nouvelle incarnation de connexion).

Supposons, par exemple, qu'une connexion soit ouverte à partir du numéro de séquence S. Supposons que cette connexion

ne soit pas beaucoup utilisée et que finalement la fonction de numéro de séquence initiale (ISN(t)) prenne une valeur égale au numéro de séquence, disons S1, du dernier segment envoyé par ce point d'extrémité TCP sur une connexion particulière. Supposons maintenant que à cet instant, l'hôte redémarre et établisse une nouvelle incarnation de la connexion. Le numéro de séquence initial choisi est S1 = ISN(t) -- dernier numéro de séquence utilisé sur l'ancienne incarnation de connexion ! Si la récupération se produit assez rapidement, n'importe quels vieux dupliqués dans le réseau portant des numéros de séquence dans le voisinage de S1 peuvent arriver et être traités comme de nouveaux paquets par le receveur de la nouvelle incarnation de la connexion.

Le problème est que l'hôte qui récupère peut ne pas savoir depuis combien de temps il était en panne avant le réamorçage et ne pas savoir non plus s'il y a encore d'anciens dupliqués dans le système provenant des incarnations précédentes de la connexion.

Une façon de résoudre ce problème est de retarder délibérément l'émission de segments pendant une MSL après la récupération d'un redémarrage -- il s'agit de la spécification d'un "temps de silence". Les hôtes qui préfèrent éviter d'attendre et qui sont prêts à risquer une confusion possible entre les anciens et les nouveaux paquets à une destination donnée peuvent choisir de ne pas attendre le "temps de silence". Les mises en œuvre peuvent donner aux utilisateurs TCP la possibilité de choisir connexion par connexion s'il faut attendre après un redémarrage, ou mettre en place de manière informelle le "temps de silence" pour toutes les connexions. Évidemment, même lorsqu'un utilisateur choisit d'attendre, ce n'est pas nécessaire une fois que l'hôte a été "actif" pendant au moins MSL secondes.

Pour résumer : chaque segment émis occupe un ou plusieurs numéros de séquence dans l'espace de numéros de séquence, et les numéros occupés par un segment sont "occupés" ou "en cours d'utilisation" jusqu'à ce que MSL secondes se soient écoulées. Au redémarrage, un bloc d'espace est occupé par les octets et les fanions SYN ou FIN de tout segment potentiellement encore en cours. Si une nouvelle connexion est démarrée trop tôt et utilise l'un des numéros de séquence dans l'empreinte spatio-temporelle des segments potentiellement encore en cours de l'incarnation précédente de la connexion, il y a une zone de chevauchement potentielle de numéros de séquence qui pourrait causer de la confusion chez le receveur.

Les cas à haute performance auront des temps de cycle plus courts que ceux des mégabits par seconde que la conception de TCP de base décrite ci-dessus prend en compte. À 1 Gbit/s, le temps de cycle est de 34 secondes, seulement 3 secondes à 10 Gbit/s, et environ un tiers de seconde à 100 Gbit/s. Dans ces cas de performances supérieures, les options d'horodatage TCP et de protection contre les retours de séquences (PAWS) [RFC7323] fournissent la capacité nécessaire pour détecter et éliminer les anciens dupliqués.

3.5 Établissement d'une connexion

La "prise de contact en trois phases" est la procédure utilisée pour établir une connexion. Cette procédure est normalement initiée par un homologue TCP et il y est répondu par un autre homologue TCP. La procédure fonctionne également si deux homologues TCP initient simultanément la procédure. En cas d'ouverture simultanée, chaque homologue TCP reçoit un segment SYN qui ne porte pas d'accusé de réception après qu'il a envoyé un SYN. Bien sûr, l'arrivée d'un ancien segment SYN dupliqué peut potentiellement faire apparaître, au receveur, qu'une initiation simultanée de connexion est en cours. L'utilisation correcte des segments "reset" peut lever l'ambiguïté de ces cas.

Voici plusieurs exemples d'initiation de connexion. Bien que ces exemples ne montrent pas la synchronisation de connexion à l'aide de segments de transport de données, c'est parfaitement légitime, tant que le point d'extrémité TCP de réception ne fournit pas les données à l'utilisateur alors qu'il n'est pas clair qu'elles sont valides (par exemple, les données sont mises en mémoire tampon chez le receveur jusqu'à ce que la connexion atteigne l'état ESTABLISHED, étant donné que la prise de contact à trois phases réduit la possibilité de fausses connexions). C'est un compromis entre la mémoire et les messages pour fournir les informations à cette vérification.

La 3WHS la plus simple est illustrée à la figure 6. Les figures devraient être interprétées de la manière suivante. Chaque ligne est numérotée à titre de référence. Les flèches vers la droite (-->) indiquent le départ d'un segment TCP de l'homologue TCP A vers l'homologue TCP B ou l'arrivée d'un segment en B à partir de A. Les flèches vers la gauche (<--) indiquent l'inverse. Les points de suspension (...) indiquent un segment qui se trouve toujours dans le réseau (retardé). Les commentaires apparaissent entre parenthèses. Les états de connexion TCP représentent l'état APRÈS le départ ou l'arrivée du segment (dont le contenu est indiqué dans le centre de chaque ligne). Le contenu des segments est présenté en abrégé avec le numéro de séquence, les fanions de contrôle et le champ ACK. Les autres champs tels que la fenêtre, les adresses, les longueurs et le texte ont été omis, dans un souci de clarté.

Homologue TCP A		Homologue TCP B
1. CLOSED		LISTEN
2. SYN-SENT	--> <SEQ=100><CTL=SYN>	--> SYN-RECEIVED
3. ESTABLISHED	<-- <SEQ=300><ACK=101><CTL=SYN,ACK>	<-- SYN-RECEIVED
4. ESTABLISHED	--> <SEQ=101><ACK=301><CTL=ACK>	--> ESTABLISHED
5. ESTABLISHED	--> <SEQ=101><ACK=301><CTL=ACK><DATA>	--> ESTABLISHED

Figure 6 : Prise de contact en trois phases de base pour la synchronisation de la connexion

À la ligne 2 de la figure 6, l'homologue TCP A commence par envoyer un segment SYN indiquant qu'il utilisera des numéros de séquence commençant par le numéro de séquence 100. À la ligne 3, l'homologue TCP B envoie un SYN et accuse réception du SYN reçu de l'homologue TCP A. Noter que le champ d'accusé de réception indique que l'homologue TCP B s'attend maintenant à voir le numéro de séquence 101, accusant réception du SYN qui occupe la séquence 100.

À la ligne 4, l'homologue TCP A répond avec un segment vide contenant un ACK pour le SYN de l'homologue TCP B ; et à la ligne 5, l'homologue TCP A envoie des données. Noter que le numéro de séquence du segment de la ligne 5 est le même que celui de la ligne 4 parce que le ACK n'occupe pas l'espace des numéros de séquence (si c'était le cas, on finirait par accuser réception des accusés de réception !).

L'initiation simultanée n'est que légèrement plus complexe, comme le montre la figure 7. L'état de connexion de chaque homologue TCP passe de CLOSED à SYN-SENT à SYN-RECEIVED à ESTABLISHED.

Homologue TCP A		Homologue TCP B
1. CLOSED		CLOSED
2. SYN-SENT	--> <SEQ=100><CTL=SYN>	...
3. SYN-RECEIVED	<-- <SEQ=300><CTL=SYN>	<-- SYN-SENT
4.	... <SEQ=100><CTL=SYN>	--> SYN-RECEIVED
5. SYN-RECEIVED	--> <SEQ=100><ACK=301><CTL=SYN,ACK>	...
6. ESTABLISHED	<-- <SEQ=300><ACK=101><CTL=SYN,ACK>	<-- SYN-RECEIVED
7.	... <SEQ=100><ACK=301><CTL=SYN,ACK>	--> ESTABLISHED

Figure 7 : Synchronisation simultanée des connexions

Une mise en œuvre de TCP DOIT prendre en charge les tentatives d'ouverture simultanée (DOIT-10).

Noter qu'une mise en œuvre de TCP DOIT garder trace de si une connexion a atteint l'état SYN-RECEIVED à la suite d'un OPEN passif ou d'un OPEN actif (DOIT-11).

La raison principale de la prise de contact en trois phases est d'éviter que les initiations de connexion en double causent la confusion. Pour traiter cela, un message de contrôle spécial, "reset" est spécifié. Si l'homologue TCP receveur est dans un état non synchronisé (c'est-à-dire SYN-SENT, SYN-RECEIVED) il retourne à LISTEN lorsqu'il reçoit une réinitialisation acceptable. Si l'homologue TCP se trouve dans l'un des états synchronisés (ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT) il interrompt la connexion et en informe son utilisateur. On discute de ce dernier cas ci-dessous sous "Connexions semi-ouvertes".

Homologue TCP A	Homologue TCP B
1. CLOSED	LISTEN
2. SYN-SENT --> <SEQ=100><CTL=SYN>	...
3. (duplicate) ... <SEQ=90><CTL=SYN>	--> SYN-RECEIVED
4. SYN-SENT <-- <SEQ=300><ACK=91><CTL=SYN,ACK>	<-- SYN-RECEIVED
5. SYN-SENT --> <SEQ=91><CTL=RST>	--> LISTEN
6. ... <SEQ=100><CTL=SYN>	--> SYN-RECEIVED
7. ESTABLISHED <-- <SEQ=400><ACK=101><CTL=SYN,ACK>	<-- SYN-RECEIVED
8. ESTABLISHED --> <SEQ=101><ACK=401><CTL=ACK>	--> ESTABLISHED

Figure 8 : Récupération à partir d'un ancien SYN dupliqué

À titre d'exemple simple de récupération à partir d'anciens dupliqués, considérons la figure 8. À la ligne 3, un ancien SYN dupliqué arrive à l'homologue TCP B. L'homologue TCP B ne peut pas dire qu'il s'agit d'un ancien dupliqué, donc il répond normalement (ligne 4). L'homologue TCP A détecte que le champ ACK est incorrect et renvoie un RST (réinitialiser) avec son champ SEQ sélectionné pour rendre le segment crédible. L'homologue TCP B, à la réception du RST, revient à l'état LISTEN. Lorsque le SYN original arrive enfin à la ligne 6, la synchronisation se déroule normalement. Si le SYN de la ligne 6 était arrivé avant le RST, un échange plus complexe aurait pu avoir lieu avec les RST envoyés dans les deux directions.

3.5.1 Connexions semi-ouvertes et autres anomalies

Une connexion établie est dite "semi-ouverte" si l'un des homologues TCP a fermé ou interrompu la connexion à son extrémité sans que l'autre le sache, ou si les deux extrémités de la connexion sont désynchronisées en raison d'une défaillance ou d'un redémarrage qui a entraîné une perte de mémoire. Ces connexions seront automatiquement réinitialisées si on tente d'envoyer des données dans une direction ou l'autre. Cependant, les connexions semi ouvertes devraient être inhabituelles.

Si, sur le site A, la connexion n'existe plus, une tentative de l'utilisateur sur le site B pour envoyer des données sur elle va entraîner que le point d'extrémité TCP du site B va recevoir un message de commande de réinitialisation. Un tel message indique au site TCP B que quelque chose ne va pas, et il est supposé interrompre la connexion.

Supposons que deux processus utilisateur A et B communiquent ensemble lorsqu'une défaillance ou un redémarrage se produit entraînant une perte de mémoire pour la mise en œuvre TCP de A. Selon le système d'exploitation qui prend en charge la mise en œuvre TCP de A, il est probable qu'il existe un mécanisme de récupération d'erreur. Lorsque le point d'extrémité TCP est à nouveau actif, A va probablement recommencer depuis le début ou à partir d'un point de reprise. Par conséquent, A essaiera probablement d'ouvrir à nouveau la connexion avec OPEN ou essayer SEND sur la connexion qu'il croit ouverte. Dans ce dernier cas, il reçoit le message d'erreur "connexion non ouverte" de la la mise en œuvre TCP locale (de A). En tentant d'établir la connexion, la mise en œuvre TCP de A enverra un segment contenant SYN. Ce scénario conduit à l'exemple illustré à la Figure 9. Après le redémarrage de l'homologue TCP A, l'utilisateur tente de rouvrir la connexion. Pendant ce temps, l'homologue TCP B pense que la connexion est ouverte.

Homologue TCP A	Homologue TCP B
1. (REBOOT)	(envoie 300, reçoit 100)
2. CLOSED	ESTABLISHED
3. SYN-SENT --> <SEQ=400><CTL=SYN>	--> (??)

4. (!!) <-- <SEQ=300><ACK=100><CTL=ACK> <-- ESTABLISHED

5. SYN-SENT --> <SEQ=100><CTL=RST> --> (Abort!!)

6. SYN-SENT CLOSED

7. SYN-SENT --> <SEQ=400><CTL=SYN> -->

Figure 9 : Découverte d'une connexion semi-ouverte

Lorsque le SYN arrive à la ligne 3, l'homologue TCP B, étant dans un état synchronisé, et le segment entrant à l'extérieur de la fenêtre, répond par un accusé de réception indiquant quelle séquence il s'attend à voir ensuite (accusé de réception 100). L'homologue TCP A constate que ce segment n'accuse réception de rien de ce qu'il envoyé et, n'étant pas synchronisé, envoie une réinitialisation (RST) car il a détecté une connexion semi-ouverte. L'homologue TCP B interrompt à la ligne 5. L'homologue TCP A continuer d'essayer d'établir la connexion ; le problème est maintenant réduit à la prise de contact en trois phases de base de la Figure 6.

Un autre cas intéressant se produit lorsque l'homologue TCP A redémarre et que l'homologue TCP B tente d'envoyer des données sur ce qu'il pense être une connexion synchronisée. C'est ce qu'illustre la Figure 10. Dans ce cas, les données arrivant à l'homologue TCP A de l'homologue TCP B (ligne 2) sont inacceptables car une telle connexion n'existe pas, de sorte que l'homologue TCP A envoie un RST. Le RST est acceptable, donc l'homologue TCP B le traite et interrompt la connexion.

Homologue TCP A	**Homologue TCP B**
1. (REBOOT)	(envoie 300, reçoit 100)
2. (??) <-- <SEQ=300><ACK=100><DATA=10><CTL=ACK>	<-- ESTABLISHED
3. --> <SEQ=100><CTL=RST>	--> (ABORT!!)

Figure 10 : Le côté actif provoque la découverte d'une connexion semi-ouverte

Dans la figure 11 sont représentés deux homologues TCP A et B avec des connexions passives dans l'attente de SYN. Un ancien dupliqué arrivant à l'homologue TCP B (ligne 2) fait entrer B en action. Un SYN-ACK est retourné (ligne 3) et amène le TCP A à générer un RST (le ACK de la ligne 3 n'est pas acceptable). L'homologue TCP B accepte la réinitialisation et revient à son état passif LISTEN.

Homologue TCP A	**Homologue TCP B**
1. LISTEN	LISTEN
2. ... <SEQ=Z><CTL=SYN>	--> SYN-RECEIVED
3. (??) <-- <SEQ=X><ACK=Z+1><CTL=SYN,ACK>	<-- SYN-RECEIVED
4. --> <SEQ=Z+1><CTL=RST>	--> (revient à LISTEN!)
5. LISTEN	LISTEN

Figure 11 : L'ancien SYN dupliqué lance une réinitialisation sur deux prises passives

Divers autres cas sont possibles, qui sont tous pris en compte par les règles suivantes pour la génération et le traitement de RST.

3.5.2 Génération de Reset

Un utilisateur ou une application TCP peut effectuer une réinitialisation sur une connexion à tout moment, bien que des événements de réinitialisation soient également générés par le protocole lui-même lorsque diverses conditions d'erreur se produisent, comme décrit ci-dessous. Le côté d'une connexion qui émet une réinitialisation devrait passer à l'état TIME-WAIT, car cela aide généralement à réduire la charge sur les serveurs occupés pour les raisons décrites dans [70].

En règle générale, la réinitialisation (RST) est envoyée chaque fois qu'un segment arrive qui n'est apparemment pas destiné à la connexion actuelle. Une réinitialisation ne doit pas être envoyée s'il n'est pas clair que c'est le cas.

Il y a trois groupes d'états :

1. Si la connexion n'existe pas (CLOSED) une réinitialisation est envoyée en réponse à tout segment entrant, à l'exception d'une autre réinitialisation. Un segment SYN qui ne correspond pas à une connexion existante est rejeté par ce moyen. Si le bit ACK est établi sur le segment entrant, la réinitialisation prend son numéro de séquence du champ ACK du segment ; sinon, la réinitialisation a le numéro de séquence zéro et le champ ACK est réglé à la somme du numéro de séquence et de la longueur du segment entrant. La connexion reste à l'état CLOSED.

2. Si la connexion est dans tout état non synchronisé (LISTEN, SYN-SENT, SYN-RECEIVED) et si le segment entrant accuse réception de quelque chose qui n'a pas encore été envoyé (le segment porte un ACK inacceptable) ou si un segment entrant a un niveau ou compartiment de sécurité (Annexe A.1) qui ne correspond pas exactement au niveau et au compartiment demandés pour la connexion, une réinitialisation est envoyée.

   Si le segment entrant comporte un champ ACK, la réinitialisation prend son numéro de séquence du champ ACK du segment ; sinon, la réinitialisation a le numéro de séquence zéro et le champ ACK est défini sur la somme du numéro de séquence et de la longueur du segment entrant. La connexion reste dans le même état.

3. Si la connexion est dans un état synchronisé (ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT) on doit répondre à tout segment inacceptable (numéro de séquence hors fenêtre ou numéro d'accusé de réception inacceptable) par un segment d'accusé de réception vide (sans aucune donnée d'utilisateur) contenant le numéro de séquence d'envoi actuel et un accusé de réception indiquant le prochain numéro de séquence attendu à recevoir, et la connexion reste dans le même état.

   Si un segment entrant a un niveau ou compartiment de sécurité qui ne correspond pas exactement au niveau et au compartiment demandé pour la connexion, une réinitialisation est envoyée et la connexion passe à l'état CLOSED. La réinitialisation prend son numéro de séquence dans le champ ACK du segment entrant.

3.5.3 Traitement de la réinitialisation

Dans tous les états sauf SYN-SENT, tous les segments de réinitialisation (RST) sont validés en vérifiant leurs champs SEQ. Une réinitialisation est valide si son numéro de séquence est dans la fenêtre. Dans l'état SYN-SENT (un RST reçu en réponse à un SYN initial) le RST est acceptable si le champ ACK accuse réception du SYN.

Le receveur d'un RST le valide d'abord, puis change d'état. Si le receveur était à l'état LISTEN, il l'ignore. Si le receveur était dans l'état SYN-RECEIVED et avait précédemment été dans l'état LISTEN, alors le receveur revient à l'état LISTEN ; sinon, le receveur interrompt la connexion et passe à l'état CLOSED. Si le receveur était dans un autre état, il interrompt la connexion, en informe l'utilisateur, et passe à l'état FERMÉ.

Les mises en œuvre de TCP DEVRAIENT permettre à un segment RST reçu d'inclure des données (DVRT-2). Il a été suggéré qu'un segment RST pourrait contenir des données de diagnostic qui expliquent la cause du RST. Aucune norme n'a encore été établie pour ces données.

3.6 Fermeture d'une connexion

CLOSE est une opération signifiant "Je n'ai plus de données à envoyer". La notion de fermeture d'une connexion bidirectionnelle est sujette à des interprétations ambiguës, bien sûr, car il n'est peut-être pas évident de savoir comment traiter le côté receveur de la connexion. On a choisi de traiter CLOSE de manière unidirectionnelle. L'utilisateur qui fait le CLOSE peut continuer à RECEVOIR jusqu'à ce que le receveur TCP soit informé que l'homologue distant est également FERMÉ. Donc, un programme pourrait initier plusieurs SEND suivis d'un CLOSE, et ensuite, continuer de RECEVOIR jusqu'à ce qu'il soit signalé qu'un RECEIVE a échoué parce que l'homologue distant est à l'état CLOSED. La mise en œuvre TCP signalera à un utilisateur, même si aucun RECEIVE n'est en instance, que l'homologue distant a fermé, afin

que l'utilisateur puisse fermer son côté en douceur. Une mise en œuvre de TCP va livrer de façon fiable tous les SENT mis en mémoire tampon avant que la connexion soit FERMÉE afin qu'un utilisateur qui n'attend pas de données en retour n'aitqu'à attendre d'entendre que la connexion a été bien CLOSED pour savoir que toutes ses données ont été reçues au point d'extrémité TCP de destination. Les utilisateurs doivent continuer de lire sur les connexions qu'ils ferment en envoi jusqu'à ce que la mise en œuvre TCP indique qu'il n'y a plus de données.

Il y a essentiellement trois cas :

1) L'utilisateur initie en demandant à la mise en œuvre TCP de FERMER la connexion (homologue TCP A dans la Figure 12).
2) Le point d'extrémité TCP distant initie en envoyant un signal de contrôle FIN (homologue TCP B dans la Figure 12).
3) Les deux utilisateurs FERMENT simultanément (Figure 13).

Cas 1 : L'utilisateur local initie la clôture

Dans ce cas, un segment FIN peut être construit et placé dans la file d'attente de segments sortants. Aucun autre SEND de l'utilisateur ne sera accepté par la mise en œuvre TCP, et il passe à l'état FIN-WAIT-1. Les RECEIVE sont autorisés dans cet état. Tous les segments précédents et incluant FIN seront retransmis jusqu'à ce qu'il en soit accusé réception. Lorsque l'autre homologue TCP a à la fois accusé réception du FIN et envoyé son propre FIN, le premier homologue TCP peut accuser réception de ce FIN. Notez qu'un point d'extrémité TCP recevant un FIN va en accuser réception mais pas envoyer son propre FIN jusqu'à ce que son utilisateur ait également FERMÉ la connexion.

Cas 2 : Le point d'extrémité TCP reçoit un FIN du réseau

Si un FIN non sollicité arrive du réseau, le point d'extrémité TCP receveur peut y faire un ACK et dire à l'utilisateur que la connexion ferme. L'utilisateur va répondre par un CLOSE, sur lequel le point d'extrémité TCP peut envoyer un FIN à l'autre homologue TCP après l'envoi de toutes données restantes. Le point d'extrémité TCP attend ensuite jusqu'à ce que son propre FIN soit acquitté, après quoi il supprime la connexion. Si un accusé de réception ne vient pas, après la fin de temporisation de l'utilisateur, la connexion est interrompue et l'utilisateur en est informé.

Cas 3 : Les deux utilisateurs se ferment simultanément

Un CLOSE simultané par les utilisateurs aux deux extrémités d'une connexion cause l'échange de segments FIN (Figure 13). Lorsque tous les segments précédant les FIN ont été traités et acquittés, chaque homologue TCP peut accuser réception du FIN reçu. À réception de ces ACK, les deux supprimeront la connexion.

Homologue TCP A	**Homologue TCP B**
1. ESTABLISHED	ESTABLISHED
2. (Close) FIN-WAIT-1 --> <SEQ=100><ACK=300><CTL=FIN,ACK> --> CLOSE-WAIT	
3. FIN-WAIT-2 <-- <SEQ=300><ACK=101><CTL=ACK> <-- CLOSE-WAIT	
4.	(Close)
TIME-WAIT <-- <SEQ=300><ACK=101><CTL=FIN,ACK> <-- LAST-ACK	
5. TIME-WAIT --> <SEQ=101><ACK=301><CTL=ACK> --> CLOSED	
6. (2 MSL) CLOSED	

Figure 12 : Séquence de fermeture normale

Homologue TCP A	**Homologue TCP B**
1. ESTABLISHED	ESTABLISHED
2. (Close)	(Close)
FIN-WAIT-1 --> <SEQ=100><ACK=300><CTL=FIN,ACK> ... FIN-WAIT-1	
<-- <SEQ=300><ACK=100><CTL=FIN,ACK> <--	
... <SEQ=100><ACK=300><CTL=FIN,ACK> -->	

```
3.  CLOSING    --> <SEQ=101><ACK=301><CTL=ACK>      ... CLOSING
                <-- <SEQ=301><ACK=101><CTL=ACK>     <--
                ... <SEQ=101><ACK=301><CTL=ACK>     -->

4.  TIME-WAIT                                       TIME-WAIT
    (2 MSL)                                         (2 MSL)
    CLOSED                                          CLOSED
```

Figure 13 : Séquence de fermeture simultanée

Une connexion TCP peut se terminer de deux manières : (1) la séquence normale de clôture de TCP à l'aide d'une prise de contact FIN (Figure 12), et (2) un "abandon" dans lequel un ou plusieurs segments RST sont envoyés et l'état de connexion est immédiatement éliminé. Si la connexion TCP locale est fermée par le côté distant en raison d'un FIN ou d'un RST reçu du côté distant, alors l'application locale DOIT être informée de si elle a fermé normalement ou a été interrompue (DOIT-12).

3.6.1 Connexions semi-fermées

La séquence de fermeture TCP normale livre les données mises en mémoire tampon de manière fiable dans les deux directions. Comme les deux directions d'une connexion TCP sont closes indépendamment, il est possible qu'une liaison soit "semi fermée", c'est-à-dire fermée dans une seule direction, et qu'il soit permis à un hôte de continuer à envoyer des données dans la direction ouverte sur une connexion semi-fermée.

Un hôte PEUT mettre en œuvre une séquence de fermeture TCP "unilatérale", afin qu'une application qui a invoqué CLOSE ne puisse pas continuer à lire les données de la connexion (PEUT-1). Si un tel hôte émet un CLOSE alors que des données reçues sont toujours en attente dans la connexion TCP, ou si de nouvelles données sont reçues après l'invocation de CLOSE, sa mise en œuvre TCP DEVRAIT envoyer un RST pour montrer que des données ont été perdues (DVRT-3). Voir la discussion au paragraphe 2.17 de la [RFC2525].

Lorsqu'une connexion est fermée activement, elle DOIT s'attarder dans l'état TIME-WAIT pour une durée de 2xMSL (durée de vie maximale du segment) (DOIT-13). Cependant, elle PEUT accepter un nouveau SYN du point d'extrémité TCP distant pour rouvrir la connexion directement à partir de l'état TIME-WAIT (PEUT-2), si :
(1) elle attribue à la nouvelle connexion un numéro de séquence initial supérieur au plus grand numéro de séquence utilisé lors de l'incarnation précédente de la connexion, et
(2) elle revient à l'état TIME-WAIT si le SYN s'avère être un vieux dupliqué.

Lorsque les options d'horodatage TCP sont disponibles, un algorithme amélioré est décrit dans la [RFC6191] afin de prendre en charge des taux d'établissement de connexion plus élevés. Cet algorithme de réduction de TIME-WAIT est une "bonne pratique courante" qui DEVRAIT être mise en œuvre puisque les options d'horodatage sont couramment utilisées, et leur utilisation pour réduire le TIME-WAIT offre des avantages pour les serveurs Internet occupés (DVRT-4).

3.7 Segmentation

Le terme "segmentation" fait référence à l'activité effectuée par TCP lors de l'ingestion d'un flux d'octets à partir d'une application d'envoi et de la mise en paquets de ce flux d'octets dans des segments TCP. Souvent, les segments TCP individuels ne correspondent pas un pour un aux appels d'envoi (ou d'écriture de prise) individuels de l'application. Les applications peuvent effectuer des écritures à la granularité des messages dans le protocole de couche supérieure, mais TCP ne garantit aucune corrélation entre les limites des segments TCP envoyés et reçus et les limites des mémoires tampons de lecture ou d'écriture des données d'application d'utilisateur. Dans certains protocoles spécifiques, tels que l'accès direct à la mémoire à distante (RDMA, *Remote Direct Memory Access*) à l'aide du placement direct de données (DDP, *Direct Data Placement*) et du tramage aligné sur la PDU de marqueur (MPA, *Marker PDU Aligned Framing*) [RFC5044], il est possible d'optimiser les performances lorsque la relation entre les segments TCP et les unités de données d'application peut être contrôlée, et MPA inclut un mécanisme spécifique pour détecter et vérifier cette relation entre les segments TCP et les structures de données de message d'application, mais cela est spécifique d'applications telles que RDMA. En général, plusieurs objectifs influencent le dimensionnement des segments TCP créés par une mise en œuvre de TCP.

Les objectifs de l'envoi de plus grands segments incluent :
- de réduire le nombre de paquets en transit au sein du réseau ;

- d’augmenter l’efficacité du traitement et les performances potentielles en permettant un plus petit nombre d’interruptions et d’interactions entre les couches ;
- de limiter les frais généraux des en-têtes TCP.

Noter que les avantages en termes de performances de l’envoi de segments plus grands peuvent diminuer à mesure que la taille augmente, et qu’il peut y avoir des limites où les avantages sont inversés. Par exemple, sur certaines architectures de mise en œuvre, 1025 octets au sein d’un segment peuvent entraîner des performances inférieures à celles de 1024 octets, en raison uniquement de l’alignement des données sur les opérations de copie.

Les objectifs de l’envoi de segments plus petits incluent :
- D’éviter d’envoyer un segment TCP qui entraînerait un datagramme IP supérieur à la plus petite MTU le long d’un chemin de réseau IP, car cela entraîne une perte ou une fragmentation de paquets. Pour aggraver les choses, certains pare-feu ou boîtiers de médiation peuvent éliminer des paquets fragmentés ou des messages ICMP liés à la fragmentation.
- D’empêcher des retards dans le flux de données d’application, en particulier lorsque TCP attend que l’application génère plus de données, ou lorsque l’application attend un événement ou une entrée de son homologue afin de générer plus de données.
- Permettre le "partage de sort" entre les segments TCP et les unités de données de couche inférieure (par exemple, en dessous de IP, pour les liaisons dont la taille de cellule ou de trame est inférieure à la MTU IP).

Pour atteindre ces ensembles d’objectifs concurrents, TCP inclut plusieurs mécanismes, notamment l’option de taille maximale de segment, la découverte de MTU de chemin, l’algorithme de Nagle, et la prise en charge des jumbogrammes IPv6, comme indiqué dans les paragraphes suivants.

3.7.1 Option de taille maximale de segment

Les points d'extrémité TCP DOIVENT mettre en œuvre l’envoi et la réception de l’option MSS (DOIT-14).

Les mises en œuvre TCP DEVRAIENT envoyer une option MSS dans chaque segment SYN lorsque la MSS reçue diffère de la MSS par défaut de 536 pour IPv4 ou 1220 pour IPv6 (DVRT-5) et elles PEUVENT toujours l’envoyer (PEUT-3).

Si aucune option MSS n’est reçue lors de l’établissement de la connexion, les mises en œuvre de TCP DOIVENT supposer une MSS d’envoi par défaut de 536 (576 - 40) pour IPv4 ou de 1220 (1280 - 60) pour IPv6 (DOIT-15).

La taille maximale d’un segment qu’un point d'extrémité TCP envoie réellement, la "MSS effective d’envoi", DOIT être le plus petit (DOIT-16) de la MSS d’envoi (qui reflète la taille de la mémoire tampon de réassemblage disponible à l’hôte distant, le EMTU_R [RFC1122]) et la plus grande taille de transmission permise par la couche IP (EMTU_S [RFC1122]) :

Eff.snd.MSS = min(SendMSS+20, MMS_S) - TCPhdrsize - IPoptionsize

où :
-SendMSS est la valeur de MSS reçue de l’hôte distant, ou la valeur par défaut 536 pour IPv4 ou 1220 pour IPv6, si aucune option MSS n’est reçue.

- MMS_S est la taille maximale d’un message de couche de transport que TCP peut envoyer.

- TCPhdrsize est la taille de l’en-tête TCP fixe et de toutes options. C’est 20 dans le cas (rare) où aucune option n’est présente, mais peut être plus grand si des options TCP doivent être envoyées. Noter que certaines options peuvent ne pas être incluses dans tous les segments, mais que pour chaque segment envoyé, l’envoyeur devrait ajuster la longueur des données en conséquence, dans Eff.snd.MSS.

IPoptionsize est la taille de toute option IPv4 ou en-tête d’extension IPv6 associé à une connexion TCP. Noter que certaines options ou en-têtes d’extension peuvent ne pas être inclus dans tous les paquets, mais que pour chaque segment envoyé, l’envoyeur devrait ajuster la longueur des données en conséquence, dans Eff.snd.MSS.

La valeur de MSS à envoyer dans une option MSS devrait être égale à la valeur de la MTU effective moins les en-têtes IP et TCP fixes. En ignorant les options IP et TCP lors du calcul de la valeur de l’option MSS, si il y a des options IP ou TCP à envoyer dans un paquet, alors l’envoyeur doit réduire la taille des données TCP en conséquence. La [RFC6691] en discute plus en détail.

La valeur de MSS à envoyer dans une option MSS doit être inférieure à ou égale à :

MMS_R – 20

où MMS_R est la taille maximale d'un message de couche de transport qui peut être reçu (et réassemblé à la couche IP) (DOIT-67). TCP obtient MMS_R et MMS_S de la couche IP ; voir l'appel générique GET_MAXSIZES au paragraphe 3.4 de la RFC 1122. Ils sont définis dans les termes de leurs MTU IP équivalentes, EMTU_R et EMTU_S [RFC1122].

Lorsque TCP est utilisé dans une situation où les en-têtes IP ou TCP ne sont pas fixes, l'envoyeur doit réduire la quantité de données TCP dans tout paquet du nombre d'octets utilisés par les options IP et TCP. Cela a historiquement été un point de confusion, comme expliqué au paragraphe 3.1 de la RFC 6691.

3.7.2 Découverte de la MTU de chemin

Une mise en œuvre TCP peut avoir connaissance de la MTU sur les liaisons directement connectées, mais aura rarement des informations sur les MTU sur l'ensemble d'un chemin du réseau. Pour IPv4, la RFC 1122 recommande une MTU effective par défaut de la couche IP inférieure ou égale à 576 pour les destinations non directement connectées, et pour IPv6, ce serait 1280. L'utilisation de ces valeurs fixes limite les performances et l'efficacité de la connexion TCP. Au lieu de cela, la mise en œuvre de la découverte de MTU de chemin (PMTUD, *Path MTU Discovery*) et de la découverte de MTU de chemin de couche de mise en paquets (PLPMTUD, *Packetization Layer Path MTU Discovery*) est fortement recommandée afin que TCP améliore les décisions de segmentation. La PMTUD et la PLPMTUD aident toutes deux TCP à choisir des tailles de segment qui évitent à la fois la fragmentation sur le chemin (pour IPv4) et à la source (IPv4 et IPv6).

La PMTUD pour IPv4 [RFC1191] ou IPv6 [RFC8201] est mise en œuvre conjointement entre TCP, IP et ICMP. Elle repose à la fois sur l'évitement de la fragmentation à la source et sur l'établissement du fanion IPv4 DF (Ne pas fragmenter) ce dernier pour inhiber la fragmentation sur le chemin. Il s'appuie sur les erreurs ICMP des routeurs le long du chemin chaque fois qu'un segment est trop grand pour traverser une liaison. Plusieurs ajustements d'une mise en œuvre TCP avec PMTUD sont décrits dans la RFC 2923 afin de faire face aux problèmes rencontrés dans la pratique [RFC2923]. La PLPMTUD [RFC4821] est une amélioration de la PMTUD sur la voie de la normalisation qui assouplit l'exigence de prendre en charge ICMP sur un chemin et améliore les performances dans les cas où ICMP n'est pas transmis de façon soutenue, mais tente toujours d'éviter la fragmentation à la source. Il est recommandé d'inclure les mécanismes de ces quatre RFC dans les mises en œuvre de TCP.

L'option TCP MSS spécifie une limite supérieure pour la taille des paquets qui peuvent être reçus (voir la [RFC6691]). Donc, régler trop petite la valeur dans l'option MSS peut impacter la capacité de la PMTUD ou de la PLPMTUD à trouver une plus grande MTU de chemin. La RFC 1191 discute de cette implication de nombreuses anciennes mises en œuvre de TCP qui règlent la MSS de TCP à 536 (correspondant à la MTU par défaut de 576 octets pour IPv4) pour les destinations non locales, plutôt que de la déduire des MTU des interfaces connectées comme recommandé.

3.7.3 Interfaces avec des valeurs de MTU variables

La MTU effective peut parfois varier, comme lorsqu'elle est utilisée avec une compression variable, par exemple, la compression d'en-tête robuste (ROHC, *RObust Header Compression*) [RFC5795]. Il est tentant pour une mise en œuvre de TCP d'annoncer la plus grande MSS possible, pour prendre en charge l'utilisation la plus efficace des charges utiles compressées. Malheureusement, certains schémas de compression ont parfois besoin de transmettre des en-têtes complets (et donc des charges utiles plus petites) pour resynchroniser l'état à leurs compresseurs/décompresseurs de point d'extrémité. Si la MTU la plus grande est utilisée pour calculer la valeur à annoncer dans l'option MSS, la retransmission TCP peut interférer avec la resynchronisation du compresseur.

Par suite, quand la MTU effective d'une interface varie d'un paquet à l'autre, les mises en œuvre de TCP DEVRAIENT utiliser la plus petite MTU effective de l'interface pour calculer la valeur à annoncer dans l'option MSS (DVRT-6).

3.7.4 Algorithme de Nagle

L'algorithme de Nagle a été décrit dans la [RFC0896] et a été recommandée dans la [RFC1122] pour atténuer un problème précoce de génération d'un trop grand nombre de petits paquets. Il a été mis en œuvre dans la plupart des bases de code TCP actuelles, parfois avec des variations mineures (voir l'Annexe A.3).

S'il y a des données non acquittées (c'est-à-dire, SND.NXT > SND.UNA) alors le point d'extrémité TCP d'envoi met en mémoire tampon toutes les données d'utilisateur (quel que soit le bit PSH) jusqu'à ce que les données restantes aient été acquittées ou jusqu'à ce que le point d'extrémité TCP puisse envoyer un segment de taille complète (Eff.snd.MSS octets).

Une mise en œuvre de TCP DEVRAIT appliquer l'algorithme de Nagle pour fusionner des segments courts (DVRT-7). Cependant, il DOIT y avoir un moyen pour une application de désactiver l'algorithme de Nagle sur une connexion individuelle (DOIT-17). Dans tous les cas, l'envoi de données est également soumis à la limitation imposée par l'algorithme de démarrage lent [RFC5681].

Comme il peut y avoir des interactions problématiques entre l'algorithme de Nagle et les accusés de réception différés, certaines mises en œuvre utilisent des variantes mineures de l'algorithme de Nagle, telles que celle décrite à l'Annexe A.3.

3.7.5 Jumbogrammes IPv6

Afin de prendre en charge les jumbogrammes TCP sur IPv6, les mises en œuvre doivent être capables d'envoyer des segments TCP supérieurs à la limite de 64 ko que ce que l'option MSS peut transporter. La [RFC2675] définit qu'une valeur de MSS de 65 535 octets doit être traitée comme infinie, et la découverte de la MTU de chemin [RFC8201] est utilisée pour déterminer la MSS réelle.

Il n'est pas nécessaire que l'option de charge utile Jumbo soit mise en œuvre ou comprise par les nœuds IPv6 qui ne prennent pas en charge le rattachement à des liaisons dont la MTU est supérieure à 65 575 [RFC2675], et la configuration actuelle des nœuds IPv6 n'inclut pas la prise en charge des jumbogrammes [RFC8504].

3.8 Communication des données

Une fois la connexion établie, les données sont communiquées par l'échange de segments. Parce que des segments peuvent être perdus en raison d'erreurs (échec de la vérification de somme de contrôle) ou de l'encombrement du réseau, TCP utilise la retransmission pour assurer la livraison de chaque segment. Des segments dupliqués peuvent arriver en raison d'une retransmission du réseau ou de TCP. Comme discuté dans le paragraphe sur les numéros de séquence (paragraphe 3.4) la mise en œuvre TCP effectue certaines vérifications sur les numéros de séquence et les numéros d'accusé de réception dans les segments pour vérifier leur acceptabilité.

L'envoyeur des données garde une trace du prochain numéro de séquence à utiliser dans la variable SND.NXT. Le receveur des données garde une trace du prochain numéro de séquence à attendre dans la variable RCV.NXT. L'envoyeur des données garde trace du numéro de séquence le plus ancien non acquitté dans la variable SND.UNA. Si le flux de données est momentanément inactif et si toutes les données envoyées ont été acquittées, alors les trois variables seront égales.

Lorsque l'envoyeur crée un segment et le transmet, l'envoyeur avance SND.NXT. Quand le receveur accepte un segment, il fait avancer le RCV.NXT et envoie un accusé de réception. Quand l'envoyeur de données reçoit un accusé de réception, il avance SND.UNA. La mesure dans laquelle les valeurs de ces variables diffèrent est une mesure du retard dans la communication. La quantité d'avancement des variables est la longueur des données et des fanions SYN ou FIN dans le segment. Noter qu'une fois dans l'état ESTABLISHED, tous les segments doivent contenir les informations d'accusé de réception courantes.

L'appel utilisateur CLOSE implique une fonction "push" (voir le paragraphe 3.9.1) tout comme le fait le fanion de contrôle FIN dans un segment entrant.

3.8.1 Fin de temporisation de retransmission

En raison de la variabilité des réseaux qui composent un système inter réseaux et le large éventail d'utilisations des connexions TCP, le temps de fin de temporisation de retransmission (RTO, *Retransmission TimeOut*) doit être déterminé dynamiquement.

Le RTO DOIT être calculé en accord avec l'algorithme de la [RFC6298], y compris l'algorithme de Karn pour la prise d'échantillons de RTT (DOIT-18).

La RFC 793 contient un premier exemple de procédure pour le calcul du RTO, fondé sur les travaux mentionnés dans l'IEN 177 [71]. Il a ensuite été remplacé par l'algorithme décrit dans la RFC 1122, qui a ensuite été mis à jour dans la RFC 2988, puis à nouveau dans la RFC 6298.

La RFC 1122 permet que si un paquet retransmis est identique à l'original (ce qui implique non seulement que les limites de données n'ont pas changé, mais aussi qu'aucun des en-têtes n'a changé) alors le même champ d'identification IPv4 PEUT être utilisé (voir le paragraphe 3.2.1.5 de la RFC 1122) (PEUT-4). Le même champ d'identification IP peut être réutilisé de toutes façons, car il n'a de sens que lorsque le datagramme est fragmenté [RFC6864]. Les mises en œuvre de TCP ne doivent pas s'appuyer sur, ou interagir normalement avec, ce champ d'en-tête IPv4 de quelque manière que ce soit. Ce n'est pas une façon raisonnable d'indiquer les segments dupliqués ni d'identifier les segments reçus en double.

3.8.2 Contrôle d'encombrement TCP

La [RFC2914] explique l'importance du contrôle de l'encombrement pour l'Internet.

La RFC 1122 exigeait la mise en œuvre des algorithmes de Van Jacobson de contrôle d'encombrement, de démarrage lent et d'évitement d'encombrement, ainsi que de retard exponentiel pour les valeurs successives de RTO pour le même segment. La RFC 2581 a fourni une description sur la voie de la normalisation de l'IETF du démarrage lent et de l'évitement d'encombrement, ainsi que de la retransmission rapide et de la récupération rapide. La RFC 5681 est la description actuelle de ces algorithmes et est la spécification actuelle sur la voie de la normalisation qui fournit des directives pour le contrôle d'encombrement TCP. La RFC 6298 décrit le retard exponentiel des valeurs de RTO, y compris la conservation de la valeur de retard jusqu'à ce qu'un segment ultérieur avec de nouvelles données ait été envoyé et acquitté sans retransmission.

Un point d'extrémité TCP DOIT mettre en œuvre les algorithmes de base de contrôle d'encombrement de démarrage lent, d'évitement d'encombrement et de retard exponentiel du RTO pour éviter de créer des conditions de collapsus d'encombrement (DOIT-19). Les RFC 5681 et RFC 6298 décrivent les algorithmes de base sur la voie de la normalisation de l'IETF qui sont largement applicables. Il existe plusieurs autres algorithmes appropriés qui ont été largement utilisés. De nombreuses mises en œuvre de TCP prennent en charge un ensemble d'algorithmes de remplacement qui peuvent être configurés pour être utilisés sur le point d'extrémité. Un point d'extrémité PEUT mettre en œuvre de tels algorithmes de remplacement à condition qu'ils soient conformes aux spécifications sur la voie de la normalisation de l'IETF de TCP tels que décrits dans la [RFC2914], la [RFC5033], et la [RFC8961] (PEUT-18).

La notification explicite d'encombrement (ECN, *Explicit Congestion Notification*) a été définie dans la RFC 3168 et constitue une amélioration sur la voie de la normalisation de l'IETF qui présente de nombreux avantages [RFC8087].

Un point d'extrémité TCP DOIT mettre en œuvre ECN comme décrit dans la RFC 3168 (DVRT-8).

3.8.3 Échecs de connexion TCP

La retransmission excessive du même segment par un point d'extrémité TCP indique une défaillance de l'hôte distant ou du chemin d'inter réseaux. Cette défaillance peut être de courte ou de longue durée. La procédure suivante DOIT être utilisée pour traiter les retransmissions excessives de segments de données (DOIT-20) :

a) Il y a deux seuils, R1 et R2, qui mesurent la quantité de retransmissions qui ont eu lieu pour le même segment. R1 et R2 peuvent être mesurés en unités de temps ou en nombre de retransmissions (avec les RTO actuels et les retards correspondants comme facteur de conversion, si nécessaire).
b) Quand le nombre de transmissions d'un même segment atteint ou dépasse le seuil R1, on donne un avis négatif (voir la paragraphe 3.3.1.4 de la [RFC1122]) à la couche IP, pour déclencher le diagnostic de passerelle morte.
c) Quand le nombre de transmissions d'un même segment atteint un seuil R2 supérieur à R1, fermer la connexion.
d) Une application DOIT (DOIT-21) être capable de régler la valeur de R2 pour une connexion particulière. Par exemple, une application peut régler R2 à "infini", ce qui donne à l'utilisateur le contrôle du moment de la déconnexion.
e) Les mises en œuvre de TCP DOIVENT informer l'application du problème de livraison (à moins que ces informations aient été désactivées par l'application ; voir à "Rapports asynchrones" (paragraphe 3.9.1.8)) lorsque R1 est atteint et avant R2 (DVRT-9). Cela permettra à un programme d'application de connexion à distance d'informer l'utilisateur, par exemple.

La valeur de R1 DEVRAIT correspondre à au moins 3 retransmissions, au RTO actuel (DVRT-10). La valeur de R2 DEVRAIT correspondre à au moins 100 secondes (DVRT-11).

Une tentative d'ouverture d'une connexion TCP pourrait échouer avec des retransmissions excessives du segment SYN ou avec la réception d'un segment RST ou d'un accès inaccessible ICMP. Les retransmissions de SYN DOIVENT être traitées de la manière générale décrite pour les retransmissions de données, y compris la notification de la couche application.

Cependant, les valeurs de R1 et R2 peuvent être différentes pour SYN et les segments de données. En particulier, R2 pour un segment SYN DOIT être suffisamment grand pour permettre la retransmission du segment pendant au moins 3 minutes (DOIT-23). L'application peut bien sûr clore plus tôt la connexion (c'est-à-dire abandonner la tentative d'ouverture).

3.8.4 Maintien en vie TCP

Une connexion TCP est dite "inactive" si pendant une longue durée aucun segment entrant n'a été reçu et qu'il n'y a pas de données nouvelles ou non acquittées à envoyer.

Les mises en œuvre PEUVENT inclure des "maintien en vie" dans leurs applications TCP (PEUT-5) bien que cette pratique ne soit pas universellement acceptée. Cependant, quelques mises en œuvre de TCP ont inclus un mécanisme de maintien en vie. Pour confirmer qu'une connexion inactive est toujours en vie, ces mises en œuvre envoient un segment de sonde conçu pour provoquer une réponse de l'homologue TCP. Un tel segment contient généralement des SEG.SEQ = SND.NXT-1 et peut ou non contenir un octet de données factices. Si des "maintien en vie" sont inclus, l'application DOIT être capable de les activer ou désactiver pour chaque connexion TCP (DOIT-24) et elle DOIT par défaut les désactiver (DOIT-25).

Les paquets "maintien en vie" DOIVENT seulement être envoyés quand aucune donnée envoyée n'est en instance, et qu'aucun paquet de données ou d'accusé de réception n'a été reçu pour la connexion dans un certain intervalle (DOIT-26). Cet intervalle DOIT être configurable (DOIT-27) et DOIT par défaut ne pas être de moins de deux heures (DOIT-28).

Il est extrêmement important de se rappeler que les segments ACK qui ne contiennent aucune donnée ne sont pas transmis de manière fiable par TCP. Par conséquent, si un mécanisme de maintien en vie est mis en place, il NE DOIT PAS interpréter l'absence de réponse à une sonde spécifique comme une connexion morte (DOIT-29).

Une mise en œuvre DEVRAIT envoyer un segment "maintien en vie" sans données (DVRT-12) ; cependant, il PEUT être configurable à envoyer un "maintien en vie" contenant un octet factice (PEUT-6) pour la compatibilité avec des mises en œuvre TCP erronées.

3.8.5 Communication d'informations urgentes

En raison des différences de mise en œuvre et des interactions entre les boîtiers de médiation, les nouvelles applications NE DEVRAIENT PAS utiliser le mécanisme urgent de TCP (DVRT-13). Cependant, les mises en œuvre de TCP DOIVENT inclure la prise en charge du mécanisme d'urgence (DOIT-30). Des informations sur la façon dont certaines mises en œuvre de TCP interprètent le pointeur urgent se trouvent dans la [RFC6093].

L'objectif du mécanisme d'urgence TCP est de permettre à l'utilisateur envoyeur d'inciter l'utilisateur receveur à accepter certaines données urgentes et de permettre au point d'extrémité TCP receveur d'indiquer à l'utilisateur receveur quand toutes les données urgentes actuellement connues ont été reçues par l'utilisateur.

Ce mécanisme permet de désigner un point du flux de données comme la fin des informations urgentes. Chaque fois que ce point est en avance sur le numéro de séquence de réception (RCV.NXT) au point d'extrémité TCP de réception, alors la mise en œuvre TCP doit indiquer à l'utilisateur de passer en "mode urgent" ; quand le numéro de séquence de réception rattrape le pointeur urgent, la mise en œuvre TCP doit dire à l'utilisateur de passer en "mode normal". Si le pointeur urgent est mis à jour alors que l'utilisateur est en "mode urgent", la mise à jour sera invisible pour l'utilisateur.

La méthode utilise un champ Urgent qui est porté dans tous les segments transmis. Le fanion de contrôle URG indique que le champ Urgent est significatif et doit être ajouté au numéro de séquence du segment pour obtenir le pointeur Urgent. L'absence de ce fanion indique qu'il y a aucune donnée urgente en instance.

Pour envoyer une indication urgente, l'utilisateur doit également envoyer au moins un octet de données. Si l'utilisateur envoyeur indique aussi un "push", la livraison en temps voulu des informations urgentes au processus de destination est améliorée. Noter que, comme les modifications apportées au pointeur urgent correspondent à des données écrites par une application envoyeuse, le pointeur urgent ne peut pas "reculer" dans l'espace des numéros de séquence, mais un receveur TCP devrait être robuste aux valeurs de pointeur urgent invalides.

Une mise en œuvre TCP DOIT prendre en charge une séquence de données urgentes de n'importe quelle longueur (DOIT-31) [RFC1122].

Le pointeur urgent DOIT pointer sur le numéro de séquence de l'octet qui suit les données urgentes (DOIT-62).

Une mise en œuvre TCP DOIT (DOIT-32) informer la couche d'application de manière asynchrone chaque fois qu'elle reçoit un pointeur Urgent et qu'il n'y avait auparavant aucune donnée urgente en attente, ou chaque fois que le pointeur Urgent avance dans le flux de données. La mise en œuvre de TCP DOIT (DOIT-33) fournir à l'application un moyen de savoir combien de données urgentes restent à lire de la connexion, ou au moins de déterminer si des données plus urgentes restent à lire [RFC1122].

3.8.6 Gestion de la fenêtre

La fenêtre envoyée dans chaque segment indique la gamme de numéros de séquence que l'envoyeur de la fenêtre (le receveur de données) est actuellement prêt à accepter. On suppose que cela est lié à l'espace de mémoire tampon de données actuellement disponible pour cette connexion.

Le point d'extrémité TCP d'envoi regroupe les données à transmettre en segments qui tiennent dans la fenêtre actuelle, et peut reconditionner les segments dans la file d'attente de retransmission. Un tel reconditionnement n'est pas exigé, mais peut être utile.

Dans une connexion avec un flux de données unidirectionnel, les informations de fenêtre vont être portées dans des segments d'accusé de réception qui ont tous le même numéro de séquence, il n'y aura donc aucun moyen de les réarranger s'ils arrivent dans le désordre. Ce n'est pas un problème grave, mais cela permettra aux informations de fenêtre d'être occasionnellement temporairement fondées sur d'anciens rapports du receveur de données. Une astuce pour éviter ce problème, est d'agir sur les informations de fenêtre des segments qui portent le plus fort numéro d'accusé de réception (c'est-à-dire, les segments avec un numéro d'accusé de réception égal ou supérieur au plus élevé précédemment reçu).

L'indication d'une grande fenêtre favorise les transmissions. Si plus de données arrivent qu'il n'en peut être acceptées, elles seront éliminées. Cela se traduira par des retransmissions excessives, ajoutant inutilement à la charge sur le réseau et les points d'extrémité TCP. L'indication d'une petite fenêtre peut restreindre la transmission de données jusqu'à introduire un délai d'aller-retour entre chaque nouveau segment transmis.

Les mécanismes fournis permettent à un point d'extrémité TCP d'annoncer une grande fenêtre et d'annoncer par la suite une fenêtre beaucoup plus petite sans avoir accepté autant de données. Ce qu'on appelle le "rétrécissement de fenêtr" est fortement déconseillé. Le principe de robustesse [RFC1122] impose que les homologues TCP ne rétrécissent pas la fenêtre eux-mêmes, mais soient prêts à un tel comportement de la part d'autres homologues TCP.

Un receveur TCP NE DEVRAIT PAS rétrécir la fenêtre, c'est-à-dire déplacer le bord de fenêtre droit vers la gauche (DVRT-14). Toutefois, un homologue TCP émetteur DOIT être robuste au rétrécissement de fenêtre, qui peut entraîner que la "fenêtre utilisable" (voir le paragraphe 3.8.6.2.1) devienne négative (DOIT-34).

Si cela se produit, l'envoyeur NE DEVRAIT PAS envoyer de nouvelles données (DVRT-15), mais DOIT retransmettre normalement les anciennes données non acquittées entre SND.UNA et SND.UNA+SND.WND (DVRT-16). L'envoyeur PEUT aussi retransmettre d'anciennes données au-delà de SND.UNA+SND.WND (PEUT-7), mais NE DEVRAIT PAS mettre la connexion en fin de temporisation si les données au-delà du bord droit de la fenêtre ne sont pas acquittées (DVRT-17). Si la fenêtre est réduite à zéro, la mise en œuvre TCP DOIT la sonder de la manière standard (décrite ci-dessous) (DOIT-35).

3.8.6.1 Sondage de la fenêtre zéro

L'homologue TCP envoyeur doit transmettre régulièrement au moins un octet de nouvelles données (si il en est de disponibles) ou retransmettre à l'homologue TCP receveur même si la fenêtre d'envoi est zéro, afin de "sonder" la fenêtre. Cette retransmission est essentielle pour garantir que lorsque l'un ou l'autre des homologues TCP a une fenêtre de zéro, la réouverture de la fenêtre sera signalée de manière fiable à l'autre. Dans d'autres documents, on parle de sondage de fenêtre zéro (ZWP, *Zero-Window Probing*).

Le sondage des fenêtres zéro (offertes) DOIT être pris en charge (DOIT-36).

Une mise en œuvre TCP PEUT garder sa fenêtre de réception offerte fermée indéfiniment (PEUT-8). Tant que l'homologue TCP receveur continue d'envoyer des accusés de réception en réponse aux segments de sonde, l'homologue TCP envoyeur DOIT laisser la connexion ouverte (DOIT-37). Ceci permet à TCP de fonctionner dans des scénarios tels que "l'imprimante est en panne de papier" décrite au paragraphe 4.2.2.17 de la [RFC1122]. Le comportement dépend des préoccupations de

gestion de ressources de la mise en œuvre, comme indiqué dans la [RFC6429].

Quand l'homologue TCP receveur a une fenêtre zéro et qu'un segment arrive, il doit envoyer tout de même un accusé de réception indiquant son prochain numéro de séquence attendu et la fenêtre actuelle (zéro).

L'hôte envoyeur DEVRAIT envoyer la première sonde de fenêtre zéro quand une fenêtre a existé pendant la période de délai de retransmission (DVRT-29) (paragraphe 3.8.1) et DEVRAIT augmenter de façon exponentielle l'intervalle entre les sondes successives (DVRT-30).

3.8.6.2 Évitement du syndrome de la fenêtre idiote

Le "syndrome de la fenêtre idiote" (SWS, *Silly Window Syndrome*) est un modèle stable de petits mouvements incrémentaires de fenêtre entraînant des performances TCP extrêmement médiocres. Les algorithmes permettant d'éviter le SWS sont décrits ci-dessous, tant du côté envoyeur que du côté receveur. La RFC 1122 contient une discussion plus détaillée du problème de la SWS. Noter que l'algorithme de Nagle et l'algorithme d'évitement de SWS de l'envoyeur jouent des rôles complémentaires dans l'amélioration des performances. L'algorithme de Nagle décourage l'envoi de petits segments lorsque les données à envoyer augmentent par petits incréments, tandis que l'algorithme d'évitement de SWS décourage les petits segments résultant de l'avancement du bord droit de la fenêtre par petits incréments.

3.8.6.2.1 Algorithme de l'envoyeur -- Quand envoyer des données

Une mise en œuvre TCP DOIT inclure un algorithme d'évitement de SWS chez l'envoyeur (DOIT-38).

L'algorithme de Nagle du paragraphe 3.7.4 décrit en outre comment fusionner des segments courts.

L'algorithme d'évitement de SWS de l'envoyeur est plus difficile que celui du receveur parce que l'envoyeur ne connaît pas (directement) l'espace de mémoire tampon total (RCV.BUFF) du receveur. Une approche qui s'est avérée efficace est pour l'envoyeur de calculer Max(SND.WND) qui est la fenêtre d'envoi maximum qu'il a vue jusqu'à présent sur la connexion, et d'utiliser cette valeur comme une estimation de la RCV.BUFF. Malheureusement, il ne peut s'agir que d'une estimation ; le receveur peut, à tout moment, temps réduire la taille de la RCV.BUFF. Pour éviter qu'il en résulte un blocage, il est nécessaire d'avoir une temporisation pour forcer la transmission des données, outrepassant l'algorithme d'évitement de SWS. En pratique, cette fin de temporisation devrait rarement se produire.

La "fenêtre utilisable" est : U = SND.UNA + SND.WND - SND.NXT

c'est-à-dire la fenêtre offerte moins la quantité de données envoyées mais pas acquittées. Si D est la quantité de données mises en file d'attente au point d'extrémité TCP d'envoi mais pas encore envoyées, alors l'ensemble de règles suivant est recommandé :

Envoi des données :
(1) si un segment de taille maximale peut être envoyé, c'est-à-dire si : min(D,U) >= Eff.snd.MSS ;
(2) ou si les données sont poussées et que toutes les données en file d'attente peuvent être envoyées maintenant, c'est-à-dire si : [SND.NXT = SND.UNA et] PUSH et D ≤ U (la condition entre crochets est imposée par l'algorithme de Nagle) ;
(3) ou si au moins une fraction de Fs de la fenêtre maximale peut être envoyée, c'est-à-dire si :
[SND.NXT = SND.UNA et] min(D,U) ≥ Fs * Max(SND.WND) ;
(4) ou si la fin de temporisation d'outrepassement survient.

Ici, Fs est une fraction dont la valeur recommandée est 1/2. La fin de temporisation d'outrepassement devrait être comprise entre 0,1 et 1,0 secondes. Il peut être pratique de combiner ce temporisateur avec celui utilisé pour sonder les fenêtres zéro (paragraphe 3.8.6.1).

3.8.6.2.2 Algorithme du receveur -- quand envoyer une mise à jour de fenêtre

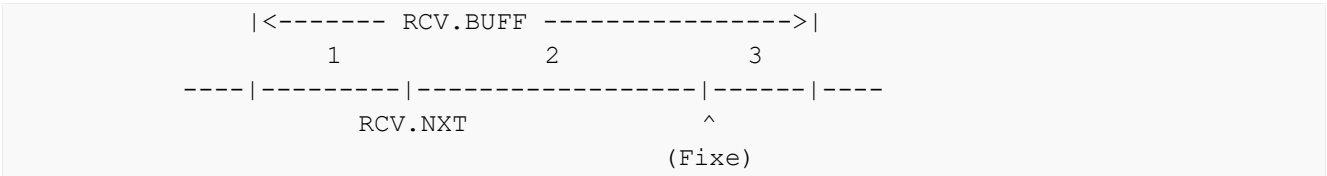
Une mise en œuvre TCP DOIT inclure un algorithme d'évitement de SWS chez le receveur (DOIT-39).

L'algorithme d'évitement de SWS du receveur détermine quand le bord droit de la fenêtre peut être avancé ; c'est habituellement appelé "mise à jour de fenêtre". Cet algorithme se combine avec l'algorithme d'ACK retardé (paragraphe 3.8.6.3) pour déterminer quand un segment ACK contenant la fenêtre actuelle sera réellement envoyé au receveur.

La solution à la SWS de receveur est d'éviter de faire avancer le bord de fenêtre droit RCV.NXT+RCV.WND par petits incréments, même si les données sont reçues du réseau en petit segments.

Supposons que l'espace total de la mémoire tampon de réception soit RCV.BUFF. À tout moment, RCV.USER octets de ce total peuvent être liés aux données qui ont été reçues et acquittées, mais que le processus utilisateur n'a pas encore consommées. Lorsque la connexion est au repos, RCV.WND = RCV.BUFF et RCV.USER = 0.

Garder le bord droit de la fenêtre fixe à mesure que les données arrivent et sont acquittées exige que le receveur offre moins que son espace tampon complet, c'est-à-dire que le receveur doit spécifiez un RCV.WND qui conserve RCV.NXT+RCV.WND constant lorsque RCV.NXT augmente. Ainsi, l'espace de mémoire tampon total RCV.BUFF est généralement divisé en trois parties :



1 - RCV.USER = données reçues mais pas encore consommées ;
2 - RCV.WND = espace annoncé à l'envoyeur ;
3 - Réduction = espace disponible mais pas encore annoncé.

L'algorithme d'évitement de SWS suggéré pour le receveur est de garder RCV.NXT+RCV.WND fixe jusqu'à ce que la réduction satisfasse : RCV.BUFF - RCV.USER - RCV.WND ≥ min(Fr * RCV.BUFF, Eff.snd.MSS) où Fr est une fraction dont la valeur recommandée est 1/2, et Eff.snd.MSS est la MSS d'envoi effectif pour la connexion (voir le paragraphe 3.7.1). Lorsque l'inégalité est satisfaite, RCV.WND est réglé à RCV.BUFF-RCV.USER.

Noter que l'effet général de cet algorithme est d'avancer RCV.WND par incréments de Eff.snd.MSS (pour les mémoires tampon de réception réalistes : Eff.snd.MSS < RCV.BUFF/2). Noter également que le receveur doit utiliser son propre Eff.snd.MSS, en supposant qu'il est le même que celui de l'envoyeur.

3.8.6.3 Accusés de réception différés -- quand envoyer un segment d'accusé de réception

Un hôte qui reçoit un flux de segments de données TCP peut augmenter l'efficacité du réseau et des hôtes en envoyant moins d'un segment d'accusé de réception par segment de données reçu ; c'est ce qu'on appelle un "ACK retardé".

Un point d'extrémité TCP DEVRAIT mettre en œuvre un accusé de réception retardé (DVRT-18), mais un ACK ne devrait pas être excessivement retardé ; en particulier, le retard DOIT être de moins de 0,5 seconde (DOIT-40). Un ACK DEVRAIT être généré au moins à chaque second segment de taille normale ou 2*RMSS octets de nouvelles données (où RMSS est la MSS spécifiée par le point d'extrémité TCP recevant les segments à acquitter ou la valeur par défaut si elle n'est pas spécifiée) (DVRT-19). Des retards excessifs sur les ACK peuvent perturber les algorithmes de temps d'aller-retour et d'horloge des paquets. Une discussion plus complète sur le comportement d'accusé de réception retardé se trouve au paragraphe 4.2 de la [RFC5681], y compris des recommandations visant à accuser immédiatement réception des segments reçus dans le désordre, des segments au-dessus d'un trou dans l'espace des numéros de séquence, ou de segments qui remplissent tout ou partie d'un trou, afin d'accélérer la récupération des pertes.

Noter qu'il existe plusieurs pratiques qui entraînent une réduction du nombre d'ACK, notamment le déchargement de réception générique (GRO, *generic receive offload*) [72], la compression d'ACK et la décimation d'ACK [RFC3449].

3.9 Interfaces

Il y a bien sûr deux interfaces préoccupantes : l'interface utilisateur/TCP et l'interface TCP/niveau inférieur. On a un modèle assez élaboré de l'interface utilisateur/TCP, mais l'interface au module de protocole de niveau inférieur n'est pas spécifié ici car il sera spécifié en détail par la spécification du protocole de niveau inférieur. Pour le cas où le niveau inférieur est IP, on note certaines des valeurs des paramètres que les mises en œuvre TCP pourraient utiliser.

3.9.1 Interface utilisateur/TCP

La description fonctionnelle suivante des commandes d'utilisateur pour la mise en œuvre TCP est au mieux, fictive, puisque chaque système d'exploitation va avoir des facilités différentes. Par conséquent, on doit avertir le lecteur que les différentes mises en œuvre de TCP peuvent avoir des interfaces d'utilisateur différentes. Cependant, toutes les mises en œuvre de TCP doivent fournir un certain ensemble minimum de services pour garantir que toutes les mises en œuvre de TCP peuvent prendre en charge la même hiérarchie de protocoles. Ce paragraphe spécifie les interfaces fonctionnelles requises de toutes les mises en œuvre de TCP.

Le paragraphe 3.1 de la [RFC8303] identifie également les primitives fournies par TCP et pourrait être utilisé comme référence supplémentaire pour les mises en œuvre.

Les paragraphes suivants caractérisent fonctionnellement une interface d'utilisateur/TCP. La notation utilisée est similaire à celle de la plupart des procédures ou invocations de fonction dans les langages de haut niveau, mais cette utilisation n'est pas destinée à exclure les invocations de service de type filtre.

Les commandes d'utilisateur décrites ci-dessous spécifient les fonctions de base que la mise en œuvre de TCP doit effectuer pour prendre en charge la communication inter processus. Les mises en œuvre individuelles doivent définir leur propre format exact et fournir des combinaisons ou des sous-ensembles des fonctions de base dans des invocations uniques. En particulier, certaines mises en œuvre peuvent souhaiter OUVRIR automatiquement une connexion lors du premier SEND ou RECEIVE émis par l'utilisateur pour une connexion donnée.

En fournissant des fonctions de communication inter processus, la mise en œuvre TCP doit non seulement accepter les commandes, mais aussi renvoyer des informations au processus qu'elle sert. Ces dernières se composent :

a) d'informations générales sur une connexion (par exemple, des interruptions, fermeture à distance, lien à une prise à distance non spécifiée).
b) de réponses à des commandes spécifiques de l'utilisateur indiquant un succès ou différents types de défaillances.

3.9.1.1 OPEN

Format : OPEN (accès local, prise distante, actif/passif [, délai d'expiration] [, champ Diffserv] [, sécurité/compartiment] [, adresse IP locale] [, options]) -> nom de la connexion locale

Si le fanion actif/passif est réglé à passif, il s'agit d'une invocation de LISTEN pour une connexion entrante. Un OPEN passif peut avoir une prise distante pleinement spécifiée pour attendre sur une connexion particulière ou une prise à distance non spécifiée à attendre pour tout appel. Un appel passif pleinement spécifié peut être rendu actif par l'exécution ultérieure d'un SEND.

Un bloc de commande de transmission (TCB, *Transmission Control Block*) est créé et rempli partiellement avec les données provenant des paramètres de la commande OPEN.

Chaque appel OPEN passif crée un nouvel enregistrement de connexion dans l'état LISTEN, ou il renvoie une erreur ; il NE DOIT PAS affecter d'enregistrement de connexion précédemment créé (DOIT-41).

Une mise en œuvre TCP qui prend en charge plusieurs connexions concurrentes DOIT fournir un appel OPEN qui permettra fonctionnellement une application de LISTEN sur un accès tandis qu'un bloc de connexion avec le même accès local est dans l'état SYN-SENT ou SYN-RECEIVED (DOIT-42).

Sur une commande OPEN active, le point d'extrémité TCP commence la procédure pour synchroniser (c'est-à-dire établir) la connexion en une seule fois.

La temporisation, si elle est présente, permet à l'appelant de configurer un délai d'expiration pour toutes les données soumises à TCP. Si la livraison des données n'est pas réussie dans le délai d'expiration, le point d'extrémité TCP va interrompre la connexion. La valeur globale actuelle par défaut est de cinq minutes.

La mise en œuvre TCP ou un composant du système d'exploitation va vérifier le droit de l'utilisateur d'ouvrir une connexion avec la valeur du champ Diffserv ou sécurité/compartiment. L'absence d'une valeur du champ Diffserv ou la spécification de sécurité/compartiment dans l'appel OPEN indique que les valeurs par défaut doivent être utilisées.

TCP n'acceptera les demandes entrantes comme correspondantes que si les informations de sécurité/compartiment sont

exactement les mêmes que dans l'appel OPEN.

La valeur du champ Diffserv indiquée par l'utilisateur n'a d'impact que sur les paquets sortants, peut être modifiée en cours de route sur le réseau et n'a pas d'incidence directe ou de relation avec les paquets reçus.

Un nom de connexion locale sera renvoyé à l'utilisateur par la mise en œuvre TCP. Le nom de la connexion locale peut ensuite être utilisé comme un terme abrégé pour la connexion définie par la paire <prise locale, prise distante>.

Le paramètre facultatif "adresse IP locale" DOIT être pris en charge pour permettre la spécification de l'adresse IP locale (DOIT-43). Cela permet aux applications de choisir l'adresse IP locale utilisée lorsque le multi-rattachements est présent.

Un appel OPEN passif avec un paramètre "adresse IP locale" spécifié attendra une demande de connexion entrante pour cette adresse. Si le paramètre n'est pas spécifié, un OPEN passif attendra une demande de connexion entrante sur n'importe quelle adresse IP locale, et va ensuite lier l'adresse IP locale de la connexion à l'adresse particulière qui est utilisée.

Pour un appel OPEN actif, un paramètre "adresse IP locale" spécifié sera utilisé pour ouvrir la connexion. Si le paramètre n'est pas spécifié, l'hôte choisira une adresse IP locale appropriée (voir la RFC 1122, paragraphe 3.3.4.2).

Si une application sur un hôte multi rattachements ne spécifie pas l'adresse IP locale lors de l'ouverture active d'une connexion TCP, alors la mise en œuvre TCP DOIT demander à la couche IP de choisir une adresse IP locale avant d'envoyer le (premier) SYN (DOIT-44). Voir la fonction GET_SRCADDR() au paragraphe 3.4 de la RFC 1122.

À tout autre moment, un segment précédent a été envoyé ou reçu sur cette connexion, et les mises en œuvre de TCP DOIVENT utiliser la même adresse locale qu'utilisée dans ces précédents segments (DOIT-45).

Une mise en œuvre TCP DOIT rejeter comme erreur un appel OPEN local pour une adresse IP distante non valide (par exemple, une adresse de diffusion ou de diffusion groupée) (DOIT-46).

3.9.1.2 SEND

Format : SEND (nom de la connexion locale, adresse de mémoire tampon, compte d'octets, fanion URGENT [, fanion PUSH] [, temporisateur])

Cet appel cause l'envoi des données contenues dans la mémoire tampon d'utilisateur indiquée sur la connexion indiquée. Si la connexion n'a pas été ouverte, le SEND est considéré comme une erreur. Quelques mises en œuvre peuvent permettre aux utilisateurs de faire SEND en premier ; dans ce cas, un OPEN automatique serait effectué. Par exemple, cela pourrait être une façon pour que les données d'application soient incluses dans les segments SYN. Si le processus d'appel n'est pas autorisé à utiliser cette connexion, une erreur est retournée.

Un point d'extrémité TCP PEUT mettre en œuvre des fanions PUSH sur les appels SEND (PEUT-15). Si les fanions PUSH ne sont pas mis en œuvre, alors l'homologue TCP envoyeur : (1) NE DOIT PAS mettre indéfiniment les données en mémoire tampon (DOIT-60), et (2) DOIT établir le bit PSH dans le dernier segment mis en mémoire tampon (c'est-à-dire lorsqu'il n'y a plus de données en file d'attente à envoyer) (DOIT-61). Le reste de la description ci-dessous suppose que le fanion PUSH est pris en charge sur les appels SEND.

Si le fanion PUSH est établi, l'application a l'intention de transmettre rapidement les données au receveur, et le bit PSH sera établi dans le dernier segment TCP créé à partir de la mémoire tampon.

Le bit PSH n'est pas un marqueur d'enregistrement et est indépendant des limites de segment. L'émetteur DEVRAIT écraser les bits suivants lorsqu'il met en paquet des données, pour envoyer le segment le plus grand possible (DVRT-27).

Si le fanion PUSH n'est pas établi, les données peuvent être combinées avec des données provenant de SEND ultérieurs pour l'efficacité de la transmission. Lorsqu'une application émet une série de SEND sans établir le fanion PUSH, la mise en œuvre TCP PEUT agréger les données en interne sans les envoyer (PEUT-16). Noter que lorsque l'algorithme de Nagle est utilisé, les mises en œuvre TCP peuvent mettre les données en mémoire tampon avant de les envoyer, sans tenir compte du fanion PUSH (voir le paragraphe 3.7.4).

Un programme d'application est logiquement tenu d'établir le fanion PUSH dans un message SEND chaque fois qu'il doit forcer la livraison des données pour éviter une impasse de communication. Cependant, une mise en œuvre de TCP DEVRAIT envoyer un segment de taille maximale chaque fois que possible (DVRT-28) pour améliorer les performances

(voir le paragraphe 3.8.6.2.1).

Les nouvelles applications NE DEVRAIENT PAS établir le fanion URGENT [RFC6093] en raison de problèmes de différences de mise en œuvre et de boîtier de médiation (DVRT-13).

Si le fanion URGENT est établi, les segments envoyés à l'homologue TCP de destination vont avoir le pointeur urgent établi. L'homologue TCP receveur va signaler la condition urgente au processus de réception si le pointeur d'urgence indique que les données précédant le pointeur urgent n'ont pas été consommées par le processus de réception. L'objectif du fanion URGENT est d'inciter le receveur à traiter les données urgentes et à indiquer au destinataire quand toutes les données urgentes connues ont été reçues. Le nombre de fois que la mise en œuvre TCP de l'utilisateur envoyeur signale l'urgence ne sera pas nécessairement égal au nombre de fois que la présence de données urgentes a été signalée à l'utilisateur receveur.

Si aucune prise distante n'a été spécifiée dans l'instruction OPEN, mais si la connexion est établie (par exemple, parce qu'une connexion LISTENING est devenue spécifique en raison de l'arrivée d'un segment distant pour la prise locale) alors la mémoire tampon désignée est envoyée à la prise distante impliquée. Les utilisateurs qui utilisent OPEN avec une prise distante non spécifiée peuvent utiliser SEND sans jamais connaître explicitement l'adresse de la prise distante.

Cependant, si une tentative de SEND est effectuée avant que la prise distante soit spécifiée, une erreur sera renvoyée. Les utilisateurs peuvent utiliser l'appel STATUS pour déterminer l'état de la connexion. Des mises en œuvre TCP peuvent notifier à l'utilisateur lorsqu'une prise non spécifiée est liée.

Si une fin de temporisation est spécifiée, le délai d'expiration actuel de l'utilisateur pour cette connexion est remplacé par le nouveau.

Dans la mise en œuvre la plus simple, SEND ne rendrait pas le contrôle au processus d'envoi jusqu'à ce que la transmission soit terminée ou que le délai de temporisation ait été dépassé. Cependant, cette méthode simple est à la fois sujette à des blocages (par exemple, les deux côtés de la la connexion pourrait essayer d'effectuer des SEND avant d'effectuer des RECEIVE) et offre de mauvaises performances, elle n'est donc pas recommandée. Un mise en œuvre plus sophistiquée reviendrait immédiatement pour permettre le processus d'exécution simultanée des entrées/sorties réseau, et en outre, pour permettre à plusieurs SEND d'être en cours. Les SEND multiples sont servis dans l'ordre du premier arrivé, premier servi, donc le point d'extrémité TCP mettra en file d'attente ceux qu'il ne peut pas traiter immédiatement.

On a implicitement supposé une interface d'utilisateur asynchrone dans laquelle un SEND suscite plus tard une sorte de SIGNAL ou pseudo-interruption du point d'extrémité TCP de service. Une solution de remplacement consiste à retourner une réponse immédiatement. Par exemple, des SEND pourraient renvoyer un accusé de réception local immédiat, même si le segment envoyé n'avait pas été acquitté par le point d'extrémité TCP distant. On pourrait être optimiste en supposant éventuellement le succès. Si on se trompe, la connexion va fermer quand même en raison du de la fin de temporisation. Dans les mises en œuvre de cette sorte (synchrone) il y aura encore des signaux asynchrones, mais ceux-ci vont s'occuper de la connexion elle-même, et non des segments ou mémoires tampons spécifiques.

Pour que le processus fasse la distinction entre les indications d'erreur et de succès pour les différents SEND, il peut être approprié que l'adresse de mémoire tampon soit retournée avec la réponse codée à la demande SEND. Les signaux de TCP à utilisateur sont décrits ci-dessous, en indiquant les renseignements qui doivent être retournés au processus d'appel.

3.9.1.3 RECEIVE

Format : RECEIVE (nom de la connexion locale, adresse de mémoire tampon, compte d'octets) -> compte d'octets, fanion URGENT [, fanion PUSH]

Cette commande alloue une mémoire tampon de réception associée à la connexion spécifiée. Si aucun OPEN ne précède cette commande ou si le processus appelant n'est pas autorisé à utiliser cette connexion, une erreur est renvoyée.

Dans la mise en œuvre la plus simple, le contrôle ne reviendrait pas au programme appelant jusqu'à ce que la mémoire tampon soit remplie ou qu'une erreur se produise, mais ce schéma est fortement sujet à des blocages. Une mise en œuvre plus sophistiquée permettrait que plusieurs RECEIVE soient en instance à la fois. Elles seraient remplies au fur et à mesure de l'arrivée des segments. Cette stratégie permet d'augmenter le débit au prix d'un schéma plus élaboré (éventuellement asynchrone) pour notifier au programme appelant qu'un PUSH a été détecté ou qu'une mémoire tampon a été remplie.

Un receveur TCP PEUT passer un bit PSH reçu à la couche d'application via le fanion PUSH dans l'interface (PEUT-17),

mais ce n'est pas exigé (cela a été précisé dans au paragraphe 4.2.2.2 de la RFC 1122). Le reste du texte décrivant l'appel RECEIVE ci-dessous suppose que la transmission de l'indication PUSH est prise en charge.

Si suffisamment de données arrivent pour remplir la mémoire tampon avant qu'un PUSH soit vu, le fanion PUSH ne sera pas établi dans la réponse au RECEIVE. La mémoire tampon sera remplie d'autant de données qu'elle peut contenir. Si un PUSH est vu avant que la mémoire tampon soit remplie, elle sera renvoyée partiellement remplie et PUSH sera indiqué.

Si il y a des données urgentes, l'utilisateur en aura été informé aussitôt qu'elles arrivent via un signal TCP à utilisateur. L'utilisateur receveur devrait donc être en "mode urgent". Si le fanion URGENT est établi, les données supplémentaires urgentes demeurent. Si le fanion URGENT est à zéro, cet appel à RECEIVE a retourné toutes les données urgentes, et l'utilisateur peut maintenant quitter le "mode urgent". Noter que les données qui suivent le pointeur Urgent (données non urgentes) ne peuvent pas être livrées à l'utilisateur dans la même mémoire tampon que les données urgentes précédentes, à moins que leur limite soit clairement marquée pour l'utilisateur.

Pour distinguer entrer plusieurs RECEIVE en cours et prendre soin du cas où une mémoire tampon n'est pas complètement remplie, le code de retour est accompagné d'un pointeur de mémoire tampon et d'un compte d'octets indiquant la longueur réelle des données reçues.

D'autres mises en œuvre de RECEIVE peuvent faire que le point d'extrémité TCP alloue un stockage de mémoire tampon ou que le point d'extrémité TCP puisse partager une mémoire tampon en anneau avec l'utilisateur.

3.9.1.4 CLOSE

Format : CLOSE (nom de la connexion locale)

Cette commande cause la fermeture de la connexion spécifiée. Si la connexion n'est pas ouverte ou si le processus d'appel n'est pas autorisé à utiliser cette connexion, une erreur est renvoyée. La fermeture des connexions est censée être une opération en douceur en ce sens que les SEND en instance seront transmis (et retransmis) lorsque le contrôle de flux le permet, jusqu'à ce que toutes soient servies. Donc, il devrait être acceptable d'effectuer plusieurs appels SEND, suivis d'un CLOSE, et s'attendre à ce que toutes les données soient envoyées à la destination. Il devrait aussi être clair que les utilisateurs devraient continuer à RECEIVE sur les connexions CLOSING car l'homologue distant peut essayer de transmettre les dernières de ses données. Donc, CLOSE signifie "Je n'ai plus rien à envoyer" mais ne signifie pas "Je ne recevrai plus rien". Il peut arriver (si le protocole au niveau utilisateur est pas bien pensé) que le côté qui ferme soit incapable de se débarrasser de toutes ses données avant la fin de temporisation. Dans ce cas, CLOSE devient un ABORT, et l'homologue TCP qui ferme abandonne.

L'utilisateur peut FERMER la connexion à tout moment à sa propre initiative ou en réponse à diverses invites de la mise en œuvre de TCP (par exemple, fermeture à distance exécutée, délai de temporisation de transmission dépassé, destination inaccessible).

Parce que la fermeture d'une connexion nécessite une communication avec l'homologue TCP distant, les connexions peuvent rester à l'état de fermeture pour un peu de temps. Tenter de rouvrir la connexion avant que l'homologue TCP réponde à la commande CLOSE va entraîner des réponses d'erreur.

Fermer implique également une fonction PUSH.

3.9.1.5 Statut

Format : STATUS (nom de la connexion locale) -> données d'état

Il s'agit d'une commande d'utilisateur dépendante de la mise en œuvre qui peut être exclue sans effet défavorable. Les renseignements retournées proviennent généralement du TCB associé à la connexion.

Cette commande renvoie un bloc de données contenant les informations suivantes :
prise locale,
prise distante,
nom de la connexion locale,
fenêtre de réception,
fenêtre d'envoi,

état de connexion,
nombre de mémoires tampon en attente d'accusé de réception,
nombre de mémoires tampons en attente de réception,
état urgentv
valeur du champ Diffserv,
sécurité/compartiment, et
délai d'expiration de la transmission.

Selon l'état de la connexion, ou l'état de la mise en œuvre elle-même, certaines de ces informations peuvent ne pas être disponibles ou significatives. Si le processus appelant n'est pas autorisé à utiliser cette connexion, une erreur est retournée. Cela empêche les processus non autorisés d'obtenir des informations sur une connexion.

3.9.1.6 ABORT

Format : ABORT (nom de la connexion locale)

Cette commande cause l'interruption de tous les SEND et RECEIVE, la suppression du TCB, et l'envoi d'un message spécial RST à l'homologue TCP distant de la connexion. Selon la mise en œuvre, les utilisateurs peuvent recevoir des indications d'interruption pour chaque SEND ou RECEIVE en cours, ou peuvent simplement recevoir un accusé de réception de ABORT.

3.9.1.7 Purge

Certaines mises en œuvre TCP ont inclus un appel FLUSH, qui videra la file d'attente d'envoi TCP de toutes les données pour lesquelles l'utilisateur a lancé des appels SEND mais qui se trouvent toujours à la droite de la fenêtre d'envoi actuelle. C'est-à-dire qu'il purge autant de données en file d'attente d'envoi que possible sans perdre la synchronisation de numéro de séquence. L'appel FLUSH PEUT être mis en œuvre (PEUT-14).

3.9.1.8 Rapports asynchrones

Il DOIT y avoir un mécanisme de rapport de conditions d'erreur TCP légères à l'application (DOIT-47). De manière générale, on suppose que cela prend la forme d'un sous programme ERROR_REPORT *(rapport d'erreur)* fourni par l'application qui peut être invoqué de manière asynchrone à partir de la couche transport :

ERROR_REPORT(nom de la connexion locale, raison, sous-raison)

Le codage précis des paramètres de raison et de sous-raison n'est pas précisé ici. Cependant, les conditions qui sont rapportées de manière asynchrone à l'application DOIVENT inclure :
- L'arrivée d'un message d'erreur ICMP (voir le paragraphe 3.9.2.2 pour une description de la gestion de chaque type de message ICMP car certains types de messages ne doivent pas générer de rapports à l'application)
- Les retransmissions excessives (voir le paragraphe 3.8.3)
- L'avance du pointeur d'urgence (voir le paragraphe 3.8.5)

Cependant, un programme d'application qui ne veut pas recevoir de tels appels de ERROR_REPORT DEVRAIT être capable de désactiver effectivement ces appels (DVRT-20).

3.9.1.9 Définir le champ Services différenciés (TOS IPv4 ou classe de trafic IPv6)

La couche application DOIT être capable de spécifier le champ Services différenciés pour les segments envoyés sur une connexion (DOIT-48). Le champ Services différenciés inclut les 6 bits de la valeur du codet de services différenciés (DSCP, *Differentiated Services Codepoint*). Ce n'est pas obligatoire, mais l'application DEVRAIT être capable de changer le champ Services différenciés pendant la durée de vie de la connexion (DVRT-21). Les mises en œuvre TCP DEVRAIENT passer la valeur actuelle du champ Services différenciés sans changement à la couche IP, quand elles envoient des segments sur la connexion (DVRT-22).

Le champ Services différenciés sera spécifié indépendamment dans chaque direction de la connexion, de sorte que l'application du receveur va spécifier le champ Services différenciés utilisé pour les segments ACK.

Les mises en œuvre TCP PEUVENT passer le champ Services différenciés le plus récemment reçu jusqu'à l'application (PEUT-9).

3.9.2 Interface TCP/de niveau inférieur

Le point d'extrémité TCP fait ses appels sur un module de protocole de niveau inférieur pour en fait envoyer et recevoir des informations sur un réseau. Les deux versions standard actuelles du protocole Internet (IP) mises en couches sous TCP sont IPv4 [RFC0791] et IPv6 [RFC8200].

Si le protocole de niveau inférieur est IPv4, il fournit des arguments pour un type de service (utilisé dans le champ Services différenciés) et pour une durée de vie. TCP utilise les réglages suivants pour ces paramètres :

Champ Diffserv : la valeur d'en-tête IP du champ Diffserv est donnée par l'utilisateur. Cela inclut les bits du codet Diffserv (DSCP).

Durée de vie (TTL) : la valeur de TTL utilisée pour l'envoi de segments TCP DOIT être configurable (DOIT-49).

Noter que la RFC 793 spécifiait une minute (60 secondes) comme constante pour lae TTL car la durée de vie maximale supposée du segment était de deux minutes. C'était destiné à demander explicitement qu'un segment soit détruit s'il ne peut être livré par le système Internet en une minute. La RFC 1122 a mis à jour la RFC 793 pour exiger que la TTL soit configurable.

Noter qu'il est permis que le champ Diffserv soit changé lors d'une connexion (paragraphe 4.2.4.2 de la RFC 1122). Cependant, l'interface d'application peut ne prend pas en charge cette capacité, et l'application n'a pas connaissances des segments TCP individuels, donc cela ne peut être fait, au mieux, que sur une granularité grossière. Cette limitation est discutée plus en détails aux paragraphes 5.1, 5.3 et à la Section 6 de la [RFC7657]. En général, une application NE DEVRAIT PAS modifier la valeur du champ Diffserv dans le cours d'une connexion (DVRT-23).

Tout protocole de niveau inférieur devra fournir l'adresse de source, l'adresse de destination, et les champs de protocole, et un moyen de déterminer la "longueur TCP", à la fois pour fournir le service fonctionnel équivalent d'IP et pour l'utiliser dans la somme de contrôle TCP.

Lorsque les options reçues sont transmises à TCP à partir de la couche IP, une mise en œuvre TCP DOIT ignorer les options qu'elle ne comprend pas (DOIT-50).

Une mise en œuvre TCP PEUT prendre en charge les options Horodatage (PEUT-10) et Enregistrement de chemin (PEUT-11).

3.9.2.1 Acheminement de source

Si le niveau inférieur est IP (ou un autre protocole qui fournit cette caractéristique) et si l'acheminement de source est utilisé, l'interface doit permettre de communiquer les informations sur le chemin. C'est particulièrement important afin que les adresses de source et de destination utilisées dans la somme de contrôle TCP soient la source d'origine et la destination finale. Il est également important de préserver le chemin de retour pour répondre aux demandes de connexion.

Une application DOIT être capable de spécifier un chemin de source quand elle ouvre activement une connexion TCP (DOIT-51) et cela DOIT avoir la priorité sur un chemin de source reçu dans un datagramme (DOIT-52).

Quand une connexion TCP est OUVERTE passivement et qu'un paquet arrive avec une option route de source IP complète (contenant un chemin de retour) les mises en œuvre TCP DOIVENT sauvegarder le chemin de retour et l'utiliser pour tous les segments envoyés sur cette connexion (DOIT-53). Si une route de source différente arrive dans un segment ultérieur, la dernière définition DEVRAIT remplacer la précédente (DVRT-24).

3.9.2.2 Messages ICMP

Les mises en œuvre TCP DOIVENT agir sur un message d'erreur ICMP transmis par la couche IP, en le dirigeant vers la connexion qui a créé l'erreur (DOIT-54). Les informations de démultiplexage nécessaires peuvent être trouvées dans l'en-tête IP contenu dans le message ICMP.

Cela s'applique à ICMPv6 en plus d'ICMP IPv4.

La [RFC5461] contient une discussion sur des messages ICMP et ICMPv6 spécifiques classés en erreurs "douces" ou "dures" qui peuvent entraîner des réponses différentes. Le traitement des classes de messages ICMP est décrit ci-dessous :

Extinction de source : les mises en œuvre TCP DOIVENT éliminer en silence tous les messages ICMP Extinction de source reçus (DOIT-55). Voir la discussion dans la [RFC6633].

Erreurs douces : pour ICMP IPv4, cela inclut : Destination inaccessible : codes 0, 1, 5 ;
Temps dépassé -- codes 0, 1 ; et
Problème de paramètre ;
pour ICMPv6, cela inclut : Destination inaccessible -- codes 0, 3 ;
Temps dépassé -- codes 0, 1 ; et
Problème de paramètre -- codes 0, 1, 2.

Comme ces messages Inaccessible indiquent des conditions d'erreur douce, une mise en œuvre TCP NE DOIT PAS interrompre la connexion (DOIT-56) et elle DEVRAIT mettre l'information à la disposition de l' application (DVRT-25).

Erreurs dures : Pour ICMP, cela inclut Destination inaccessible - codes 2 à 4

Il s'agit de conditions d'erreur dures, donc les mises en œuvre TCP DEVRAIENT interrompre la connexion (DVRT-26). La [RFC5461] note que certaines mises en œuvre n'interrompent pas les connexions quand une erreur dure ICMP est reçue pour une connexion qui est dans un des états synchronisés.

Noter que la Section 4 de la [RFC5461] décrit un comportement généralisé de mise en œuvre qui traite les erreurs douces comme des erreurs dures lors de l'établissement de la connexion.

3.9.2.3 Validation de l'adresse de source

La RFC 1122 exige que les adresses soient validées dans les paquets SYN entrants :

Un SYN entrant avec une adresse de source invalide DOIT être ignoré soit par TCP, soit par la couche IP [(DOIT-63)] (voir le paragraphe 3.2.1.3).
Une mise en œuvre TCP DOIT supprimer en silence un segment SYN entrant qui est adressé à une adresse de diffusion ou de diffusion groupée [(DOIT-57)].

Cela empêche que l'état de la connexion et les réponses soient générés par erreur, et les mises en œuvre devraient noter que ces conseils s'appliquent à tous les segments entrants, pas seulement aux SYN, comme spécifiquement indiqué dans la RFC 1122.

3.10 Traitement des événements

Le traitement décrit dans ce paragraphe est un exemple d'une mise en œuvre possible. D'autres mises en œuvre peuvent avoir des séquences de traitement légèrement différentes, mais elles devraient différer de celles de ce paragraphe seulement par des détails, pas par leur substance.

L'activité du point d'extrémité TCP peut être caractérisée comme une réponse à des événements. Les événements qui se produisent peuvent être classés en trois catégories : les appels d'utilisateurs, les segments d'arrivée et les fins de temporisation. Cette section décrit le traitement fait par le point d'extrémité TCP en réponse à chacun des événements. Dans de nombreux cas, le traitement nécessaire dépend de l'état de la connexion.

Événements qui se produisent :

- Appels utilisateur
  OPEN
  SEND
  RECEIVE
  CLOSE
  ABORT
  STATUT

- Segments d'arrivée
  SEGMENT ARRIVE
- Fins de temporisation
  FIN DE TEMPORISATION DE L'UTILISATEUR
  FIN DE TEMPORISATION DE LA RETRANSMISSION
  FIN DE TEMPORISATION DE DÉLAI D'ATTENTE

Le modèle de l'interface TCP/utilisateur est que les commandes d'utilisateur reçoivent un retour immédiat et éventuellement une réponse retardée via un événement ou une pseudo-interruption. Dans les descriptions qui suivent, le terme "signaler" signifie provoquer une réponse retardée.

Les réponses d'erreur dans ce document sont identifiées par des chaînes de caractères. Par exemple, les commandes d'utilisateur qui font référence à des connexions qui n'existent pas reçoivent "Erreur : connexion non ouverte".

Noter que dans ce qui suit, toutes les arithmétiques sur les numéros de séquence, numéros d'accusé de réception, fenêtres, etc., sont modulo 2^{32} (la taille de l'espace de numéro de séquence). Noter également que "≤" signifie inférieur ou égal à (modulo 2^{32}).

Une façon naturelle d'envisager le traitement des segments entrants est d'imaginer qu'on vérifie d'abord si ils ont le bon numéro de séquence (c'est-à-dire, que leur contenu se situe dans la plage de la "fenêtre de réception" attendue dans l'espace des numéros de séquence) et ensuite qu'ils sont généralement mis en file d'attente et traités dans l'ordre des numéros de séquence.

Lorsqu'un segment chevauche d'autres segments déjà reçus, on reconstruit le segment pour qu'il contienne uniquement les nouvelles données et pour ajuster les champs d'en-tête afin qu'ils soient cohérents.

Noter que si aucun changement d'état n'est mentionné, la connexion TCP reste dans le même état.

3.10.1 Appel OPEN

ÉTAT CLOSED (c'est-à-dire, la TCB n'existe pas)

- Créer un nouveau bloc de commande de transmission (TCB) pour contenir les informations sur l'état de la connexion. Remplir les informations d'identifiant de prise locale, de prise distante, de champ Diffserv, de sécurité/compartiment et de temporisation de l'utilisateur. Noter que certaines parties de la prise distante peuvent être non spécifiées dans un OPEN passif et doivent être remplies par les paramètres du segment SYN entrant. Vérifier que la sécurité et les valeurs Diffserv demandées sont autorisées pour cet utilisateur, sinon, renvoyer une "Erreur : valeur Diffserv non admise" ou "Erreur : sécurité/compartiment non autorisé". Si il est passif, entrer dans l'état LISTEN et retour. Si il est actif et si la prise distante n'est pas spécifiée, retourner "Erreur : prise distante non spécifiée" ; si il est actif et si la prise distante est spécifiée, émettre un segment SYN. Choisir un numéro de séquence d'envoi initial (ISS). Envoyer un segment SYN de la forme <SEQ=ISS><CTL=SYN>. Régler SND. UNA à l'ISS, SND. NXT à ISS+1, entrer à l'état SYN-SENT, et retour.

- Si l'appelant n'a pas accès à la prise locale spécifiée, retourner "Erreur : connexion illégale pour ce processus". Si il n'y a pas de place pour créer une nouvelle connexion, retourner "Erreur : ressources insuffisantes".

ÉTAT LISTEN

- Si l'appel OPEN est actif et si la prise distante est spécifiée, changer la connexion de passive à active, choisir un ISS. Envoyer un segment SYN, régler SND.UNA à l'ISS, SND.NXT à ISS+1. Entrer dans l'état SYN-SENT. Les données associées à SEND peuvent être envoyées avec un segment SYN ou mises en file d'attente pour la transmission après l'entrée dans l'état ESTABLISHED. Le bit Urgent, si il est demandé dans la commande, doit être envoyé avec les segments de données envoyés par suite de cette commande. S'il n'y a pas de place pour mettre la demande en file d'attente, répondre "Erreur : ressources insuffisantes". Si la prise distante n'a pas été spécifiée, renvoyer "Erreur : prise distante non spécifiée".

ÉTAT SYN-SENT
ÉTAT SYN-RECEIVED
ÉTAT ESTABLISHED

ÉTAT FIN-WAIT-1
ÉTAT FIN-WAIT-2
ÉTAT CLOSE-WAIT
ÉTAT CLOSING
ÉTAT LAST-ACK
ÉTAT TIME-WAIT

- Retourner "Erreur : la connexion existe déjà".

3.10.2 Appel SEND

ÉTAT FERMÉ (c'est-à-dire que la TCB n'existe pas)

Si l'utilisateur n'a pas accès à une telle connexion, retourner "Erreur : connexion illégale pour ce processus".
Sinon, renvoyer "Erreur : la connexion n'existe pas".

ÉTAT LISTEN
Si la prise distante est spécifiée, changer la connexion de passive à active, choisir un ISS. Envoyer un segment SYN, régler SND.UNA à l'ISS, SND.NXT à ISS+1. Entrer dans l'état SYN-SENT. Les données associées à SEND peuvent être envoyées avec un segment SYN ou mises en file d'attente pour transmission après l'entrée dans l'état ESTABLISHED. Le bit Urgent si il est demandé dans la commande doit être envoyé avec les segments de données envoyés par suite de cette commande. S'il n'y a pas de place pour mettre la demande dans la file d'attente, répondre "Erreur : ressources insuffisantes". Si la prise distante n'a pas été spécifiée, alors renvoyer "Erreur : prise distante non spécifiée".

ÉTAT SYN-SENT
ÉTAT SYN-RECEIVED
Mettre les données en file d'attente pour transmission après être entré dans l'état ESTABLISHED. S'il n'y a pas d'espace pour la mise en file d'attente, répondre "Erreur : ressources insuffisantes".

ÉTAT ESTABLISHED
ÉTAT CLOSE-WAIT
Segmenter la mémoire tampon et l'envoyer avec un accusé de réception porté (valeur de l'accusé de réception = RCV.NXT). Si il y a un espace insuffisant pour mémoriser cette mémoire tampon, retourner simplement "Erreur : ressources insuffisantes".
Si le fanion URGENT est établi, alors SND.UP <- SND.NXT et établir le pointeur Urgent dans les segments sortants.

ÉTAT FIN-WAIT-1
ÉTAT FIN-WAIT-2
ÉTAT CLOSING
ÉTAT LAST ACK
ÉTAT TIME-WAIT

- Retourner "Erreur : fermeture de la connexion" et ne pas traiter la demande.

3.10.3 Appel RECEIVE

ÉTAT CLOSED (c'est-à-dire, la TCB n'existe pas)
Si l'utilisateur n'a pas accès à une telle connexion, retournez "Erreur : connexion illégale pour ce processus".
Sinon, renvoyez "Erreur : la connexion n'existe pas".

ÉTAT D'ÉCOUTE
ÉTAT SYN-SENT
ÉTAT SYN-REÇU

- Mettre en file d'attente pour le traitement après l'entrée dans l'état ESTABLISHED. S'il y a n'a pas de place pour mettre cette demande en file d'attente, répondre "Erreur : ressources insuffisantes".

ÉTAT ESTABLISHED
ÉTAT FIN-WAIT-1
ÉTAT FIN-WAIT-2

- Si il n'y a pas suffisamment de segments entrants dans la file d'attente pour satisfaire la demande, mettre la demande en file d'attente. S'il n'y a pas d'espace dans la file d'attente pour mémoriser le RECEIVE, répondre "Erreur : ressources

insuffisantes".
- Réassembler les segments entrants en file d'attente dans la mémoire tampon de réception et les retourner à l'utilisateur. Marquer "push vu" (PUSH) si c'est le cas.
- Si RCV.UP est en avance sur les données actuellement transmises à l'utilisateur, notifier l'utilisateur de la présence de données urgentes.
- Lorsque le point d'extrémité TCP prend la responsabilité de la livraison des données à l'utilisateur, ce fait doit être communiqué à l'envoyeur via un accusé de réception. La formation d'un tel accusé de réception est décrite ci-dessous dans la discussion sur le traitement d'un segment entrant.

ÉTAT CLOSE-WAIT
- Comme le côté distant a déjà envoyé FIN, les RECEIVE doivent être satisfaits par les données déjà disponibles, mais pas encore livrées à l'utilisateur. Si aucun texte n'est en attente de livraison, le RECEIVE va avoir une réponse "Erreur : fermeture de connexion". Autrement, toutes les données restantes peuvent être utilisées pour satisfaire le RECEIVE.

ÉTAT CLOSING
ÉTAT LAST-ACK
ÉTAT TIME-WAIT
Retourner "Erreur : fermeture de la connexion".

3.10.4 Appel CLOSE

ÉTAT CLOSE (c'est-à-dire, la TCB n'existe pas)
Si l'utilisateur n'a pas accès à une telle connexion, retourner "Erreur : connexion illégale pour ce processus".
Autrement, retourner "Erreur : la connexion n'existe pas".

ÉTAT LISTEN
Tous les RECEIVE en attente sont renvoyés avec "Erreur : fermeture". Supprimer le TCB, entrer dans l'état CLOSED, puis retour.

ÉTAT SYN-SENT
Supprimez le TCB et renvoyez la réponse "Erreur : fermeture" à tout SEND ou RECEIVE en file d'attente.

ÉTAT SYN-RECEIVED
Si aucun SEND n'a été émis et qu'il n'y a pas de données en attente à envoyer, former un segment FIN et l'envoyer, et entrer dans l'état FIN-WAIT-1 ; sinon, mettre en file d'attente pour traitement après être passé à l'état ESTABLISHED.

ÉTAT ESTABLISHED
Mettre cela en file d'attente jusqu'à ce que tous les SEND précédents aient été segmentés, puis former un segment FIN et l'envoyer. Dans tous les cas, entrer dans l'état FIN-WAIT-1.

ÉTAT FIN-WAIT-1
ÉTAT FIN-WAIT-2
À proprement parler, il s'agit d'une erreur et devrait recevoir une réponse "Erreur : fermeture de la connexion". Une réponse "ok" serait aussi acceptable, pour autant qu'un second FIN n'est pas émis (le premier FIN peut cependant être retransmis).

ÉTAT CLOSE-WAIT
Mettre cette demande en file d'attente jusqu'à ce que tous les SEND précédents aient été segmentés ; puis envoyer un segment FIN, entrer l'état LAST-ACK.

ÉTAT CLOSING
ÉTAT LAST-ACK
ÉTAT TIME-WAIT
Répondre par "Erreur : fermeture de la connexion".

3.10.5 Appel ABORT

ÉTAT CLOSE (c'est-à-dire, la TCB n'existe pas)
Si l'utilisateur ne devrait pas avoir accès à une telle connexion, retourner "Erreur : connexion illégale pour ce processus".

Sinon, retourner "Erreur : la connexion n’existe pas".

ÉTAT LISTEN
Tous les RECEIVE en attente doivent être renvoyés avec "Erreur : réinitialisation de la connexion". Supprimer le TCB, entrer à l’état CLOSED, et retour.

ÉTAT SYN-SENT
Tous les SEND et RECEIVE en file d’attente devraient recevoir une notification "Connexion réinitialisée". Supprimer le TCB, entrer à l’état CLOSED, et retour.

ÉTAT SYN-RECEIVED
ÉTAT ESTABLISHED
ÉTAT FIN-WAIT-1
ÉTAT FIN-WAIT-2
ÉTAT CLOSE-WAIT
Envoyer un segment de réinitialisation : <SEQ=SND.NXT><CTL=RST>
Tous les SEND et RECEIVE en file d’attente devraient recevoir une notification"Réinitialisation de connexion" ; tous les segments mis en file d’attente pour transmission (à l’exception du RST formé ci-dessus) ou la retransmission doit être purgée. Supprimer le TCB, entrer à l’état CLOSED, puis retour.

ÉTAT CLOSING
ÉTAT LAST-ACK
ÉTAT TIME-WAIT
Répondre "ok" et supprimer le TCB, entrer dans l’état CLOSED, et retour.

3.10.6 Appel STATUS

ÉTAT FERMÉ (c’est-à-dire, le TCB n’existe pas)
Si l’utilisateur ne devrait pas avoir accès à une telle connexion, retourner "Erreur : connexion illégale pour ce processus".
Sinon, retourner "Erreur : la connexion n’existe pas".

ÉTAT LISTEN
Retourner "État = LISTEN" et le pointeur de TCB.

ÉTAT SYN-SENT
Retourner "État = SYN-SENT" et le pointeur de TCB.

ÉTAT SYN-RECEIVED
Retourner "État = SYN-RECEIVED" et le pointeur de TCB.

ÉTAT ESTABLISHED
Retourner "État = ESTABLISHED" et le pointeur de TCB.

ÉTAT FIN-WAIT-1
Retourner "État = FIN-WAIT-1" et le pointeur de TCB.

ÉTAT FIN-WAIT-2
Retourner "État = FIN-WAIT-2" et le pointeur de TCB.

ÉTAT CLOSE-WAIT
Retourner "État = CLOSE-WAIT" et le pointeur de TCB.

ÉTAT CLOSING
Retourner "État = CLOSING" et le pointeur de TCB.

ÉTAT LAST-ACK
Retourner "État = LAST-ACK" et le pointeur de TCB.

ÉTAT TIME-WAIT
Retourner "État = TIME-WAIT" et le pointeur de TCB.

3.10.7 SEGMENT ARRIVE

3.10.7.1 État CLOSE

Si l'état est CLOSE (c'est-à-dire, le TCB n'existe pas) alors :
Toutes les données du segment entrant sont éliminées. Un segment contenant un RST est éliminé. Un segment entrant qui ne contient pas de RST entraîne l'envoi d'un RST en réponse. Les valeurs de champ d'accusé de réception et de numéro de séquence sont choisies pour rendre la réinitialisation de séquence acceptable pour le point d'extrémité TCP qui a envoyé le segment en infraction.
Si le bit ACK est à zéro, le numéro de séquence zéro est utilisé,
<SEQ=0><ACK=SEG. SEQ+SEG. LEN><CTL=RST,ACK>
Si le bit ACK est établi, <SEQ=SEG. ACK><CTL=RST>
Retour.

3.10.7.2 État LISTEN

Si l'état est LISTEN, alors :
Tout d'abord, chercher si il y a un RST :
- Un segment RST entrant pourrait ne pas être valide car il n'aurait pas pu être envoyé en réponse à quoi que ce soit envoyé par cette incarnation de la connexion. Un RST entrant devrait être ignoré. Retour.

Ensuite, vérifier la présence d'un accusé de réception :
- Tout accusé de réception est mauvais si il arrive sur une connexion encore à l'état LISTEN. Un segment de réinitialisation acceptable devrait être formé pour tout segment portant un ACK. Le RST devrait être formaté comme suit :
    - <SEQ=SEG. ACK><CTL=RST>

  Retour.

Troisièmement, vérifier la présence d'un SYN :
- Si le bit SYN est établi, vérifier la sécurité. Si la sécurité/compartiment sur le segment entrant ne correspond pas exactement à la sécurité/compartiment dans le TCB, alors envoyer une réinitialisation et retour.
    - <SEQ=0><ACK=SEG.SEQ+SEG.LEN><CTL=RST,ACK>
- Régler RCV.NXT à SEG.SEQ+1, IRS est réglé à SEG.SEQ, et tout autre contrôle ou texte devrait être mis en file d'attente pour être traité ultérieurement. ISS devrait être choisi et un segment SYN envoyé sous la forme :
    - <SEQ=ISS><ACK=RCV. NXT><CTL=SYN,ACK>
- SND.NXT est réglé à ISS+1 et SND.UNA à ISS. L'état de la connexion devrait être changé en SYN-RECEIVED. Noter que tout autre contrôle ou données entrants (combinés à SYN) seront traités dans l'état SYN-RECEIVED, mais le traitement de SYN et ACK ne devrait pas être répété. Si le LISTEN n'a pas été pleinement spécifié (c'est-à-dire, si la prise distante n'était pas été pleinement spécifiée) alors les champs non spécifiés devraient être complétés.

Quatrièmement, autres données ou contrôle :
- Cela ne devrait pas arriver. Éliminer le segment et retourner. Tout autre segment de contrôle ou de données (ne contenant pas de SYN) doit avoir un ACK et aurait donc été éliminé par le traitement d'accusé de réception à la deuxième étape, à moins qu'il n'ait d'abord été éliminé par la vérification de RST dans la première étape.

3.10.7.3 ÉTAT SYN-SENT

Si l'état est SYN-SENT, alors
Tout d'abord, vérifiez le bit ACK :
- Si le bit ACK est établi,

Si SEG.ACK ≤ ISS ou SEG.ACK > SND.NXT, envoyer une réinitialisation (sauf si le bit RST est établi ; si c'est le cas, éliminer le segment et retour)
<SEQ=SEG.ACK><CTL=RST>
et supprimez le segment. Retour.
Si SND.UNA < SEG.ACK ≤ SND.NXT, alors le ACK est acceptable. Une partie du code TCP déployé a utilisé la vérification SEG.ACK == SND.NXT (en utilisant "==" plutôt que "≤") mais cela n'est pas approprié quand la pile capable d'envoyer des données sur le SYN car l'homologue TCP peut ne pas accepter et accuser réception de toutes les données sur le SYN.

Deuxièmement, vérifiez le bit RST :
Si le bit RST est établi,
- Une attaque potentielle de réinitialisation aveugle est décrite dans la [RFC5961]. L'atténuation décrite dans ce document a une applicabilité spécifique qui y est expliquée et ne remplace pas la protection cryptographique (par

exemple, par IPsec ou TCP-AO). Une mise en œuvre TCP qui prend en charge l'atténuation décrite dans la RFC 5961 DEVRAIT vérifier d'abord que le numéro de séquence correspond exactement au RCV.NXT avant d'exécuter l'action du paragraphe suivant.

- Si le ACK était acceptable, signaler alors à l'utilisateur "Erreur : réinitialisation de connexion", éliminer le segment, entrer dans l'état CLOSED, supprimer le TCB, et retour. Sinon (pas de ACK) éliminer le segment et retour.

Troisièmement, vérifiez la sécurité :

- Si la sécurité/compartiment dans le segment ne correspond pas exactement à la sécurité/compartiment dans le TCB, envoyez une réinitialisation :
  - Si il y a un ACK,
    - <SEQ=SEG. ACK><CTL=RST>

  Autrement
    - <SEQ=0><ACK=SEG. SEQ+SEG. LEN><CTL=RST,ACK>

  Si une réinitialisation a été envoyée, supprimez le segment et retour.

Quatrièmement, vérifiez le bit SYN :

Cette étape ne devrait être atteinte que si l'accusé de réception est correct ou si il y a il n'y avait pas d'accusé de réception et si le segment ne contenait pas de RST.

- Si le bit SYN est établi et si la sécurité/compartiment est acceptable, alors RCV.NXT est réglé à SEG.SEQ+1, IRS est réglé à SEG.SEQ. SND.UNA devrait être avancé à être égal à SEG.ACK (si il y a un ACK) et tous les segments de la file d'attente de retransmission qui sont ainsi acquittés devraient être supprimés.
- Si SND.UNA > ISS (notre SYN a été acquitté) changer l'état de la connexion à ESTABLISHED, former un segment ACK
    - <SEQ=SND.NXT><ACK=RCV.NXT><CTL=ACK>

  et l'envoyer. Les données ou contrôles qui ont été mis en file d'attente pour transmission PEUVENT être inclus. Certaines mises en œuvre TCP suppriment l'envoi de ce segment quand le segment reçu contient des données qui vont de toutes façons générer un accusé de réception dans les étapes de traitement ultérieures, évitant l'envoi de cet accusé de réception supplémentaire du SYN. Si il y a d'autres contrôles ou texte dans le segment, alors on poursuit le traitement à la sixième étape du paragraphe 3.10.7.4 où le bit URG est vérifié ; sinon, retour.
- Sinon, entrez à l'état SYN-RECEIVED, former un segment SYN,ACK
  <SEQ=ISS><ACK=RCV.NXT><CTL=SYN,ACK>
  et l'envoyer. Définir les variables :
  SND.WND <- SEG.WND
  SND.WL1 <- SEG.SEQ
  SND.WL2 <- SEG.ACK

  Si il y a d'autres commandes ou du texte dans le segment, les mettre en file d'attente pour traitement après que l'état ESTABLISHED a été atteint, retour.

  Noter qu'il est légal d'envoyer et recevoir des données d'application sur des segments SYN (c'est le "texte dans le segment" mentionné ci-dessus). Il y a eu beaucoup de désinformation et d'incompréhension sur ce sujet dans le passé. Certains pare-feu et dispositifs de sécurité considèrent cela comme suspect. Cependant, cette capacité a été utilisée dans T/TCP [RFC1644] et est utilisé dans l'ouverture rapide de TCP (TFO, *TCP Fast Open*) [RFC7413], il est donc important que les mises en œuvre et les périphériques réseau le permettent.

Cinquièmement, si ni le bit SYN ni le bit RST n'est établi, alors éliminer le segment et retour.

3.10.7.4 Autres états

Autrement

- Tout d'abord, vérifiez le numéro de séquence :
  - ÉTAT SYN-RECEIVED
  - ÉTAT ESTABLISHED
  - ÉTAT FIN-WAIT-1
  - ÉTAT FIN-WAIT-2
  - ÉTAT CLOSE-WAIT
  - ÉTAT CLOSING
  - ÉTAT LAST-ACK
  - ÉTAT TIME-WAIT

- Les segments sont traités dans l'ordre. Les essais initiaux à l'arrivée sont utilisés pour éliminer les anciens dupliqués, mais un traitement ultérieur est fait dans l'ordre de SEG.SEQ. Si le contenu d'un segment chevauche la limite entre l'ancien et le nouveau, seules les nouvelles parties sont traitées.
- En général, le traitement des segments reçus DOIT être mis en œuvre pour agréger les segments ACK chaque fois que

possible (DOIT-58). Par exemple, si le point d'extrémité TCP traite une série de segments en file d'attente, il DOIT les traiter tous avant d'envoyer des segments d'accusé de réception (DOIT-59).

- Il y a quatre cas pour l'essai d'acceptabilité d'un segment :

Tableau 6 : Essais d'acceptabilité des segments

Longueur du segment	Fenêtre de réception	Essai
0	0	SEG.SEQ = RCV.NXT
0	>0	RCV.NXT =< SEG.SEQ < RCV.NXT+RCV.WND
>0	0	Inacceptable
>0	>0	RCV.NXT =< SEG.SEQ < RCV.NXT+RCV.WND ou RCV.NXT =< SEG.SEQ+SEG.LEN-1 < RCV.NXT+RCV.WND

Pour mettre en œuvre la validation du numéro de séquence décrite ici, prendre note de l'annexe A.2.

- Si le RCV.WND est zéro, aucun segment ne sera acceptable, mais une tolérance spéciale devrait être accordée pour accepter les ACK, les URG et les RST valides.

- Si un segment entrant n'est pas acceptable, un accusé de réception devrait être envoyé en réponse (à moins que le bit RST soit établi, si c'est le cas, éliminer le segment et retour) :
  <SEQ=SND.NXT><ACK=RCV.NXT><CTL=ACK>

- Après l'envoi de l'accusé de réception, supprimer le segment inacceptable et retour.

Noter que pour l'état TIME-WAIT, il existe un algorithme amélioré décrit dans la [RFC6191] pour le traitement des segments SYN entrants qui utilisent des horodatages plutôt que de s'appuyer sur la vérification du numéro de séquence décrite ici. Quand l'algorithme amélioré est mis en œuvre, la logique ci-dessus n'est pas applicable pour les segments SYN entrants avec les options Horodatage, reçus sur une connexion dans l'état TIME-WAIT.

Dans ce qui suit, on suppose que le segment est le segment idéal qui commence à RCV.NXT et ne dépasse pas la fenêtre. On pourrait adapter les segments réels à cette hypothèse en coupant toutes les parties qui se trouvent à l'extérieur de la fenêtre (y compris SYN et FIN) et ne traiter ensuite que si le segment commence à RCV.NXT. Les segments avec un début de numéros de séquence plus élevé DEVRAIENT être conservés pour un traitement ultérieur (DVRT-31).

Deuxièmement, vérifier le bit RST :
La Section 3 de la [RFC5961] décrit une attaque potentielle de réinitialisation aveugle et une approche d'atténuation facultative. Cela ne fournit pas de protection cryptographique (par exemple, comme dans IPsec ou TCP-AO) mais peut être applicable dans les situations décrites dans la RFC 5961. Pour les piles de protocole qui mettent en œuvre la protection décrite dans la RFC 5961, les trois vérifications ci-dessous s'appliquent ; autrement, le traitement pour ces états est indiqué plus loin.

1) Si le bit RST est établi et si le numéro de séquence se trouve en dehors de la fenêtre de réception actuelle, éliminer le segment en silence.
2) Si le bit RST est établi et si le numéro de séquence correspond exactement au prochain numéro de séquence attendu (RCV.NXT) alors les points d'extrémité TCP DOIVENT réinitialiser la connexion de la manière prescrite ci-dessous en fonction de l'état de la connexion.
3) Si le bit RST est établi et si le numéro de séquence ne correspond pas exactement à la prochaine valeur de numéro de séquence attendue, mais se trouve dans la fenêtre de réception actuelle, les points d'extrémité TCP DOIVENT envoyer un accusé de réception (ACK de défi) :
   <SEQ=SND.NXT><ACK=RCV.NXT><CTL=ACK>
   Après l'envoi de l'ACK de défi, les points de terminaison TCP DOIVENT supprimer le segment inacceptable et arrêter de traiter le paquet entrant. Noter que la RFC 5961 et l'errata ID 4772 [99] contiennent des considérations supplémentaires pour la limitation d'ACK dans une mise en œuvre.

ÉTAT SYN-RECEIVED
Si le bit RST est établi,

- Si cette connexion a été initiée par un système OPEN passif (c'est-à-dire, provenait de l'état LISTEN) alors renvoyer cette connexion à LISTEN et retour. Il n'est pas nécessaire d'en informer l'utilisateur. Si cette connexion a été initiée avec un OPEN actif (c'est-à-dire, venait de l'état SYN-SENT) alors la connexion a été refusée ; signaler à l'utilisateur "Connexion refusée". Dans les deux cas, la file d'attente de retransmission doit être purgée. Et dans le cas de OPEN

actif, entrer dans l'état CLOSED, supprimer le TCB, et retour.

ÉTAT ESTABLISHED
ÉTAT FIN-WAIT-1
ÉTAT FIN-WAIT-2
ÉTAT CLOSE-WAIT
Si le bit RST est défini, tous les RECEIVE et SEND en attente devraient recevoir des réponses "réinitialisation". Toutes les files d'attente de segments doivent être purgées. Les utilisateurs devraient également recevoir un signal général non sollicité "réinitialisation de la connexion". Entrer dans l'état CLOSED, supprimer le TCB, et retour.

ÉTAT CLOSING
ÉTAT LAST-ACK
ÉTAT TIME-WAIT
Si le bit RST est établi, entrez dans l'état CLOSED, supprimez le TCB, et retour.

Troisièmement, vérifier la sécurité :

ÉTAT SYN-REÇU
Si la sécurité/compartiment du segment ne correspond pas exactement à la sécurité/compartiment du TCB, alors envoyer une réinitialisation et retour.

ÉTAT ESTABLISHED
ÉTAT FIN-WAIT-1
ÉTAT FIN-WAIT-2
ÉTAT CLOSE-WAIT
ÉTAT CLOSING
ÉTAT LAST-ACK
ÉTAT TIME-WAIT
Si la sécurité/compartiment dans le segment ne correspond pas exactement à la sécurité/compartiment du TCB, alors envoyer une réinitialisation ; tous les RECEIVE et SEND en attente devraient recevoir des réponses "reset". Toutes les files d'attente de segments devraient être purgées. Les utilisateurs devraient aussi recevoir un signal général non sollicité "Réinitialisation de la connexion". Entrer dans l'état CLOSED, supprimer le TCB, et retour.

Noter que cette vérification est placée après la vérification de séquence pour éviter qu'un segment d'une ancienne connexion entre ces numéros de accès avec une sécurité différente provoque un abandon de la connexion actuelle.

Quatrièmement, vérifier le bit SYN :

ÉTAT SYN-RECEIVED
Si la connexion a été initiée avec un OPEN passif, alors ramener cette connexion à l'état LISTEN et retour. Autrement, traiter selon les instructions pour les états synchronisés.

ÉTAT ESTABLISHED
ÉTAT FIN-WAIT-1
ÉTAT FIN-WAIT-2
ÉTAT CLOSE-WAIT
ÉTAT CLOSING
ÉTAT LAST-ACK
ÉTAT TIME-WAIT
- Si le bit SYN est défini dans ces états synchronisés, il peut s'agir soit d'une nouvelle tentative de connexion légitime (par exemple, dans le cas de TIME-WAIT) soit d'une erreur où la connexion devrait être réinitialisée, soit du résultat d'une tentative d'attaque, comme décrit dans la [RFC5961]. Pour l'état TIME-WAIT, de nouvelles connexions peuvent être acceptées si l'option Horodatage est utilisée et répond aux attentes (selon la [RFC6191]). Pour tous les autres cas, la RFC 5961 fournit une atténuation avec applicabilité à certaines situations, bien qu'il existe également des solutions de remplacement qui offrent une protection cryptographique (voir la Section 7). La RFC 5961 recommande que dans ces états synchronisés, si le bit SYN est établi, quel que soit le numéro de séquence, les points d'extrémité TCP DOIVENT envoyer un "ACK de défi" à l'homologue distant :
  <SEQ=SND. NXT><ACK=RCV.NXT><CTL=ACK>

- Après l'envoi de l'accusé de réception, les mises en œuvre TCP DOIVENT éliminer le segment inacceptable et arrêter

le traitement. Noter que la RFC 5961 et l'errata ID 4772 [99] contiennent des notes supplémentaires de limitation des ACK pour une mise en œuvre.

- Pour les mises en œuvre qui ne suivent pas la RFC 5961, le comportement d'origine décrit dans la RFC 793 suit dans ce paragraphe. Si le SYN est dans la fenêtre, c'est une erreur : envoyer un "reset", tous les RECEIVE et SEND en attente devraient recevoir des réponses "reset", toutes les files d'attente de segments devraient être purgées, l'utilisateur devrait aussi recevoir un signal général non sollicité de "Réinitialisation de la connexion", entrer dans l'état CLOSED, supprimer le TCB, et retour.

- Si le SYN n'est pas dans la fenêtre, cette étape ne devrait pas atteinte et un ACK aurait été vu à la première étape (vérification du numéro de séquence).

Cinquièmement, vérifier le champ ACK :
- si le bit ACK est à zéro, éliminer le segment, et retour.
- si le bit ACK est établi,
  - La Section 5 de la [RFC5961] décrit une attaque potentielle par injection de données à l'aveugle et l'atténuation que les mises en œuvre PEUVENT choisir d'inclure (PEUT-12). Les piles TCP qui mettent en œuvre la RFC 5961 DOIVENT ajouter une vérification d'entrée que la valeur de ACK n'est acceptable que si elle est dans la gamme de ((SND.UNA - MAX.SND.WND) ≤ SEG.ACK ≤ SND.NXT). Tous les segments entrants dont la valeur d'accusé de réception ne satisfait pas à la condition ci-dessus DOIVENT être éliminés et un ACK renvoyé. La nouvelle variable d'état MAX.SND.WND est définie comme la plus grande fenêtre que l'envoyeur local ait jamais reçue de son homologue (sous réserve de la mise à l'échelle de la fenêtre) ou peut être codée à une valeur de fenêtre maximale autorisée. Lorsque la valeur ACK est acceptable, le traitement par état ci-dessous s'applique :

    ÉTAT SYN-RECEIVED
    + Si SND.UNA < SEG.ACK ≤ SND.NXT, alors entrer dans l'état ESTABLISHED et poursuivre le traitement avec les variables ci-dessous définies sur :
      SND.WND <- SEG.WND
      SND.WL1 <- SEG.SUIV
      SND.WL2 <- SEG.ACK
    + Si le segment d'accusé de réception n'est pas acceptable, former un segment Réinitialiser
      <SEQ=SEG.ACK><CTL=RST>
    + et l'envoyer.

    ÉTAT ESTABLISHED
    + Si SND.UNA < SEG.ACK ≤ SND.NXT, puis réglez SND.UNA <- SEG.ACK. Tous les segments de la file d'attente de retransmission qui sont ainsi entièrement acquittés sont supprimés. Les utilisateurs devraient recevoir des accusés de réception positifs pour les mémoires tampons qui ont été SEND et entièrement acquittées (c'est-à-dire que la mémoire tampon SEND doit être retournée avec la réponse "ok"). Si le ACK est un dupliqué (SEG. ACK ≤ SND.UNA) il peut être ignoré. Si le ACK accuse réception de quelque chose qui n'a pas encore été envoyé (SEG.ACK > SND.NXT) alors envoyer un ACK, éliminer le segment, et retour.

    + Si SND.UNA ≤ SEG.ACK ≤ SND.NXT, la fenêtre d'envoi devrait être actualisée. Si (SND.WL1 < SEG.SEQ ou (SND.WL1 = SEG.SEQ et SND.WL2 ≤ SEG.ACK)) régler SND.WND <- SEG.WND, régler SND.WL1 <- SEG.SEQ, et régler SND.WL2 <- SEG.ACK.

    Notez que SND.WND est un décalage de SND.UNA, que SND.WL1 enregistre le numéro de séquence du dernier segment utilisé pour la mise à jour de SND.WND, et que SND.WL2 enregistre le numéro d'accusé de réception du dernier segment utilisé pour mettre à jour SND.WND. La vérification empêche ici l'utilisation d'anciens segments pour mettre à jour la fenêtre.

    ÉTAT FIN-WAIT-1
    + En plus du traitement pour l'état ESTABLISHED, si le segment FIN est maintenant acquitté, alors entrer à l'état FIN-WAIT-2 et continuer traitement dans cet état.

    ÉTAT FIN-WAIT-2
    En plus du traitement pour l'état ESTABLISHED, si la file d'attente de retransmission est vide, le CLOSE de l'utilisateur peut être acquitté ("ok") mais on ne supprime pas le TCB.

    ÉTAT CLOSE-WAIT

Effectuer le même traitement que pour l'état ESTABLISHED.

ÉTAT CLOSING
En plus du traitement pour l'état ESTABLISHED, si le ACK accuse réception du FIN, alors entrer dans l'état TIME-WAIT ; sinon, ignorer le segment.

ÉTAT LAST-ACK
La seule chose qui peut arriver dans cet état est un accusé de réception du FIN. Si le FIN est maintenant acquitté, supprimer le TCB, entrer à l'état CLOSED, et retour.

ÉTAT TIME-WAIT
La seule chose qui peut arriver dans cet état est une retransmission du FIN distant. En accuser réception et redémarrer le temporisateur de 2 MSL.

Sixièmement, vérifier le bit URG :
ÉTAT ESTABLISHED
ÉTAT FIN-WAIT-1
ÉTAT FIN-WAIT-2
Si le bit URG est établi, RCV.UP <- max(RCV.UP,SEG.UP) et signaler à l'utilisateur que le côté distant dispose de données urgentes si le pointeur Urgent (RCV.UP) est en avance sur les données consommées. Si l'utilisateur a déjà eu la signalisation (ou se trouve toujours dans le mode "urgent") pour cette séquence continue de données urgentes, ne pas signaler à nouveau à l'utilisateur.

ÉTAT CLOSE-WAIT
ÉTAT CLOSING
ÉTAT LAST-ACK
ÉTAT TIME-WAIT
Cela ne devrait pas se produire puisqu'un FIN a été reçu du côté distant. Ignorer le URG.

Septièmement, traiter le texte du segment :
ÉTAT ESTABLISHED
ÉTAT FIN-WAIT-1
ÉTAT FIN-WAIT-2
- Une fois à l'état ESTABLISHED, il est possible de livrer le segment de données aux mémoires tampon de réception de l'utilisateur. Les données des segments peuvent être déplacées dans les mémoires tampon jusqu'à ce qu'elles soient pleines ou que le segment soit vide. Si le segment se vide et porte un fanion PUSH, alors l'utilisateur est informé, quand la mémoire tampon est retournée, qu'un PUSH été reçu.
- Lorsque le point d'extrémité TCP prend la responsabilité de livrer les données à l'utilisateur, il doit également accuser réception des données.
- Une fois que le point d'extrémité TCP a pris la responsabilité des données, il avance RCV.NXT sur les données acceptées, et ajuste RCV.WND comme approprié à la disponibilité actuelle de la mémoire tampon. Le total de RCV.NXT et RCV.WND ne devrait pas être réduit.
- Une mise en œuvre TCP PEUT envoyer un segment ACK accusant réception de RCV.NXT lorsqu'un sgment valide arrive qui est dans la fenêtre mais pas comme bord gauche de la fenêtre (PEUT-13).
- Noter les suggestions de gestion des fenêtres du paragraphe 3.8.
- Envoyer un accusé de réception de forme :
  <SEQ=SND.NXT><ACK=RCV.NXT><CTL=ACK>
- Cet accusé de réception devrait être porté sur un segment transmis si possible sans subir de retard indu.

ÉTAT CLOSE-WAIT
ÉTAT CLOSING
ÉTAT LAST-ACK
ÉTAT TIME-WAIT
Cela ne devrait pas se produire puisqu'un FIN a été reçu du côté distant. Ignorer le texte du segment.

Huitièmement, vérifier le bit FIN :
- Ne pas traiter le FIN si l'état est CLOSED, LISTEN ou SYN-SENT car la SEG.SEQ ne peut pas être validée ; éliminer le segment, et retour.
- Si le bit FIN est établi, signaler à l'utilisateur "Fermeture de la connexion" et retourner tous RECEIVE en attente avec le même message, avancer RCV.NXT sur le FIN, et envoyer un accusé de réception pour le FIN. Noter que FIN

implique PUSH pour tout segment de texte qui n’a pas encore été livré à l' utilisateur.

ÉTAT SYN-REÇU
ÉTAT ESTABLISHED
Entrer dans l’état CLOSE-WAIT.

ÉTAT FIN-WAIT-1
Si le FIN a été acquitté (peut-être dans ce segment) alors entrer dans l’état TIME-WAIT, démarrer le temporisateur d’attente, désactiver les autres temporisateurs ; sinon, entrer à l’état CLOSING.

ÉTAT FIN-WAIT-2
Entrer à l’état TIME-WAIT. démarrer le temporisateur d’attente, désactiver les autres temporisateurs.

ÉTAT CLOSE-WAIT
Rester à l’état CLOSE-WAIT.

ÉTAT CLOSING
Rester à l’état CLOSING.

ÉTAT LAST-ACK
Rester à l’état LAST-ACK.

ÉTAT TIME-WAIT
Rester à l’état TIME-WAIT. redémarrer le temporisateur d’attente de 2 MSL, et retour.

3.10.8 Temporisateurs

FIN DE TEMPORISATION DE L’UTILISATEUR
Pour tout état si la temporisation de l’utilisateur arrive à expiration, purger toutes les files d’attente, signaler à l’utilisateur "Erreur : connexion interrompue en raison de la fin de la temporisation d’utilisateur" en général et pour tout appel en cours, supprimer le TCB, entrer à l’état CLOSED, et retour.

FIN DE TEMPORISATION DE LA RETRANSMISSION
Pour tout état si le temporisateur de retransmission arrive à expiration sur un segment dans la file d’attente de retransmission, renvoyer le segment à l’avant de la file d’attente de retransmission, réinitialiser le temporisateur de retransmission, et retour.

FIN DE TEMPORISATION DE DÉLAI D’ATTENTE
Si le délai d’attente arrive à expiration sur une connexion, supprimer le TCB, entrer l’état CLOSED, et retour.

-

4. Glossaire

Accès : partie d’un identifiant de connexion utilisée pour le démultiplexage des connexions à un point d’extrémité.

ACK : Un bit de contrôle (accusé de réception) n’occupant pas d’espace de numéro de séquence, qui indique que le champ d’accusé de réception de ce segment spécifie le numéro de séquence suivant que l’envoyeur de ce segment s’attend à recevoir, accusant ainsi réception de tous les numéros de séquence précédents.

Accusé de réception du segment : le numéro de séquence dans le champ Accusé de réception du segment arrivant.

Adresse de destination : Adresse de la couche réseau du point d'extrémité destiné à recevoir un segment.

Adresse de source : adresse de couche réseau du point d'extrémité d’envoi.

Adresse Internet : adresse de couche réseau.

Connexion : Chemin de communication logique identifié par une paire de prises.

Datagramme : Message envoyé dans un réseau de communication informatique à commutation de paquets.

Datagramme Internet : unité de données échangée entre les hôtes Internet, avec l'en-tête Internet, cela permet au datagramme d'être acheminé de la source à la destination.

En-tête : informations de contrôle au début d'un message, d'un segment, d'un fragment, paquet ou bloc de données.

Fenêtre d'envoi : elle représente les numéros de séquence que le point d'extrémité TCP (receveur) est disposé à recevoir. C'est la valeur du champ de fenêtre spécifié en segments à partir du point de terminaison TCP distant (receveur de données). La gamme des nouveaux numéros de séquence qui peuvent être émis par une mise en œuvre TCP se trouve entre SND.NXT et SND.UNA + SND.WND - 1. (Les retransmissions de numéros de séquence entre SND.UNA et SND.NXT sont attendues, bien sûr.)

Fenêtre de réception : elle représente les numéros de séquence que le point d'extrémité TCP local (receveur) est disposé à recevoir. Ainsi, le point d'extrémité TCP local considère que les segments qui chevauchent la gamme de RCV.NXT à RCV.NXT + RCV.WND - 1 transportent des données ou une commande acceptables. Les segments contenant des numéros de séquence entièrement en dehors de cette gamme sont considérés comme des dupliqués ou des attaques par injection et sont éliminés.

FIN : Un bit de contrôle (fin) occupant un numéro de séquence, qui indique que l'envoyeur n'enverra plus de données ou de commandes occupant l'espace de numéros de séquence.

Fragment : partie d'une unité logique de données. En particulier, un fragment Internet est une partie d'un datagramme Internet.

Fragment Internet : partie des données d'un datagramme Internet avec un en-tête Internet.

Hôte : ordinateur. En particulier, une source ou une destination de messages du point de vue du réseau de communication.

Identifiant : champ de protocole Internet. Cette valeur d'identification attribuée par l'envoyeur aide à assembler les fragments d'un datagramme.

IP : protocole Internet. Voir la [RFC0791] et la [RFC8200].

IRS : numéro de séquence de réception initial. Premier numéro de séquence utilisé par l'envoyeur sur une connexion.

ISN : numéro de séquence initial. Premier numéro de séquence utilisé sur une connexion (ISS ou IRS). Sélectionné d'une manière unique dans une période donnée et imprévisible pour les attaquants.

ISS : numéro de séquence d'envoi initial. Premier numéro de séquence utilisé par l'envoyeur sur une connexion.

Longueur du segment : quantité d'espace de numéro de séquence occupée par un segment, y compris toutes commandes qui occupent l'espace de séquence.

Module : mise en œuvre, généralement d'un logiciel, d'un protocole ou d'une autre procédure.

MSL : durée de vie maximale de segment, durée pendant laquelle un segment TCP peut exister dans le système inter réseaux. Défini arbitrairement comme étant de 2 minutes.

Numéro de séquence suivant à recevoir : c'est le prochain numéro de séquence que le point d'extrémité TCP local s'attend à recevoir.

Octet : octet de huit bits.

Options : un champ Option peut contenir plusieurs options, et chaque option peut avoir une longueur de plusieurs octets.

Paquet : paquet de données avec un en-tête qui peut ou non être logiquement complet. Plus souvent un emballage physique qu'un empaquetage logique de données.

Pointeur Urgent : champ de contrôle qui n'a de sens que lorsque le bit URG est établi. Ce champ communique la valeur du pointeur Urgent qui indique l'octet de données associé à l'appel urgent de l'utilisateur envoyeur.

Prise (ou numéro de prise, ou adresse de prise, ou identifiant de prise) : adresse qui inclut spécifiquement un identifiant d'accès, qui est l'enchaînement d'une adresse Internet et d'un accès TCP.

Processus : programme en cours d'exécution. Une source ou une destination de données du point de vue du point d'extrémité TCP ou d'un autre protocole d'hôte à hôte.

Purger : pour supprimer tout le contenu (données ou segments) d'une mémorisation (mémoire tampon ou file d'attente).

PUSH : bit de contrôle n'occupant pas d'espace de numéro de séquence, indiquant que ce segment contient des données qui doivent être poussées jusqu'à l'utilisateur receveur.

RCV.NXT : Prochain numéro de séquence à recevoir.

RCV.UP : pointeur de données urgentes

RCV.WND : fenêtre de réception

RST : bit de contrôle (reset) n'occupant aucun espace de numéros de séquence, indiquant que le receveur devrait supprimer la connexion sans autre interaction. Le receveur peut déterminer, en fonction des numéros de séquence et des champs d'accusé de réception du segment entrant, si il doit honorer la commande RST ou l'ignorer. En aucun cas, la réception d'un segment contenant un RST ne donne lieu à un RST en réponse.

SEG.ACK : segment d'accusé de réception

SEG.LEN : longueur du segment

SEG.SEQ : séquence de segments

SEG.UP : champ de pointeur de segment urgent

SEG.WND : champ Fenêtre de segment

Segment : unité logique de données. En particulier, un segment TCP est l'unité de données transférées entre une paire de modules TCP.

Séquence gauche : numéro de séquence suivant à être acquitté par le point d'extrémité TCP de réception de données (ou le le plus bas numéro de séquence actuellement non acquitté) qui est parfois appelé bord gauche de la fenêtre d'envoi.

Séquence de segments : Numéro dans le champ de séquence du segment arrivant.

Séquence d'envoi : prochain numéro de séquence que le point d'extrémité TCP local (d'envoi) va utiliser sur la connexion. Il est initialement sélectionné à partir d'une courbe initiale de numéros de séquence (ISN) et est incrémenté pour chaque octet de données ou contrôle séquencé transmis.

SND.NXT : séquence d'envoi.

SND.UNA : bord gauche de séquence.

SND.UP : Pointeur de données urgentes à l'émission.

SND.WL1 : Numéro de séquence de segment lors de la dernière mise à jour de la fenêtre

SND.WL2 : Numéro d'accusé de réception du segment lors de la dernière mise à jour de la fenêtre

SND.WND : fenêtre d'envoi

SYN : un bit de contrôle dans le segment entrant, occupant un numéro de séquence, utilisé lors de l'initialisation d'une connexion pour indiquer où la numérotation séquentielle commencera.

TCB : bloc de contrôle de transmission, la structure de données qui enregistre l'état d'une connexion.

TCP *(Transmission Control Protocol)* : protocole d'hôte à hôte pour la communication fiable dans les environnements inter réseaux.

ToS : Type de service, champ IPv4 obsolète. Les mêmes bits d'en-tête sont actuellement utilisés pour le champ Services différenciés [RFC2474] contenant la valeur du codet de services différenciés (DSCP) et le codet ECN de 2 bits [RFC3168].

Type de service : voir "ToS".

URG : bit de contrôle (urgent) n'occupant pas d'espace de numéro de séquence, utilisé pour indiquer que l'utilisateur destinataire devrait être informé de faire un traitement urgent pour autant qu'il y a des données à consommer avec des numéros de séquence inférieurs à la valeur indiquée par le pointeur Urgent.

5. Modifications par rapport à la RFC 793

Ce document rend obsolètes les RFC 793 ainsi que les RFC 6093 et 6528, qui ont mis à jour la RFC0793. Dans tous les cas, seules les spécifications et exigences normatives du protocole ont été incorporées dans le présent document, et du texte d'information avec contexte et justification peut n'avoir pas été inclus. Le contenu informatif de ces documents est toujours précieux pour l'apprentissage et la compréhension de TCP, et ils constituent des références informatives valides, même si leur contenu normatif a été incorporé dans le présent document.

Le corps principal de ce document a été adapté de la section 3 de la RFC 793, intitulée "Spécification fonctionnelle", en tentant de garder le formatage et la mise en page aussi proches que possible.

La collection d'errata aux RFC applicables qui ont été signalés et qui ont été acceptés ou conservés pour mise à jour de la RFC 793 a été incorporée (identifiants d'errata : 573 [73], 574 [74], 700 [75], 701 [76], 1283 [77], 1561 [78], 1562 [79], 1564 [80], 1571 [81], 1572 [82], 2297 [83], 2298 [84], 2748 [85], 2749 [86], 2934 [87], 3213 [88], 3300 [89], 3301 [90], 6222 [91]). Certains errata n'étaient pas applicables en raison d'autres changements (ID d'errata : 572 [92], 575 [93], 1565 [94], 1569 [95], 2296 [96], 3305 [97], 3602 [98]).

Les modifications apportées à la spécification du pointeur urgent décrites dans les RFC 1011, 1122 et 6093 ont été incorporées. Voir la RFC 6093 pour une discussion détaillée sur les raisons pour lesquelles ces changements étaient nécessaires.

La discussion sur le RTO de la RFC 793 a été mise à jour pour faire référence à la RFC 6298. Le texte sur le RTO dans la RFC 1122 a initialement remplacé le texte de la RFC 793 ; cependant, la RFC 2988 aurait dû mettre à jour la RFC 1122 et a ensuite été rendue obsolète par la RFC 6298.

La [RFC1011] contient un certain nombre de commentaires sur la RFC 793, y compris quelques modifications nécessaires à la spécification de TCP. Ceux-ci sont développés dans la RFC 1122, qui contient une collection d'autres modifications et clarifications à la RFC 793. Les éléments normatifs ayant un impact sur le protocole ont été incorporés ici, bien que certains conseils de mise en œuvre historiquement utiles et la discussion informative de la RFC 1122 ne soient pas inclus ici. Le présent document, qui est maintenant la spécification TCP plutôt que la RFC 793, met à jour la RFC 1011, et les commentaires notés dans la RFC 1011 ont été incorporés.

La RFC 1122 contient plus que les exigences pour TCP, donc ce document ne peut pas complètement rendre obsolète la RFC 1122. Il n'est marqué que comme "mise à jour" de la RFC 1122 ; cependant, il doit être compris comme rendant effectivement obsolète tout le matériel sur TCP trouvé dans la RFC 1122.

L'algorithme de génération de numéros de séquence initiaux, plus sécurisé, de la RFC 6528 a été incorporé. Voir la RFC 6528 pour une discussion sur les attaques que cela atténue, ainsi que des conseils sur la sélection des algorithmes de PRF et la gestion des données de clé secrète.

Une note fondée sur la RFC 6429 a été ajoutée pour préciser explicitement que les problèmes de gestion des ressources système permettent de récupérer les ressources de connexion. La RFC 6429 est obsolète dans le sens où la clarification qu'elle décrit a été reflétée dans cette spécification TCP de base.

La description de la mise en œuvre du contrôle d'encombrement a été ajoutée sur la base de l'ensemble des documents qui sont des bonnes pratiques actuelles ou sur la voie de la normalisation de l'IETF sur le sujet et de l'état actuel des mises en œuvre courantes.

6. Considérations relatives à l'IANA

Dans le registre "Fanions d'en-tête du protocole de contrôle de transmission (TCP)", l'IANA a apporté plusieurs modifications, comme décrit dans cette section.

La RFC 3168 a créé à l'origine ce registre, mais ne l'a rempli qu'avec les nouveaux bits définis dans la RFC 3168, négligeant les autres bits qui avaient été précédemment décrits dans la RFC 793 et d'autres documents. Le bit 7 a depuis également été mis à jour par la [RFC8311].

La colonne "Bit" a été renommée ci-dessous en colonne "Décalage de bit", car elle fait référence au décalage de chaque fanion d'en-tête dans la vue alignée sur 16 bits de l'en-tête TCP de la figure 1. Les bits dans les décalages 0 à 3 sont le champ Décalage de données du segment TCP, et non des fanions d'en-tête.

L'IANA a ajouté une colonne pour les "Notes d'allocation".

L'IANA a attribué les valeurs comme indiqué ci-dessous.

Tableau 7 : fanions d'en-tête TCP

Décalage de bits	Nom	Référence	Notes d'allocation
4	Réservé pour une utilisation future	RFC 9293	
5	Réservé pour une utilisation future	RFC 9293	
6	Réservé pour une utilisation future	RFC 9293	
7	Réservé pour une utilisation future	RFC 8311	Précédemment utilisé par la RFC 3540 historique sous le nom de NS (Nonce Sum).
8	LRS (fenêtre d'encombrement réduite)	RFC 3168	
9	ECE (écho ECN)	RFC 3168	
10	Le champ de pointeur urgent est significatif (URG)	RFC 9293	
11	Le champ d'accusé de réception est significatif (ACK)	RFC 9293	
12	Fonction push (PSH)	RFC 9293	
13	Réinitialiser la connexion (RST)	RFC 9293	
14	Synchroniser les numéros de séquence (SYN)	RFC 9293	
15	Plus de données de l'envoyeur (FIN)	RFC 9293	

Le registre "Fanions d'en-tête TCP" a également été déplacé à un sous-registre du registre global "Paramètres du protocole de contrôle de transmission (TCP)" <https://www.iana.org/assignments/tcp-parameters/>.

La procédure d'enregistrement du registre reste Action de normalisation, mais la référence a été mise à jour pour pointer sur le présent document et la note a été supprimée.

7. Considérations relatives à la sécurité et à la confidentialité

La conception de TCP ne comprend que des fonctionnalités de sécurité rudimentaires qui améliorent la robustesse et la fiabilité des connexions et du transfert de données d'application, mais il n'existe pas de capacités cryptographiques intégrées pour prendre en charge toute forme de confidentialité, d'authentification ou d'autres fonctions typiques de sécurité. Des améliorations non cryptographiques (par exemple, la [RFC5961]) ont été mises au point pour améliorer la

robustesse des connexions TCP face à des types particuliers d'attaques, mais l'applicabilité et les protections des améliorations non cryptographiques sont limitées (par exemple, voir le paragraphe 1.1 de la [RFC5961]). Les applications utilisent généralement des protocoles de couche inférieure (par exemple, IPsec) et de couche supérieure (par exemple, TLS) pour assurer la sécurité et la confidentialité des connexions TCP et des données d'application transportées dans TCP. Des méthodes fondées sur les options TCP ont également été développées pour prendre en charge certaines capacités de sécurité.

Afin d'assurer pleinement la confidentialité, la protection de l'intégrité et l'authentification des connexions TCP (y compris leurs fanions de contrôle) IPsec est la seule méthode efficace à l'heure actuelle. Pour la protection de l'intégrité et l'authentification, l'option d'authentification TCP (TCP-AO) [RFC5925] est disponible, avec une proposition d'extension pour assurer également la confidentialité de la charge utile du segment. D'autres méthodes abordées dans la présente section peuvent assurer la confidentialité ou la protection de l'intégrité la charge utile, mais pour l'en-tête TCP, ne couvrir qu'un sous-ensemble des champs (par exemple, tcpcrypt [RFC8548]) ou aucun (par exemple, TLS). D'autres fonctions de sécurité qui ont été ajoutées à TCP (par exemple, génération d'ISN, vérifications des numéros de séquence, etc.) ne sont capables que d'entraver partiellement les attaques.

Les applications utilisant des flux TCP à longue durée de vie ont été vulnérables aux attaques qui exploitent le traitement des fanions de contrôle décrits dans les spécifications TCP précédentes [RFC4953]. TCP-MD5 était une option TCP couramment mise en œuvre pour prendre en charge l'authentification de certaines de ces connexions, mais présentait des défauts et est maintenant déconseillé. TCP-AO offre la possibilité de protéger les connexions TCP à longue durée de vie contre les attaques et possède des propriétés supérieures à TCP-MD5. Il n'assure aucune confidentialité pour les données de l'application ou pour les en-têtes TCP.

L'extension expérimentale "tcpcrypt" [RFC8548] de TCP offre la possibilité de protéger cryptographiquement les données de connexion. Les aspects de métadonnées du flux TCP sont toujours visibles, mais le flux d'application est bien protégé. Dans l'en-tête TCP, seuls le pointeur urgent et le fanion FIN sont protégés par tcpcrypt.

La feuille de route de TCP [RFC7414] comprend des notes sur plusieurs RFC relatives à la sécurité de TCP. De nombreuses améliorations apportées par ces RFC ont été intégrées dans le présent document, notamment la génération d'ISN, l'atténuation des attaques aveugles dans la fenêtre et l'amélioration de la gestion des erreurs douces et des paquets ICMP. Tout cela est discuté plus en détail dans les RFC référencées qui décrivaient à l'origine les modifications nécessaires aux spécifications TCP antérieures. De plus, voir la [RFC6093] qui discute des considérations de sécurité relatives au champ Pointeur urgent, qui déconseille aussi aux nouvelles applications d'utiliser le pointeur urgent.

Comme TCP est souvent utilisé pour les flux de transfert en vrac, certaines attaques sont possibles qui abusent de la logique de contrôle d'encombrement de TCP. Les attaques de type "division de ACK" en sont un exemple. Les mises à jour qui ont été apportées aux spécifications de contrôle d'encombrement de TCP incluent des mécanismes tels que le comptage d'octets appropriés (ABC, *Appropriate Byte Counting*) [RFC3465] qui agissent pour atténuer ces attaques.

D'autres attaques visent à épuiser les ressources d'un serveur TCP. Des exemples sont l'inondation de SYN [RFC4987] ou le gaspillage des ressources sur des connexions qui ne progressent pas [RFC6429]. Les systèmes d'exploitation mettent couramment en œuvre des mesures d'atténuation pour ces attaques. Certaines défenses courantes utilisent également des mandataires, des pare-feu à états pleins et d'autres technologies en dehors de la mise en œuvre TCP de l'hôte d'extrémité.

Le concept d'une "image filaire" d'un protocole est décrit dans la [RFC8546], qui explique comment les en-têtes en clair de TCP exposent aux nœuds du chemin plus de métadonnées que ce qui est strictement nécessaire pour acheminer les paquets vers leur destination. Les adversaires sur le chemin peuvent être capables d'exploiter ces métadonnées. Les enseignements tirés de TCP à cet égard ont été appliqués à la conception de nouveaux transports comme QUIC [RFC9000]. De plus, en partie sur la base de l'expérience acquise avec TCP et ses extensions, il existe des considérations qui pourraient s'appliquer aux futures extensions à TCP et à d'autres transports que l'IETF a documentés dans la [RFC9065], ainsi que les recommandations de l'IAB dans les [RFC8558] et [RFC9170].

Il y a aussi des méthodes de "prise d'empreinte digitale" qui peuvent être utilisées pour déduire les informations de version ou de plate-forme de la mise en œuvre (système d'exploitation) TCP de l'hôte. Elles collectent des observations de plusieurs aspects, comme les options présentes dans les segments, l'ordre des options, les comportements spécifiques dans le cas de conditions diverses, le cadencement des paquets, la taille des paquets et d'autres aspects du protocole qui sont à déterminer par une mise en œuvre, et peuvent utiliser ces observations pour identifier des informations sur l'hôte et la mise en œuvre.

Comme le traitement des messages ICMP peut aussi interagir avec les connexions TCP, il existe un risque d'attaques fondées sur ICMP contre les connexions TCP. Cela est discuté dans la [RFC5927], ainsi que les mesures d'atténuation qui ont été mises en œuvre.

8. Références

8.1 Références normatives

[RFC0791] J. Postel, éd., "Protocole Internet - Spécification du protocole du programme Internet", STD 5, DOI 10.17487/RFC0791, septembre 1981.

[RFC1191] J. Mogul et S. Deering, "Découverte de la MTU de chemin", DOI 10.17487/RFC1191, novembre 1990.

[RFC2119] S. Bradner, "Mots clés à utiliser dans les RFC pour indiquer les niveaux d'exigence", BCP 14, mars 1997. DOI 10.17487/RFC2119, *(MàJ par* RFC8174)

[RFC2474] K. Nichols, S. Blake, F. Baker et D. Black, "Définition du champ Services différenciés (DS) dans les en-têtes IPv4 et IPv6", DOI 10.17487/RFC2474, décembre 1998. *(P.S. ; MàJ par* RFC3168, RFC3260, RFC8436)

[RFC2914] S. Floyd, "Principes du contrôle d'encombrement", BCP 41, DOI 10.17487/RFC2914, septembre 2000.

[RFC3168] K. Ramakrishnan et autres, "Ajout de la notification explicite d'encombrement (ECN) à IP", DOI 10.17487/RFC3168, septembre 2001. *(P.S. ; MàJ par* RFC8311)

[RFC5033] S. Floyd, M. Allman, "Spécification de nouveaux algorithmes de contrôle d'encombrement", DOI 10.17487/RFC5033, août 2007. (BCP0133)

[RFC5681] M. Allman, V. Paxson, E. Blanton, "Contrôle d'encombrement de TCP", DOI 10.17487/RFC5681, septembre 2009. *(Remplace* RFC2581) *(D.S.)*

[RFC5961] A. Ramaiah, R. Stewart, M. Dalal, "Amélioration de la robustesse de TCP aux attaques sur fenêtre aveugle", DOI 10.17487/RFC5961, août 2010. *(P.S. ; MàJ par* RFC9293*)*

[RFC6298] V. Paxson, M. Allman, J. Chu, M. Sargent, "Calcul du temporisateur de retransmission de TCP", DOI 10.17487/RFC6298, juin 2011. *(Remplace la* RFC2988) *(MàJ la* RFC1122) *(P.S.)*

[RFC6633] F. Gont, "Les messages Extinction de source ICMP sont déconseillés", DOI 10.17487/RFC6633, mai 2012. (*P.S. ; MàJ les* RFC0792, 1122, 1812)

[RFC8174] B. Leiba, "Ambiguïté des mots clés en majuscules ou minuscules dans la RFC2119", DOI 10.17487/RFC8174, mai 2017. BCP14. *(MàJ RFC*2119)

[RFC8200] S. Deering, R. Hinden, "Spécification du protocole Internet version 6 (IPv6)", juillet 2017, DOI 10.17487/RFC8200. STD 86. *(Remplace* 2460)

[RFC8201] J. McCann, et autres, "Découverte de la MTU de chemin pour IPv6", DOI 10.17487/RFC8201, juillet 2017. STD 87. *(Remplace* RFC1981)

[RFC8961] M. Allman, "Exigences pour la détection de pertes fondée sur l'heure", DOI : 10.17487/RFC8961, novembre 2020. (BCP0233)

8.2 Références pour information

[IANA] IANA, "Paramètres du protocole TCP", <https://www.iana.org/assignments/tcp-parameters/ >.

[RFC0793] J. Postel (éd.), "Protocole de commande de transmission – Spécification du protocole du programme Internet DARPA", STD 7, DOI 10.17487/RFC0793, septembre 1981. (*Remplacée par* RFC9293)

[RFC0896] J. Nagle, "Contrôle de l'encombrement dans l'inter-réseau IP/TCP", DOI 10.17487/RFC0896, janvier 1984. *(Historique)*

[RFC1011] J. Reynolds et J. Postel, "Protocoles officiels de l'Internet", DOI 10.17487/RFC1011, mai 1987.

[RFC1122] R. Braden, "Exigences pour les hôtes Internet – couches de communication", STD 3, DOI 10.17487/RFC1122, octobre 1989. *(MàJ par* RFC6633, 8029, 9293)

[RFC1349] P. Almquist, "Type de service dans la suite de protocole Internet", DOI 10.17487/RFC1349, juillet 1992. *(Remplacée par RFC2474)*

[RFC1644] R. Braden, "T/TCP – Extensions à TCP pour spécification fonctionnelle de transactions", DOI 10.17487/RFC1644, juillet 1994. *(Exp.)*

[RFC2018] M. Mathis et autres, "Options d'accusé de réception sélectif sur TCP", DOI 10.17487/RFC2018, octobre 1996. (*Remplace* RFC1072) *(P.S.)*

[RFC2525] V. Paxson et autres, "Problèmes connus de mise en œuvre de TCP", DOI 10.17487/RFC2525, mars 1999. *(Information)*

[RFC2675] D. Borman, S. Deering, R. Hinden, "Jumbogrammes IPv6", DOI 10.17487/RFC2675, août 1999. *(P.S.)*

[RFC2873] X. Xiao et autres, "Traitement du champ de préséance IPv4 dans TCP", DOI 10.17487/RFC2873, juin 2000. *(P.S. ; remplacée par* RFC9293*)*

[RFC2883] S. Floyd et autres, "Extension à l'option d'accusé de réception sélectif (SACK) pour TCP", DOI 10.17487/RFC2883, juillet 2000. *(P.S.)*

[RFC2923] K. Lahey, "Problèmes de TCP avec la découverte de MTU de chemin", DOI 10.17487/RFC2923, septembre 2000. *(Information)*

[RFC3449] H. Balakrishnan et autres, "Implications de l'asymétrie du chemin de réseau sur les performance de TCP", DOI 10.17487/RFC3449, décembre 2002. (BCP0069)

[RFC3465] M. Allman, "Contrôle d'encombrement sur TCP avec compte d'octets appropriés (ABC)", DOI 10.17487/RFC3465, février 2003. *(Expérimentale)*

[RFC4727] B. Fenner, "Valeurs expérimentales dans les en-têtes IPv4, IPv6, ICMPv4, ICMPv6, UDP, et TCP", DOI 10.17487/RFC4727, novembre 2006. *(P.S.)*

[RFC4821] M. Mathis, J. Heffner, "Découverte de la MTU de chemin de couche de mise en paquet", DOI 10.17487/RFC4821, mars 2007. *(P.S.)*

[RFC4953] J. Touch, "Défendre TCP contre les attaques par usurpation d'identité", DOI 10.17487/RFC4953, juillet 2007. *(Information)*

[RFC4987] W. Eddy, "Attaques par inondation de segment SYN contre TCP et contre-mesures courantes", DOI 10.17487/RFC4987, août 2007. *(Information)*

[RFC5044] P. Culley et autres, "Tramage verrouillé sur la PDU de marqueur pour la spécification de TCP", DOI 10.17487/RFC5044, octobre 2007. *(P.S. ; MàJ par* RFC6581, RFC7146 *)*

[RFC5461] F. Gont, "Réaction de TCP aux erreurs douces", DOI 10.17487/RFC5461, février 2009. *(Information)*

[RFC5570] M. StJohns, R. Atkinson, G. Thomas, "Architecture commune d'étiquette d'option de sécurité IPv6 (CALIPSO)", DOI 10.17487/RFC5570, juillet 2009. *(Info)*

[RFC5795] K. Sandlund, G. Pelletier, L-E. Jonsson, "Cadre de la compression d'en-tête RObust (ROHC)", DOI 10.17487/RFC5795,, mars 2010. (*Remplace* RFC4995). *(P. S.)*

[RFC5925] J. Touch, A. Mankin, R. Bonica, "Option Authentification de TCP", DOI 10.17487/RFC5925, juin 2010. (*Remplace* RFC2385). *(P. S.)*

[RFC5927] F. Gont, "Attaques ICMP contre TCP", DOI 10.17487/RFC5927, juillet 2010. *(Information)*

[RFC6093] F. Gont, A. Yourtchenko, "À propos de la mise en œuvre du mécanisme Urgent de TCP", DOI 10.17487/RFC6093, janvier 2011. (*MàJ RFC* 0793, *RFC*1011, *RFC*1122) *(P.S. ; remplacée par* RFC9293*)*

[RFC6191] F. Gont, "Réducion de l'état TIME-WAIT en utilisant des horodatages", DOI 10.17487/RFC6191, avril 2011. (BCP0159)

[RFC6429] M. Bashyam, M. Jethanandani, A. Ramaiah, "Éclaircissements sur l'envoyeur TCP pour la condition Persist", DOI 10.17487/RFC6429, décembre 2011. *(Information ; remplacée par* RFC9293*)*

[RFC6528] F. Gont, S. Bellovin, "Se défendre contre les attaques de numéro de séquence", DOI 10.17487/RFC6528, février 2012. (*P.S. ; Remplace la* RFC1948 ; *MàJ la* RFC0793 ; *remplacée par* RFC9293*)*

[RFC6691] D. Borman, "Options TCP et taille maximum de segment (MSS)", DOI 10.17487/RFC6691, juillet 2012. (*MàJ les* RFC0879, RFC2385) *(Information ; remplacée par* RFC9293*)*

[RFC6864] J. Touch, "Mise à jour de la spécification du champ d'ID IPv4", DOI 10.17487/RFC6864, février 2013.

RFC6994] J. Touch, "Utilisation partagée des options expérimentales de TCP", DOI 10.17487/RFC6994, août 2013 *(P.S.)*

[RFC7094] D. McPherson et autres, "Considérations architecturales sur l'envoi IP à la cantonade", DOI 10.17487/RFC7094, janvier 2014. *(Information)*

[RFC7323] D. Borman, et autres, "Extensions à TCP pour l'amélioration des performances", DOI 10.17487/RFC7323, septembre 2014. *(P.S.)*

[RFC7413] Y. Cheng, et autres, "Ouverture rapide de TCP", DOI 10.17487/RFC7413, décembre 2014. *(Expérimental)*

[RFC7414] M. Duke, et autres, "Plan pour les documents de spécification du protocole de contrôle de transmission", DOI 10.17487/RFC7414, février 2015. *(Information, MàJ par* RFC7805)

[RFC7657] D. Black, P. Jones, "Services différenciés (Diffserv) et communication en temps réel", DOI 10.17487/RFC7657, novembre 2015. *(Info)*

[RFC8087] G. Fairhust, M. Welzl, "Avantages de l'utilisation de la notification explicite d'encombrement (ECN)", DOI 10.17487/RFC8087, mars 2017. *(Information)*

[RFC8095] G. Fairhurst et autres, "Services fournis par le protocole de transport de l'IETF et mécanismes de contrôle de l'encombrement", DOI 10.17487/RFC8095, mars 2017. *(Information)*

[RFC8303] M.Welel, M. Tuexen, N. Khademi, "Utilisation des caractéristiques de transport fournies par les protocoles de l'IETF", DOI 10.17487/RFC8303, février 2018. *(Information)*

[RFC8311] D. Black, "Expérimentation d'assouplissement des restrictions à ECN", DOI 10.17487/RFC8311, janvier 2018. *(P.S. ; MàJ* RFC3168, 4341, 5622, 6679*)*

[RFC8504] T. Chown, J. Loughney, T. Winters, "Exigences pour les nœuds IPv6", DOI 10.17487/RFC8546, janvier 2019. BCP 220. *(Remplace la* RFC 6434)

[RFC8546] B. Trammell, M. Kuehlewind, "Image filaire d'un protocole réseau", DOI 10.17487/RFC8546, avril 2019. *(Information)*

[RFC8548] A. Bittau, et autres, "Protection cryptographique des flux TCP (tcpcrypt)", DOI 10.17487/RFC8548, mai 2019. *(Expérimentale)*

[RFC8558] T. Hardie, "Signalisation de chemin de protocole de transport", DOI 10.17487/RFC8558, avril 2019.

(Information)

[RFC8684] A. Ford, et autres, "Extensions TCP pour fonctionnement multichemins avec plusieurs adresses", DOI 10.17487/RFC8684, mars 2020. *(P.S. ; Remplace RFC*6824)

[RFC9000] J. Iyengar, Ed., M. Thomson, "QUIC : transport multiplexé et sécurisé fondé sur UDP" mai 2021. (DOI : 10.17487/RFC9000) *(P.S.)*

[RFC9065] G. Fairhurst, C. Perkins, "Considérations sur la confidentialité d'en-tête de transport, le fonctionnement du réseau, et l'évolution des protocoles de transport Internet", juillet 2021. (DOI : 10.17487/RFC9065) *(Info.)*

[RFC9170] M. Thomson, T. Pauly, "Viabilité à long terme des mécanismes d'extension de protocole", décembre 2021. (DOI : 10.17487/RFC9170) *(Information)*

[63] Gont, F., "Traitement des informations de compartiment IP de sécurité/préséance par TCP", Travail en cours, 29 mars 2012, <https://datatracker.ietf.org/doc/html/draft-gont-tcpm-tcp-seccomp-prec-00 >.

[64] Gont, F. et D. Borman, "Sur la validation des numéros de séquence TCP", Travail en cours, 11 mars 2019, <https://datatracker.ietf.org/doc/html/draft-gont-tcpm-tcp-seq-validation-04 >.

[65] Touch, J. et W. M. Eddy, "Option de décalage de données étendu de TCP", Travail en cours,, 15 avril 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-tcpm-tcp-edo-12 >.

[66] McQuistin, S., Groupe, V., Jacob, D.et C. Perkins, "Description des unités de données de protocole avec des diagrammes d'en-tête de paquet augmentés", Travail en cours,7 mars 2022, <https://datatracker.ietf.org/doc/html/draft-mcquistin-augmented-ascii-diagrams-10 >.

[68] Minshall, G., "Modification suggérée à l'algorithme de Nagle", Travail en cours,, 18 juin 1999, <https://datatracker.ietf.org/doc/html/draft-minshall-nagle-01 >.

[69] Dalal, Y., "Gestion des connexions dans les protocoles de transport", Computer Networks, vol. 2, n° 6, p. 454-473, DOI 10.1016/0376-5075(78)90053-3, décembre 1978, <https://doi.org/10.1016/0376-5075(78)90053-3 >.

[70] Faber, T., Touch, J.et W. Yui, "L'état TIME-WAIT dans TCP et son effet sur les serveurs occupés", Actes de IEEE INFOCOM, pp. 1573-1583, DOI 10.1109/INFCOM.1999.752180, mars 1999, <https://doi.org/10.1109/INFCOM.1999.752180 >.

[71] J. Postel., "Commentaires sur les mesures de suivi de la réunion de janvier", IEN 177, mars 1981, <https://www.rfc-editor.org/ien/ien177.txt >.

[72] **"**Déchargements de segmentation", La documentation du noyau Linux, <https://www.kernel.org/doc/html/latest/networking/segmentation-offloads.html >.

[73] RFC Errata, Erratum ID 573, RFC 793, <https://www.rfc-editor.org/errata/eid573>.
[74] RFC Errata, Erratum ID 574, RFC 793, <https://www.rfc-editor.org/errata/eid574>.
[75] RFC Errata, Erratum ID 700, RFC 793, <https://www.rfc-editor.org/errata/eid700>.
[76] RFC Errata, Erratum ID 701, RFC 793, <https://www.rfc-editor.org/errata/eid701>.
[77] RFC Errata, Erratum ID 1283, RFC 793, <https://www.rfc-editor.org/errata/eid1283>.
[78] RFC Errata, Erratum ID 1561, RFC 793, <https://www.rfc-editor.org/errata/eid1561>.
[79] RFC Errata, Erratum ID 1562, RFC 793, <https://www.rfc-editor.org/errata/eid1562>.
[80] RFC Errata, Erratum ID 1564, RFC 793, <https://www.rfc-editor.org/errata/eid1564>.
[81] RFC Errata, Erratum ID 1571, RFC 793, <https://www.rfc-editor.org/errata/eid1571>.
[82] RFC Errata, Erratum ID 1572, RFC 793, <https://www.rfc-editor.org/errata/eid1572>.
[83] RFC Errata, Erratum ID 2297, RFC 793, <https://www.rfc-editor.org/errata/eid2297>.
[84] RFC Errata, Erratum ID 2298, RFC 793, <https://www.rfc-editor.org/errata/eid2298>.
[85] RFC Errata, Erratum ID 2748, RFC 793, <https://www.rfc-editor.org/errata/eid2748>.
[86] RFC Errata, Erratum ID 2749, RFC 793, <https://www.rfc-editor.org/errata/eid2749>.
[87] RFC Errata, Erratum ID 2934, RFC 793, <https://www.rfc-editor.org/errata/eid2934>.
[88] RFC Errata, Erratum ID 3213, RFC 793, <https://www.rfc-editor.org/errata/eid3213>.
[89] RFC Errata, Erratum ID 3300, RFC 793, <https://www.rfc-editor.org/errata/eid3300>.

[90] RFC Errata, Erratum ID 3301, RFC 793, <https://www.rfc-editor.org/errata/eid3301>.
[91] RFC Errata, Erratum ID 6222, RFC 793, <https://www.rfc-editor.org/errata/eid6222>.
[92] RFC Errata, Erratum ID 572, RFC 793, <https://www.rfc-editor.org/errata/eid572>.
[93] RFC Errata, Erratum ID 575, RFC 793, <https://www.rfc-editor.org/errata/eid575>.
[94] RFC Errata, Erratum ID 1565, RFC 793, <https://www.rfc-editor.org/errata/eid1565>.
[95] RFC Errata, Erratum ID 1569, RFC 793, <https://www.rfc-editor.org/errata/eid1569>.
[96] RFC Errata, Erratum ID 2296, RFC 793, <https://www.rfc-editor.org/errata/eid2296>.
[97] RFC Errata, Erratum ID 3305, RFC 793, <https://www.rfc-editor.org/errata/eid3305>.
[98] RFC Errata, Erratum ID 3602, RFC 793, <https://www.rfc-editor.org/errata/eid3602>.
[99] RFC Errata, Erratum ID 4772, RFC 5961, <https://www.rfc-editor.org/errata/eid4772>.

Annexe A. Autres notes de mise en œuvre

Cette section comprend des notes et des références supplémentaires sur les décisions de mise en œuvre de TCP qui ne font actuellement pas partie de la série des RFC ou qui ne sont pas incluses dans la norme TCP. Ces éléments peuvent être pris en compte par les responsables de la mise en œuvre, mais il n'y avait pas encore de consensus pour les inclure dans la norme.

A.1 Compartiment de sécurité IP et préséance

La spécification IPv4 [RFC0791] inclut une valeur de préséance dans le champ Type de service (TOS) (désormais obsolète). Il a été modifié dans la [RFC1349] et ensuite rendu obsolète par la définition de services différenciés (Diffserv) [RFC2474]. L'établissement et le transport du ToS entre la couche réseau, la mise en œuvre TCP et les applications est obsolète et est remplacé par Diffserv dans la spécification TCP actuelle.

La RFC 793 exigeait la vérification du compartiment de sécurité IP et de la préséance sur les segments TCP entrants pour la cohérence au sein d'une connexion et avec les demandes d'application. Chacun de ces aspects de IP est devenu désuet, sans mises à jour spécifiques de la RFC 793. Les questions relatives à la préséance étaient réglées par la [RFC2873], qui est sur la voie de la normalisation, et donc la présente spécification de TCP inclut ces modifications. Cependant, l'état des options de sécurité IP qui peuvent être utilisées par les systèmes de sécurité multi niveaux (MLS, *Multi-Level Secure*) n'est pas aussi évident dans l'IETF actuellement.

La réinitialisation des connexions lorsque les paquets entrants ne satisfont pas aux attentes du compartiment de sécurité ou de préséance a été reconnue comme un possible vecteur d'attaque [63], et il y a eu une discussion sur l'amendement de la spécification TCP pour empêcher les connexions d'être interrompues en raison d'une non correspondance entre le compartiment de sécurité IP et les valeurs de codets Diffserv.

A.1.1 Préséance

Dans Diffserv, les anciennes valeurs de préséance sont traitées comme des codets de sélecteur de classe, et les méthodes de traitement compatible sont décrites dans l'architecture Diffserv. La spécification de TCP définie par les RFC 793 et 1122 incluait une logique visant à faire en sorte que les connexions utilisent la préséance la plus élevée demandée par l'un ou l'autre des points d'extrémité d'application, et pour maintenir la préséance cohérente tout au long d'une connexion. Cette logique de ToS obsolète n'est pas applicable à Diffserv et ne devrait pas être incluse dans les mises en œuvre de TCP, bien que les changements des valeurs Diffserv au sein d'une connexion soient déconseillées. Pour une discussion à ce sujet, voir les paragraphes 5.1 et 5.3 et la Section 6 de la [RFC7657].

Les règles de traitement de ToS obsolètes dans TCP supposaient des valeurs de préséance bidirectionnelles (ou symétriques) utilisées sur une connexion, mais l'architecture Diffserv est asymétrique. Les problèmes rencontrés avec l'ancienne logique de TCP à cet égard ont été décrits dans la [RFC2873], et la solution décrite consiste à ignorer la préséance IP dans TCP. Comme la RFC 2873 est un document sur la voie de la normalisation (bien qu'elle ne soit pas marquée comme mise à jour de la RFC 793) les mises en œuvre actuelles devraient être robustes dans ces conditions. Noter que la valeur du champ Diffserv utilisée dans chaque direction fait partie de l'interface entre TCP et la couche réseau, et les valeurs utilisées peuvent être indiquées dans les deux sens entre TCP et l'application.

A.1.2 Systèmes MLS

L'option de sécurité IP (IPSO) et le compartiment définis dans la [RFC0791] a été précisée dans la RFC 1038, qui a ensuite été rendue obsolète par la RFC 1108. L'option de sécurité IP commerciale (CIPSO) est définie dans la norme FIPS-188 (retirée par le NIST en 2015) et est prise en charge par certains fournisseurs et systèmes d'exploitation. La RFC 1108 est maintenant Historique, bien que la RFC 791 elle-même n'ait pas été mise à jour pour supprimer l'option IP de sécurité. Pour IPv6, une option similaire (CALIPSO, *Common Architecture Label IPv6 Security Option*)) a été définie dans la [RFC5570]. La RFC 793 comporte une logique qui inclut les informations de sécurité/compartiment IP dans le traitement des segments TCP. Les références à la "sécurité/compartiment" IP dans le présent document peuvent être pertinentes pour les mises en œuvre de systèmes de sécurité multi niveaux (MLS, *Multi-Level Secure*) mais peuvent être ignorés pour les mises en œuvre non MLS, en cohérence avec le code exécuté sur l'Internet. Voir plus de détails à l'annexe A.1. Noter que la RFC 5570 décrit certains scénarios de mise en réseau MLS où IPSO, CIPSO ou CALIPSO peuvent être utilisés. Dans ces cas particuliers, les mises en œuvre de TCP devraient voir le paragraphe 7.3.1 de la RFC 5570 et suivre les instructions de ce document.

A.2 Validation du numéro de séquence

Dans certains cas, les règles de validation du numéro de séquence TCP peuvent empêcher le traitement des champs ACK. Cela peut entraîner des problèmes de connexion, comme décrit dans [64], qui comprend des descriptions de problèmes potentiels dans des conditions d'ouverture, d'auto connexion, de fermeture simultanée et de sondes de fenêtre simultanées. Le document décrit également les modifications potentielles apportées à la spécification TCP pour atténuer le problème en élargissant les numéros de séquence acceptables.

Dans l'utilisation de TCP sur Internet, ces conditions se produisent rarement. Les systèmes d'exploitation courants incluent différentes solutions d'atténuation de remplacement, et la norme n'a pas encore été mise à jour pour codifier l'une d'entre elles, mais les responsables de la mise en œuvre doivent prendre en compte les problèmes décrits dans [64].

A.3 Modification de Nagle

Dans les systèmes d'exploitation courants, l'algorithme de Nagle et les accusés de réception différés sont mis en œuvre et activés par défaut. TCP est utilisé par de nombreuses applications qui ont un style de demande-réponse de communication, où la combinaison de l'algorithme de Nagle et des accusés de réception différés peut entraîner de mauvaises performances de l'application. Une modification de l'algorithme de Nagle est décrite dans [68] qui améliore la situation pour ces applications.

Cette modification est mise en œuvre dans certains systèmes d'exploitation courants et n'a pas d'impact sur l'interopérabilité de TCP. De plus, de nombreuses applications désactivent simplement Nagle, car cela est généralement pris en charge par une option de prise. La norme TCP n'a pas été mise à jour pour inclure cette modification de Nagle, mais les mises en œuvre peuvent trouver bénéfique de la prendre en compte.

A.4 Réglages de filigrane numérique faible

Certaines mises en œuvre TCP du noyau du système d'exploitation incluent des options de prise qui permettent de spécifier le nombre d'octets dans la mémoire tampon jusqu'à ce que la couche prise transmette les données envoyées à TCP (SO_SNDLOWAT) ou à l'application à la réception (SO_RCVLOWAT).

De plus, une autre option de prise (TCP_NOTSENT_LOWAT) peut être utilisée pour contrôler la quantité d'octets non envoyés dans la file d'attente d'écriture. Cela peut aider une application TCP émettrice à éviter de créer de grandes quantités de données mises en mémoire tampon (et la latence correspondante). À titre d'exemple, cela peut être utile pour les applications qui multiplexent des données de plusieurs flux de niveau supérieur sur une connexion, en particulier lorsque les flux peuvent être un mélange de transfert de données interactives/en temps réel et en vrac.

Appendice B. Résumé des exigences de TCP

Cette section est adaptée de la RFC 1122.
Noter qu'il n'y a pas d'exigence relative à la PLPMTUD dans cette liste, mais que la PLPMTUD est recommandée.

Tableau 8 : Résumé des exigences en matière de TCP

Caractéristique	ReqID	DOIT	DEVRAIT	PEUT	NE DEVRAIT PAS	NE DOIT PAS
Fanion PUSH						
Agréger ou mettre en file d'attente les données non poussées	PEUT-16			X		
Réduire les bits PSH successifs de l'envoyeur	DVRT-27		X			
L'appel SEND peut spécifier PUSH	PEUT-15			X		
• * Si ce n'est pas le cas : mémoire tampon de l'envoyeur indéfiniment	DOIT-60					X
• * Si ce n'est pas le cas : PSH dernier segment	DOIT-61	X				
Notifier la réception de l'ALP[1] de PSH	PEUT-17			X		
Envoyer le segment de taille maximale lorsque cela est possible	DVRT-28		X			
Fenêtre						
Traiter comme un nombre non signé	DOIT-1	X				
Traiter comme nombre de 32 bits	REC-1		X			
Réduire la fenêtre à partir de la droite	DVRT-14				X	
• * Envoi de nouvelles données lorsque la fenêtre se rétrécit	DVRT-15				X	
• * Retransmettre les anciennes données non confirmées dans la fenêtre	DVRT-16		X			
• * Fin de temporisation pour les données au-delà du bord droit	DVRT-17				X	
Robuste contre le rétrécissement de la fenêtre	DOIT-34	X				
La fenêtre du receveur s'est fermée indéfiniment	PEUT-8			X		
Utiliser une logique de sondage standard	DOIT-35	X				
L'envoyeur sonde la fenêtre zéro	DOIT-36	X				
• * Première sonde après RTO	DVRT-29		X			
• * Recul exponentiel	DVRT-30		X			
Autoriser la fenêtre à rester zéro indéfiniment	DOIT-37	X				
Retransmettre les anciennes données au-delà de SND.UNA+SND.WND	PEUT-7			X		
Traiter RST et URG même avec une fenêtre zéro	DOIT-66	X				
Données urgentes						
Inclure la prise en charge du pointeur urgent	DOIT-30	X				
Le pointeur indique le premier octet non urgent	DOIT-62	X				
Séquence de données urgente de longueur arbitraire	DOIT-31	X				
Informer ALP[1] de manière asynchrone des données urgentes	DOIT-32	X				
ALP[1] peut apprendre si/combien de données urgentes	DOIT-33	X				
ALP utilise le mécanisme d'urgence	DVRT-13				X	
Options TCP						
Prise en charge de l'ensemble d'options obligatoires	DOIT-4	X				
Option de réception TCP dans tout segment	DOIT-5	X				
Ignorer les options non prises en charge	DOIT-6	X				

Caractéristique	ReqID	DOIT	DEVRAIT	PEUT	NE DEVRAIT PAS	NE DOIT PAS
Inclure la longueur pour toutes les options sauf EOL+NOP	DOIT-68	X				
S’accommoder de la durée illégale des options	DOIT-7	X				
Traiter les options indépendamment de l’alignement sur le mot	DOIT-64	X				
Mettre en œuvre l’option MSS d’envoi et de réception	DOIT-14	X				
Option IPv4 d’envoi MSS sauf 536	DVRT-5		X			
Option d’envoi MSS IPv6 sauf 1220	DVRT-5		X			
Envoyer toujours l’option MSS	PEUT-3			X		
La valeur par défaut d’IPv4 Send-MSS est 536	DOIT-15	X				
La valeur par défaut d’IPv6 Send-MSS est 1220	DOIT-15	X				
Calculer la taille effective de la séquence d’envoi	DOIT-16	X				
MSS tient compte de MTU variables	DVRT-6		X			
MSS non envoyée sur les segments non-SYN	DOIT-65					X
Valeur de MSS fondée sur MMS_R	DOIT-67	X				
Bourrage avec zéro	DOIT-69	X				
Sommes de contrôle TCP						
L’envoyeur calcule la somme de contrôle	DOIT-2	X				
Le receveur vérifie la somme de contrôle	DOIT-3	X				
Choix de l'ISN						
Inclure un composant de générateur ISN piloté par horloge	DOIT-8	X				
Générateur d’ISN sécurisé avec un composant PRF	DVRT-1		X			
PRF calculable depuis l’extérieur de l’hôte	DOIT-9					X
Ouverture des connexions						
Prise en charge des tentatives d’ouverture simultanées	DOIT-10	X				
SYN-RECEIVED se souvient du dernier état	DOIT-11	X				
L’appel OPEN passif interfère avec d’autres	DOIT-41					X
Fonction : LISTEN simultanés pour le même accès	DOIT-42	X				
Demander à IP l’adresse de src pour SYN si nécessaire	DOIT-44	X				
* Sinon, utilisez l’adresse locale de connexion	DOIT-45	X				
OPEN à l’adresse IP de diffusion/multidiffusion	DOIT-46					X
Supprimer en silence les segments pour adresse de diffusion/diffusion groupée	DOIT-57	X				
Fermeture des connexions						
RST peut contenir des données	DVRT-2		X			
Informer l’application de connexion avortée	DOIT-12	X				
Connexions closes en semi-duplex	PEUT-1			X		
* Envoyer RST pour indiquer des données perdues	DVRT-3		X			
En état TIME-WAIT pendant 2MSL secondes	DOIT-13	X				
* Accepter SYN dans l’état TIME-WAIT	PEUT-2			X		
* Utilisez des horodatages pour réduire TIME-WAIT	DVRT-4		X			
Retransmissions						

Caractéristique	ReqID	DOIT	DEVRAIT	PEUT	NE DEVRAIT PAS	NE DOIT PAS
Mettre en œuvre le ralentissement exponentiel, le démarrage lent et l'évitement de l'encombrement	DOIT-19	X				
Retransmettre avec la même identité IP	PEUT-4			X		
Algorithme de Karn	DOIT-18	X				
Génération d'accusés de réception						
Agrégez chaque fois que possible	DOIT-58	X				
Mettre en file d'attente les segments déclassés	DVRT-31		X			
Traitez toute la file d'attente avant d'envoyer le ACK	DOIT-59	X				
Envoyer un ACK pour un segment déclassé	PEUT-13			X		
ACK retardés	DVRT-18		X			
• * Délai < 0,5 seconde	DOIT-40	X				
• * Tous les 2 segments de pleine taille ou 2*RMSS acquittés	DVRT-19		X			
Algorithme d'évitement SWS du receveur	DOIT-39	X				
Envoi de données						
TTL configurable	DOIT-49	X				
Algorithme d'évitement SWS de l'envoyeur	DOIT-38	X				
Algorithme de Nagle	DVRT-7		X			
• * L'application peut désactiver l'algorithme de Nagle	DOIT-17	X				
Échecs de connexion						
Avis négatif à l'IP sur les retransmissions R1	DOIT-20	X				
Cloture de connexion sur les retransmissions R2	DOIT-20	X				
ALP[1] peut définir R2	DOIT-21	X				
Informer ALP de R1<=retxs<R2	DVRT-9		X			
Valeur recommandée pour R1	DVRT-10		X			
Valeur recommandée pour R2	DVRT-11		X			
Même mécanisme pour les SYN	DOIT-22	X				
• * R2 au moins 3 minutes pour SYN	DOIT-23	X				
Envoyer des paquets de maintien en vie						
Envoi de paquets Maintien en vie :	PEUT-5		X			
• * L'application peut demander	DOIT-24	X				
• * La valeur par défaut est "désactivé"	DOIT-25	X				
• * Envoyer seulement si inactif pendant l'intervalle	DOIT-26	X				
• * Intervalle configurable	DOIT-27	X				
• * Par défaut au moins 2 heures.	DOIT-28	X				
• * Tolérant aux ACK perdus	DOIT-29	X				
• * Envoi sans données	DVRT-12		X			
• * Configurable pour envoyer un octet factice	PEUT-6			X		
IP Options						
Ignorer les options que TCP ne comprend pas	DOIT-50	X				
Prise en charge de l'horodatage	PEUT-10		X			
Prise en charge de l'enregistrement de chemin	PEUT-11		X			

Caractéristique	ReqID	DOIT	DEVRAIT	PEUT	NE DEVRAIT PAS	NE DOIT PAS
Route de source :						
• * ALP[1] peut spécifier	DOIT-51	X				
• * Remplace route de source dans le datagramme	DOIT-52	X				
• * Construit chemin de retour à partir de route de source	DOIT-53	X				
• * Outrepassement ultérieur de route de source	DVRT-24		X			
Réception de messages ICMP à partir de IP						
Réception de messages ICMP de IP	DOIT-54	X				
• * Dest injoign (0,1,5) => informer ALP	DVRT-25	X				
• * Abandon sur dest injoignable (0,1,5)	DOIT-56					X
• * Dest injoignable (2-4) => interruption connexion	DVRT-26		X			
• * Extinction de source => rejet en silence	DOIT-55	X				
• * Interruption en cas de dépassement de temps	DOIT-56					X
• * Interruption pour problème de paramètre	DOIT-56					X
Validation d'adresse						
Rejet d'appel OPEN vers une adresse IP invalide	DOIT-46	X				
Rejet de SYN à partir d'une adresse IP invalide	DOIT-63	X				
Supprimer en silence SYN vers adresse diff/diff group	DOIT-57	X				
Services d'interface TCP/ALP						
Mécanisme de rapport d'erreurs	DOIT-47	X				
ALP peut désactiver la routine de rapport d'erreurs	DVRT-20		X			
ALP peut spécifier le champ Diffserv pour l'envoi	DOIT-48	X				
• * Transmis inchangé à IP	DVRT-22		X			
ALP peut modifier le champ Diffserv durant la connexion	DVRT-21		X			
ALP change généralement Diffserv pendant la connexion	DVRT-23				X	
Passe le champ Diffserv reçu jusqu'à ALP	PEUT-9			X		
Appel FLUSH	PEUT-14			X		
Paramètre adr IP locale facultatif dans OPEN	DOIT-43	X				
Prise en charge de la RFC 5961						
Met en œuvre la protection contre l'injection de données	PEUT-12			X		
Notification explicite d'encombrement						
Prise en charge de ECN	DVRT-8		X			
Contrôle d'encombrement de remplacement						
Met en œuvre des algorithmes conformes de rempl.	PEUT-18			X		

Note : (1) "ALP" désigne le programme de couche application.

Remerciements

Ce document est en grande partie une révision de la RFC 793, dont Jon Postel était l'éditeur. Grâce à son excellent travail, il a pu durer trois décennies avant qu'on ne ressente le besoin de le réviser.

André Oppermann a contribué à éditer la première révision de ce document.

Nous sommes reconnaissants de l’aide des présidents du groupe de travail TCPM de l’IETF au cours du travail sur ce document : Michael Scharf, Yoshifumi Nishida, Pasi Sarolahti, Michael Tüxen.

Au cours des discussions de ce travail sur la liste de diffusion TCPM, dans les réunions du groupe de travail, et via des revues de zone, des commentaires utiles, critiques, et des révisions ont été reçues de (classés par ordre alphabétique de nom de famille) : Praveen Balasubramanian, David Borman, Mohamed Boucadair, Bob Briscoe, Neal Cardwell, Yuchung Cheng, Martin Duke, Francis Dupont, Ted Faber, Gorry Fairhurst, Fernando Gont, Rodney Grimes, Yi Huang, Rahul Jadhav, Markku Kojo, Mike Kosek, Juhamatti Kuusisaari, Kevin Lahey, Kevin Mason, Matt Mathis, Stephen McQuistin, Jonathan Morton, Matt Olson, Tommy Pauly, Tom Petch, Hagen Paul Pfeifer, Kyle Rose, Anthony Sabatini, Michael Scharf, Greg Skinner, Joe Touch, Michael Tüxen, Reji Varghese, Bernie Volz, Tim Wicinski, Lloyd Woodet Alex Zimmermann.

Joe Touch a fourni une aide supplémentaire pour clarifier la description des paramètres de taille de segment et les recommandations de PMTUD/PLPMTUD. Markku Kojo a aidé à rédiger le texte de la section sur le contrôle de l'encombrement TCP.

Ce document comprend le contenu d’errata qui ont été signalés par (classés par ordre chronologique) : Yin Shuming, Bob Braden, Morris M. Keesan, Pei-chun Cheng, Constantin Hagemeier, Vishwas Manral, Mykyta Yevstifeyev, EungJun Yi, Botong Huang, Charles Deng, Merlin Buge.

Adresse de l’éditeur

Wesley M. Eddy
MTI Systems
USA
mél : wes@mti-systems.com