Groupe de travail Réseau
Request for Comments : 2813
RFC mise à jour : 1459
Catégorie : Information

C. Kalt
avril 2000

Traduction Claude Brière de L'Isle

Relais pour la causette Internet : protocole de serveur

Statut de ce mémoire
Le présent mémoire fournit des informations pour la communauté de l'Internet. Il ne spécifie aucune norme de l'Internet. La distribution du présent mémoire n'est soumise à aucune restriction.
(La présente traduction incorpore les errata 383 et 2870)

Résumé
Bien que fondé sur le modèle client-serveur, le protocole de relais de la causette Internet (IRC, *Internet Relay Chat*) permet aux serveurs de se connecter les uns avec les autres pour former un réseau effectif.

Le présent document définit le protocole utilisé par les serveurs pour parler ensemble. Il était à l'origine un sur-ensemble du protocole client mais a connu une évolution différente.

Documenté pour la première fois de façon formelle en mai 1993 au titre de la [RFC1459], la plupart des changements apportés depuis se trouvent dans le présent document car les développements se sont concentrés sur une meilleure adaptabilité du protocole. Cette meilleure adaptabilité a permis aux réseaux existant à l'échelle mondiale de continuer leur croissance et d'atteindre des tailles qui sont un défi pour l'ancienne spécification.

Table des matières

1. Introduction....2
2. Base de données globale....2
  2.1 Serveurs....2
  2.2 Clients....2
  2.3 Canaux....3
3. Spécification du serveur IRC....3
  3.1 Généralités....3
  3.2 Codes de caractères....3
  3.3 Messages....3
  3.4 Réponses numériques....4
4. Détails du message....4
  4.1 Enregistrement de connexion....5
  4.2 Opérations du canal....8
5. Détails de mise en œuvre....9
  5.1 Vivacité de la connexion....9
  5.2 Acceptation d'une connexion de client à serveur....10
  5.3 Établissement d'une connexion serveur-serveur....10
  5.4 Terminaison d'une connexion serveur-client....11
  5.5 Terminaison des connexions serveur-serveur....11
  5.6 Suivi des changements de pseudonyme....11
  5.7 Suivi des pseudonymes récemment utilisés....11
  5.8 Contrôle des flux de clients....12
  5.9Recherches non bloquantes....12
6. Problèmes en cours....12
  6.1 Adaptabilité....12
  6.2 Étiquettes....13
  6.3 Algorithmes....13
7. Considérations pour la sécurité....13
  7.1 Authentification....13
  7.2 Intégrité....14
8. Prise en charge et disponibilité actuelle....14

9. Remerciements....14
10. Références....14
11. Adresse de l'auteur....15
12. Déclaration complète de droits de reproduction....15

1. Introduction

Le présent document est destiné aux personnes qui travaillent à la mise en œuvre d'un serveur IRC mais sera aussi utile à toute personne qui met en œuvre un service IRC.

Les serveurs fournissent les trois services de base exigés pour le système de conférence en temps réel défini par la [RFC2810] "Relais pour la causette Internet : architecture" : le localisateur de client (via le protocole client de la [RFC2812]), le relais de message (via le protocole serveur défini dans le présent document) et l'hébergement et la gestion de canal (suivant les règles spécifiques de la [RFC2811]).

Dans le présent document, les mots clés "DOIT", "NE DOIT PAS", "EXIGÉ", "DEVRA" NE DEVRA PAS", "DEVRAIT" NE DEVRAIT PAS", "RECOMMANDÉ", "PEUT", et "FACULTATIF" sont à interpréter comme décrit dans la [RFC2119].

2. Base de données globale

Bien que le protocole IRC définisse un modèle très réparti, chaque serveur entretient une "base de données d'état globale" sur l'ensemble du réseau IRC. Cette base de données est, en théorie, identique sur tous les serveurs.

2.1 Serveurs

Les serveurs sont identifiés de façon univoque par leur nom qui a une longueur maximum de soixante trois (63) caractères. Voir dans les règles de grammaire du protocole au paragraphe 3.3.1 ce qui peut ou non être utilisé dans un nom de serveur.

Chaque serveur est normalement connu de tous les autres serveurs, cependant il est possible de définir un "gabarit d'hôte" pour regrouper les serveurs en fonction de leur nom. À l'intérieur de la zone délimitée par le gabarit d'hôte, tous les serveurs ont un nom qui correspond au gabarit d'hôte, et tout autre serveur qui a un nom qui correspond au gabarit d'hôte NE DEVRA PAS être connecté au réseau IRC en dehors de la zone délimitée par le gabarit d'hôte. Les serveurs qui sont en-dehors de la zone n'ont pas connaissance des serveurs individuels présents à l'intérieur de la zone, ils sont au contraire présentés avec un serveur virtuel qui a pour nom le gabarit d'hôte.

2.2 Clients

Pour chaque client, tous les serveurs DOIVENT avoir les informations suivantes : un identifiant unique à l'échelle du réseau (dont le format dépend du type de client) et le serveur auquel le client est connecté.

2.2.1 Utilisateurs

Chaque utilisateur se distingue des autres utilisateurs par un pseudonyme *(nickname)* unique qui a une longueur maximum de neuf (9) caractères. Voir au paragraphe 3.3.1 sur les règles de grammaire du protocole ce qui peut être ou non utilisé dans un pseudonyme. En plus du pseudonyme, tous les serveurs DOIVENT avoir les informations suivantes sur tous les utilisateurs : le nom de l'hôte sur lequel fonctionne l'utilisateur, le nom d'utilisateur *(username)* de l'utilisateur sur cet hôte, et le serveur auquel le client est connecté.

2.2.2 Services

Chaque service se distingue des autres services par un nom de service composé d'un pseudonyme et d'un nom de serveur. Le pseudonyme a une longueur maximum de neuf (9) caractères. Voir au paragraphe 3.3.1 sur les règles de grammaire du protocole ce qui peut être ou non utilisé dans un pseudonyme. Le nom de serveur utilisé pour composer le nom de service est le nom du serveur auquel est connecté le service. En plus de ce nom de service, tous les serveurs DOIVENT connaître le type du service.

Les services diffèrent des utilisateurs par le format de leur identifiant, mais plus important, les services et les utilisateurs n'ont pas le même type d'accès au serveur : les services peuvent demander une partie ou la totalité des informations d'état globales qu'entretient un serveur, mais ont un ensemble plus restreint de commandes disponibles (voir les détails dans la

[RFC2812] "Protocole de client IRC") et il ne leur est pas permis de se joindre aux canaux. Finalement, les services ne sont normalement pas soumis au mécanisme de "contrôle de flux" décrit au paragraphe 5.8.

2.3 Canaux

Comme les services, les canaux ont une portée [RFC2811] et ne sont pas nécessairement connus de tous les serveurs. Lorsque l'existence d'un canal est connue d'un serveur, celui-ci DOIT garder trace des membres du canal, ainsi que des modes du canal.

3. Spécification du serveur IRC

3.1 Généralités

Le protocole tel qu'il est décrit ici est à utiliser sur les connexions de serveur à serveur. Pour les connexions de client à serveur, voir la spécification de la {RFC2812] sur le protocole de client IRC.

Il existe cependant plus de restrictions sur les connexions de client (qui sont considérées comme n'étant pas dignes de confiance) que sur les connexions de serveur.

3.2 Codes de caractères

Aucun jeu de caractères spécifique n'est spécifié. Le protocole se fonde sur un jeu de codes qui sont composés de huit (8) bits, faisant un octet. Chaque message peut être composé d'un nombre quelconque de ces octets ; cependant, certaines valeurs d'octet sont utilisées pour des codes de commandes qui agissent comme délimiteurs de message.

Indépendamment du fait d'être un protocole à 8 bits, les délimiteurs et les mots-clés sont tels que le protocole est principalement utilisable à partir d'un terminal US-ASCII et d'une connexion telnet.

À cause de l'origine scandinave de l'IRC, les caractères {}|^ sont considérés respectivement comme l'équivalent en minuscules des caractères []\~. Ceci est d'une importance critique pour la détermination de l'équivalence de deux pseudonymes ou de deux noms de canal.

3.3 Messages

Serveurs et clients s'envoient l'un l'autre des messages qui peuvent ou non générer une réponse. La plupart des communications entre serveurs ne génèrent pas de réponse, car les serveurs effectuent principalement des tâches d'acheminement pour les clients.

Chaque message IRC peut consister en jusqu'à trois parties principales : le préfixe (FACULTATIF), la commande, et les paramètres de commande (un maximum de quinze (15)). Le préfixe, la commande, et tous les paramètres sont séparés chacun par un caractère ASCII espace (0x20).

La présence d'un préfixe est indiquée par un seul caractère ASCII deux point (':', 0x3a) en tête, qui DOIT être le premier caractère du message lui-même. Il NE DOIT PAS y avoir de trou (d'espace) entre les deux points et le préfixe. Le préfixe est utilisé par les serveurs pour indiquer la véritable origine du message. Si le préfixe manque dans le message, on suppose qu'il a été généré à partir de la connexion d'où il a été reçu. Les clients NE DEVRAIENT PAS utiliser un préfixe lors de l'envoi d'un message à partir d'eux-mêmes ; si ils en utilisent un, le seul préfixe valide est le pseudonyme enregistré associé au client.

Lorsque un serveur reçoit un message, il DOIT identifier sa source en utilisant le préfixe (éventuellement supposé). Si le préfixe ne peut pas être trouvé dans la base de données interne du serveur, il DOIT être éliminé, et si le préfixe indique que le message provient d'un serveur (inconnu) la liaison d'où le message a été reçu DOIT être abandonnée. Abandonner une liaison dans de telles circonstances est un peu excessif mais nécessaire pour conserver l'intégrité du réseau et pour empêcher des problèmes futurs. Une autre condition d'erreur courante est que le préfixe trouvé dans la base de données interne du serveur identifie une source différente (normalement une source enregistrée à partir d'une liaison différente de celle sur laquelle le message est arrivé). Si le message a été reçu d'une liaison de serveur et si le préfixe identifie un client, un message KILL DOIT être produit pour le client et envoyé à tous les serveurs. Dans les autres cas, la liaison d'où le message est arrivé DEVRAIT être abandonnée pour les clients, et DOIT être abandonnée pour les serveurs. Dans tous les cas, le message DOIT être éliminé.

La commande DOIT être une commande IRC valide ou un numéro de trois (3) chiffres valide représenté dans un texte ASCII.

Les messages IRC sont toujours des lignes de caractères terminées par une paire CR-LF (Retour chariot – Saut à la ligne) et

ces messages NE DEVRONT PAS excéder 512 caractères, en comptant tous les caractères y compris le CR-LF de queue. Donc, il y a 510 caractères maximum qui sont permis pour la commande et ses paramètres. Aucune disposition n'existe pour la continuation des lignes de message. Voir à la section 5 d'autres détails sur les mises en œuvre actuelles.

3.3.1 Format de message en BNF augmenté

Les messages du protocole doivent être extraits du flux contigu d'octets. La solution actuelle est de désigner deux caractères, CR et LF, comme séparateurs de messages. Les messages vides sont ignorés en silence, ce qui permet l'utilisation de la séquence CR-LF entre les messages sans poser d'autre problème.

Le message extrait est analysé selon ses composants <préfixe>, <commande> et liste de paramètres (<params>).

La représentation en BNF augmenté de cela se trouve dans la [RFC2812] "Protocole client IRC".

Le préfixe étendu (["!" utilisateur "@" hôte]) NE DOIT PAS être utilisé dans les communications de serveur à serveur et n'est destiné qu'aux messages de client afin de fournir aux clients des informations utiles sur la provenance d'un message sans qu'il soir besoin d'interrogations supplémentaires.

3.4 Réponses numériques

La plupart des messages envoyés au serveur génèrent une réponse d'une certaine sorte. La réponse la plus courante est la réponse numérique, utilisée aussi bien pour les réponses d'erreur que pour les réponses normales. La réponse numérique DOIT être envoyée comme un message consistant en le préfixe de l'envoyeur, le numéro de trois chiffres, et la cible de la réponse. Une réponse numérique n'est pas permise à partir d'un client ; un tel message reçu par un serveur sera éliminé en silence. Sous tous les autres aspects, une réponse numérique est tout comme un message normal, sauf que le mot-clé est constitué des trois chiffres numériques plutôt que d'une chaîne de lettres. Une liste des différentes réponses est fournie dans la [RFC2812] "Protocole de client IRC".

4. Détails du message

Tous les messages reconnus par le serveur et le client IRC sont décrits dans la [RFC2812] "Protocole de client IRC".

Lorsque la réponse ERR_NOSUCHSERVER est retournée, elle signifie que la cible du message n'a pas pu être trouvée. Le serveur NE DOIT PAS envoyer d'autres réponses après cette erreur pour cette commande.

Le serveur auquel un client est connecté est obligé d'analyser le message complet, et de retourner toutes les erreurs appropriées. Si le serveur rencontre une erreur fatale lors de l'analyse d'un message, une erreur DOIT être renvoyée au client et l'analyse s'arrête. Une erreur fatale peut résulter d'une commande incorrecte, d'une destination qui est par ailleurs inconnue du serveur (les noms de serveur, client ou canal rentrent dans cette catégorie) d'une insuffisance de paramètres ou de privilèges incorrects.

Si un ensemble complet de paramètres est présenté, chacun DOIT alors être vérifié quant à sa validité et les réponses appropriées renvoyées au client. Dans le cas de messages qui utilisent des listes de paramètres avec la virgule comme séparateur d'élément, une réponse DOIT être envoyée pour chaque élément.

Dans les exemples ci-dessous, certains messages apparaissent avec le format complet :

:Nom COMMANDE liste_de_paramètres

De tels exemples représentent un message provenant de "nom" en transit entre des serveurs, où il est essentiel d'inclure le nom de l'envoyeur d'origine du message afin que les serveurs distants puissent renvoyer une réponse le long du chemin correct.

Les détails du message pour les communications de client à serveur sont décrites dans la [RFC2812] "Protocole de client IRC". Certains paragraphes des pages qui suivent s'appliquent à certains de ces messages, ce sont des ajouts aux spécifications de messages qui ne relèvent que de la communication de serveur à serveur, ou de la mise en œuvre de serveur. Les messages qui sont introduits ici ne sont utilisés que pour la communication de serveur à serveur.

4.1 Enregistrement de connexion

Les commandes décrites ici sont utilisées pour enregistrer une connexion avec un autre serveur IRC.

4.1.1 Message Mot de passe

Commande : PASS
Paramètres : <mot_de_passe> <version> <fanions> [<options>]

La commande PASS est utilisée pour établir un 'mot de passe de connexion'. Le mot de passe DOIT être établi avant toute tentative d'enregistrement de la connexion. En fait, cela signifie que les serveurs DOIVENT envoyer une commande PASS avant toute commande SERVER. Seule une (1) commande PASS DEVRA être acceptée de la part d'une connexion.

Les trois (3) derniers paramètres DOIVENT être ignorés si ils sont reçus d'un client (par exemple un utilisateur ou un service). Ils ne sont pertinents que lorsque ils sont reçus d'un serveur.

Le paramètre <version> est une chaîne d'au moins quatre (4) caractères, et jusqu'à quatorze (14) caractères. Les quatre (4) premiers caractères DOIVENT être des chiffres et indiquent la version du protocole connue par le serveur qui produit le message. Le protocole décrit dans le présent document est la version 2.10 qui est codée "0210". Les caractères FACULTATIFS restants dépendent de la mise en œuvre et devraient décrire le numéro de version du logiciel.

Le paramètre <fanions> est une chaîne pouvant aller jusqu'à cent (100) caractères. Elle est composée de deux sous chaînes séparées par le caractère "|" (%x7C). Si elle est présente, la première sous chaîne DOIT être le nom de la mise en œuvre. La mise en œuvre de référence (voir la Section 8, "Prise en charge et disponibilité actuelle") utilise la chaîne "IRC". Si une mise en œuvre différente est rédigée, et qu'elle a besoin d'un identifiant, cet identifiant devrait alors être enregistré par la publication d'une RFC. La seconde sous chaîne dépend de la mise en œuvre. Les deux sous chaînes sont FACULTATIVES, mais le caractère "|" est EXIGÉ. Le caractère "|" NE DOIT PAS apparaître dans l'une ou l'autre des sous chaînes.

Finalement, le dernier paramètre, <options>, est utilisé pour les options de la liaison. Les seules options définies par le protocole sont la compression de la liaison (qui utilise le caractère "Z") et un fanion de protection contre les abus (qui utilise le caractère "P"). Voir respectivement aux paragraphes 5.3.1.1 (Liaisons compressées de serveur à serveur) et 5.3.1.2 (Protections contre les abus) des informations complémentaires sur ces options.

Réponses numériques :
ERR_NEEDMOREPARAMS *(461)* ERR_ALREADYREGISTRED *(462)*

Exemple :
PASS moresecretpassword 0210010000 IRC|aBgH$ Z

4.1.2 Message Server

Commande : SERVER
Paramètres : <nom_de_serveur> <compte_de_bonds> <jeton> <info>

La commande SERVER est utilisée pour enregistrer un nouveau serveur. Une nouvelle connexion se présente elle-même comme un serveur à ses homologues. Ce message est aussi utilisé pour passer les données du serveur à l'ensemble du réseau. Lorsque un nouveau serveur est connecté au réseau, les informations le concernant DOIVENT être diffusées à l'ensemble du réseau.

Le paramètre <info> peut contenir des caractères espace.

<compte_de_bonds> est utilisé pour donner à tous les serveurs des informations internes sur la distance à laquelle se trouve chaque serveur. Les homologues locaux ont une valeur de 0, et chaque serveur passé incrémente la valeur. Avec une liste complète des serveurs, il serait possible de construire une carte de la totalité de l'arborescence des serveurs, mais les gabarits d'hôte en empêchent la réalisation.

Le paramètre <jeton> est un nombre non signé utilisé par les serveurs comme identifiant. Cet identifiant est ensuite utilisé pour référencer un serveur dans les messages NICK et SERVICE envoyés entre les serveurs. Les jetons de serveur n'ont de signification que pour l'échange de trafic en point à point pour lequel ils sont utilisés et DOIVENT être univoques pour cette connexion. Ils ne sont pas mondiaux.

Le message SERVER DOIT seulement être accepté de la part (a) d'une connexion qui est déjà à enregistrer et tente de s'enregistrer comme un serveur, ou (b) d'une connexion existante avec un autre serveur, auquel cas le message SERVER introduit un nouveau serveur derrière ce serveur.

La plupart des erreurs qui surviennent à la réception d'un message SERVER résultent en la clôture de la connexion par l'hôte de destination (cible de SERVER). À cause de la sévérité d'un tel événement, les réponses d'erreur sont normalement envoyées en utilisant la commande "ERROR" plutôt qu'un numéro.

Si un message SERVER est analysé et si il tente de présenter un serveur qui est déjà connu du serveur receveur, la connexion d'où ce message est arrivé DOIT être close (en suivant les procédures correctes) car un chemin dupliqué vers un serveur a été formé et la nature acyclique de l'arborescence IRC est rompue. Dans certaines conditions, la connexion d'où s'est enregistré le serveur déjà connu PEUT être fermée plutôt que l'autre. On notera que ce genre d'erreur peut aussi être le résultat du fonctionnement d'un second serveur, problème qui ne peut pas être réglé au sein du protocole et exige normalement une intervention humaine. Ce type de problème est particulièrement insidieux, car il peut très facilement déboucher sur l'isolement d'une partie du réseau IRC, avec un des deux serveurs connecté à chaque partition rendant donc impossible la réunification des deux parties.

Réponse numérique : ERR_ALREADYREGISTRED *(462)*

Exemples :
SERVER test.oulu.fi 1 1 :Experimental server ; Le nouveau serveur test.oulu.fi se présente et tente de s'enregistrer.
:tolsun.oulu.fi SERVER csd.bu.edu 5 34 :BU Central Server ; Le serveur tolsun.oulu.fi est la liaison montante pour csd.bu.edu qui est cinq bonds plus loin. Le jeton "34" sera utilisé par tolsun.oulu.fi lors de la présentation de nouveaux utilisateurs ou services connectés à csd.bu.edu.

4.1.3 Nick

Commande : NICK
Paramètres : <pseudonyme> <compte_de_bonds> <nom_d'utilisateur> <hôte> <jeton_de_serveur> <umode> <nom_réel>

Cette forme du message NICK NE DOIT PAS être permise aux connexions d'utilisateur. Cependant, elle DOIT être utilisée au lieu de la paire NICK/USER pour notifier aux autres serveurs les nouveaux utilisateurs qui se joignent au réseau IRC.

Ce message est en fait la combinaison de trois messages distincts : NICK, USER et MODE [RFC2812].

Le paramètre <compte de bonds> est utilisé par les serveurs pour indiquer à quelle distance se trouve un utilisateur de son serveur de rattachement. Une connexion locale a un compte de bonds de 0. La valeur du compte de bonds est incrémentée par chaque serveur passé.

Le paramètre <jeton_de_serveur> remplace le paramètre <nom_de_serveur> du message USER (voir au paragraphe 4.1.2 plus d'informations sur les jetons de serveur).

Exemples :
NICK syrk 5 kalt millennium.stealth.net 34 +i :Christophe Kalt ; Nouvel utilisateur avec le pseudonyme "syrk", le nom d'utilisateur "kalt", connecté de l'hôte "millennium.stealth.net" au serveur "34" ("csd.bu.edu" selon l'exemple précédent).
:krys NICK syrk ; Autre forme du message NICK, comme définie dans la [RFC2812] "Protocole de client IRC" et utilisée entre serveurs : krys a changé son pseudonyme en syrk.

4.1.4 Message Service

Commande : SERVICE
Paramètres : <nom_de_service> <jeton_de_serveur> <distribution> <type> <compte_de_bonds> <info>

La commande SERVICE est utilisée pour présenter un nouveau service. Cette forme du message SERVICE NE DEVRAIT PAS être permise à partir des connexions client (enregistrées ou non). Cependant, elle DOIT être utilisée entre serveurs pour notifier aux autres serveurs les nouveaux services qui se joignent au réseau IRC.

Le paramètre <jeton_de_serveur> est utilisé pour identifier le serveur auquel le service est connecté. (Voir au paragraphe 4.1.2 plus d'informations sur les jetons de serveur).

Le paramètre <compte_de_bonds> est utilisé par les serveurs pour indique à quelle distance se trouve un service de son

serveur de rattachement. Une connexion locale a un compte de bonds de 0. La valeur du compte de bonds est incrémentée par chaque serveur traversé.

Le paramètre <distribution> est à utiliser pour spécifier la visibilité d'un service. Le service peut n'être connu que des serveurs qui ont un nom correspondant à distribution. Pour qu'un serveur qui correspond ait connaissance du service, le chemin réseau entre ce serveur et le serveur auquel le service est connecté DOIT être composé de serveurs dont le noms correspond au gabarit. Le caractère générique "*" est utilisé lorsque on souhaite ne fixer aucune restriction.

Le paramètre <type> est actuellement réservé pour une utilisation future.

Réponses numériques :
ERR_ALREADYREGISTRED *(462)* ERR_NEEDMOREPARAMS *(461)*
ERR_ERRONEUSNICKNAME *(432)*
RPL_YOURESERVICE *(383)* RPL_YOURHOST *(002)*
RPL_MYINFO *(004)*

Exemple :
SERVICE dict@irc.fr 9 *.fr 0 1 : Le dictionnaire français r" enregistré sur le serveur "9" est annoncé à un autre serveur. Ce service ne sera disponible qu'aux serveurs dont le nom correspond à "*.fr".

4.1.5 Message Quit

Commande : QUIT
Paramètres : [<message QUIT>]

Une session client se termine avec un message quit. Le serveur DOIT clore la connexion avec un client qui envoie un message QUIT. Si un "message QUIT" est donné, il sera envoyé à la place du message par défaut, qui est le pseudonyme ou le nom de service.

Quand survient un "partage de réseau" (voir au paragraphe 4.1.6) le "message QUIT" se compose du nom des deux serveurs impliqués, séparés par une espace. Le premier nom est celui du serveur qui est toujours connecté et le second nom est soit celui du serveur qui a été déconnecté, soit celui du serveur auquel le client en partance était connecté :

<message QUIT> = ":" nom_du_serveur ESPACE nom_du_serveur

Comme le "message QUIT" a une signification particulière pour les "partages de réseau", les serveurs NE DEVRAIENT PAS permettre à un client d'utiliser un <message QUIT> dans le format décrit ci-dessus.

Si, pour quelque autre raison, une connexion de client est close sans que le client produise une commande QUIT (par exemple le client meurt et EOF s'affiche sur la prise) il est EXIGÉ du serveur qu'il remplisse le message QUIT avec une sorte de message reflétant la nature de l'événement qui l'a causé. Normalement, cela se fait en rapportant une erreur spécifique du système.

Réponses numériques : Aucune.

Exemple :
:WiZ QUIT :Parti déjeuner ; Format de message préféré.

4.1.6 Message Server quit

Commande : SQUIT
Paramètres : <serveur> <commentaire>

Le message SQUIT a deux utilisations distinctes.

La première (décrite dans la [RFC2812] "Relais pour la causette Internet : protocole client") permet aux opérateurs de casser une liaison de serveur locale ou distante. Cette forme du message est aussi éventuellement utilisée par les serveurs pour casser une liaison de serveur distante.

La seconde utilisation de ce message est nécessaire pour informer les autres serveurs lorsque survient un "partage de réseau", en d'autres termes, pour informer les autres serveurs des serveurs morts ou en partance. Si un serveur souhaite casser la connexion avec un autre serveur, il DOIT envoyer un message SQUIT à l'autre serveur, en utilisant le nom de

l'autre serveur comme paramètre <serveur>, qui va alors clore sa connexion avec le serveur en partance.

Le paramètre <commentaire> est rempli par les serveurs qui DEVRAIENT y placer une erreur ou message similaire.

Il est EXIGÉ des deux serveurs qui sont sur l'un et l'autre côté de la connexion en cours de clôture qu'ils envoient un message SQUIT (à toutes les autres connexions de serveur) pour tous les autres serveurs qui sont considérés comme étant derrière cette liaison.

De même, un message QUIT PEUT être envoyé aux autres serveurs toujours connectés au nom de tous les clients qui sont derrière cette liaison en cours de fermeture. En plus de cela, tous les membres du canal d'un canal qui a perdu un membre à cause du "partage" DOIVENT être destinataires d'un message QUIT. Les messages aux membres du canal sont générés par le serveur local de chaque client.

Si une connexion de serveur se termine de façon prématurée (par exemple, le serveur de l'autre côté de la liaison est mort) il est EXIGÉ du serveur qui détecte cette déconnexion qu'il informe le reste du réseau que la connexion a fermé et qu'il remplisse le champ <commentaire> avec quelque chose d'approprié.

Lorsque un client est retiré par suite d'un message SQUIT, le serveur DEVRAIT ajouter le pseudonyme à la liste des indisponibles temporaires pour essayer d'empêcher de futures collisions de pseudonymes. Voir au paragraphe 5.7 (Suivi des pseudonymes récemment utilisés) des informations complémentaires sur cette procédure.

Réponses numériques :
ERR_NOPRIVILEGES *(481)* ERR_NOSUCHSERVER *(402)*
ERR_NEEDMOREPARAMS *(451)*

Exemple :
SQUIT tolsun.oulu.fi :Mauvaise liaison ? ; la liaison de serveur tolson.oulu.fi a été clôturée à cause d'une "mauvaise liaison".
:Trillian SQUIT cm22.eng.umd.edu :Serveur hors contrôle ; message de Trillian pour déconnecter "cm22.eng.umd.edu" du réseau parce que le "serveur est hors contrôle".

4.2 Opérations du canal

Ce groupe de messages concerne la manipulation des canaux, leurs propriétés (modes de canal) et leur contenu (normalement des utilisateurs). Un certain nombre de conditions de concurrence sont inévitables dans la mise en œuvre de cela lorsque les utilisateurs à l'extrémité opposée du réseau envoient des commandes qui vont finir par s'opposer. Il est aussi EXIGÉ que les serveurs conservent un historique des pseudonymes pour s'assurer que partout où un paramètre <pseudonyme> est donné, le serveur vérifie dans son historique qu'il n'a pas été changé récemment.

4.2.1 Message Join

Commande : JOIN
Paramètres : <canal>[%x7 <modes>] *("," <canal>[%x7 <modes>])

La commande JOIN est utilisée par le client pour commencer à écouter sur un canal spécifique. La vérification qu'un client est ou non autorisé à se joindre à un canal n'est effectuée que par le serveur local auquel le client est connecté ; tous les autres serveurs ajoutent automatiquement l'utilisateur au canal lorsque la commande est reçue d'autres serveurs.

Facultativement, le statut d'utilisateur (modes de canal 'O', 'o', et 'v') sur le canal peut être ajouté au nom de canal en utilisant un contrôle G (^G ou ASCII 7) comme séparateur. De telles données DOIVENT être ignorées si le message n'est pas reçu d'un serveur. Ce format NE DOIT PAS être envoyé aux clients, il ne peut être utilisé qu'entre les serveurs et DEVRAIT être évité.

La commande JOIN DOIT être diffusée à tous les serveurs afin que chacun sache où trouver les utilisateurs qui sont sur le canal. Cela permet une livraison optimale des messages PRIVMSG et NOTICE sur le canal.

Réponses numériques :
ERR_NEEDMOREPARAMS *(461)* ERR_BANNEDFROMCHAN *(474)* ERR_INVITEONLYCHAN *(473)*
ERR_BADCHANNELKEY *(475)* ERR_CHANNELISFULL *(471)* ERR_BADCHANMASK *(476)*
ERR_NOSUCHCHANNEL *(403)* ERR_TOOMANYCHANNELS *(405)* ERR_TOOMANYTARGETS *(407)*
ERR_UNAVAILRESOURCE *(437)* RPL_TOPIC *(332)*

Exemple :
:WiZ JOIN #Twilight_zone ; message JOIN provenant de WiZ.

4.2.2 Message Njoin

Commande : NJOIN
Paramètres : <canal> ["@@" / "@"] ["+"] <pseudonyme> *("," ["@@" / "@"] ["+"] <pseudonyme>)

Le message NJOIN est utilisé seulement entre serveurs. Si un tel message est reçu d'un client, il DOIT être ignoré. Il est utilisé lorsque deux serveurs se connectent ensemble pour échanger la liste des membres du canal de chaque canal.

Bien que la même fonction puisse être effectuée en utilisant une succession de JOIN, ce message DEVRAIT être utilisé à la place car il est plus efficace. Le préfixe "@@" indique que l'utilisateur est le "créateur du canal", le caractère "@" seul indique un "opérateur de canal", et le caractère '+' indique que l'utilisateur a le privilège de parler.

Réponses numériques :
ERR_NEEDMOREPARAMS *(461)* ERR_NOSUCHCHANNEL *(403)* ERR_ALREADYREGISTRED *(462)*

Exemples :
:ircd.stealth.net NJOIN #Twilight_zone :@WiZ,+syrk,avalon ; message NJOIN de ircd.stealth.net annonçant les usagers qui se joignent au canal #Twilight_zone : WiZ avec le statut d'opérateur du canal, syrk avec le privilège de parler et avalon sans aucun privilège.

4.2.3 Message Mode

Le message MODE est une commande au double objet dans IRC. Elle permet aussi bien aux noms d'utilisateur qu'aux canaux de changer leur mode.

Lors de l'analyse des messages MODE, il est RECOMMANDÉ que le message entier soit d'abord analysé, et ensuite les changements qui résultent de sa passation.

Il est EXIGÉ des serveurs qu'ils soient capables de changer les modes de canal afin que puissent être créés le "créateur du canal" et les "opérateurs de canal".

5. Détails de mise en œuvre

Au moment de la rédaction du présent document, la seule mise en œuvre de ce protocole est le serveur IRC, version 2.10. Les versions antérieures peuvent mettre en œuvre tout ou partie des commandes décrites dans ce document avec des messages NOTICE à la place de nombreuses réponses numériques. Malheureusement, du fait des exigences de rétro compatibilité, la mise en œuvre de certaines parties du présent document varie selon ce qui est affiché. Une différence notable est :

* la reconnaissance que tout LF ou CR en tout point d'un message marque la fin de ce message (au lieu d'exiger un CR-LF);

Le reste de cette section traite de questions qui n'ont d'importance que pour ceux qui souhaitent mettre en œuvre un serveur mais certaines parties s'appliquent aussi directement aux clients.

5.1 Vivacité de la connexion

Pour détecter quand une connexion est morte ou ne répond plus, le serveur DOIT interroger chacune de ses connexions. La commande PING (voir la [RFC2812] "Protocole client IRC") est utilisée si le serveur n'obtient pas une réponse de son homologue dans un certain délai.

Si une connexion ne répond pas dans les temps, sa connexion est close en utilisant les procédures appropriées.

5.2 Acceptation d'une connexion de client à serveur

5.2.1 Utilisateurs

Lorsque un serveur réussit à enregistrer une nouvelle connexion d'utilisateur, il est EXIGÉ qu'il envoie à l'utilisateur un message non ambigu déclarant les identifiants d'utilisateur sous lesquels il a été enregistré (RPL_WELCOME) le nom du serveur et sa version (RPL_YOURHOST) les informations sur la naissance du serveur (RPL_CREATED) les modes disponibles d'utilisateur et de canal (RPL_MYINFO) et il PEUT envoyer tout message d'introduction qu'il pourrait juger approprié.

En particulier le serveur DEVRA envoyer le compte actuel de service/serveur/utilisateur (suivant la réponse LUSERS) et finalement le MOTD (si il en est un, conformément à la réponse MOTD).

Après avoir traité l'enregistrement, le serveur DOIT ensuite envoyer aux autres serveurs le pseudonyme du nouvel utilisateur (message NICK) les autres informations fournies par lui-même (message USER) et celles que le serveur pourrait découvrir (à partir des serveurs du DNS). Le serveur NE DOIT PAS envoyer ces informations avec une paire de messages NICK et USER, comme défini dans la [RFC2812] "Protocole de client IRC", mais DOIT plutôt tirer parti du message NICK étendu défini au paragraphe 4.1.3.

5.2.2 Services

Ayant réussi à enregistrer une nouvelle connexion de service, le serveur est soumis aux mêmes sortes d'exigences qu'un utilisateur. Les services étant quelque peu différents, seules les réponses suivantes sont envoyées : RPL_YOURESERVICE *(383)*, RPL_YOURHOST *(002)*, RPL_MYINFO *(004)*.

Après avoir traité cela, le serveur DOIT ensuite envoyer aux autres serveurs (message SERVICE) le pseudonyme et les autres informations sur le nouveau service telles que fournies par le service (message SERVICE) et telles que le serveur pourrait les découvrir (auprès des serveurs du DNS).

5.3 Établissement d'une connexion serveur-serveur

Le processus d'établissement d'une connexion de serveur à serveur est plein de dangers car il y a de nombreux domaines possibles où les problèmes peuvent survenir – dont le moindre est celui d'une condition de concurrence.

Après qu'un serveur a reçu une connexion suivie par une paire PASS/SERVER qui a été reconnue comme valide, le serveur DEVRAIT alors répondre avec ses propres informations PASS/SERVER pour cette connexion ainsi que toutes les autres informations d'état dont il a connaissance comme décrit ci-dessous.

Lorsque le serveur qui a l'initiative reçoit une paire PASS/SERVER, il vérifie lui aussi que le serveur qui répond est correctement authentifié avant d'accepter la connexion avec ce serveur.

5.3.1 Options de liaison

Les liaisons de serveur se fondent sur un protocole commun (défini par le présent document) mais une liaison particulière PEUT établir des options spécifiques en utilisant le message PASS (voir au paragraphe 4.1.1).

5.3.1.1 Liaisons compressées de serveur à serveur

Si un serveur souhaite établir une liaison compressée avec son homologue, il DOIT établir le fanion 'Z' dans les paramètres d'options du message PASS. Si les deux serveurs demandent la compression et si les deux serveurs sont capables d'initialiser les deux flux compressés, alors le reste de la communication sera compressé. Si un des serveurs échoue à initialiser le flux, il va envoyer un message ERROR non compressé à son homologue et clore la connexion.

Le format de données utilisé pour la compression est décrit par les [RFC1950], [RFC1951] et [RFC1952].

5.3.1.2 Protections contre les abus

La plupart des serveurs mettent en œuvre diverses sortes de protection contre de possibles comportements abusifs de la part de parties qui ne sont pas de confiance (normalement des utilisateurs). Sur certains réseaux, de telles protections sont indispensables ; sur d'autres, elles sont superflues. Pour exiger que tous les serveurs mettent en œuvre et activent de tels dispositifs sur un réseau particulier, le fanion 'P' est utilisé lorsque deux serveurs se connectent. Si ce fanion est présent, il signifie que les protections de serveur sont activées, et que le serveur EXIGE que toutes ses liaisons de serveur les activent aussi.

Les protections qu'on trouve habituellement sont décrites aux paragraphes 5.7 (Suivi des pseudonymes utilisés récemment) et 5.8 (Contrôle des flux de clients).

5.3.2 Échange d'informations d'état à la connexion

L'ordre des informations d'état échangées entre les serveurs est essentiel. L'ordre EXIGÉ est le suivant :
* tous les serveurs connus ;
* toutes les informations de client connues ;
* toutes les informations de canal connues.

Les informations qui concernent les serveurs sont envoyées via des messages SERVER supplémentaires, les informations de client avec les messages NICK et SERVICE, et les informations des canaux avec les messages NJOIN/MODE.

Note : Les sujets des canaux NE DEVRAIENT PAS être échangés ici parce que la commande TOPIC écrase toutes les anciennes informations de sujet, de sorte qu'au mieux, les deux côtés de la connexion vont échanger leurs sujets.

En passant en premier les informations d'état sur les serveurs, toute collision avec les serveurs qui existe déjà survient avant les collisions de pseudonyme causées par un second serveur qui introduit un pseudonyme particulier. Du fait que le réseau IRC n'est capable d'exister que comme un graphe acyclique, il est possible que le réseau se soit déjà reconnecté dans une autre localisation. Dans ce cas, le lieu où survient la collision de serveurs indique où le réseau doit se partager.

5.4 Terminaison d'une connexion serveur-client

Lorsque une connexion client ferme de façon inattendue, un message QUIT est généré au nom du client par le serveur auquel le client était connecté. Aucun autre message n'est à générer ni utiliser.

5.5 Terminaison des connexions serveur-serveur

Si une connexion serveur-serveur est close, via une commande SQUIT ou pour des causes "naturelles", le reste du réseau IRC connecté DOIT avoir ses informations mises à jour par le serveur qui a détecté la fermeture. Le serveur qui termine envoie alors une liste de messages SQUIT (un pour chaque serveur derrière cette connexion). (Voir au paragraphe 4.1.6 (SQUIT).)

5.6 Suivi des changements de pseudonyme

Il est EXIGÉ de tous les serveurs IRC qu'ils conservent un historique des récents changements de pseudonymes. C'est important pour permettre au serveur d'avoir une chance de rester au contact lorsque survient une condition de concurrence avec des commandes qui les manipulent. Les messages qui DOIVENT faire le suivi des changements de pseudonymes sont :
* KILL (le pseudonyme à déconnecter)
* MODE (+/- o,v sur les canaux)
* KICK (le pseudonyme à retirer du canal)

Aucune autre commande n'a besoin de vérifier les changements de pseudonyme.

Dans les cas ci-dessus, il est exigé du serveur qu'il vérifie d'abord l'existence du pseudonyme, puis qu'il vérifie dans son historique quel pseudonyme (s'il en est !) lui appartient maintenant. Cela réduit les chances de conditions de concurrence mais elles peuvent encore survenir lorsque la clôture du serveur affecte le mauvais client. En effectuant un suivi de changement à partir d'une des commandes ci-dessus, il est RECOMMANDÉ qu'une gamme de temps soit donnée et que les entrées qui sont trop anciennes soient ignorées.

Pour un historique raisonnable, un serveur DEVRAIT être capable de conserver les pseudonymes précédents pour tous les clients qu'il connaît si ils décidaient tous de changer. Cette taille est limitée par d'autres facteurs (comme celle de la mémoire, etc.).

5.7 Suivi des pseudonymes récemment utilisés

Ce mécanisme est couramment appelé "délai de pseudonyme", et il a été prouvé qu'il réduit significativement le nombre de collisions de pseudonymes résultant des"partages de réseau"/reconnexions ainsi que des abus.

En plus de faire le suivi des changements de pseudonymes, les serveurs DEVRAIENT garder trace des pseudonymes qui

ont été récemment utilisés et ont été libérés par suite d'un "partage de réseau" ou d'un message KILL. Ces pseudonymes sont alors indisponibles pour les clients locaux du serveur et ne peuvent pas être réutilisés (même si ils ne sont pas actuellement utilisés) pendant un certain temps.

La durée pendant laquelle un pseudonymes reste indisponible DEVRAIT être réglée en prenant de nombreux facteurs en considération parmi lesquels sont la taille (en nombre d'utilisateurs) du réseau IRC, et la durée habituelle des "partages de réseau". Il DEVRAIT être uniforme sur tous les serveurs d'un même réseau IRC.

5.8 Contrôle des flux de clients

Avec un grand réseau de serveurs IRC interconnectés, il est assez facile à un seul client rattaché au réseau de fournir un flux continu de messages qui résultent non seulement en l'inondation du réseau, mais aussi à dégrader le niveau de service fourni aux autres. Plutôt que d'exiger de toutes les 'victimes' qu'elles assurent leur propre protection, la protection contre l'inondation é été inscrite dans le serveur et s'applique à tous les clients sauf les services. L'algorithme actuel est le suivant :

* vérifier si le "temporisateur de message" du client est inférieur à l'heure actuelle (le régler égal si il l'est) ;
* lire toutes les données présentes provenant du client ;
* lorsque le temporisateur est inférieur à dix (10) secondes avant l'heur actuelle, analyser tous les messages présents et pénaliser le client de deux (2) secondes pour chaque message ;
* des pénalités supplémentaires PEUVENT être utilisées pour des commandes spécifiques qui génèrent beaucoup de trafic à travers le réseau.

Cela signifie essentiellement que le client peut envoyer un (1) message toutes les deux (2) secondes sans être affecté. Les services PEUVENT aussi être soumis à ce mécanisme.

5.9 Recherches non bloquantes

Dans un environnement de temps réel, il est essentiel qu'un processeur de serveur subisse le moins d'attente possible afin que tous les clients soient servis de façon équitable. Cela exige évidemment un fonctionnement non bloquant sur toutes les opérations de lecture/écriture du réseau. Pour les connexions normales de serveur, cela n'est pas difficile, mais il y a d'autres opérations de prise en charge qui peuvent causer le blocage du serveur (comme des lectures de disque). Lorsque possible, de telles activités DEVRAIENT être effectuées avec une brève temporisation.

5.9.1 Recherches de noms d'hôte (DNS)

L'utilisation des bibliothèques standard de résolveur de Berkeley et autres a signifié de gros retards dans certains cas où les réponses ont dépassé la temporisation. Pour éviter cela, un ensemble distinct de sous-programmes du DNS a été rédigé pour les mises en œuvre actuelles. Les sous-programmes ont été établis pour des opérations IO non bloquantes avec antémémoire locale, et ensuite interrogés à partir de l'intérieur de la boucle IO du serveur principal.

5.9.2 Recherches de nom d'utilisateur (Ident)

Bien qu'il y ait de nombreuses bibliothèques d'identités (qui mettent en œuvre le "protocole d'identification" de la [RFC1413]) à utiliser et inclure dans les autres programmes, cela a causé des problèmes car elles fonctionnent de manière synchrone et il en a résulté de fréquents délais. La solution a été là encore d'écrire un ensemble de sous-programmes qui coopèrent avec le reste du serveur et fonctionnent en utilisant un IO non bloquant.

6. Problèmes en cours

Un certain nombre de problèmes ont été reconnus sur ce protocole, dont on espère que tous seront résolus dans un futur proche lors de sa révision. Les travaux sont actuellement en cours pour trouver des solutions à ces problèmes.

6.1 Adaptabilité

Il est largement reconnu que ce protocole ne s'adapte pas suffisamment bien lorsque il est utilisé dans une grande zone. Le principal problème vient de l'exigence que tous les serveurs sachent tout sur tous les autres serveurs et clients et que les informations les concernant soient mises à jour aussitôt qu'elles changent. Il est aussi souhaitable de garder faible le nombre de serveurs afin que la longueur du chemin entre deux points quelconques reste minimale et que l'arborescence de développement ait autant d'embranchements que possible.

6.2 Étiquettes

Le protocole IRC actuel a quatre types d'étiquettes : le pseudonyme, le nom de canal, le nom de serveur et le nom de service. Chacun de ces quatre types a son propre domaine et aucune duplication n'est permise à l'intérieur de ce domaine. Actuellement, il est possible aux utilisateurs de prendre une étiquette pour l'un des trois premiers types, d'où résultent des collisions. Il est largement reconnu que cela doit être retravaillé, avec un plan pour l'unicité des noms pour des pseudonymes qui n'entrent pas en collision ainsi qu'une solution permettant une arborescence cyclique.

6.2.1 Pseudonymes

L'idée du pseudonyme sur IRC est d'utilisation très pratique pour les utilisateurs pour parler les uns aux autres en dehors d'un canal, mais il n'y a qu'un espace fini de pseudonymes, et étant donné ce qu'ils sont, il n'est pas anormal que plusieurs personnes veuillent utiliser le même pseudonyme. Si un pseudonyme est choisi par deus personnes qui utilisent ce protocole, une des deux va échouer ou les deux seront retirées par l'utilisation de KILL (voir le paragraphe 3.7.1 de la [RFC2812] "Protocole client IRC").

6.2.2 Canaux

La disposition actuelle des canaux exige que tous les serveurs connaissent tous les canaux, leurs habitants et leurs propriétés. En plus de la mauvaise adaptation, la question de la confidentialité est aussi problématique. Une collision de canaux est traitée comme un événement inclusif (des gens des deux réseaux sur un canal avec un nom commun sont considérés comme en étant membres) plutôt qu'exclusif comme on le fait pour résoudre les collisions de pseudonymes.

Le présent protocole définit des "canaux sûrs" qui sont très peu soumis à des collisions de canaux. Les autres types de canaux sont conservés pour la rétro compatibilité.

6.2.3 Serveurs

Bien que le nombre de serveurs soit normalement faible par rapport au nombre d'utilisateurs et de canaux, il est aussi EXIGÉ actuellement qu'ils soient connus mondialement, chacun séparément ou caché derrière un gabarit.

6.3 Algorithmes

Dans certains endroits du code serveur, il n'a pas été possible d'éviter des algorithmes N^2 comme pour vérifier la liste des canaux d'un ensemble de clients.

Dans les versions de serveur actuelles, il n'y a que peu de vérifications de cohérence des bases de données, la plupart du temps, chaque serveur suppose qu'un serveur du voisinage est correct. Cela ouvre la porte à de gros problèmes si un serveur qui se connecte est fautif ou essaye autrement d'introduire des contradictions dans le réseau existant.

Actuellement, à cause du manque d'étiquettes internes et mondiales univoques, il y a une multitude de conditions de concurrence. Ces conditions de concurrence proviennent généralement du problème du temps qu'il faut pour que les messages traversent et affectent le réseau IRC. Même en passant à des étiquettes univoques, il y a des problèmes avec l'interruption de commandes en rapport avec le canal.

7. Considérations pour la sécurité

7.1 Authentification

Les serveurs n'ont que deux moyens d'authentifier les connexions entrantes : le mot de passe en clair, et les recherches dans le DNS. Bien que ces méthodes soient faibles et qu'il soit largement reconnu qu'elles ne sont pas sûres, leur combinaison a été prouvée suffisante par le passé :

* les réseaux public permettent normalement les connexions d'utilisateurs avec peu de restrictions, sans exiger d'authentification précise ;
* les réseaux privés qui fonctionnent dans un environnement contrôlé utilisent souvent un mécanisme d'authentification développé en interne non disponible sur l'Internet, des serveurs d'identité fiables [RFC1413], ou d'autres mécanismes brevetés.

Les mêmes commentaires s'appliquent à l'authentification des opérateurs IRC.

On devrait aussi noter que bien qu'il n'y ait pas eu de réelle demande au fil des ans pour une plus forte authentification, et

pas d'effort réel pour fournir de meilleurs moyens pour authentifier les utilisateurs de façon sûre, le protocole actuel offre suffisamment pour être capable d'adjoindre facilement des méthodes d'authentification externes fondées sur les informations qu'un client peut soumettre au serveur au moment de la connexion : pseudonyme, nom d'utilisateur, mot de passe.

7.2 Intégrité

Comme les messages PASS et OPER du protocole IRC sont envoyés en clair, un mécanisme de chiffrement de flux (comme "le protocole TLS" [RFC2246]) pourrait être utilisé pour protéger ces transactions.

8. Prise en charge et disponibilité actuelle

Listes de diffusion de discussion en rapport avec IRC :
Discussion générale : ircd-users@irc.org
Développement de protocole : ircd-dev@irc.org

Mises en œuvre de logiciels :
ftp://ftp.irc.org/irc/server
ftp://ftp.funet.fi/pub/unix/irc
ftp://coombs.anu.edu.au/pub/irc

Groupe de nouvelles : alt.irc

9. Remerciements

Certaines parties du présent document ont été copiées de la [RFC1459] qui a été la première à documenter formellement le protocole IRC. Il a aussi bénéficié de nombreuses séances de révision et de commentaires. En particulier, les personnes suivantes ont fourni des contributions significatives à ce document : Matthew Green, Michael Neumayer, Volker Paulsen, Kurt Roeckx, Vesa Ruokonen, Magnus Tjernstrom, Stefan Zehl.

10. Références

[RFC1413] M. St. Johns, "Protocole d'identification", février 1993. *(P.S.)*

[RFC1459] J. Oikarinen et D. Reed, "Protocole Internet de relais de débats", mai 1993. *(Exp., MàJ par 2810-13)*

[RFC1950] P. Deutsch et J-L Gailly, "Spécification du format ZLIB de données compressées, version 3.3", mai 1996.

[RFC1951] P. Deutsch, "Spécification du format DEFLATE de données compressées, version 1.3", mai 1996.

[RFC1952] P. Deutsch, "Spécification du format de fichier GZIP version 4.3", mai 1996. *(Information)*

[RFC2119] S. Bradner, "Mots clés à utiliser dans les RFC pour indiquer les niveaux d'exigence", BCP 14, mars 1997.

[RFC2234] D. Crocker et P. Overell, "BNF augmenté pour les spécifications de syntaxe : ABNF", novembre 1997. (*Obsolète, voir* RFC5234)

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

[RFC2810] C. Kalt, "Relais pour la causette Internet : architecture", avril 2000. *(Information)*

[RFC2811] C. Kalt, "Relais pour la causette Internet : gestion de canal", avril 2000. *(Information)*

[RFC2812] C. Kalt, "Relais pour la causette Internet : protocole client", avril 2000. *(Information)*

11. Adresse de l'auteur

Christophe Kalt
99 Teaneck Rd, Apt #117
Ridgefield Park, NJ 07660
USA
mél : kalt@stealth.net

12. Déclaration complète de droits de reproduction

Remerciement
Le financement de la fonction d'éditeur des RFC est actuellement assuré par la Internet Society.