Groupe de travail Réseau
Request for Comments : 2371
Catégorie : En cours de normalisation
Traduction Claude Brière de L'Isle

J. Lyon, Microsoft
K. Evans & J. Klein, Tandem Computers
juillet 1998

Protocole Internet de transaction (TIP) version 3.0

Statut du présent mémoire
Le présent document spécifie un protocole de l'Internet en cours de normalisation pour la communauté de l'Internet, et appelle à des discussions et suggestions pour son amélioration. Prière de se référer à l'édition en cours des "Normes officielles des protocoles de l'Internet" (STD 1) pour connaître l'état de la normalisation et le statut de ce protocole. La distribution du présent mémoire n'est soumise à aucune restriction.

Résumé
Dans de nombreuses applications où différents nœuds coopèrent à certains travaux, il y a un besoin de garantie que le travail se fasse de façon atomique ; c'est à dire que chaque nœud doit arriver à la même conclusion sur l'achèvement du travail, même en présence de défaillances. Le présent document propose un protocole simple, facile à mettre en œuvre pour réaliser cet objectif.

Table des matières

1. Introduction

La méthode standard pour satisfaire à une obligation de réaliser un traitement atomique est le protocole d'engagement à deux phases ; voir dans [1] une introduction à l'engagement de traitement atomique et les protocoles d'engagement à deux phases.

De nombreux protocoles d'engagement à deux phases ont été mis en œuvre au fil des ans. Cependant, aucun d'eux n'est devenu d'utilisation courante dans l'Internet, principalement du fait de leur complexité. La plus grande partie de cette complexité vient du fait que le protocole d'engagement à deux phases est lié à un protocole spécifique de communication de programme à programme, et que ce protocole fonctionne par dessus une très grosse infrastructure.

Le présent mémoire propose un protocole très simple d'engagement à deux phases. Il réalise cette simplicité en spécifiant seulement comment les différents nœuds se mettent d'accord sur le résultat d'une transaction ; cela permet (et même exige) que le sujet qui fait l'objet de l'accord des nœuds soit communiqué via d'autres protocoles. En faisant ainsi, on évite tous les problèmes qui se rapportent à la sémantique de communication d'application et à la représentation des données (pour n'en nommer que quelques uns). Indépendamment du protocole de communication d'application, un gestionnaire de transaction peut utiliser le protocole de sécurité de la couche Transport [3] pour authentifier les autres gestionnaires de transaction et chiffrer les messages.

Il est envisagé que ce protocole soit principalement utilisé pour un gestionnaire de transaction sur un nœud Internet pour communiquer avec un gestionnaire de transaction sur un autre nœud. Bien qu'il soit possible d'utiliser ce protocole pour que les programmes d'application et/ou les gestionnaires de ressources parlent aux gestionnaires de transaction, cette communication est normalement intra-nœud, et la plupart des gestionnaires de transaction ont déjà des interfaces très adéquates pour cette tâche.

Bien qu'on ne s'attende pas à ce que ce protocole remplace ceux qui existent, on espère qu'il sera relativement facile à de nombreux gestionnaires de transaction hétérogènes existants de mettre en œuvre ce protocole pour leur communication des uns avec les autres.

On trouvera des informations supplémentaires concernant le protocole TIP dans [5].

2. Exemple d'usage

Le panier d'achat électronique est aujourd'hui une métaphore courante de nombreux sites d'achats électroniques. Les consommateurs naviguent à travers un catalogue électronique, choisissent des biens et les placent dans un panier d'achats électronique. Les serveurs HTTP [2] fournissent divers moyens allant du codage d'URL au mouchard de contexte pour garder la trace du contexte du client (par exemple, le panier d'achats d'un consommateur) et le reprendre lors des prochaines demandes du consommateur.

Une fois qu'un consommateur a terminé ses achats, il peut décider de s'engager sur sa sélection et passer les ordres associés. La plupart des ordres peuvent n'avoir pas de relation les uns avec les autres, sauf d'être exécutés au titre de la même transaction d'achats ; d'autres peuvent dépendre les uns des autres (par exemple, si ils sont faits au titre d'une offre spéciale). Sans considération de ces détails, un consommateur va s'attendre à ce que tous les ordres aient été bien passés à réception d'un accusé de réception positif. Les sites d'achat électronique d'aujourd'hui doivent mettre en œuvre leurs propres protocoles spéciaux pour coordonner de telles passations de tous les ordres. Cette programmation est particulièrement complexe lorsque les ordres sont passés à travers plusieurs sites d'achats électroniques. Cette complexité limite l'utilité potentielle des applications internet, et en restreint le développement. Le protocole décrit dans le présent document vise à fournir un standard pour les serveurs Internet pour réaliser l'accord sur une unité de travail partagé (par exemple, la passation d'ordres dans un panier d'achats électronique). Le serveur (par exemple, un programme CGI) qui passe l'ordre peut vouloir commencer une transaction en invoquant son gestionnaire de transaction local, et demander aux autres serveurs participant au travail de se joindre à la transaction. Le serveur qui passe les ordres passe une référence à la transaction sous forme de données d'utilisateur sur les demandes HTTP aux autres serveurs. Les autres serveurs invoquent leurs gestionnaires de transaction pour commencer une transaction locale et leur demander de se joindre à la transaction distante en utilisant le protocole défini dans le présent document. Une fois que tous les ordres sont passés, l'exécution du protocole d'engagement à deux phases est déléguée aux gestionnaires de transaction impliqués. Si la transaction engage, tous les ordres sont bien passés et le consommateur obtient un accusé de réception positif. Si la transaction s'interrompt, aucun ordre n'est passé et le consommateur sera informé du problème.

La prise en charge de transaction simplifie considérablement la programmation de ces applications car le traitement des exceptions et la récupération des défaillances sont délégués à un composant particulier. Les utilisateurs finaux aussi ne sont pas abandonnés en face des conséquences d'un succès seulement partiel. Bien que cet exemple montre comment le

protocole peut être utilisé par des serveurs HTTP, les applications peuvent utiliser le protocole quand elles accèdent à une base de données distante (par exemple, via ODBC) ou en invoquant des services distants en utilisant d'autres protocoles déjà existants (par exemple, RPC). Le protocole rend plus facile aux applications dans un réseau hétérogène de participer à la même transaction, même si elles utilisent des protocoles de communication différents.

3. Transactions

"Transaction" est le terme donné au modèle de programmation par lequel le travail de calcul effectué a une sémantique atomique. C'est-à-dire que tout le travail se termine avec succès et les changements sont rendus permanents (la transaction engage) ou si un travail n'est pas réussi, les changements sont défaits (la transaction s'interrompt). Le travail comportant une transaction (unité de travail) est défini par l'application.

4. Connexions

Le protocole Internet de transaction (TIP) exige un transport de flux ordonnés fiable avec de faibles coûts d'établissement de connexion. Dans un environnement Internet (IP), TIP fonctionne sur TCP, utilisant facultativement TLS pour fournir une connexion sûre et authentifiée, et utilisant facultativement un protocole pour multiplexer des connexions légères sur la même connexion TCP ou TLS.

Les gestionnaires de transaction qui partagent les transactions établissent une connexion TCP (et facultativement TLS). Le protocole utilise une connexion différente pour chaque transaction simultanée partagée par deux gestionnaires de transaction. Après la fin d'une transaction, la connexion peut être réutilisée par une transaction différente.

Facultativement, au lieu d'associer une connexion TCP ou TLS à une seule transaction, deux gestionnaires de transaction peuvent se mettre d'accord sur un protocole pour multiplexer des connexions légères sur la même connexion TCP ou TLS, et associer chaque transaction simultanée à une connexion légère séparée. L'utilisation de connexions légères réduit la latence et la consommation des ressources associées à l'exécution de transactions simultanées. Des techniques similaires à celles décrites ici sont largement utilisées par les systèmes existants de traitement de transactions. Voir à l'Appendice A un exemple d'un tel protocole.

Noter que bien que le protocole TIP lui-même ne soit décrit qu'en termes de TCP et TLS, il n'y a rien qui empêche l'utilisation de TIP avec d'autres protocoles de transport. Cependant, il appartient aux mises en œuvre de s'assurer que le transport choisi fournit une sémantique équivalente à celle de TCP, et de transposer de façon appropriée le protocole TIP.

Dans le présent document, les termes "connexion" ou "connexion TCP" peuvent se référer à une connexion TIP TCP, à une connexion TIP TLS, ou à une connexion TIP multiplexée (sur TCP ou TLS). Il n'y a pas de différence car le comportement est le même dans les deux cas. Lorsque il y a des différences de comportement entre les types de connexion cela est mentionné explicitement.

5. Identifiants de transaction

Malheureusement, il n'y a pas une seule norme acceptée mondialement sur le format d'un identifiant de transaction ; il y a plusieurs normes et formats propriétaires. Les formats admis pour un identifiant de transaction sont décrits plus loin dans la section 8 "Localisateurs de ressource universels TIP". Un gestionnaire de transaction peut transposer ses identifiants de transaction internes dans ce format TIP de toute manière qui lui paraît convenir. De plus, chaque partie dans une relation supérieur/subordonné doit allouer son propre identifiant à la transaction ; ces identifiants sont échangés lorsque la relation est établie pour la première fois. Donc, un gestionnaire de transaction va utiliser son propre format d'identifiant de transaction en interne, mais il doit se souvenir d'un identifiant de transaction étranger pour chaque relation supérieur/subordonné dans laquelle il est impliqué.

6. Transactions poussées ou tirées

Supposons qu'un programme sur le nœud "A" ait créé une transaction, et veuille qu'un programme sur le nœud "B" fasse un certain travail au titre de cette transaction. Il y a deux moyens classiques pour faire cela, qu'on appelle le modèle "poussé" et le modèle "tiré".

Dans le modèle "poussé", le programme sur A demande d'abord à son gestionnaire de transaction d'exporter la transaction

au nœud B. Le gestionnaire de transaction (TM, *Transaction Manager*) de A envoie un message au TM de B pour lui demander d'instancier la transaction comme subordonné de A, et retourne son nom pour la transaction. Le programme sur A envoie alors un message à son homologue sur B avec l'ordre de "Faire un travail, et l'incorporer à la transaction que votre gestionnaire de transaction connaît déjà son le nom de ...". Parce que le TM de A sait qu'il a envoyé la transaction au TM de B, le TM de A sait impliquer le TM de B dans le processus d'engagement en deux phases.

Dans le modèle "tiré", le programme sur A envoie simplement un message à B avec l'ordre de "Faire un certain travail, et l'incorporer à la transaction que mon TM connaît sous le nom de ...". Le programme sur B demande à son TM de s'enrôler dans la transaction. À ce moment, le TM de B va "tirer" la transaction de A. Par suite de ce tirage, le TM de A sait impliquer le TM de B dans le processus d'engagement en deux phases.

Le protocole décrit ici prend en charge les deux modèles "poussé" et "tiré".

7. Identification de gestionnaire de transaction TIP et établissement de connexion

Pour que les gestionnaires de transaction TIP se connectent, ils doivent être capables de s'identifier et se localiser l'un l'autre. Les informations nécessaires pour ce faire sont décrites par l'adresse du gestionnaire de transaction TIP.

[La présente spécification ne prescrit pas la façon dont les gestionnaires de transaction TIP obtiennent initialement l'adresse du gestionnaire de transaction (ce qui sera probablement via un mécanisme de configuration spécifique de la mise en œuvre).]

Les adresses de gestionnaire de transaction TIP sont de la forme :

<accès_d'hôte><chemin>

Le composant <accès_d'hôte> comporte :

<hôte>[:<accès>]

où <hôte> est soit un <nom dns> soit une <adresse ip>; et <accès> est un nombre décimal qui spécifie l'accès auquel le gestionnaire de transaction (ou son mandataire) est à l'écoute des demandes d'établissement de connexions TIP. Si le numéro d'accès est omis, on utilise le numéro d'accès TIP standard (3372).

Un <nom dns> est un nom standard, acceptable pour le service des noms de domaines. Il doit être suffisamment qualifié pour être utile au receveur de la commande.

Une <adresse ip> est une adresse IP, sous la forme usuelle : quatre nombres décimaux séparés chacun par un caractère point.

Le composant <accès d'hôte> définit la portée (locale) du composant <chemin>.

Le composant <chemin> de l'adresse du gestionnaire de transaction contient des données qui identifient le gestionnaire de transaction TIP spécifique, à la localisation définie par <accès d'hôte>.

Le composant <chemin> prend la forme :

"/" [segments_de_chemin]

segments_de_chemin = segment *("/" segment)
segment = *pchar *(";" param)
param = *pchar
pchar = nonréservé | échappé | ":" | "@" | "&" | "=" | "+"
nonréservé = octets de caractère ASCII avec des valeurs dans la gamme (inclusive) : 48-57, 65-90, 97-122 | "$" | "-" | "_" |
"." | "!" | "~" | "*" | "'" | "(" | ")" | ","
échappé = "%" hex hex
hex = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" | "A" | "B" | "C" | "D" | "E" | "F" | "a" | "b" | "c" | "d" | "e" | "f"

Le composant <chemin> peut consister en une séquence de segments de chemin séparés par un seul caractère barre oblique "/". Au sein d'un segment de chemin, les caractères "/", ";", "=", et "?" sont réservés. Chaque segment de chemin peut inclure une séquence de paramètres, indiqués par le caractère point-virgule ";". Les paramètres ne sont pas significatifs

pour l'analyse des références relatives.

[Il est entendu que la forme de l'adresse du gestionnaire de transaction suit le schéma proposé pour les identifiants de ressource uniformes (URI) [8].]

L'adresse du gestionnaire de transaction TIP fournit donc à l'initiateur de la connexion (le principal) l'identifiant de point d'extrémité à utiliser pour la connexion TCP (<accès d'hôte>) et au receveur de la connexion (le secondaire) le chemin à utiliser pour localiser le gestionnaire de transaction TIP spécifique (<chemin>). Ce sont toutes les informations requises pour que soit établie la connexion entre les gestionnaires de transaction TIP principal et secondaire.

Après l'établissement d'une connexion, la partie principale produit une commande IDENTIFY. Cette commande comporte comme paramètres deux adresses de gestionnaire de transaction : l'adresse du gestionnaire de transaction principal, et l'adresse du gestionnaire de transaction secondaire.

L'adresse du gestionnaire de transaction principal identifie le gestionnaire de transaction TIP qui a initié la connexion. Cette information est requise dans certains cas après une défaillance de la connexion, lorsque une des parties à la connexion doit rétablir une nouvelle connexion avec l'autre partie afin de d'achever le protocole d'engagement à deux phases. Si la partie principale a besoin de rétablir la connexion, la tâche est aisée : une connexion est établie de la même manière que l'a été la connexion d'origine. Cependant, si la partie secondaire a besoin de rétablir la connexion, elle doit savoir comment contacter l'initiateur de la connexion originale. Cette information est fournie au secondaire via l'adresse de gestionnaire de transaction du principal sur la commande IDENTIFY. Si une adresse de gestionnaire de transaction de principal n'est pas fournie, la partie principale ne doit pas effectuer d'action qui exigerait qu'une connexion soit rétablie (par exemple, effectuer des actions de récupération).

L'adresse de gestionnaire de transaction du secondaire identifie le gestionnaire de transaction TIP receveur. Dans le cas de communication TIP via des serveurs mandataires intermédiaires, cet URL peut être utilisé par les serveurs mandataires pour identifier correctement et se connecter au gestionnaire de transaction TIP requis.

8. Localisateurs de ressource uniformes TIP

Les transactions et les gestionnaires de transaction sont des ressources associées au protocole TIP. Les gestionnaires de transaction et les transactions sont localisés en utilisant le schéma d'adresse du gestionnaire de transaction. Une fois qu'une connexion a été établie, les commandes TIP peuvent être envoyées pour fonctionner sur les transactions associées aux gestionnaires de transaction respectifs.

Les applications qui veulent tirer une transaction d'un nœud distant doivent fournir une référence à la transaction distante qui permette au gestionnaire de transaction local (c'est-à-dire au gestionnaire de transaction qui tire la transaction) de se connecter au gestionnaire de transaction distant et d'identifier cette transaction particulière. Les applications qui veulent pousser une transaction sur un nœud distant doivent fournir une référence au gestionnaire de transaction distant (c'est-à-dire le gestionnaire de transaction auquel la transaction est à pousser) qui permette au gestionnaire de transaction local de localiser le gestionnaire de transaction distant. Le protocole TIP définit un schéma d'URL [4] qui permet aux applications et aux gestionnaires de transaction d'échanger des références aux gestionnaires de transaction et aux transactions.

Un URL TIP prend la forme suivante : tip://<adresse du gestionnaire de transaction>?<chaîne de transaction>

où <adresse du gestionnaire de transaction> identifie le gestionnaire de transaction TIP (comme défini à la Section 7 ci-dessus) ; et <chaîne de transaction> spécifie un identifiant de transaction, qui peut prendre une des deux formes (standard ou non standard) :

a) "urn:" <NID> ":" <NSS>

Un identifiant de transaction standard se conforme à la norme Internet proposée pour les noms de ressource uniformes (URN) telle que spécifiée dans la RFC2141 ; où <NID> est l'identifiant d'espace de noms, et <NSS> est la chaîne spécifique de l'espace de noms (NSS, *Namespace Specific String*). L'identifiant d'espace de noms détermine l'interprétation syntaxique de la chaîne spécifique de l'espace de noms. La chaîne spécifique de l'espace de noms est une séquence de caractères représentant un identifiant de transaction (tel que défini par <NID>). Les règles pour le contenu de ces champs sont spécifiées dans [6] (caractères valides, codages, etc.).

Ce format de <chaîne de transaction> peut être utilisé pour exprimer les identifiants de transaction globaux en termes de représentations standard. Des exemples pour <NID> pourraient être <iso> ou <xopen>.

Par exemple, tip://123.123.123.123/?urn:xopen:xid

Noter que les identifiants d'espaces de noms requièrent l'enregistrement. Voir dans [7] les détails sur la façon de le faire.

b) <identifiant de transaction>

Une séquence de caractères ASCII imprimables (octets avec des valeurs dans la gamme de 32 à 126 inclus (en excluant ":") représentant un identifiant de transaction. Dans ce cas non standard, c'est la combinaison de <adresse du gestionnaire de transaction> et de <identifiant de transaction> qui assure l'unicité mondiale. Par exemple,

tip://123.123.123.123/?transid1

Pour créer un URL TIP non standard à partir d'un identifiant de transaction, on remplace d'abord tous les caractères réservés dans l'identifiant de transaction par leur séquence d'échappement équivalente, puis en insérant l'adresse de gestionnaire de transaction appropriée. Si l'identifiant de transaction est de votre création, insérer votre propre adresse de gestionnaire de transaction. Si c'est un identifiant de transaction que vous avez reçu sur une connexion TIP que vous avez initiée, utilisez l'adresse de gestionnaire de transaction secondaire qui a été envoyée dans la commande IDENTIFY. Si l'identifiant de transaction a été reçu sur une connexion TIP que vous n'avez pas initiée, utiliser l'adresse de gestionnaire de transaction principal qui a été reçue dans la commande IDENTIFY.

L'unicité mondiale des URL TIP doit être garantie pour toujours. Cette contrainte d'unicité assure que les URL TIP ne sont jamais dupliqués, empêchant pas là des comportements non déterministes possibles. La réalisation de l'unicité est spécifique de la mise en œuvre. Par exemple, l'identifiant unique universel (UUID, *Universally Unique Identifier*), aussi appelé identifiant unique au monde (GUID, *Globally Unique Identifier*) (voir la référence [9])) pourrait être utilisé au titre de la <chaîne de transaction>. Noter aussi que certains identifiants de transactions standard peuvent définir leurs propres règles pour assurer l'unicité mondiale (par exemple, les identifiants d'action atomique CCR d'OSI).

Excepté lorsque décrit autrement ci-dessus, le schéma d'URL TIP suit les règles pour les caractères réservés comme défini dans [4], et utilisent les séquences d'échappement définies à la section 5 de [4].

Noter que le protocole TIP lui-même n'utilise pas le schéma d'URL de TIP (il n'utilise pas le schéma d'adresse de gestionnaire de transaction). Le schéma d'URL TIP est proposé comme une manière standard de passer les informations d'identification de transaction à travers d'autres protocoles. Par exemple, entre des processus d'application coopératifs. L'URL TIP peut alors être utilisé pour communiquer au gestionnaire de transaction local les informations nécessaires pour associer l'application à une transaction TIP particulière. Par exemple, pour tirer (PULL) la transaction de chez un gestionnaire de transaction distant. On prévoit que chaque mise en œuvre de TIP va fournir un ensemble d'API à cette fin ([5] comporte des exemples de telles API).

9. États d'une connexion

À tout instant, il est permis à une seule partie sur une connexion d'envoyer des commandes, alors que l'autre partie ne peut que répondre aux commandes qu'elle reçoit. Tout au long du présent document, la partie à qui il est permis d'envoyer des commandes est appelée "principal" ; l'autre partie est appelée "secondaire". Initialement, la partie qui initie la connexion est le principal ; cependant, quelques commandes causent la commutation des rôles. Une connexion retourne à sa polarité d'origine chaque fois qu'est atteint l'état Repos *(iddle)*.

Lorsque le multiplexage est utilisé, ces règles s'appliquent de façon indépendante à chaque connexion "virtuelle", sans considération de la polarité de la connexion sous-jacente (qui sera dans l'état Multiplexage).

Noter que des commandes peuvent être envoyées "hors bande" par le secondaire via l'utilisation de traitements en parallèle *(pipelining)*. Cela n'affecte pas la polarité de la connexion (c'est-à-dire que les rôles de principal et de secondaire ne s'échangent pas). Voir les détails à la section 12.

Dans le cas normal, les connexions TIP ne devraient être closes que par le principal, lorsque il est dans l'état Initial. Il n'est en général pas souhaitable pour une connexion d'être close par le secondaire, bien que cela puisse être nécessaire dans certains cas d'erreur.

À tout instant, une connexion est dans un des états suivants. Du point de vue de la partie secondaire, l'état change lorsque elle envoie une réponse ; du point de vue de la partie principale, l'état change lorsque elle reçoit une réponse.

Initial : La connexion initiale commence dans l'état Initial. À l'entrée dans cet état, la partie qui a initié la connexion

devient principale, et l'autre partie devient secondaire. Il n'y a pas de transaction associée à la connexion dans cet état. À partir de cet état, le principal peut envoyer une commande IDENTIFY ou TLS.

Repos *(Idle)* : Dans cet état, le principal et le secondaire se sont mis d'accord sur une version de protocole et le principal a fourni un identifiant au secondaire pour se reconnecter après une défaillance. Il n'y a pas de transaction associée à la connexion dans cet état. À l'entrée dans cet état, la partie qui a initié la connexion devient le principal, et l'autre partie devient le secondaire. À partir de cet état, le principal peut envoyer n'importe laquelle des commandes suivantes : BEGIN, MULTIPLEX, PUSH, PULL, QUERY et RECONNECT.

Commencé *(Begun)* : Dans cet état, une connexion est associée à une transaction active, qui peut seulement être achevée par un protocole à une phase. Une réponse BEGUN à une commande BEGIN place une connexion dans cet état. La défaillance d'une connexion dans l'état Commencé implique que la transaction sera interrompue. À partir de cet état, le principal peut envoyer une commande ABORT, ou COMMIT.

Enlisté : Dans cet état, la connexion est associée à une transaction active, qui peut être achevée par un protocole à une phase ou par un protocole à deux phases. Une réponse PUSHED à une commande PUSH, ou une réponse PULLED à une commande PULL, place la connexion dans cet état. Une défaillance de la connexion dans l'état Enlisté implique que la transaction va être interrompue. À partir de cet état, le principal peut envoyer une commande ABORT, COMMIT, ou PREPARE.

Préparé : Dans cet état, une connexion est associée à une transaction qui a été préparée. Unr réponse PREPARED à une commande PREPARE, ou une réponse RECONNECTED à une commande RECONNECT place une connexion dans cet état. À la différence des autres états, la défaillance d'une connexion dans cet état ne cause pas l'interruption automatique de la transaction. À partir de cet état, le principal peut envoyer une commande ABORT, ou COMMIT.

Multiplexage : Dans cet état, la connexion est utilisée par un protocole de multiplexage, qui fournit son propre ensemble de connexions. Dans cet état, aucune commande TIP n'est possible sur la connexion. (Bien sûr, les commandes TIP sont possibles sur les connexions fournies par le protocole de multiplexage.) La connexion ne peut jamais quitter cet état.

Tls : Dans cet état, la connexion est utilisée par le protocole TLS, qui fournit sa propre connexion sécurisée. Dans cet état, aucune commande TIP n'est possible sur la connexion. (Bien sûr, les commandes TIP sont possibles sur les connexions fournies par le protocole TLS.) La connexion ne peut jamais quitter cet état.

Erreur : Dans cet état, une erreur de protocole est survenue, et la connexion n'est plus utile. La connexion ne peut jamais quitter cet état.

10. Versions du protocole

Le présent document décrit la version 3 du protocole. Afin de s'accommoder des futures versions, la partie principale envoie un message qui indique le plus faible et le plus fort numéro de version qu'elle comprend. Le secondaire répond par le plus fort numéro de version qu'il comprend.

Après un tel échange, la communication peut avoir lieu en utilisant le plus petit des plus forts numéros de version (c'est-à-dire, le plus haut numéro de version que tous deux comprennent). Cet échange est obligatoire et survient en utilisant la commande IDENTIFY (et la réponse IDENTIFIED).

Si la plus forte version supportée par une partie est considérée comme obsolète et n'est plus acceptée par l'autre partie, aucune communication utile ne peut avoir lieu. Dans ce cas, la partie la plus récente devrait simplement abandonner la connexion.

11. Commandes et réponses

Toutes les commandes et réponses consistent en une ligne de texte ASCII, en utilisant seulement des octets avec des valeurs dans la gamme de 32 à 126 inclus, suivis soit par un CR (un octet de valeur 13) ou un LF (un octet de valeur 10). Chaque ligne peut être coupée en un ou plusieurs "mots", où les mots successifs sont séparés par un ou plusieurs octets espace (valeur 32).

Un nombre arbitraire d'espaces au début et/ou à la fin de chaque ligne est admis, et sont ignorées.

Les lignes qui sont vides, ou consistent entièrement d'espaces sont ignorées. (Une implication de cela est qu'on peut

terminer les lignes aussi bien avec un CR qu'un LF si on le désire ; le LF sera traité comme terminant une ligne vide, et ignoré.)

Dans tous les cas, le premier mot de chaque ligne indique le type de commande ou réponse ; toutes les commandes et réponses définies consistent seulement en lettres majuscules.

Pour certaines commandes et réponses, les mots suivants portent des paramètres pour la commande ou réponse ; chaque commande et réponse prend un nombre fixe de paramètres.

Tous les mots sur une ligne de commande ou de réponse après (et incluant) le premier mot indéfini sont totalement ignorés. Cela peut être utilisé pour passer des informations lisibles par l'homme pour des besoins de débogage ou autres.

12. Traitement de commandes en parallèle

Afin de réduire la latence de la communication et d'améliorer l'efficacité, il est possible que plusieurs "lignes" TIP (commandes ou réponses) soient traitées en parallèle (concaténées) et envoyées dans un seul message. Les lignes peuvent aussi être envoyées "en avance" (par le secondaire) pour traitement ultérieur par le principal). Des exemples sont une commande ABORT suivie immédiatement d'une commande BEGIN, ou une réponse COMMITTED immédiatement suivie d'une commande PULL.

L'envoi de lignes en parallèle est une option de mise en œuvre. Tout comme le choix des lignes qui sont envoyées en parallèle. Généralement, il doit être certain que la ligne envoyée en parallèle sera valide pour l'état de la connexion au moment du traitement de la ligne par le receveur. Il incombe à l'envoyeur de le déterminer.

Toutes les mises en œuvre doivent prendre en charge la réception des lignes en parallèle – dont les règles de traitement sont décrites ci-dessous :

Lorsque l'état de connexion est tel qu'une ligne devrait être lue (qu'elle soit de commande ou de réponse) cette ligne est alors traitée (si elle est reçue). Aucune autre ligne n'est lue à partir de la connexion jusqu'à ce que le traitement atteigne à nouveau un tel état. Si une ligne est reçue sur une connexion lorsque ce n'est pas au tour de l'autre partie d'envoyer, cette ligne n'est pas rejetée. La ligne est plutôt conservée et sera traitée lorsque l'état de connexion l'exigera à nouveau. La partie qui reçoit doit traiter les lignes et produire les réponses dans l'ordre des lignes reçues. Si une ligne cause une erreur, la connexion entre dans l'état Erreur, et toutes les lignes suivantes sur la connexion sont éliminées.

13. Commandes TIP

Les commandes relèvent soit des connexions soit des transactions. Les commandes qui relèvent des connexions sont : IDENTIFY, MULTIPLEX et TLS. Les commandes qui relèvent des transactions sont : ABORT, BEGIN, COMMIT, PREPARE, PULL, PUSH, QUERY, et RECONNECT.

Voici une liste de toutes les commandes valides, et de toutes les réponses possibles à chacune d'elles:

ABORT
Cette commande est valide dans les états Commencé, Enlisté, et Préparé. Elle informe le secondaire que la transaction en cours de la connexion va s'interrompre. Les réponses possibles sont :

ABORTED
La transaction s'est interrompue; la connexion entre dans l'état Repos.

ERROR
La commande a été produite dans le mauvais état, ou était mal formée. La connexion entre dans l'état Erreur.

BEGIN
Cette commande n'est valide que dans l'état Repos. Elle demande au secondaire de créer une nouvelle transaction et de l'associer à la connexion. La transaction nouvellement créée sera achevée par un protocole à une phase. Les réponses possibles sont :

BEGUN <identifiant de transaction>
Une nouvelle transaction a bien été commencée, et la transaction est maintenant la transaction en cours de la connexion. La connexion entre dans l'état Commencé.

NOTBEGUN
Une nouvelle transaction n'a pas pu commencer ; la connexion reste dans l'état Repos.

ERROR
La commande a été produite dans le mauvais état, ou était mal formée. La connexion entre dans l'état Erreur.

COMMIT
Cette commande est valide dans les états Commencé, Enlisté ou Préparé. Dans les états Commencé ou Enlisté, elle demande au secondaire d'essayer d'engager la transaction ; dans l'état Préparé, elle informe le secondaire que la transaction s'est engagée. Noter que dans l'état Enlisté, cette commande représente un protocole à une phase, et ne devrait être faite que lorsque l'envoyeur : 1) n'a pas de ressource locale récupérable impliquée dans la transaction, et 2) n'a qu'un seul subordonné (l'envoyeur ne sera pas impliqué dans un processus de récupération de transaction). Les réponses possibles sont :

ABORTED
Cette réponse n'est possible qu'à partir des états Commencé et Enlisté. Elle indique qu'une des parties s'est opposée à l'engagement de la transaction, de sorte qu'elle a été interrompue au lieu de s'engager. La connexion entre dans l'état Repos.

COMMITTED
Cette réponse indique que la transaction a été engagée, et que le principal n'a plus aucune responsabilité envers le secondaire par rapport à la transaction. La connexion entre dans l'état Repos.

ERROR
La commande a été produite dans le mauvais état, ou était mal formée. La connexion entre dans l'état Erreur.

ERROR
Cette commande est valide dans tous les états ; elle informe le secondaire qu'une réponse précédente n'a pas été reconnue ou était mal formée. Un secondaire ne devrait pas répondre à cette commande. La connexion entre dans l'état Erreur.

IDENTIFY <plus faible version de protocole> <plus haute version de protocole> <adresse du gestionnaire de transaction principal> | "-" <adresse du gestionnaire de transaction secondaire>

Cette commande n'est valide que dans l'état Initial. La partie principale informe la partie secondaire de : 1) la plus faible et la plus forte version de protocole acceptée (toutes les versions entre la plus faible et la plus forte doivent être prises en charge ; 2) facultativement, un identifiant de la partie principale à laquelle la partie secondaire peut rétablir une connexion s'il en était besoin (l'adresse du gestionnaire de transaction principal) et 3) un identifiant qui peut être utilisé par des serveurs mandataires intermédiaires pour se connecter au gestionnaire de transaction TIP requis (l'adresse du gestionnaire de transaction secondaire). Si une adresse de gestionnaire de transaction principal n'est pas fournie, la partie secondaire va répondre par un ABORTED ou READONLY à toute commande PREPARE. Les réponses possibles sont :

IDENTIFIED <version du protocole>
La partie secondaire a bien été contactée et a sauvegardé l'adresse du gestionnaire de transaction principal. La réponse contient la plus forte version de protocole acceptée par la partie secondaire. Toutes les communications ultérieures sont supposées avoir lieu en utilisant la plus faible des versions de protocole contenues dans la commande IDENTIFY et la réponse IDENTIFIED. La connexion entre dans l'état Repos.

NEEDTLS
La partie secondaire ne veut communiquer que sur une connexion sécurisée par TLS. La connexion entre dans l'état Tls, et toute la suite de la communication est comme défini par le protocole TLS. Ce protocole va commencer par le premier octet après la terminaison de ligne de la commande IDENTIFY (pour les données envoyées par la partie principale) et le premier octet après la terminaison de ligne de la réponse NEEDTLS (pour les données envoyées par la partie secondaire). Cela implique qu'une mise en œuvre ne doit pas envoyer à la fois un octet CR et un octet LF après la commande IDENTIFY ou la réponse NEEDTLS,de peur que l'octet LF soit confondu avec le premier octet du protocole TLS. La connexion fournie par le protocole TLS commence dans l'état Initial. Après que TLS a été négocié, la partie principale doit renvoyer la commande IDENTIFY. Si la partie principale ne peut pas accepter (ou refuse d'utiliser) le protocole TLS, elle clôt la connexion.

ERROR
La commande a été produite dans le mauvais état, ou était mal formée. Cette réponse survient aussi si la partie secondaire n'accepte aucune version du protocole dans la gamme prise en charge par la partie principale. La connexion entre dans l'état Erreur. La partie principale devrait clore la connexion.

MULTIPLEX <identifiant-de-protocole>
Cette commande n'est valide que dans l'état Repos. La commande cherche à établir un accord pour utiliser la connexion avec un protocole de multiplexage qui va fournir un grand nombre de connexions sur la connexion existante. Le principal suggère un protocole de multiplexage particulier. La partie secondaire peut accepter ou rejeter l'utilisation de ce protocole.

À présent, le seul identifiant de protocole défini est "TMP2.0", qui se réfère au protocole de multiplexage TIP, version 2.0. Voir à l'Appendice A les détails de ce protocole. D'autres identifiants de protocole pourront être définis à l'avenir.

Si la commande MULTIPLEX est acceptée, le protocole de multiplexage spécifié va totalement contrôler la connexion sous-jacente. Ce protocole va commencer par le premier octet après la terminaison de ligne de la commande MULTIPLEX (pour les données envoyées par l'initiateur) et par le premier octet après la terminaison de ligne de la réponse MULTIPLEXING (pour les données reçues par l'initiateur). Cela implique qu'une mise en œuvre ne doit pas envoyer à la fois un octet CR et un octet LF après la commande MULTIPLEX ou la réponse MULTIPLEXING, de peur que l'octet LF soit pris pour le premier octet du protocole de multiplexage.

Noter que quand on utilise TMP V2.0, une seule commande TIP (message d'application TMP) doit être entièrement contenue dans un seul paquet TMP (le fanion PUSH TMP n'est pas utilisé par TIP). Les réponses possibles à la commande MULTIPLEX sont :

MULTIPLEXING
La partie secondaire accepte d'utiliser le protocole de multiplexage spécifié. La connexion entre dans l'état Multiplexage, et toute la suite de la communication est comme défini par ce protocole. Toutes les connexions créées par le protocole de multiplexage commencent dans l'état Repos.

CANTMULTIPLEX
La partie secondaire ne peut pas accepter (ou refuse d'utiliser) le protocole de multiplexage spécifié. La connexion reste dans l'état Repos.

ERROR
La commande a été produite dans le mauvais état, ou était mal formée. La connexion entre dans l'état Erreur.

PREPARE
Cette commande n'est valide que dans l'état Enlisté ; elle demande au secondaire de préparer l'engagement de la transaction (première phase de l'engagement en deux phases). Les réponses possibles sont :

PREPARED
Le subordonné a préparé la transaction ; la connexion entre dans l'état Préparé.

ABORTED
Le subordonné a interdit l'engagement de la transaction. La connexion entre dans l'état Repos. Après cette réponse, le supérieur n'a plus de responsabilités envers le subordonné à l'égard de la transaction.

READONLY
Le subordonné ne se soucie plus de savoir si la transaction s'engage ou s'interrompt. La connexion entre dans l'état Repos. Après cette réponse, le supérieur n'a plus de responsabilités envers le subordonné à l'égard de la transaction.

ERROR
La commande a été produite dans le mauvais état, ou était mal formée. La connexion entre dans l'état Erreur.

PULL <identifiant de transaction du supérieur> <identifiant de transaction du subordonné>

Cette commande n'est valide que dans l'état Repos. Cette commande cherche à établir une relation supérieur/subordonné dans une transaction, avec la partie principale de la connexion comme subordonné (c'est-à-dire qu'il va tirer une transaction de la partie secondaire). Noter que la valeur entière de <chaîne de transaction> (comme définie à la Section 8 "Localisateurs de ressource uniforme TIP") doit être spécifiée comme identifiant de transaction. Les réponses possibles sont :

PULLED
La relation a été établie. À réception de cette réponse, la transaction spécifiée devient la transaction en cours de la connexion, et la connexion entre dans l'état Enlisté. De plus, les rôles de principal et de secondaire sont inversés. (C'est à dire que le supérieur devient le principal pour la connexion.)

NOTPULLED
La relations n'a pas été établie (peut-être parce que la partie secondaire n'a plus la transaction demandée). La connexion reste dans l'état repos.

ERROR
La commande a été produite dans le mauvais état, ou était mal formée. La connexion entre dans l'état Erreur.

PUSH <identifiant de transaction du supérieur>
Cette commande n'est valide que dans l'état Repos. Elle cherche à établir une relation supérieur/subordonné dans une transaction avec le principal comme supérieur. Noter que la valeur entière de <chaîne de transaction> (comme définie dans la section 8 "Localisateurs de ressource uniforme TIP") doit être spécifiée comme l'identifiant de transaction. Les réponses possibles sont :

PUSHED <identifiant de transaction du subordonné>
La relation a été bien établie, et l'identifiant par lequel le subordonné connaît la transaction est retourné. La transaction devient la transaction en cours pour la connexion, et la connexion entre dans l'état Enlisté.

ALREADYPUSHED <identifiant de transaction du subordonné>
La relation a été bien établie, et l'identifiant par lequel le subordonné connaît la transaction est retourné. Cependant, le subordonné connaît déjà la transaction, et il attend l'arrivée du protocole d'engagement à deux phases via une connexion différente. Dans ce cas, la connexion reste dans l'état Repos.

NOTPUSHED
La relation n'a pas pu être établie. La connexion reste dans l'état Repos.

ERROR
La commande a été produite dans le mauvais état, ou était mal formée. La connexion entre dans l'état Erreur.

QUERY <identifiant de transaction du supérieur>
Cette commande n'est valide que dans l'état Repos. Un subordonné utilise cette commande pour déterminer si une transaction spécifique existe toujours chez le supérieur. Les réponses possibles sont :

QUERIEDEXISTS
La transaction existe toujours. La connexion reste dans l'état Repos.

QUERIEDNOTFOUND
La transaction n'existe plus. La connexion reste dans l'état Repos.

ERROR
La commande a été produite dans le mauvais état, ou était mal formée. La connexion entre dans l'état Erreur.

RECONNECT <identifiant de transaction du subordonné>
Cette commande n'est valide que dans l'état Repos. Un supérieur utilise la commande pour rétablir une connexion pour une transaction, lorsque la connexion a été perdue durant l'état Préparé. Les réponses possibles sont :

RECONNECTED
Le subordonné accepte la reconnexion. La connexion entre dans l'état Préparé.

NOTRECONNECTED
Le subordonné ne connaît plus la transaction. La connexion reste dans l'état Repos.

ERROR
La commande a été produite dans le mauvais état, ou était mal formée. La connexion entre dans l'état Erreur.

TLS
Cette commande n'est valide que dans l'état Initial. Un principal utilise cette commande pour tenter d'établir une connexion sûre utilisant TLS.

Si la commande TLS est acceptée, le protocole TLS va contrôler totalement la connexion sous-jacente. Ce protocole va commencer avec le premier octet après la terminaison de ligne de la commande TLS (pour les données envoyées par le principal) et avec le premier octet après la terminaison de ligne de la réponse TLSING (pour les données reçues par le principal). Cela implique qu'une mise en œuvre ne doit pas envoyer à la fois un octet CR et un octet LF après la commande TLS ou la réponse TLSING, de peur que l'octet LF soit pris pour le premier octet du protocole TLS.

Les réponses possibles à la commande TLS sont :

TLSING
La partie secondaire accepte d'utiliser le protocole TLS [3]. La connexion entre dans l'état Tls, et toute la communication suivante est comme défini par le protocole TLS. La connexion fournie par le protocole TLS débute dans l'état Initial.

CANTTLS
La partie secondaire ne peut pas prendre en charge (ou refuse d'utiliser) le protocole TLS. La connexion reste dans l'état Initial.

ERROR
La commande a été produite dans le mauvais état, ou était mal formée. La connexion entre dans l'état Erreur.

14. Traitement des erreurs

Si l'une ou l'autre partie reçoit une ligne qu'elle ne peut pas comprendre, elle clôt la connexion. Si l'une ou l'autre partie (sur une commande ou une réponse) reçoit une indication ERROR ou une réponse ERROR sur une connexion, la connexion entre dans l'état Erreur et aucune autre communication n'est possible sur cette connexion. Une mise en œuvre peut décider de clore la connexion. La clôture d'une connexion est traitée par l'autre partie comme une défaillance de communication.

La réception d'une indication ERROR ou d'une réponse ERROR indique que l'autre partie croit que vous n'avez pas mis correctement en œuvre le protocole.

15. Défaillance et récupération de connexion

Une défaillance de connexion peut être causée par une défaillance de communication, ou par la clôture de la connexion par une des parties. On suppose que les mises en œuvre de TIP vont utiliser un mécanisme privé pour détecter les défaillances de connexion TIP (par exemple, de conservation d'activation de prise, ou un schéma de temporisation).

Selon l'état d'une connexion, les gestionnaires de transaction vont devoir effectuer diverses actions lors d'une défaillance de connexion.

Si la défaillance de la connexion a lieu dans l'état Initial ou Repos, la connexion ne se réfère pas à une transaction. Aucune action n'est nécessaire.

Si la défaillance de la connexion a lieu dans l'état Multiplexage, toutes les connexions fournies par le protocole de multiplexage sont supposées défaillantes. Chacune d'elles sera traitée indépendamment.

Si la défaillance de la connexion a lieu dans l'état Commencé ou Enlisté et si COMMIT a été envoyé, l'achèvement de la transaction a alors été délégué au subordonné (le supérieur n'est pas impliqué) ; le résultat de la transaction n'est pas connu du supérieur (il est connu chez le subordonné). Le supérieur utilise des moyens spécifiques de l'application pour déterminer le résultat de la transaction (noter que l'intégrité de la transaction n'est pas compromise dans ce cas car le supérieur n'a pas de ressources récupérables impliquées dans la transaction). Si la défaillance de la connexion a lieu dans l'état Commencé ou Enlisté et si COMMIT n'a pas été envoyé, la transaction va s'interrompre.

Si la défaillance de la connexion a lieu dans l'état Préparé, l'action appropriée est alors différente pour le supérieur et pour le subordonné dans la transaction.

Si le supérieur détermine que la transaction engage, il doit alors éventuellement établir une nouvelle connexion avec le subordonné, et envoyer une commande RECONNECT pour la transaction. Si il reçoit une réponse NOTRECONNECTED, il n'a rien à faire d'autre. Cependant, si il reçoit une réponse RECONNECTED, il doit envoyer une demande COMMIT et recevoir une réponse COMMITTED.

Si le supérieur détermine que la transaction est interrompue, il lui est permis (mais il n'y est pas obligé) d'établir une nouvelle connexion et d'envoyer une commande RECONNECT pour la transaction. Si il reçoit une réponse RECONNECTED, il devrait envoyer une commande ABORT.

La définition ci-dessus permet au supérieur de rétablir la connexion avant qu'il connaisse le résultat de la transaction, si il l'estime convenable. Ayant réussi à une commande RECONNECT, la connexion revient à l'état Préparé, et le supérieur peut envoyer une commande COMMIT ou ABORT selon ce qui est approprié lorsque il sait le résultat de la transaction.

Noter qu'il est possible qu'une commande RECONNECT soit reçue par le subordonné avant qu'il sache que la connexion précédente a eu une défaillance. Dans ce cas, le subordonné traite la commande RECONNECT comme une indication de défaillance et purge toutes les ressources associées à la connexion, et associe l'état de la transaction à la nouvelle connexion.

Si un subordonné remarque une défaillance de connexion dans l'état Préparé, il devrait alors tenter périodiquement de créer une nouvelle connexion avec le supérieur et envoyer une commande QUERY pour la transaction. Il devrait continuer à faire cela jusqu'à ce qu'un des deux événements suivants arrive :

1. Il reçoit une réponse QUERIEDNOTFOUND de la part du supérieur. Dans ce cas, le subordonné devrait interrompre la transaction.

2. Le supérieur, sur une connexion initiée par lui, envoie une commande RECONNECT pour la transaction au subordonné. Dans ce cas, le subordonné peut s'attendre à apprendre le résultat de la transaction sur cette nouvelle connexion. Si cette nouvelle connexion devait être défaillante avant que le subordonné apprenne le résultat de la transaction, il devrait recommencer d'envoyer des commandes QUERY.

Noter que si un système TIP reçoit une commande QUERY ou RECONNECT, et si pour une raison quelconque il est dans l'incapacité de satisfaire la demande (par exemple, les informations de récupération nécessaires ne sont pas actuellement disponibles) la connexion devrait alors être abandonnée.

16. Considérations pour la sécurité

La présente section est destinée à informer les développeurs d'applications, les développeurs de gestionnaire de transaction, et les utilisateurs, des implications pour la sécurité de TIP tel que décrit dans le présent document. L'exposé ne comporte pas de solutions définitives aux problèmes décrits, bien qu'il fasse des suggestions pour réduire les risques pour la sécurité.

Comme avec tous les protocoles d'engagement en deux phases, tous les mécanismes de sécurité appliqués au protocole de communication d'application sont sujets à la subversion sauf si les mécanismes correspondants sont appliqués au protocole d'engagement. Par exemple, toute authentification entre les parties qui utilisent le protocole d'application doit être soutenue par la sécurité des échanges TIP avec au moins le même niveau de certitude.

16.1 TLS, authentification mutuelle et autorisation

TLS fournit l'authentification facultative du côté client, l'authentification facultative du côté serveur, et le chiffrement de paquet facultatif.

Une mise en œuvre TIP peut refuser de fournir le service si TLS n'est pas utilisé. Elle peut refuser de fournir le service si le chiffrement de paquet n'est pas utilisé. Elle peut refuser de fournir le service à moins que la partie distante n'ait été authentifiée (via TLS).

Une mise en œuvre TIP devrait être d'accord pour s'authentifier (via TLS). Ceci est vrai que la mise en œuvre agisse comme client ou comme serveur.

Une fois qu'une partie distante a été authentifiée, un gestionnaire de transaction TIP peut utiliser l'identité de cette partie distante pour décider quelles opérations permettre.

La décision d'utiliser TLS sur une connexion, et si elle est prise, comment utiliser TLS, et quelles opérations seront ensuite permises, est déterminée par les politiques de sécurité des gestionnaires de transaction TIP qui se connectent les uns à l'égard de chacun des autres. Comment sont définies ces politiques de sécurité, et comment un gestionnaire de transaction TIP les apprend est totalement du domaine de la mise en œuvre et sort du domaine d'application du présent document.

16.2. Attaques de déni de service fondées sur PULL

Supposons qu'un utilisateur malveillant connaisse l'identité d'une transaction qui est actuellement active dans un gestionnaire de transaction. Si le malfaiteur ouvre une connexion TIP avec le gestionnaire de transaction, envoie une commande PULL, puis clôt la connexion, il peut causer l'interruption de cette transaction. Il en résulte un déni de service pour le possesseur légitime de la transaction.

Une mise en œuvre peut éviter cette attaque en refusant les commandes PULL sauf si elles utilisent TLS, si la partie distante a été authentifiée, et si la partie distante est de confiance.

16.3 Attaques de déni de service fondées sur PUSH

Lorsque la connexion entre deux gestionnaire de transactions est close alors qu'une transaction est dans l'état Préparé, chaque gestionnaire de transaction doit se souvenir des informations sur la transaction jusqu'à ce qu'une connexion puisse être rétablie.

Si un utilisateur malveillant exploite ce fait pour créer des transactions de façon répétée, les met dans l'état Préparé et abandonne la connexion, il peut causer l'épuisement des ressources d'un gestionnaire de transaction, déniant par là le service à tous les utilisateurs légitimes de ce gestionnaire de transaction.

Une mise en œuvre peut éviter cette attaque en refusant les commandes PUSH sauf si TLS est utilisé, si la partie distante a été authentifiée, et si la partie distante est de confiance.

16.4 Attaque de corruption de transaction

Si un gestionnaire de transaction subordonné a perdu sa connexion pour une transaction préparée particulière, un utilisateur malveillant peut initier une commande TIP au gestionnaire de transaction, et envoyer une commande RECONNECT suivie par une commande COMMIT ou ABORT pour la transaction. L'utilisateur malveillant pourrait ainsi causer l'engagement d'une partie d'une transaction alors qu'elle aurait dû être interrompue, ou vice versa.

Une mise en œuvre peut éviter cette attaque en enregistrant l'identité authentifiée de son supérieur dans une transaction, et en refusant les commandes RECONNECT sauf si TLS est utilisé et si l'identité authentifiée de la partie distante est la même que l'identité du supérieur d'origine.

16.5 Attaques par reniflage de paquet

Si un utilisateur malveillant peut intercepter du trafic sur une connexion TIP, il peut être capable de déduire des informations utiles pour monter d'autres attaques. Par exemple, si les champs de commentaire comportent le nom du produit et le numéro de version d'un gestionnaire de transaction, un utilisateur malveillant pourrait être capable d'utiliser ces informations pour déterminer quelles failles à la sécurité existent dans la mise en œuvre.

Une mise en œuvre peut éviter cette attaque en utilisant toujours TLS pour fournir le chiffrement de session, et en ne mettant aucune information personnalisée sur la paire commande/réponse TLS/TLSING.

16.6 Attaque par interposition

Si un utilisateur malveillant peut intercepter et altérer du trafic sur une connexion TIP, il peut causer des dommages de diverses façons. Par exemple, il pourrait remplacer une commande COMMIT par une commande ABORT.

Une mise en œuvre peut éviter cette attaque en utilisant toujours TLS pour fournir le chiffrement de session et l'authentification de la partie distante.

17. Références

[1] Gray, J. and A. Reuter (1993), "Transaction Processing: Concepts and Techniques". San Francisco, CA: Morgan Kaufmann Publishers. (ISBN 1-55860-190-2).

[2] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee, "Protocole de transfert Hypertexte -- HTTP/1.1", RFC2068, janvier 1997. (*Obsolète, voir* RFC2616) *(P.S.)*

[3] T. Dierks et C. Allen, "Protocole TLS version 1.0", RFC2246, janvier 1999.

[4] T. Berners-Lee et autres, "Localisateurs uniformes de ressource (URL)", RFC1738, décembre 1994. *(P.S., Obsolète, voir les RFC4248 et 4266)*

[5] K. Evans, J. Klein, J. Lyon, "Protocole Internet de transaction – Exigences et informations supplémentaires", RFC2372, juillet 1998. *(Information)*

[6] R. Moats, "Syntaxe des URN", RFC2141, mai 1997.

[7] L. Daigle, D. van Gulik, R. Iannella, P. Faltstrom, "Mécanismes de définition d'espace de noms de noms de ressource

uniforme (URN)", RFC3406, octobre 2002. (BCP0066)

[8] T. Berners-Lee, R. Fielding et L. Masinter, "Identifiants de ressource uniformes (URI) : Syntaxe générique", RFC2396, août 1998. (*Obsolète, voir* RFC3986)

[9] P. Leach et autres, "Espace de noms d'URN d'identifiant unique au monde (UUID)", RFC4122, juillet 2005. *(P.S.)*

18. Adresse des auteurs

Jim Lyon
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399, USA

téléphone : +1 (206) 936 0867
fax : +1 (206) 936 7329
mél : JimLyon@Microsoft.Com

Keith Evans
Tandem Computers, Inc.
5425 Stevens Creek Blvd
Santa Clara, CA 95051-7200, USA

téléphone : +1 (408) 285 5314
fax : +1 (408) 285 5245
mél : Keith.Evans@Tandem.Com

Johannes Klein
Tandem Computers Inc.
10555 Ridgeview Court
Cupertino, CA 95014-0789, USA

téléphone : +1 (408) 285 0453
fax : +1 (408) 285 9818
mél : Johannes.Klein@Tandem.Com

19. Commentaires

Prière d'envoyer les commentaires sur ce document aux auteurs aux adresse de messagerie ci-dessus , ou à la liste de diffusion de TIP à <Tip@Lists.Tandem.Com>. On peut s'abonner à la liste de diffusion de TIP en envoyant un message à <Listserv@Lists.Tandem.Com> avec une ligne "subscribe tip <nom complet>" quelque part dans le corps du message.

Appendice A. Protocole de multiplexage TIP version 2.0.

Le présent appendice décrit la version 2.0 du protocole de multiplexage TIP (TMP, *TIP Multiplexing Protocol*). TMP est uniquement destiné à être utilisé avec le protocole TIP et fait partie de la spécification du protocole TIP (bien que sa mise en œuvre soit facultative). TMP V2.0 est le seul protocole de multiplexage supporté par TIP V3.0.

Résumé
TMP fournit un mécanisme simple pour créer plusieurs connexions légères sur une seule connexion TCP. Plusieurs de ces connexions légères peuvent être actives simultanément. TMP fournit un service en mode octet, mais permet que les limites de message soient marquées.

A.1 Introduction

Il existe plusieurs protocoles de large utilisation dans l'Internet qui créent une seule connexion TCP pour chaque transaction. Malheureusement, comme ces transactions sont brèves, le coût d'établissement et d'acquisition de ces connexions TCP devient significatif, à la fois en termes de ressources utilisées et de délais associés aux mécanismes de contrôle d'encombrement de TCP.

Le protocole de multiplexage TIP (TMP, *TIP Multiplexing Protocol*) est un protocole simple qui fonctionne par dessus TCP et qui peut être utilisé pour créer plusieurs connexions légères sur une seule connexion de transport. TMP fournit donc une utilisation plus efficace des connexions TCP. Les données provenant de plusieurs connexions TMP différentes peuvent être entrelacées, et les frontières de message ainsi que les fins de marqueurs de flux peuvent être fournies.

Comme TMP fonctionne par dessus un service de transport ordonné par octet fiable, il peut éviter la plus grande partie du travail supplémentaire que doit subir TCP pour assurer la fiabilité. Par exemple, les connexions TMP n'ont pas besoin d'être confirmées, de sorte qu'il n'y a pas besoin d'attendre que la prise de contact soit terminée pour envoyer les données.

Note : TMP n'est pas destiné à être un protocole de multiplexage généralisé. Si vous concevez un protocole différent qui a besoin du multiplexage, TMP peut être ou non approprié. Les protocoles qui ont de gros messages peuvent dépasser les capacités de mise en mémoire tampon du receveur, et dans certaines conditions cela peut causer un blocage. Lorsque TMP est utilisé avec TIP, il ne subit pas ce problème car TIP est un protocole de demande/réponse, et tous les messages sont courts.

A.2 Modèle du protocole

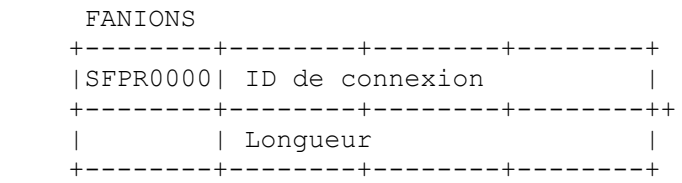
Le modèle de protocole de base est celui de multiples connexions légères fonctionnant sur un flux d'octets fiable. La partie qui a initié la connexion est désignée comme le principal et la partie qui a accepté la connexion est désignée comme le secondaire.

Les connexions peuvent être unidirectionnelle ou bidirectionnelles ; chaque extrémité d'une connexion bidirectionnelle peut être close séparément. Les connexions peuvent être closes normalement, ou être rétablies pour indiquer une libération interruptive. L'interruption d'une connexion clôt les deux flux de données.

Une fois qu'une connexion a été ouverte, les applications peuvent envoyer des messages sur elle, et signaler la fin des messages de niveau application. Les messages d'application sont encapsulés dans des paquets TMP et transférés sur le flux d'octets. Une seule commande TIP (message d'application TMP) doit être entièrement contenue au sein d'un seul paquet TMP.

A.3 Format du paquet TMP

Un paquet TMP comporte un en-tête de 64 bits suivi par zéro, un ou plusieurs octets de données. L'en-tête contient trois champs ; un octet de fanion, l'identifiant de connexion, et la longueur de paquet. Tous deux entiers, l'identifiant de connexion et la longueur de paquet doivent être envoyées dans l'ordre des octets du réseau.



A.3.1 Détails des fanions

Nom	Gabarit	Description
SYN	1xxx|0000	Ouvre une nouvelle connexion
FIN	x1xx|0000	Clôt une connexion existante
PUSH	xx1x|0000	Marque la limite du message au niveau application
RESET	xxx1|0000	Interrompt la connexion

A.4 Identifiants de connexion

Chaque connexion TMP est identifiée par un entier de 24 bits. Les connexions TMP créées par la partie qui a initié la connexion TCP sous-jacente doivent avoir des identifiants pairs ; celles créées par l'autre partie doivent avoir des identifiants impairs.

A.5 États de connexion TMP

Les connexions TMP peuvent exister dans plusieurs états différents ; Clos *(Closed)*, OuvertÉcriture *(OpenWrite*, OuvertSynLecture *(OpenSynRead)*, OuvertSynRétabli *(OpenSynReset)*, OuvertLectureÉcriture*) (OpenReadWrite)* , FerméÉcriture *(CloseWrite)*, et FerméLecture *(CloseRead)*. Une connexion peut changer son état en réponse à la réception d'un paquet avec le bit SYN, FIN, ou RESET mis, ou en réponse à une invocation d'API par l'application. Les invocations d'API disponibles sont ouvert, fermé, et interrompu.

La signification de la plupart des états est évidente (par exemple, OuvertÉcriture signifie qu'une connexion a été ouverte pour l'écriture). La signification des états OuvertSynLecture *(OpenSynRead)* et OuvertRétabliLecture *(OpenResetRead)* exigent quelques explications.

Dans l'état OuvertSynLecture, un principal a ouvert et immédiatement clos le flux de données de sortie d'une connexion, et attend maintenant une réponse SYN de la part du secondaire pour ouvrir le flux de données en entrée pour la lecture.

Dans l'état OuvertRétabliLecture, un principal a ouvert et immédiatement interrompu une connexion, et attend maintenant une réponse SYN de la part du secondaire pour clore finalement la connexion.

A.6 Priorités d'événement et transitions d'état

Le tableau des états qui figure ci-dessous décrit les actions et les transitions d'état qui surviennent en réponse à un événement donné. Les événements acceptés par chaque état sont énumérés dans l'ordre de priorité, la plus forte en premier. Si plusieurs événements sont présents dans un message, les événements qui correspondent à la liste sont traités. Si plusieurs événements correspondent, l'événement qui a la plus forte priorité est accepté et traité en premier. Tous les événements restants sont traités dans l'état résultant suivant.

Par exemple, si une connexion TMP est au secondaire dans l'état Clos, et si le secondaire reçoit un paquet qui contient un événement SYN, un événement FIN, et un événement de données d'entrée (c'est-à-dire DATA-IN) le secondaire accepte d'abord l'événement SYN (parce que c'est le seul correspondant dans l'état Clos). Le secondaire accepte la connexion, envoie un événement SYN et entre dans l'état LectureÉcriture. L'événement SYN est retiré de la liste des événements en instance. Les événements restants sont FIN et DATA-IN. Dans l'état LectueÉcriture, le secondaire lit les données d'entrée (c'est-à-dire que l'événement DATA-IN est traité d'abord parce qu'il une plus forte priorité que l'événement FIN). Une fois que les données ont été lues et que l'événement DATA-IN a été retiré de la liste des événements en instance, l'événement FIN est traité, et le secondaire entre dans l'état FerméÉcriture.

Si le secondaire reçoit un paquet qui contient un événement SYN, et si pour une raison quelconque, il n'est pas capable d'accepter la connexion (par exemple, des ressources insuffisantes) il devrait rejeter la demande en envoyant un événement SYN suivi par un événement RESET. Noter que les deux événements peuvent être envoyés au titre du même paquet TMP.

Si l'une ou l'autre partie reçoit un paquet TMP qu'il ne comprend pas, ou un événement dans un état incorrect, il clôt la connexion TCP.

État d'entrée	Événement	Action	État de sortie
Clos	SYN	SYN	LectureÉcriture
	OPEN	SYN	OuvertÉcriture
OuvertÉcriture	SYN	Accepte	LectureÉcriture
	WRITE	DATA-OUT	OuvertÉcriture
	CLOSE	FIN	OuvertSynLecture
	ABORT	RESET	OuvertSynRétabli
OuvertSynLecture	SYN	Accepte	FerméLecture
OuvertSynRétabli	SYN	Accepte	Clos
LectureÉcriture	DATA-IN	Accepte	LectureÉcriture
	FIN	Accepte	FerméÉcriture
	RESET	Accepte	Clos
	WRITE	DATA-OUT	LectureÉcriture
	CLOSE	FIN	FerméLecture
	ABORT	RESET	Clos
FerméÉcriture	RESET	Accepte	Clos
	WRITE	DATA-OUT	FerméÉcriture
	CLOSE	FIN	Clos
	ABORT	RESET	Clos
FerméÉcriture	DATA-IN	Accepte	FerméLecture
	FIN	Accepte	Clos
	RESET	Accepte	Clos
	ABORT	RESET	Clos

Priorités et transitions d'état d'événement TMP

Déclaration complète de droits de reproduction