SCTP : Stream Control Transmission Protocol
Protocole et Services

EFORT
http://www.efort.com

Les protocoles usuels de transport de l'information dans les réseaux IP sont TCP (Transmission Control Protocol) et UDP (User Datagram Protocol). Pour répondre aux besoins du transport des protocoles de signalisation téléphonique sur IP notamment dans le contexte NGN (Next Generation Network), l'IETF (Internet Engineering Task Force) a élaboré un protocole spécifique très fiable, SCTP (Stream Control Transmission Protocol) qui est présent au même niveau que TCP et UDP. Ainsi il est possible de remplacer le transport coûteux et complexe des protocoles de signalisation INAP, ISUP, MAP, CAP traditionnellement sur SS7, Q.931 traditionnellement sur LAPD et V5.2 traditionnellement sur LAPv5, par un transport économique sur IP. Comme TCP et UDP ne sont pas assez fiables pour prendre en charge ce transport, SCTP a été défini avec un ensemble de fonctions critiques par rapport aux contraintes de fiabilité et redondance, de sécurité, de rapidité du transport de la signalisation téléphonique. SCTP est aussi considéré comme nouveau transport pour certains protocoles de signalisation et de contrôle du réseau de 4ème génération mobile EPS (Evolved Packet System) tels que S1-AP, X2-AP et DIAMETER.
Le but de ce tutoriel est de présenter les caractéristiques du protocole SCTP, le comparer avec les protocoles de même niveau, à savoir TCP et UDP, et de décrire l'établissement d'association SCTP, le transfert de données SCTP et la libération d'association SCTP.

1 Protocole SCTP

1.1 Caractéristiques de base SCTP

SCTP est un protocole **unicast** et permet l'échange de données en mode bidirectionnel entre deux endpoints SCTP.
SCTP fournit un **transport fiable**, détecte le rejet, la duplication de données ainsi que les données erronées et retransmet les données corrompues. A ce propos, SCTP gère des **temporisateurs plus courts que ceux de TCP** car il s 'agit de transporter des données de signalisation qui ont des contraintes de temps de livraison plus strictes que celles liées aux données classiques.
Alors que dans TCP un flux fait référence à une séquence d'octets, un flux SCTP fait référence à une séquence de messages. SCTP est donc plus simple a interpréter à la réception.
Le nom Stream Control Transmission Protocol découle de la fonction **multi-streaming** fournie par SCTP. Un stream (flot) est un canal logique unidirectionnel permettant l'échange de messages entre terminaisons SCTP. Lors de l'établissement d'une association SCTP, il est nécessaire de spécifier le nombre de streams que comportera cette association. La fonction multi-streaming permet de partitionner les données dans différents streams de telle sorte que la perte d'un message dans un des streams n'ait d'impact sur le transport des données que sur ce stream.
Une des fonctionnalités principales du protocole SCTP est le **multi-homing**, c'est à dire la capacité pour un endpoint SCTP de supporter plusieurs adresses IP. Ceci est un avantage comparé à TCP. Une connexion TCP est définie par une paire d'adresses de transport (Adresse IP + numéro de port TCP). Chaque endpoint d'une association SCTP fournit à l'autre extrémité une liste d'adresses IP avec un unique numéro de port SCTP. L'endpoint

est donc l'extrémité logique du protocole de transport SCTP. Une association SCTP associe toutes les combinaisons d'adresses source et destination entre les deux nœuds impliqués. Chaque endpoint SCTP peut être adressé par un autre endpoint SCTP à travers plusieurs chemins correspondant à plusieurs adresses de transport. La fonctionnalité de multi-homing est utilisée à des fins de redondance et non pour permettre un partage de charge entre différentes routes IP.
L'état de chaque chemin est supervisé par SCTP en ce qui concerne son accessibilité, le délai et le nombre de retransmissions consécutives. La supervision du chemin (path monitoring), l'utilisation d'un chemin alternatif pour des retransmissions et la sélection d'un chemin à partir de son état font de SCTP un protocole plus robuste que TCP lors de défaillances partielles du réseau.

1.2 Comparaison entre SCTP, TCP et UDP

Les principaux avantages de SCTP par rapport à TCP sont (Tableau 1):

- Le support du multihoming qui est directement supporté par SCTP alors qu'avec TCP l'application devra s'appuyer sur plusieurs connexions TCP et gérer le basculement si une connexion TCP est perdue.
- Le support du multi-streaming qui est proposé par SCTP permettant d'émuler plusieurs flots parallèles alors que TCP ne gère qu'un seul flot. Il est possible d'émuler cette fonction au niveau applicatif si TCP est le transport. L'application devra établir plusieurs connexions TCP et répartir la charge sur l'ensemble des connexions.
- La sécurité intrinsèque à SCTP alors que TCP ne fournit aucune sécurité. Avec SCTP il y a authentification au moment de l'établissement de l'association SCTP et par ailleurs chaque paquet transmis par toute entité SCTP émettrice doit contenir une signature qui doit être validée par l'entité SCTP réceptrice..

Caractéristique	**SCTP**	**TCP**	**UDP**
Transfert de données fiable	Oui	Oui	non
Contrôle de congestion	Oui	Oui	non
Découverte MTU	Oui	Oui	non
Multiplexage de message	Oui	Oui	non
Support du Multi-homing	Oui	Non	non
Support du Multi-streaming	Oui	Non	non
Livraison de données désordonnée	Oui	Non	oui
Sécurité	Oui	Non	non
Heartbeat intégré	Oui	Non	non

Tableau 1 : Comparaison entre les trois protocoles de transport SCTP, TCP et UDP

1.3 Multihoming

- Dans la terminologie IP, un nœud de communication ou une machine est appelé "multi-homed" si il peut être adressé par plusieurs adresses. "Multi-homed" signifie que la machine host a été installée avec plusieurs cartes d'interface, chacune ayant une adresse IP.
- Un endpoint SCTP est une entité Emettrice/ Réceptrice logique de paquets SCTP.
- Un endpoint une adresse de transport qui est un couple "adresse IP" et "port SCTP" sur une machine qui possède une adresse IP. Un exemple de endpoint est le couple [192.25.13.26 : 120]. Un endpoint SCTP est dit "Multi-Homed" s'il a été établi sur une machine possédant plusieurs adresses IP et si lui-même agrège plusieurs adresses IP (host multi-homed). Un exemple de endpoint multi-homed est [192.25.13.26, 192.27.14.23, 192.21.13.14:120]
- Puisque SCTP est un protocole de transport fonctionnant en mode connecté, deux endpoints SCTP doivent initier une procédure d'établissement de la communication.

Cette relation de communication est appelée association SCTP. Un exemple d'association SCTP est le couple de endpoints {[192.25.13.26, 192.27.14.23, 192.21.13.14:2905] : [128.120.11.14, 128.120.11.44 :2905]}. On remarquera que les deux endpoints qui constituent l'association sont multi-homed.

1.4 Le paquet SCTP

La PDU (Protocol Data Unit) SCTP est appelée un paquet SCTP. Le paquet SCTP est encapsulé dans un paquet IP, qui est routé à la destination.
Le paquet SCTP est composé d'un en-tête commun et de Chunks. Un Chunk contient soit des données de contrôle soit des données utilisateur (Figure 1).
Plusieurs Chunks peuvent être multiplexés dans un même paquet SCTP sauf dans le cas des Chunks de contrôle INIT, INIT ACK et SHUTDOWN COMPLETE. Ces derniers ne peuvent pas être regroupés avec d'autres chunks dans un même paquet SCTP.
Si un message utilisateur (e.g., message ISUP) ne peut pas être contenu dans un seul paquet SCTP du fait de sa taille, il est possible de fragmenter le message en plusieurs chunks qui seront encapsulés dans différents paquets SCTP.

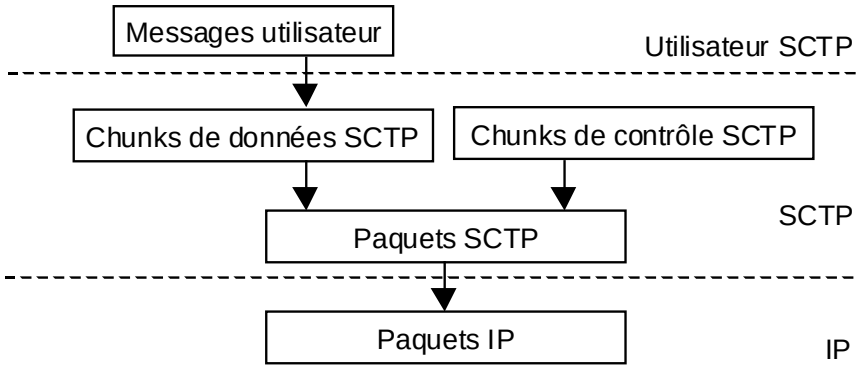

Figure 1 : Chunks SCTP

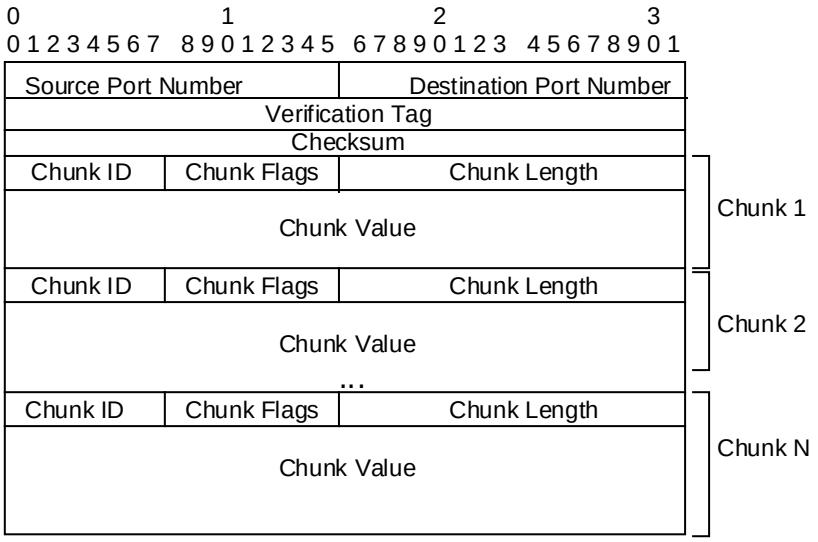

Figure 2 : Format d'un paquet SCTP

L'en-tête SCTP, d'une longueur de 12 octets identifie une association SCTP à travers le même concept de port utilisé par TCP et UDP (Figure 2). Des numéros de port en émission

(2 octets) et réception (2 octets) présents dans l'en-tête combinés aux numéros d'adresses IP en émission et réception (inclus dans l'en-tête du paquet IP) identifient sans ambiguïté les endpoints s'échangeant des paquets SCTP.

Pour la détection des erreurs de transmission, chaque paquet SCTP est protégé par un total de contrôle (checksum) sur 4 octets qui est plus robuste que le checksum TCP ou UDP d'une longueur de 2 octets. Un paquet SCTP dont le checksum est invalide est rejeté. L'en-tête contient enfin une marque de vérification (verification tag) dont le rôle est décrit plus loin. Chaque Chunk débute par un champ chunkID indiquant le type de Chunk afin de distinguer les Chunks de données et les différents Chunks de contrôle. Dans le cas d'un chunk de données, la valeur de ChunkID est égale à 0. Suivent des fanions de Chunk (Chunk Flags), la longueur de Chunk (Chunk Length) nécessaire du fait de la taille variable d'un chunk, et la valeur de Chunk (Chunk Value) qui contient les données utiles du chunk.

Le tableau 2 définit l'ensemble des identificateurs de Chunks (Chunk ID) utilisés afin de déterminer le type de Chunk.

Chunk ID	**Chunk Type**
00000000	Payload Data (DATA)
00000001	Initiation (INIT)
00000010	Initiation Acknowledgement (INIT ACK)
00000011	Selective Acknowledgement (SACK)
00000100	Heartbeat Request (HEARTBEAT)
00000101	Heartbeat Acknowledgement (HEARTBEAT ACK)
00000110	Abort (ABORT)
00000111	Shutdown (SHUTDOWN)
00001000	Shutdown Acknowledgement (SHUTDOWN ACK)
00001001	Operation Error (ERROR)
00001010	State Cookie (COOKIE ECHO)
00001011	Cookie Acknowledgement (COOKIE ACK)
00001100	Reserved for Explicit Congestion Notification Echo (ECNE)
00001101	Reserved for Congestion Window Reduced (CWR)
00001110	Shutdown Complete (SHUTDOWN COMPLETE)
00001111 à 11111111	Réservé par l'IETF

Tableau 2: Types de Chunk SCTP

1.5 Les Chunks SCTP

Le Chunk INIT permet l'initiation d'une association entre deux endpoints SCTP. Comme indiqué plus haut, le paquet SCTP contenant ce Chunk ne peut contenir d'autres Chunks. Le Chunk INIT ACK permet à l'endpoint récepteur du Chunk INIT de confirmer l'initiation de l'association. Comme pour le Chunk INIT, le paquet SCTP encapsulant le Chunk INIT ACK ne contient pas d'autres Chunks.
Le Chunk DATA contient l'information de la couche cliente de SCTP (e.g., M3UA, M2UA, SUA, IUA, V5UA). Le Chunk SACK acquitte la réception d'un ensemble de Chunks DATA et informe éventuellement l'émetteur d'absence de Chunks DATA intermédiaires grâce au numéro de séquence TSN (Transmission Sequence Number) présent dans chaque Chunk DATA.
Le Chunk HEARTBEAT est émis par un endpoint SCTP à un autre endpoint afin de sonder l'accessibilité d'une adresse de transport particulière associée à cet autre endpoint.
Le Chunk HEARTBEAT ACK est la réponse au CHUNK HEARTBEAT. Il est toujours émis à l'adresse IP source du paquet IP contenant le message HEARTBEAT.
Le Chunk ABORT permet l'abandon de l'association en indiquant au récepteur la raison de l'abandon. Tout Chunk de contrôle à l'exception d'INIT, INIT ACK et SHUTDOWN

COMPLETE peut être regroupé avec un Chunk ABORT dans le même paquet SCTP mais doit être placé devant le Chunk ABORT, sinon il est ignoré.
Le Chunk SHUTDOWN permet de libérer proprement l'association. L'entité souhaitant libérer l'association arrête d'émettre des Chunks DATA et doit attendre d'avoir reçu tous les acquittements SACK correspondant aux données déjà émises. Le Chunk SHUTDOWN qui est alors envoyé contient le numéro de séquence du dernier Chunk DATA reçu. Le récepteur répond par un message SHUTDOWN COMPLETE.
Le Chunk ERROR est utilisé afin d'informer un endpoint SCTP d'erreurs en indiquant les causes d'erreur. Par exemple, la réception d'un Chunk avec des paramètres obligatoires invalides ou avec absence de paramètres obligatoires conduisent à l'envoi d'un Chunk ERROR.
Le Chunk COOKIE ECHO est émis par l'entité SCTP qui initie l'association SCTP afin de terminer le processus d'initialisation.
Lorsqu'un endpoint SCTP reçoit un Chunk INIT, il répond par un Chunk INIT ACK contenant un champ *State Cookie*. Ce dernier indique entre autres l'instant de création du champ *State Cookie* et la durée de vie de ce champ. A la réception de ce Chunk, l'entité initiatrice de l'association copie l'information du champ *State Cookie* et l'insère dans un Chunk COOKIE ECHO qui est alors transmis.
Ce Chunk doit être émis avant l'envoi de tout Chunk DATA dans l'association mais peut partager le même paquet SCTP que des Chunks DATA qui suivent. Si le Chunk COOKIE ECHO est reçu sans erreur (avec le même champ *State Cookie* que celui envoyé dans le message INIT ACK), et avant l'expiration de la durée de vie du *State Cookie*, le récepteur du Chunk COOKIE ECHO renvoie un message COOKIE ACK. Dans le cas contraire, un Chunk ERROR est retourné.

2 Fonctionnement du protocole SCTP

2.1 Etablissement d'une association SCTP

L'établissement d'une association entre deux entités SCTP se fait généralement par avance, comparable à la mise en service d'un canal sémaphore dans le réseau SS7. Il requiert l'échange de quatre chunks : INIT, INIT ACK, COOKIE ECHO et COOKIE ACK.

2.1.1 Chunk INIT

Chaque entité SCTP est initialement dans l'état « Closed ». L'entité qui souhaité établir l'association émet un Chunk de contrôle nommé INIT (Figure 3). Le champ *Chunk Flags* a une valeur égale à 0. La valeur du champ *InitateTag* est générée aléatoirement et différente de 0. Le paquet SCTP contenant le Chunk INIT a une étiquette de vérification (Verification Tag) dont la valeur est égale à 0. L'entité réceptrice du Chunk INIT stocke la valeur du champ *InitiateTag*. Le champ Verification Tag de chaque paquet SCTP émis par l'entité réceptrice sur cette association aura pour valeur celle du champ *InitiateTag*.
Le Chunk INIT contient par ailleurs un champ *Advertised Receiver Credit Window (a_rwnd)* qui indique l'espace mémoire en nombre d'octets que l'émetteur a alloué pour l'association qui sera établie. Pendant la durée de vie de l'association, cet espace mémoire ne peut diminuer mais peut augmenter.
Le paramètre *Number of Outbound Streams (OS)* spécifie le nombre de flux unidirectionnels (streams) que l'émetteur souhaite créer dans cette association. La valeur 0 ne peut pas être utilisée.
Le paramètre *Number of Inbound Streams (MIS)* définit le nombre maximum de flux unidirectionnels (streams) que l'émetteur permet au récepteur de créer dans cette association. La valeur 0 ne peut pas être utilisée.

Le paramètre *Initial TSN* indique le numéro de TSN (Transmission Sequence Number) initial que l'émetteur utilisera lors de l'envoi du premier Chunk DATA dans cette association. Le paramètre TSN présent dans chaque Chunk DATA permet de numéroter de façon incrémentale ces Chunks dans le contexte d'une association.
Le Chunk INIT peut par ailleurs contenir un nombre de paramètres optionnels ou de longueur variable (Optional / Variable-Length parameters). Parmi ces paramètres figurent l'adresse IPv4 ou IPv6 de l'émetteur. Combinée au numéro de port SCTP de l'émetteur présent dans le paquet SCTP, cette adresse permet à l'émetteur d'indiquer au récepteur l'adresse de transport qu'il supporte pour cette association.
Plus d'un paramètre d'adresse Ipv4 ou IPv6 peut être inclus dans un Chunk INIT si l'entité SCTP émettrice est de type « multi-homed »

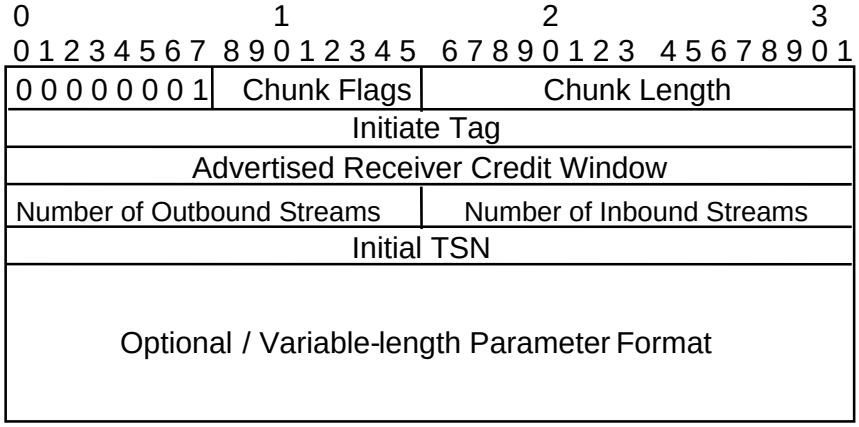

Figure 3: Chunk INIT

2.1.2 Chunk INIT ACK

A la réception du Chunk INIT, l'entité SCTP réceptrice retourne un Chunk INIT ACK (Figure 4). Ce dernier a un format similaire à celui du Chunk INIT. Il rajoute deux paramètres, Cookie State et Unrecognized.
Le récepteur rappelle les valeurs des paramètres qu'il a reçu dans le Chunk INIT.
Il ne lui est pas possible de négocier une valeur du paramètre *Number of Inbound Streams* (MIS) supérieure à celle reçue.
La valeur du paramètre *Initial TSN* indique la valeur de TSN que l'émetteur du message INIT ACK utilisera lors de l'envoi de son premier Chunk DATA dans cette association.

Le Chunk INIT ACK contient un nombre de paramètres de longueur variable (Optional / Variable-Length parameters) et éventuellement des paramètres optionnels.
Le seul paramètre obligatoire dont la taille est variable est State Cookie.
Parmi les paramètres optionnels figurent la ou les adresses IPv4 et/ou IPv6 de l'entité réceptrice qui seront utilisées dans cette association, ainsi que Unrecognized. Ce dernier sera inclus dans le Chunk INIT ACK lorsque l'entité réceptrice du Chunk INIT ne reconnaît par un des paramètres.

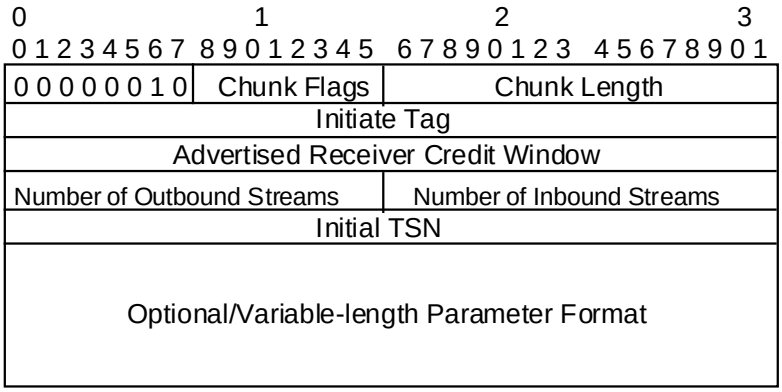

Figure 4 : Chunk INIT ACK

2.1.3 Chunk COOKIE ECHO

A la réception du Chunk INIT ACK, l'initiateur de l'association renvoie un Chunk COOKIE ECHO afin de finaliser l'établissement de l'association (Figure 5). Ce Chunk doit précéder l'envoi de tout Chunk DATA mais peut partager le même paquet SCTP que des Chunks DATA en étant le premier Chunk du paquet. La valeur du champ Chunk Flags est égale à 0 et ignorée à la réception. Le Chunk COOKIE ECHO doit contenir le paramètre Cookie dont la valeur est celle du paramètre State Cookie reçu dans le Chunk INIT ACK.

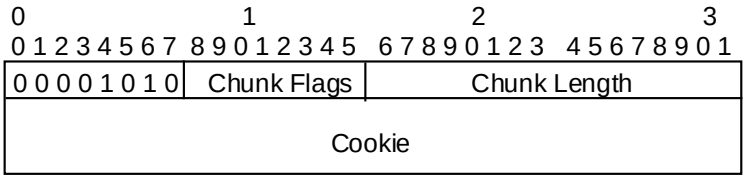

Figure 5 : Chunk COOKIE ECHO

2.1.4 Chunk COOKIE ACK

A la réception du Chunk COOKIE ECHO, un Chunk COOKIE ACK est retourné. La valeur du champ Chunk Flags est positionnée à 0 alors que la longueur du Chunk (Chunk Length) est égale à 4 (Figure 6). Ce Chunk doit précéder tout Chunk DATA envoyé par l'émetteur de ce Chunk COOKIE ECHO, et avant tout message SACK acquittant des messages DATA reçus. Par contre, il peut partager le même paquet SCTP que ces Chunks DATA et SACK, mais en première position dans ce paquet.

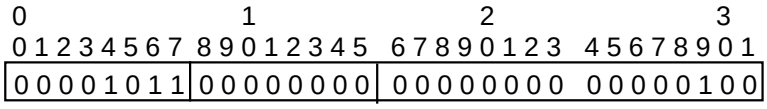

Figure 6 : Chunk COOKIE ACK

2.2 Transfert de données SCTP

2.2.1 Le Chunk DATA

Le Chunk DATA est utilisé afin de transporter les données provenant de la couche cliente (Figure 7).

Le bit *U* (Unordered) s'il est positionné à 1 indique que les données doivent être délivrées par SCTP à la couche cliente réceptrice, dans l'ordre reçu.
Il est possible de fragmenter un message utilisateur dont la taille est supérieure à celle du paquet SCTP. Les bits *B* et *E* sont utilisés afin d'informer le récepteur de cette fragmentation (Tableau 3).
Le bit B (Beginning) s'il est positionné à la valeur 1 indique le premier fragment du message utilisateur. Le bit E (end) mis à la valeur 1 précise qu'il s'agit du dernier fragment du message. Pour un message non fragmenté, les bits B et E ont pour valeur 1.

Bit B	Bit E	Signification
1	0	Première partie d'un message fragmenté
0	0	Partie intermédiaire d'un message fragmenté
0	1	Dernière partie d'un message fragmenté
1	1	Message non fragmenté

Tableau 3 : Signification des bits B et E

Lorsqu'un message utilisateur est fragmenté en plusieurs Chunks DATA, le champ TSNs (Transmission Sequence Number) est utilisé afin de réassembler le message. Ce champ TSN identifie le message utilisateur dans le contexte d'une association indépendamment d'un stream particulier et est incrémenté à chaque envoi d'un Chunk DATA modulo $2^{32} - 1$.
Le champ *Stream Identifier* (S) indique le flux (stream) auquel appartient ce Chunk DATA.

Le champ *Stream Sequence Number n* indique la position de ce Chunk DATA dans le flux.
Lorsqu'un message utilisateur est fragmenté par SCTP, toutes les parties du message encapsulé dans des Chunks DATA ont le champ SSN positionné à la même valeur.

Le champ *Payload Protocol Identifier* représente un identificateur de protocole de la couche cliente. La valeur associée est passée par la couche cliente émettrice à SCTP et retournée à la couche cliente réceptrice afin de lui permettre d'identifier le type d'information transportée dans ce DATA Chunk.

Le champ *User Data* contient le message utilisateur sur un nombre d'octets multiple de 4. Si la longueur du message utilisateur n'est pas un multiple de 4, des octets de bourrage (au maximum 3) pourront alors être insérés dont la valeur sera « 00000000 ».

Le champ *Length* indique la longueur du Chunk DATA en octets, à partir du premier champ (Chunk Type dont la valeur est 00000000) jusqu'à la fin du champ User Data en excluant les octets de bourrage. Un Chunk DATA sans données utilisateur à un champ Length positionné à la valeur 16 (00010000).

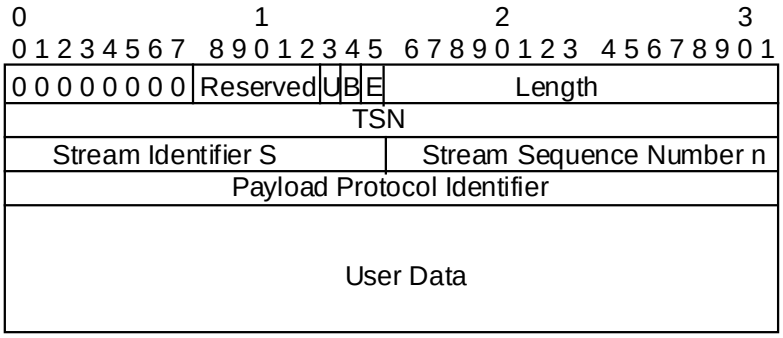

Figure 7 : Chunk DATA

Prenons l'exemple d'une fragmentation d'un message utilisateur SCTP, provenant de la couche M3UA (MTP3 User Adaptation). Le message doit être délivré sur le stream 5 dans l'ordre d'émission (Figure 8).
Ce message est décomposé en quatre fragments. Les quatre fragments sont des chunks DATA SCTP. Ils ont leur champ U positionné à 0. Les données doivent être réordonnées à l'arrivée avant d'être délivrées à la couche supérieure.
Les bit B/E sont positionnées à 1 0 pour le premier fragment. Il s'agit du premier fragment mais pas du dernier. D'autres fragments suivent. Les second et troisième fragments ont leur bit B/E positionnés à 0 0. Ils ne sont ni le premier ni le dernier fragment du message. Les bits B et E du quatrième fragment sont mis à 0 1 puisqu'il ne s'agit pas du premier mais du dernier fragment. Tous les fragments doivent être émis sur le stream numéro 5. Le numéro de séquence est le même pour tous les fragments puisqu'il s'agit de fragments d'un même message. Par contre, le champ TSN est incrémenté de 1 à chaque nouveau chunk émis sur l'association SCTP. Enfin le payload Id indique le type de contenu des chunks DATA. Dans l'exemple le contenu des chunks DATA provient de la couche M3UA dont le type est 3.

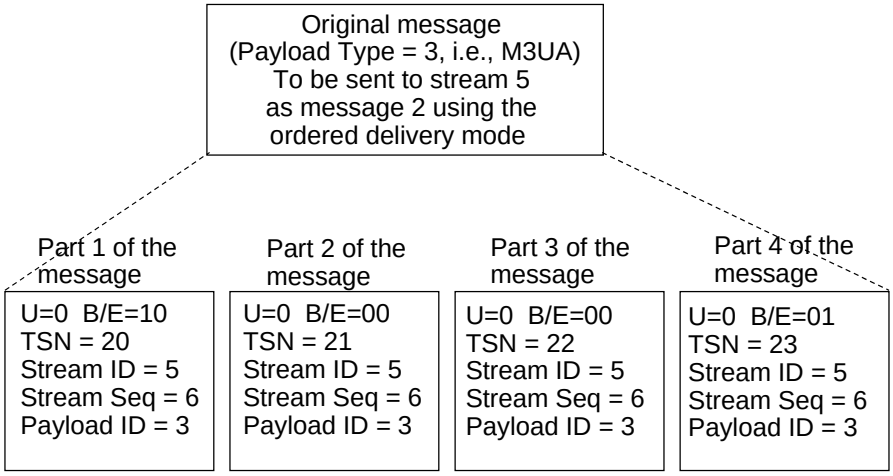

Figure 8 : Fragmentation SCTP

2.2.2 Chunk SACK

Le Chunk SACK est émis par une entité SCTP afin d'acquitter les Chunks DATA reçus de l'autre entité SCTP et de l'informer d'éventuels « gaps » ou absences dans les Chunks DATA consécutifs reçus. Sa structure est présentée à la Figure 9.

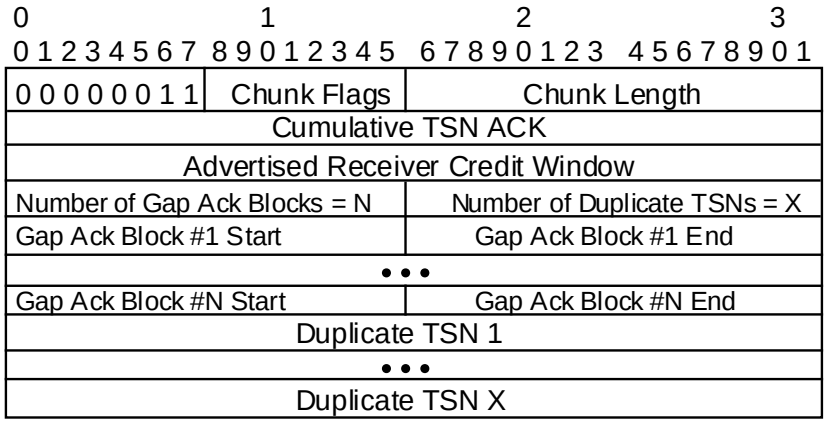

Figure 9 : Chunk SACK

Afin d'illustrer l'utilisation du Chunk SACK considérons l'exemple suivant tiré du RFC SCTP : L'émetteur transmet 8 Chunks DATA consécutifs numérotés de 10 à 17. Le récepteur reçoit les Chunks DATA numérotés 10, 11, 12, 14, 15 et 17 par leur TSN (Transmission Sequence Number). Les Chunks DATA 13 et 16 sont manquants. Les Chunks DATA 12 et 15 sont reçus quant à eux deux fois (Figure 10).

Le message SACK retourné par l'entité réceptrice des Chunks DATA contient les champs suivants (Figure 11):

- Le champ *Cumulative TSN ACK* indique le TSN du dernier Chunk DATA reçu en séquence avant un « gap ». Dans notre exemple, la valeur de ce champs sera 12.
- Le champ *Advertised Receiver Credit Window (a_rwnd)* indique l'espace mémoire en réception mis à jour par l'émetteur du Chunk SACK. Cela permet au récepteur du Chunk SACK de gérer la quantité de données émises afin de ne pas générer de débordement de tampons et donc de pertes de données.
- Le champ *Number of Gap Ack Blocks* indique le nombre de « gaps » présents dans la séquence à acquitter. L'exemple en présente 2.
- Le champ *Number of Duplicate TSNs* précise le nombre de TSNs reçus plus d'une fois. Dans l'exemple considéré, ce champ est positionné à la valeur 2 correspondant à 2 TSNs, numérotés 12 et 15.
- Le champ *Gap Ack Block # 1 Start* indique la valeur 2 qui correspond à la différence entre Cumulative TSN Ack qui vaut 12 et la plus petite valeur TSN reçue après le premier « gap » qui vaut 14.
- Le champ Gap Ack Block # 1 End indique la valeur 3 qui correspond à la différence entre Cumulative TSN Ack qui vaut 12 et la plus grande valeur TSN reçue après le premier « gap » qui vaut 15.
- Le champ *Gap Ack Block # 2 Start* indique la valeur 5 qui correspond à la différence entre Cumulative TSN Ack qui vaut 12 et la plus petite valeur TSN reçue après le second « gap » qui vaut 17.
- Le champ Gap Ack Block # 1 End indique la valeur 5 qui correspond à la différence entre Cumulative TSN Ack qui vaut 12 et la plus grande valeur TSN reçue après le second « gap » qui vaut 15.
- Le champ *Duplicate TSNs* indique les TSNs reçus plus d'une fois depuis l'envoi du dernier SACK. Chaque fois qu'un même TSN est délivré plus d'une fois à un récepteur, ce dernier le rajoute à une liste de TSNs dupliqués. Le compteur Number of Duplicate TSNs est réinitialisé à 0 après l'envoi de tout Chunk SACK. Si un TSN est reçu trois fois, alors la valeur de ce TSN est répétée deux fois dans le champ « Duplicate TSN » du Chunk SACK.

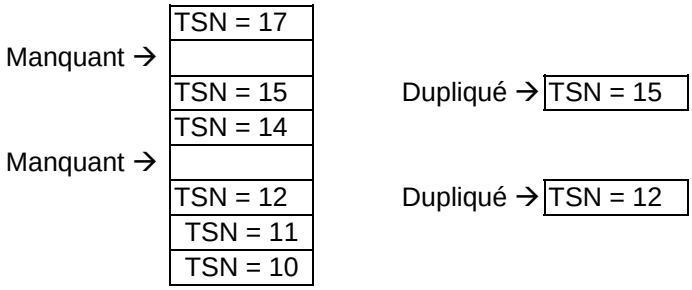

Figure 10 : Exemple avec données manquantes et données dupliquées

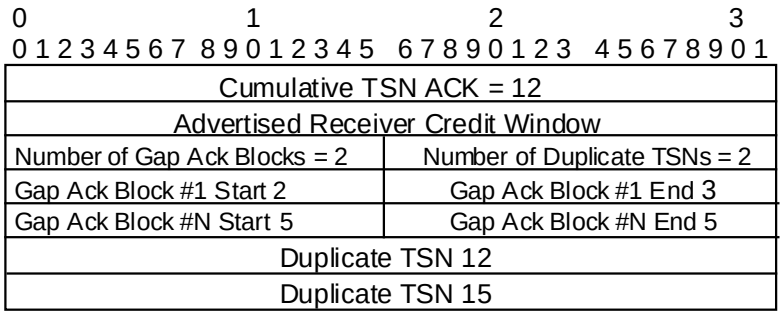

Figure 11 : Chunk SACK de l'exemple

2.3 Libération d'une association SCTP

L'association est libérée proprement par un chunk SHUTDOWN ou abandonnée brutalement par un chunk ABORT.
Quand un utilisateur/application désire fermer le socket SCTP proprement, il utilise le chunk SHUTDOWN. L'endpoint SCTP envoie alors toutes les données encore dans ses mémoires tampon, et ensuite émet le chunk SHUTDOWN. Quand le destinataire reçoit le chunk SHUTDOWN, il arrête d' accepter des données provenant de l'application et cesse d'envoyer des données. Une fois obtenus tous les SACK pour les données, il enverra un chunk SHUTDOWN ACK, et une fois que le côté qui libère l'association a reçu ce chunk, il répondra par un chunk SHUTDOWN COMPLETE. L'association est dorénavant complètement libérée.

Une autre façon de terminer l'association est d'utiliser le chunk ABORT. Mais c'est un moyen plus brutal de terminer une association SCTP. Quand une des deux parties désire terminer une association SCTP instantanément, elle émet ce chunk ABORT. Toutes les données dans les tampons sont alors supprimées et l'association terminée. Le destinataire fait de même après vérification du chunk ABORT.

2.3.1 Chunk SHUTDOWN

Le chunk SHUTDOWN apparaît quand un des endpoints d'une association désire fermer l'association en cours. L'endpoint qui le transmet doit vider tous ses tampons avant d'expédier le chunk SHUTDOWN, et ne doit pas envoyer d'autres chunks DATA par la suite. Le destinataire doit également vider ses tampons d'émission et ensuite expédier le chunk SHUTDOWN ACK correspondant.
Le chunk SHUTDOWN contient les champs suivants (Figure 12):

- Le champ Chunk Flags dont la valeur est positionnée à 0
- Le champ Chunk Length qui indique la longueur du Chunk et dont la valeur est égale à 8
- Le champ *Cumulative TSN ACK* indique le TSN du dernier Chunk DATA reçu en séquence avant un « gap ».

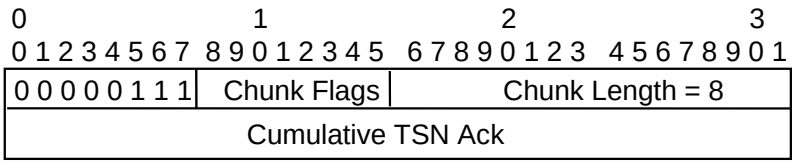

Figure 12 : Chunk SHUTDOWN

2.3.2 Chunk SHUTDOWN ACK

Le chunk SHUTDOWN ACK est utilisé pour accuser réception d'un bloc SHUTDOWN reçu. Avant que le bloc SHUTDOWN ACK soit envoyé, toutes les données dans les tampons d'envoi doivent être expédiées, les tampons ne doivent plus accepter aucune donnée provenant de l'application.
Le chunk SHUTDOWN ACK contient les champs suivants (Figure 13):

- Le champ Chunk Flags dont la valeur est positionnée à 0
- Le champ Chunk Length qui indique la longueur du Chunk et dont la valeur est égale à 4

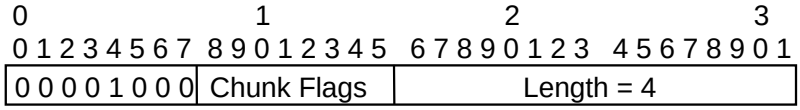

Figure 13 : Chunk SHUTDOWN ACK

2.3.3 Chunk SHUTDOWN COMPLETE

Le chunk SHUTDOWN COMPLETE est envoyé, par l'expéditeur du SHUTDOWN, en réponse au chunk SHUTDOWN ACK. Il est expédié pour accuser réception que l'association est totalement fermée.
Le chunk SHUTDOWN COMPLETE contient les champs suivants (Figure 14):

- Le champ Chunk Flags dont la valeur est positionnée à 0
- Le champ Chunk Length qui indique la longueur du Chunk et dont la valeur est égale à 4.

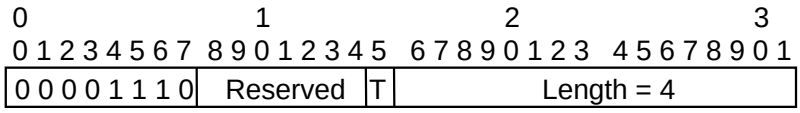

Figure 14 : Chunk SHUTDOWN COMPLETE

2.3.4 Chunk ABORT

Le chunk ABORT est utilisé pour abandonner une association (Figure 15).
Parmi les causes d'erreur qui conduisent à l'envoi du chunk ABORT figurent :
Out of resource : L'endpoint émettant l'ABORT n'est pas en mesure de maintenir l'association à cause de problèmes de ressources internes.
Unresolvable Address Error : L'endpoint a reçu une adresse de hostname de l'autre endpoint pendant la phase d'initialisation de l'association. Or cette adresse de hostname ne peut pas être résolue par le DNS en une adresse IP.
Invalid Mandatory Parameter : Il est possible qu'un des chunks reçus présente un paramètre obligatoire invalide. Par exemple, l'émetteur du chunk INIT a positionné le champ "nombre de streams" à la valeur 0 qui n'est pas une valeur acceptable. Le récepteur du chunk INIT l'acquitte alors par un ABORT en précisant la cause de l'erreur.
No User Data Error : Si un endpoint SCTP émet un chunk DATA sans données utilisateur, le récepteur est dans l'obligation de retourner un chunk ABORT et de supprimer l'association.

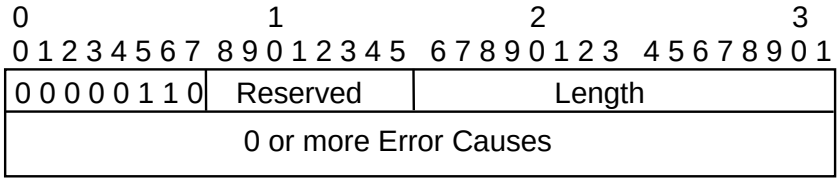

Figure 15 : Chunk ABORT

Références

RFC 4960, R. Stewart et al., « Stream Control Transmission Protocol », Sept 2007.
RFC 3286, L.Ong et al. “An Introduction to the Stream Control Transmission Protocol (SCTP)”, May 2002.
Randall R. Stewart, Qiaobing Xie, “Stream Control Transmission Protocol : A reference Guide”, Addison Wesley, Dec 2001.