UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

UIT-T V.44

(11/2000)

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

SÉRIE V: COMMUNICATIONS DE DONNÉES SUR LE RÉSEAU TÉLÉPHONIQUE

Contrôle d'erreur

Procédures de compression de données

Recommandation UIT-T V.44

(Antérieurement Recommandation du CCITT)

RECOMMANDATIONS UIT-T DE LA SÉRIE V

COMMUNICATIONS DE DONNÉES SUR LE RÉSEAU TÉLÉPHONIQUE

Considérations générales	V.1–V.9
Interfaces et modems pour la bande vocale	V.10–V.34
Modems à large bande	V.35–V.39
Contrôle d'erreur	**V.40–V.49**
Qualité de transmission et maintenance	V.50–V.59
Transmission simultanée de données et d'autres signaux	V.60–V.99
Interfonctionnement avec d'autres réseaux	V.100–V.199
Spécifications de la couche interface pour les communications de données	V.200–V.249
Procédures de commande	V.250–V.299
Modems sur circuits numériques	V.300–V.399

Pour plus de détails, voir la Liste des Recommandations de l'UIT-T.

Recommandation UIT-T V.44

Procédures de compression de données

Résumé

La présente Recommandation décrit un algorithme de compression de données à utiliser dans les équipements ETCD. Cet algorithme est plus performant que l'algorithme V.42 *bis* pour de nombreux types de données. En plus de la méthode normale de type flux, l'algorithme comporte une méthode qui peut compresser efficacement les données déjà contenues dans des paquets.

Source

La Recommandation V.44 de l'UIT-T, élaborée par la Commission d'études 16 (2001-2004) de l'UIT-T, a été approuvée le 17 novembre 2000 selon la procédure définie dans la Résolution 1 de l'AMNT.

AVANT-PROPOS

L'UIT (Union internationale des télécommunications) est une institution spécialisée des Nations Unies dans le domaine des télécommunications. L'UIT-T (Secteur de la normalisation des télécommunications) est un organe permanent de l'UIT. Il est chargé de l'étude des questions techniques, d'exploitation et de tarification, et émet à ce sujet des Recommandations en vue de la normalisation des télécommunications à l'échelle mondiale.

L'Assemblée mondiale de normalisation des télécommunications (AMNT), qui se réunit tous les quatre ans, détermine les thèmes d'étude à traiter par les Commissions d'études de l'UIT-T, lesquelles élaborent en retour des Recommandations sur ces thèmes.

L'approbation des Recommandations par les Membres de l'UIT-T s'effectue selon la procédure définie dans la Résolution 1 de l'AMNT.

Dans certains secteurs des technologies de l'information qui correspondent à la sphère de compétence de l'UIT-T, les normes nécessaires se préparent en collaboration avec l'ISO et la CEI.

NOTE

Dans la présente Recommandation, l'expression "Administration" est utilisée pour désigner de façon abrégée aussi bien une administration de télécommunications qu'une exploitation reconnue.

DROITS DE PROPRIÉTÉ INTELLECTUELLE

L'UIT attire l'attention sur la possibilité que l'application ou la mise en œuvre de la présente Recommandation puisse donner lieu à l'utilisation d'un droit de propriété intellectuelle. L'UIT ne prend pas position en ce qui concerne l'existence, la validité ou l'applicabilité des droits de propriété intellectuelle, qu'ils soient revendiqués par un Membre de l'UIT ou par une tierce partie étrangère à la procédure d'élaboration des Recommandations.

A la date d'approbation de la présente Recommandation, l'UIT avait été avisée de l'existence d'une propriété intellectuelle protégée par des brevets à acquérir pour mettre en œuvre la présente Recommandation. Toutefois, comme il ne s'agit peut-être pas de renseignements les plus récents, il est vivement recommandé aux responsables de la mise en œuvre de consulter la base de données des brevets du TSB.

TABLE DES MATIÈRES

Recommandation UIT-T V.44

Procédures de compression de données

1 Domaine d'application

1.1 Généralités

La présente Recommandation décrit une procédure de compression de données sans perte à utiliser avec les équipements de terminaison de circuit de données (ETCD) de la série V.

Les principales caractéristiques de la procédure de compression de données sont les suivantes:

a) une procédure de compression fondée sur un algorithme qui code des chaînes de caractères provenant d'un équipement terminal de traitement de données (ETTD) sous forme de codes binaires de longueur variable;

b) une procédure de décodage qui redonne les chaînes de caractères à partir des codes binaires de longueur variable reçus;

c) un mécanisme de constitution de chaînes qui étend rapidement les chaînes existantes;

d) un mode de fonctionnement transparent invoqué automatiquement lorsque des données incompressibles sont détectées.

L'Annexe B décrit la mise en œuvre de cette procédure de compression de données dans des réseaux de transmission par paquet.

Le paragraphe 8 récapitule l'ensemble des paramètres utilisés dans la présente Recommandation.

Le paragraphe I.7 récapitule les principales différences entre les algorithmes des UIT-T V.44 et V.42 *bis*.

La présente Recommandation contient des exemples donnés à titre d'illustration; chaque fois qu'il semble y avoir une contradiction entre un exemple et le texte normatif, c'est le texte normatif qui prime.

1.2 Procédures de correction d'erreur requises

Pour que la fonction de compression de données soit exécutée correctement, il faut implémenter une procédure de correction d'erreur entre les deux entités utilisant la présente Recommandation. Dans le cas des Recommandations de la série V, il faut mettre en œuvre les procédures de correction d'erreur de la procédure d'accès de liaison pour modems (LAPM, *link access procedure for modems*) définies dans l'UIT-T V.42 ou les procédures de correction d'erreur de l'UIT-T V.76 ou V.120.

NOTE – Les erreurs binaires non détectées entraîneront une mauvaise exécution de la fonction de compression de données. L'utilisation d'une séquence de contrôle de trame (FCS, *frame check sequence*) de 32 bits, telle que définie dans l'ISO/IEC 13239, réduit nettement la probabilité que de telles erreurs se produisent et peut être souhaitable dans des environnements où les dégradations sont importantes. Cette séquence FCS de 32 bits est facultative dans la procédure LAPM V.42.

1.3 Equipement ETCD employant la compression de données

La fonction de compression de données peut être employée dans un équipement ETCD utilisant une correction d'erreur, comme indiqué sur la Figure 1 et décrit au paragraphe 5. Les éléments d'un équipement ETCD de type série V utilisant une correction d'erreur sont spécifiés dans l'UIT-T V.42.

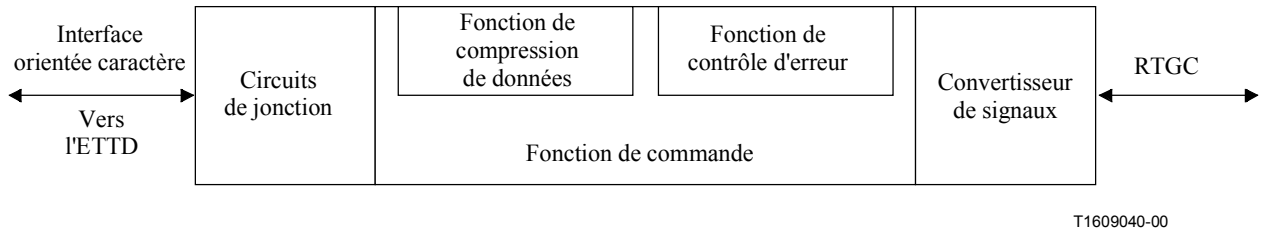

Figure 1/V.44 – Equipement ETCD employant une compression de données et un contrôle d'erreur

2 Références normatives

La présente Recommandation se réfère à certaines dispositions des Recommandations UIT-T et textes suivants qui, de ce fait, en sont partie intégrante. Les versions indiquées étaient en vigueur au moment de la publication de la présente Recommandation. Toute Recommandation ou tout texte étant sujet à révision, les utilisateurs de la présente Recommandation sont invités à se reporter, si possible, aux versions les plus récentes des références normatives suivantes. La liste des Recommandations de l'UIT-T en vigueur est régulièrement publiée.

– UIT-T V.42 (1996), *Procédures de correction d'erreur pour les équipements de terminaison de circuits de données utilisant la conversion asynchrone/synchrone*.

– UIT-T V.42 *bis* (1990), *Procédures de compression des données pour les équipements de terminaison du circuit de données (ETCD) utilisant des procédures de correction d'erreur*.

– UIT-T V.120 (1996), *Prise en charge par un RNIS d'un équipement terminal de traitement de données muni d'interfaces de type série V permettant un multiplexage statistique*.

– ISO/CEI 13239:2000 *Technologies de l'information – Télécommunications et échange d'information entre systèmes – Procédures de commande de liaison de données à haut niveau (HDLC)*.

3 Définitions

La présente Recommandation définit les termes suivants:

3.1 alphabet: ensemble de tous les caractères possibles. Dans la présente Recommandation, les caractères vont de 0 à $N_4 – 1$ (voir paragraphe 8).

3.2 adjoindre: créer une nouvelle chaîne en utilisant un caractère sans correspondant. Si celui-ci suit une chaîne mise en correspondance, la nouvelle chaîne est constituée de la chaîne mise en correspondance et du caractère sans correspondant ajouté à la fin; si le caractère sans correspondant suit un caractère, la nouvelle chaîne est constituée des deux caractères.

3.3 caractère: élément de données unique transmis de l'ETTD au codeur et utilisant un nombre prédéfini de bits N_3 (voir paragraphe 8).

3.4 code: séquence de bits produite par le codeur et représentant une commande ou une information. Les types de code définis sont les suivants: codes de commande, ordinaux, mots de code et longueurs d'extension de chaîne.

3.5 préfixe de code: Un ou deux bits précédant un code et indiquant le type de code qui suit. Ils sont définis au paragraphe 6.6.3.

3.6 **mot de code**: nombre binaire compris entre 4 et $N_2 - 1$ qui représente une chaîne de caractères consécutifs. Dans le codeur, il correspond à un certain segment de chaîne. Dans le décodeur, il correspond à une chaîne entière (voir paragraphe 8).

3.7 **mode comprimé**: mode de fonctionnement dans lequel les données provenant de l'ETTD sont transmises sous forme de codes binaires.

3.8 **fonctionnement avec compression**: état de l'ETCD dans lequel la fonction de compression de données est active. Le fonctionnement avec compression a deux modes: mode comprimé et mode transparent. Les transitions entre ces modes peuvent être automatiques et fondées sur le contenu des données provenant de l'ETTD (voir paragraphe 6.5).

3.9 **code de commande**: nombre binaire compris entre 0 et 3, dont l'usage est réservé à la signalisation, entre ETCD, d'informations de commande associées à la fonction de compression lorsque le mode de fonctionnement est le mode comprimé (voir paragraphe 7.2).

3.10 **décodeur**: sous-fonction de compression de données qui décompresse la sortie d'un codeur.

3.11 **dictionnaire**: structure de données qui représente les chaînes créées lors du codage ou du décodage. Pour le codeur, le dictionnaire est constitué d'une rangée de racines, d'un arbre de nœuds et d'un historique; pour le décodeur, il est constitué d'une collection de chaînes et d'un historique.

3.12 **codeur**: sous-fonction de compression de données qui compresse les données.

3.13 **échappement**: en mode transparent, indication du début d'une séquence de commandes au décodeur.

3.14 **étendre**: créer une nouvelle chaîne, constituée d'une chaîne mise en correspondance et d'un ou de plusieurs caractères ajoutés à la fin.

3.15 **vidage**: le codeur termine le traitement des caractères d'entrée jusqu'à ce stade, transfère les codes résultants, met à jour le dictionnaire selon qu'il convient et établit l'alignement d'octets. Le décodeur établit alors l'alignement d'octets.

3.16 **historique**: la mémoire-tampon de l'historique, appelée ici "l'historique", est la structure de données qui contient les caractères d'entrée dans le codeur (ou, ce qui est équivalent, les caractères décodés par le décodeur) depuis la plus récente réinitialisation des structures de données. Les caractères entrent dans l'historique dans l'ordre dans lequel ils entrent dans le codeur (ou dans l'ordre dans lequel ils sont décodés).

3.17 **méthode multipaquet**: compression de données qui sont mises sous forme de paquets (ou de trames), telle que les deux extrémités de la transmission de données peuvent identifier les frontières de paquet (ou de trame) et plusieurs paquets ou parties de paquets sont traités les uns à la suite des autres. La méthode multipaquet est décrite dans l'Annexe B.

3.18 **nœud**: dans une structure arborescente, point qui représente un segment de chaîne. Il correspond à un mot de code dans le codeur.

3.19 **arbre de nœuds**: dans le codeur, structure de données qui représente l'ensemble des segments de chaîne et leurs relations. La combinaison de cette structure et de la rangée de racines représente les structures arborescentes du codeur. Un mot de code correspond à un segment de chaîne donné et à un nœud donné.

3.20 **ordinal**: l'ordinal d'un caractère est l'équivalent numérique de ce caractère.

3.21 **méthode paquet**: compression de données qui sont mises sous forme de paquets (ou de trames), telle que les deux extrémités de la transmission de données peuvent identifier les frontières de paquet (ou de trame) et chaque paquet est traité séparément. La méthode paquet est décrite dans l'Annexe B.

3.22 mode paramètre: mode de fonctionnement dans lequel des paramètres de compression sont transférés entre homologues de compression de données.

3.23 réception: lors de la négociation de paramètres, sens correspondant au décodeur d'une entité.

3.24 racine: point situé à la base d'une structure arborescente et qui représente, dans le contexte de la présente Recommandation, le premier caractère d'une chaîne (voir Figure 2 et 6.2.1). Chaque caractère de l'alphabet est associé à une racine.

3.25 rangée de racines: structure de données qui contient l'ensemble de toutes les racines du codeur.

3.26 méthode flux: compression de données qui sont transmises en continu sur une liaison avec remise garantie. Cette méthode constitue le principal sujet de l'UIT-T V.44.

3.27 chaîne: séquence non brisée de deux caractères ou davantage. Dans le dictionnaire du codeur, elle est constituée d'un premier caractère (la racine) suivi d'un ou de plusieurs segments de chaîne.

3.28 extension de chaîne: séquence d'un ou de plusieurs caractères par laquelle une chaîne est étendue pour créer une nouvelle chaîne plus longue.

3.29 longueur d'extension de chaîne: code binaire indiquant le nombre de caractères par lesquels la chaîne associée au mot de code immédiatement précédent est étendue.

3.30 segment de chaîne: section de chaîne associée à un mot de code donné dans l'arbre de nœuds du codeur.

3.31 collection de chaînes: dans le décodeur, structure de données qui représente les chaînes décodées.

3.32 émission: lors de la négociation de paramètres, sens correspondant au codeur d'une entité.

3.33 mode transparent: mode de fonctionnement dans lequel les caractères provenant de l'ETTD sont transmis sous forme non comprimée. La fonction de compression de données est active, mais n'a pas d'incidence sur la transmission de données.

3.34 structure arborescente: dans le codeur, structure de données abstraite qui est utilisée dans la présente Recommandation pour représenter un ensemble de chaînes ayant toutes le même caractère initial (voir Figure 2 et 6.2.1).

3.35 fonctionnement sans compression: état de l'ETCD dans lequel la fonction de compression de données est inactive.

3.36 caractère sans correspondant: caractère qui met fin au processus de mise en correspondance de chaîne ou d'extension de chaîne car il ne correspond à aucun caractère de l'historique. Le caractère sans correspondant devient alors le premier caractère d'une éventuelle nouvelle chaîne mise en correspondance.

3.37 { }: {A} est la représentation vectorielle d'un code binaire. Elle a autant de composantes qu'il y a de bits dans le code. La composante A_1 correspond au bit de plus faible poids du code binaire.

4 Abréviations

La présente Recommandation utilise les abréviations suivantes:

ECM passer au mode comprimé (*enter compressed mode*): commande du mode transparent définie au 7.2

EID échappement dans les données (*ESCAPE in data*): commande du mode transparent définie au 7.2

EPM passer au mode paramètre (*enter parameter mode*): commande du mode transparent définie au 7.2

ETCD équipement de terminaison de circuit de données

ETM passer au mode transparent (*enter transparent mode*): code de commande définie au 7.2

ETTD équipement terminal de traitement de données

5 Description fonctionnelle d'un ETCD

Le présent paragraphe décrit un ETCD employant une fonction de compression de données.

5.1 Généralités

Un ETCD employant une compression de données, comme illustré sur la Figure 1, contient les composants suivants:

a) des circuits de jonction ETTD/ETCD;

b) un convertisseur de signaux;

c) une fonction de commande;

d) une fonction de contrôle d'erreur;

e) une fonction de compression de données.

5.2 Circuits de jonction ETTD/ETCD

Ce composant doit avoir les capacités décrites dans l'UIT-T V.42.

5.3 Convertisseur de signaux

Ce composant doit avoir les capacités décrites dans l'UIT-T V.42.

5.4 Fonction de commande

Ce composant doit avoir les capacités décrites au 6.2/V.42. En outre, il doit exécuter les opérations suivantes:

a) négociation des modes de fonctionnement de la fonction de compression de données avec l'ETCD distant et négociation de paramètres associés au fonctionnement de cette fonction;

b) instigation de l'initialisation ou de la réinitialisation de la fonction de compression de données;

c) coordination de l'établissement d'une connexion avec contrôle d'erreur à utiliser par les fonctions de compression de données homologues;

d) coordination de la remise des données entre l'interface ETTD/ETCD et la fonction de compression de données, conformément aux procédures définies aux 6.2 et 8.4/V.42, y compris les procédures de contrôle de flux qui y sont définies;

e) coordination de la remise des données entre la fonction de compression de données et la fonction de contrôle d'erreur;

f) mesure consécutive à la détection d'une anomalie.

5.5 Fonction de contrôle d'erreur

Ce composant doit avoir les capacités décrites dans l'UIT-T V.42.

5.6 Fonction de compression de données

Elle doit mettre en œuvre les procédures définies dans la présente Recommandation afin de coder efficacement les données avant leur transmission sur une connexion avec contrôle d'erreur. Elle doit en outre exécuter les opérations suivantes:

a) initialisation et réinitialisation des dictionnaires du codeur et du décodeur;

b) codage et décodage des données;

c) commutation entre les modes de fonctionnement comprimé et transparent;

d) configuration du codeur et du décodeur conformément aux paramètres négociés par la fonction de commande.

6 Procédures de la fonction de compression de données

6.1 Aperçu général de la fonction de compression de données

La fonction de compression de données comprend un codeur et un décodeur. Une connexion de données prend généralement en charge la transmission de données dans les deux sens et peut donc prendre en charge la compression de données dans les deux sens. Par conséquent, chaque homologue de la connexion de données peut avoir un codeur et un décodeur. Le codeur transfère les données comprimées à son décodeur homologue à l'autre extrémité de la connexion et le décodeur décompresse les données comprimées reçues de son codeur homologue. Les valeurs des paramètres de compression de données du couple codeur-décodeur pour chaque sens doivent être coordonnées, établies par négociation; toutefois, à une extrémité donnée, les valeurs des paramètres du codeur et du décodeur peuvent être différentes, puisqu'elles sont associées à des sens de transmission différents.

Les caractères provenant de l'entrée ETTD du codeur sont comparés à toutes les chaînes de caractères précédemment identifiées. Si une chaîne correspond aux caractères, le mot de code représentant la chaîne est transféré puis on tente d'étendre la chaîne mise en correspondance pour coder d'autres caractères et créer une nouvelle chaîne plus longue. Si aucune chaîne correspondant aux caractères n'est trouvée, l'ordinal associé au premier caractère est transféré.

Le codeur procède continuellement à des ajouts dans son ensemble de chaînes disponibles pour la mise en correspondance, en mettant des caractères d'entrée dans l'historique et en ajoutant des nœuds à l'arbre de nœuds. Lorsque l'un des deux est plein, le dictionnaire est réinitialisé et l'opération de codage continue comme avant.

Le décodeur crée des chaînes qui sont des répliques des chaînes créées par son codeur homologue, en utilisant les mêmes mots de code assignés. Grâce à ces chaînes, il décompresse le flux de données comprimées reçues. Le dictionnaire du décodeur est réinitialisé à la réception d'un code de commande REINIT.

6.2 Structure des dictionnaires

Pour pouvoir décrire en détail l'algorithme de compression de données, il est utile de définir les structures de données suivantes.

6.2.1 Dictionnaire du codeur

Le codeur utilise un dictionnaire comprenant trois parties:

1) *rangée de racines*: elle contient une entrée pour chaque caractère de l'alphabet. La taille de l'alphabet, N_4, est fonction de la taille des caractères provenant de la fonction de commande: pour un alphabet de caractères à 8 bits, il y a 256 entrées. Chaque entrée est indexée par son caractère et sert de racine pour une structure arborescente; elle contient un indice descendant

qui pointe sur un nœud de l'arbre de nœuds. Cet indice est le point de départ pour descendre dans la structure arborescente.

2) *arbre de nœuds*: il contient les mots de code créés pendant le fonctionnement du codeur et les segments de chaîne auxquels ils sont associés. Les nœuds de l'arbre de nœuds définissent un ensemble de structures arborescentes et les segments de chaîne associés. Chaque nœud contient un mot de code, la position dans l'historique du premier caractère du segment de chaîne, le nombre de caractères du segment de chaîne et deux indices permettant d'établir des liens avec d'autres nœuds. "L'indice descendant", s'il est valide, pointe sur un nœud représentant un segment de chaîne qui suit le segment de chaîne considéré. "L'indice latéral", s'il est valide, pointe sur un nœud représentant un segment de chaîne situé au même niveau que le segment de chaîne considéré (autrement dit qui suit aussi la racine ou le segment de chaîne qui précède le segment de chaîne considéré). Le nombre de mots de code ne peut pas dépasser N_{2T}.

3) *historique*: il contient tous les caractères d'entrée du codeur, ordonnés, depuis la réinitialisation la plus récente du dictionnaire. Les segments de chaîne de l'arbre de nœuds sont référencés par la position de leur premier caractère dans l'historique et par leur longueur. Le nombre de caractères de l'historique ne peut pas dépasser N_{8T}.

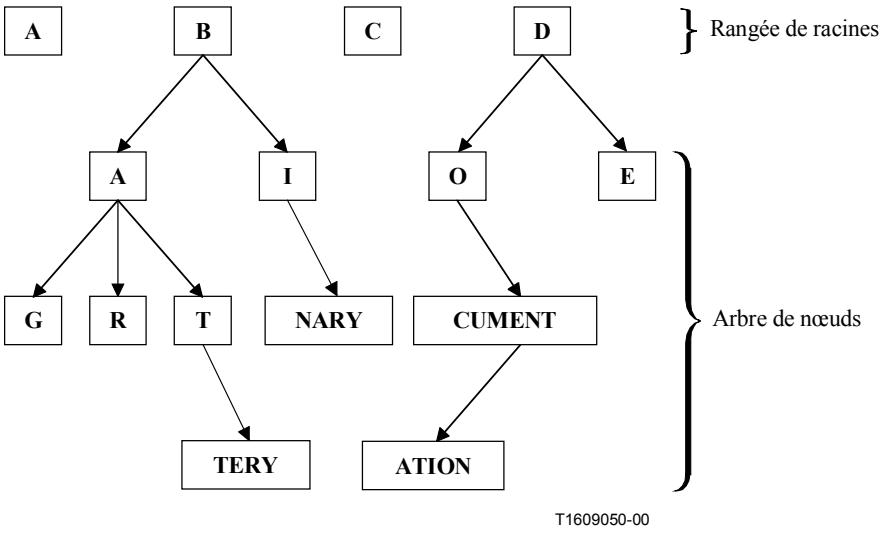

Figure 2/V.44 – Exemple de structures arborescentes

Cette figure montre uniquement les structures arborescentes à partir des racines A, B, C et D.

Dans les structures arborescentes de la Figure 2, on peut voir les segments de chaîne, représentés chacun par un nœud:

- A et I, qui suivent le caractère racine B;
- G, R et T, qui suivent le segment de chaîne A;
- TERY, qui suit le segment de chaîne T;
- NARY, qui suit le segment de chaîne I;
- O et E, qui suivent le caractère racine D;
- CUMENT, qui suit le segment de chaîne O;
- ATION, qui suit le segment de chaîne CUMENT.

Les chaînes représentées sont les suivantes: BA, BAG, BAR, BAT, BATTERY, BI, BINARY, DO, DOCUMENT, DOCUMENTATION, DE. Pour la mise en correspondance d'une chaîne complète, on commence à la racine en utilisant le caractère suivant à coder et on procède à la mise en correspondance des segments de chaîne successifs en descendant dans l'arbre avec les caractères d'entrée suivants.

6.2.2 Dictionnaire du décodeur

Le décodeur utilise un dictionnaire comprenant deux parties:

1) *collection de chaînes*: elle définit les chaînes créées pendant le processus de décodage qui correspondent aux segments de chaîne créés par le processus de codage. Chaque entrée contient un mot de code, la position dans l'historique du dernier caractère de la chaîne et la longueur totale de la chaîne. Le nombre de mots de code ne peut pas dépasser N_{2R}.

2) *historique*: il contient tous les caractères décodés, ordonnés, depuis la réinitialisation la plus récente du dictionnaire. Il est identique à l'historique du codeur; toutefois, il est généré au moyen des caractères résultant du processus de décodage. Le nombre de caractères de l'historique ne peut pas dépasser N_{8R}.

6.3 Codage

L'algorithme de codage tente de trouver une chaîne de caractères déjà traités qui corresponde aux caractères suivants à traiter; Si une telle chaîne est trouvée, le mot de code la représentant est transféré. A la réception du mot de code, le décodeur peut ensuite régénérer correctement cette chaîne. La compression de données aboutit si la chaîne codée nécessite le transfert d'un moins grand nombre de bits que les caractères non comprimés d'origine.

Les caractères d'entrée sont placés dans les endroits disponibles suivants de l'historique et traités par le codeur comme suit:

a) utilisant le caractère d'entrée et le ou les caractères suivants, le codeur tente de trouver la chaîne correspondante la plus longue dans le dictionnaire;

b) si aucune chaîne correspondante n'est trouvée, le codeur transfère l'ordinal associé au caractère d'entrée et retourne à l'étape a) avec le caractère suivant;

c) si une chaîne correspondante est trouvée, le codeur transfère le mot de code assigné au dernier segment de chaîne mis en correspondance en totalité de la chaîne correspondante la plus longue;

d) le codeur tente ensuite d'étendre la chaîne correspondante la plus longue, jusqu'à la longueur maximale de chaîne. Pour cela, il essaie de mettre en correspondance le ou les caractères suivants avec le ou les caractères de l'historique suivant immédiatement le dernier caractère du dernier segment de chaîne mis en correspondance en totalité de la chaîne correspondante la plus longue;

e) si la chaîne correspondante la plus longue est étendue avec succès par un ou plusieurs caractères, le codeur transfère une longueur d'extension de chaîne indiquant le nombre de caractères par lesquels la chaîne a été étendue.

Utilisant le caractère sans correspondant (c'est-à-dire le premier caractère ne faisant pas partie de la chaîne correspondante ni de l'extension) comme caractère suivant, le codeur retourne à l'étape a).

6.3.1 Procédure de mise en correspondance de chaîne

Une chaîne du dictionnaire du codeur est constituée d'un premier caractère suivi d'un ou de plusieurs segments de chaîne de l'arbre de nœuds; par exemple, sur la Figure 2, la chaîne "BATTERY" a pour premier caractère "B", suivi du segment de chaîne "A", suivi du segment de chaîne "T", suivi du segment de chaîne "TERY". La longueur totale de la chaîne ne peut pas dépasser N_{7T}.

La procédure de mise en correspondance de chaîne utilise la structure de dictionnaire décrite au 6.2 pour trouver la plus longue chaîne correspondante comme suit:

– chaque racine de la rangée de racines a un indice descendant: s'il a pour valeur un indice valide, le nœud associé donne le premier segment de chaîne pour la comparaison;

– chaque nœud de l'arbre de nœuds a un indice latéral utilisé par le codeur pour chercher un autre segment de chaîne qui se raccorde sur le même nœud précédent ou sur la même racine précédente (un autre nœud situé au même niveau);

– chaque nœud de l'arbre de nœuds a un indice descendant utilisé par le codeur pour chercher un segment de chaîne pour la suite de la comparaison (un nœud situé au niveau suivant).

Pour chercher une chaîne correspondante, le codeur examine la racine associée au premier caractère d'entrée et compare les caractères d'entrée qui suivent avec les segments de chaîne associés aux nœuds se raccordant directement à cette racine. Si les caractères d'entrée qui suivent correspondent exactement à un segment de chaîne complet (les correspondances partielles ne sont pas valides), le codeur compare des caractères d'entrée additionnels avec les segments de chaîne des nœuds se raccordant sur ce nœud. Par conséquent, le codeur continue à descendre dans la structure arborescente jusqu'à ce qu'il ne puisse plus trouver d'autres correspondances ou jusqu'à ce qu'il n'y ait plus de nœuds de raccordement.

La procédure de mise en correspondance de chaîne échoue si la racine associée au premier caractère n'a pas d'indice descendant valide ou si aucun nœud de deuxième niveau ne correspond au deuxième caractère. Dans ce cas, l'ordinal associé au premier caractère est transféré et la procédure de mise en correspondance de chaîne recommence avec le caractère (sans correspondant) suivant.

Autrement, la procédure de mise en correspondance de chaîne trouve un nœud associé à la plus longue chaîne du dictionnaire qui correspond aux caractères d'entrée. Le codeur transfère le mot de code associé à ce nœud. (Il est à noter que le mot de code est en fait associé directement au dernier segment de chaîne mis en correspondance en totalité de la chaîne correspondante la plus longue, étant donné que le codeur n'associe pas explicitement de mots de code à des chaînes complètes.)

Le codeur peut transférer tout mot de code qu'il a créé. Si la valeur du mot de code est égale à la valeur de C_1 du décodeur, cela signifie qu'on a un mot de code qui n'est pas encore créé dans le décodeur; néanmoins, le décodeur est tenu d'interpréter correctement toutes les valeurs de mot de code jusqu'à C_1. Voir 6.4.1.

Il peut également être mis fin à la procédure de mise en correspondance de chaîne pour les motifs suivants: l'historique est plein; ou une demande FLUSH est reçue. Le traitement est semblable à celui qui est effectué lorsque le caractère d'entrée suivant ne correspond à aucun caractère dans le segment de chaîne courant: le mot de code du dernier segment de chaîne mis en correspondance en totalité (s'il y en a un) est transféré; puis, s'il existe un segment de chaîne mis en correspondance partiellement, les caractères mis en correspondance avec succès sont transférés sous la forme d'une longueur d'extension de chaîne et une entrée associée est créée dans l'arbre de nœuds.

NOTE 1 – Pour qu'un segment de chaîne corresponde aux caractères d'entrée, tous les caractères doivent correspondre: les correspondances partielles ne sont pas valides. Toutefois, une correspondance partielle de segment de chaîne peut être utilisée pour étendre la chaîne, si aucune correspondance complète de segment de chaîne n'est trouvée au même niveau. Dans ce cas, la procédure de mise en correspondance de chaîne a déjà procédé aux comparaisons nécessaires pour trouver la plus longue extension de chaîne possible spécifiée au 6.3.2 et la longueur de cette extension est précisément le nombre de caractères mis en correspondance du

segment de chaîne mis en correspondance partiellement. Il n'est donc pas nécessaire de lancer explicitement la procédure d'extension de chaîne, étant donné que le codeur peut simplement transférer la longueur d'extension de chaîne associée au nombre de caractères mis en correspondance du segment de chaîne mis en correspondance partiellement.

NOTE 2 – Il est avantageux d'ordonner les nœuds à un même niveau qui se raccordent à un nœud donné, alphabétiquement en fonction du premier caractère, pour que la recherche de segments de chaîne correspondants soit plus efficace. Parmi les segments de chaîne qui se raccordent à un nœud donné, il se peut que plusieurs aient le même premier caractère.

6.3.2 Procédure d'extension de chaîne

Si la procédure de mise en correspondance de chaîne aboutit (c'est-à-dire si on a trouvé la plus longue chaîne correspondante) et si la longueur totale de la chaîne est inférieure à N_{7T}, le codeur lance la procédure d'extension de chaîne. Autrement dit, il tente d'étendre la plus longue chaîne correspondante en comparant le ou les caractères suivants au ou aux caractères de l'historique qui suivent immédiatement le dernier caractère du dernier segment de chaîne mis en correspondance en totalité.

La réception d'une demande C-FLUSH émanant de la fonction de commande, immédiatement après une mise en correspondance de chaîne fructueuse et avant que la procédure d'extension de chaîne ait traité un caractère, met fin à la procédure d'extension de chaîne. La réception du caractère qui suit immédiatement la demande C-FLUSH marque le début d'une nouvelle chaîne correspondante et le dictionnaire est mis à jour comme s'il s'agissait d'un caractère sans correspondant.

Par exemple, supposons que la chaîne "BATTERY" soit la plus longue chaîne correspondante. Le codeur compare le caractère suivant au caractère de l'historique qui suit le segment de chaîne "TERY". Il existe deux possibilités:

- le caractère suivant ne correspond pas au caractère de l'historique qui suit immédiatement le "Y" du segment de chaîne "TERY". La procédure d'extension de chaîne échoue et ce caractère suivant est adjoint à "TERY" pour créer une nouvelle chaîne plus longue et devient par là-même le premier caractère d'une éventuelle nouvelle chaîne correspondante;
- le caractère suivant correspond au caractère de l'historique qui suit immédiatement le "Y" du segment de chaîne "TERY". La procédure d'extension de chaîne continue en comparant le caractère d'entrée suivant avec le deuxième caractère de l'historique suivant le "Y" du segment de chaîne "TERY", le caractère d'entrée suivant au troisième caractère de l'historique suivant le "Y" et ainsi de suite. Le nombre de caractères mis en correspondance avec succès est transféré sous la forme d'une longueur d'extension de chaîne. Le codeur crée un nouveau segment de chaîne dans l'arbre de nœuds, constitué de l'extension de chaîne (la séquence de caractères mis en correspondance après la plus longue chaîne correspondante). Ce segment de chaîne se voit assigner le mot de code disponible suivant et a une longueur égale à la longueur de l'extension. Son indice d'historique fait référence à la position dans l'historique du premier caractère d'entrée parmi les plus récents utilisés pour étendre la chaîne.

Il peut aussi être mis fin à la procédure d'extension de chaîne pour les motifs suivants: l'historique est plein ou une demande FLUSH est reçue ou la longueur de la chaîne totale a atteint N_{7T}. Le traitement opéré est semblable au cas où le caractère d'entrée suivant ne correspond pas au caractère suivant de l'historique: les caractères qui ont été mis en correspondance avec succès pour étendre la chaîne sont transférés sous la forme d'une longueur d'extension de chaîne et une entrée associée est créée dans l'arbre de nœuds.

6.3.3 Création de segments de chaîne

Le codeur crée continuellement de nouvelles chaînes en vue d'éventuelles correspondances de chaînes futures. Les deux méthodes de création de nouveaux segments de chaîne sont les suivantes:

adjoindre

échec de la mise en correspondance de chaîne:

– si la mise en correspondance de chaîne échoue car le premier caractère n'a pas d'indice descendant valide dans la rangée de racines, un segment de chaîne à un caractère est créé, reliant le caractère suivant à la racine du premier caractère;

– si la mise en correspondance de chaîne échoue car le caractère suivant ne correspond à aucun nœud existant se raccordant à la racine du premier caractère, un segment de chaîne à un caractère est créé au moyen du caractère sans correspondant, se raccordant à la racine du premier caractère;

échec de l'extension de chaîne:

– un segment de chaîne à un caractère est créé pour relier le caractère sans correspondant au dernier segment de chaîne mis en correspondance en totalité de la chaîne correspondante la plus longue;

étendre

succès de l'extension de chaîne:

un segment de chaîne est créé pour étendre la chaîne correspondante la plus longue par un ou plusieurs caractères. Le segment de chaîne ainsi créé a une longueur égale au nombre de caractères de l'extension et est relié au dernier segment de chaîne mis en correspondance en totalité.

6.3.4 Résumé pour le codage

Tableau 1/V.44 – Procédures de codage

Procédure	Résultat	Code transféré	Nouveau segment de chaîne	Utilisation du caractère sans correspondant
Mise en correspondance de chaîne	Echec	Ordinal correspondant au premier caractère	Adjoindre le caractère sans correspondant à la racine du premier caractère	Comme premier caractère pour la mise en correspondance de chaîne
Mise en correspondance de chaîne	Succès	Mot de code associé au dernier segment de chaîne mis en correspondance en totalité	Aucun	Pour l'extension de chaîne
Extension de chaîne	Echec	Rien	Adjoindre le caractère sans correspondant au dernier segment de chaîne mis en correspondance en totalité	Comme premier caractère pour la mise en correspondance de chaîne
Extension de chaîne	Succès	Longueur d'extension de chaîne	Tous les caractères de l'extension, reliés au dernier segment de chaîne mis en correspondance en totalité	Comme premier caractère pour la mise en correspondance de chaîne

Le Tableau 1 résume les procédures de codage. L'Appendice II illustre le fonctionnement de l'algorithme explicitement en termes de structure de dictionnaire.

6.4 Décodage

Dans le décodeur, une chaîne est définie par la position dans l'historique de son dernier caractère et par sa longueur totale. Elle est associée à un mot de code donné qui, par construction, correspond dans le codeur au dernier segment de cette chaîne.

En mode comprimé, les codes binaires reçus par le décodeur sont analysés en codes de commande, ordinaux, mots de code et longueurs d'extension de chaîne, par un examen des préfixes de code. Le décodeur doit rester synchronisé avec les chaînes et les mots de code qui sont créés par le codeur, mettant à jour la collection de chaînes et l'historique en fonction des codes reçus. Le décodeur est plus simple que le codeur, car le codeur doit chercher des chaînes correspondantes et créer de nouvelles chaînes plus longues, tandis que le décodeur n'a besoin que de garder la trace des chaînes qu'il a créées.

Le décodeur doit être capable de fonctionner en mode transparent comme en mode comprimé.

6.4.1 Codes de traitement

En mode comprimé, la fonction de décodage doit procéder comme suit:

1) à la réception d'un ordinal, le décodeur place le caractère associé dans la sortie décomprimée et dans l'historique;

2) à la réception d'un mot de code inférieur à C_1, le décodeur copie la chaîne représentée par le mot de code dans la sortie décomprimée et dans l'historique;

3) à la réception d'un mot de code égal à C_1, si le précédent code reçu était un mot de code, le décodeur copie la chaîne représentée par ce précédent mot de code dans la sortie décomprimée et dans l'historique, et place ensuite le premier caractère de la chaîne représentée par ce précédent mot de code dans la sortie décomprimée et dans l'historique;

4) à la réception d'un mot de code égal à C_1, si le précédent code reçu était un ordinal, le décodeur traite le caractère associé conformément à l'étape 3) ci-dessus, comme s'il s'agissait d'une chaîne de longueur un;

5) à la réception d'une longueur d'extension de chaîne, le décodeur utilise le mot de code précédent pour accéder aux caractères de l'historique qui suit immédiatement la chaîne représentée par ce mot de code. Le nombre de caractères indiqués par la longueur de l'extension de chaîne sont copiés dans la sortie décomprimée et dans l'historique;

6) la réception d'un mot de code supérieur à C_1 constitue une erreur de procédure (voir 7.15).

6.4.2 Création de nouvelles chaînes

Les règles de création de nouvelles chaînes sont les suivantes:

Tableau 2/V.44 – Règles de création de chaîne

Code courant	Code précédent	Nouvelle chaîne créée
Ordinal	Ordinal	Le caractère associé est adjoint au caractère précédent pour créer une chaîne à 2 caractères.
Ordinal	Mot de code	Le caractère associé est adjoint à la chaîne précédente pour créer une chaîne plus longue. (Note)
Ordinal	Longueur d'extension de chaîne	Le caractère associé n'est pas adjoint à la chaîne étendue.
Ordinal	Code de commande FLUSH	Le caractère associé est traité comme si la commande FLUSH n'avait pas eu lieu. L'action est déclenchée comme si le code qui précède la commande FLUSH était le "code précédent".
Ordinal	REINIT, ECM ou initialisation	Aucune: le dictionnaire est vide.
Mot de code	Ordinal	Le premier caractère de la chaîne associée au mot de code est adjoint au caractère précédent pour créer une chaîne à 2 caractères.
Mot de code	Mot de code	Le premier caractère de la chaîne associé au mot de code est adjoint à la chaîne précédente pour créer une chaîne plus longue. (Note)
Mot de code	Longueur d'extension de chaîne	Le premier caractère de la chaîne associé au mot de code n'est pas adjoint à la chaîne étendue.
Mot de code	Code de commande FLUSH	Le premier caractère de la chaîne associée au mot de code est traité comme si la commande FLUSH n'avait pas eu lieu. L'action est déclenchée comme si le code qui précède la commande FLUSH était le "code précédent".
Longueur d'extension de chaîne	Mot de code	La chaîne précédente est étendue pour créer une nouvelle chaîne plus longue. (Note)
NOTE – Les chaînes de longueur supérieure à N_{7R} ne doivent pas être créées.		

L'intervention d'un code de commande STEPUP n'a pas d'incidence sur la création de chaîne.

6.5 Mode transparent

La surcharge liée au fonctionnement en mode comprimé peut parfois dépasser les économies réalisées par la compression. Lorsqu'une telle situation se produit, le codeur peut passer au mode de fonctionnement transparent pour transférer les caractères d'entrée non codés, sans nécessiter une nouvelle connexion.

En mode transparent, l'activité de codage peut se poursuivre, afin de déterminer quand le passage à nouveau au mode comprimé serait avantageux; mais la sortie du codeur n'est pas transférée. Les caractères d'entrée du codeur sont transférés sans modification, avec alignement des octets. Le codeur doit réinitialiser le dictionnaire au moment de l'abandon du mode transparent pour passer à nouveau au mode comprimé.

En mode transparent, le décodeur doit traiter les caractères et commandes mais ne doit mettre à jour ni l'historique ni la collection de chaînes. Le dictionnaire du décodeur doit être mis dans un état compatible avec celui du dictionnaire du codeur homologue, en procédant à une réinitialisation au moment de l'abandon du mode transparent.

6.5.1 Passage du mode comprimé au mode transparent

En mode comprimé, si le codeur détermine que le flux de données n'est pas compressible, il doit:

a) faire en sorte que tous les caractères d'entrée jusqu'à ce stade soient transférés conformément aux procédures de codage normales;

b) transférer le code de commande ETM (*enter transparent mode*) pour indiquer une transition vers le mode transparent au décodeur homologue;

c) transmettre suffisamment de bits 0 pour établir l'alignement des octets (voir Tableau 6);

d) passer au mode transparent.

En mode comprimé, à la réception d'un code de commande ETM, le décodeur passe au mode transparent.

6.5.2 Passage du mode transparent au mode comprimé

En mode transparent, si le codeur détermine que le flux de données est compressible, il doit:

a) transférer en mode transparent tous les caractères d'entrée jusqu'à ce stade;

b) réinitialiser son dictionnaire;

c) transférer la valeur courante de ESCAPE pour indiquer une séquence de commandes;

d) transférer la commande ECM;

e) passer au mode comprimé.

En mode transparent, à la réception d'un ESCAPE suivi immédiatement d'une commande ECM, le décodeur réinitialise son dictionnaire et passe au mode comprimé.

6.6 Transfert

En mode transparent, les caractères doivent être transférés à la fonction de commande pour transmission sous une forme dans laquelle les octets sont alignés, au moyen d'une indication C-TRANSFER. Ils peuvent être transférés individuellement au cours des procédures de mise en correspondance de chaîne et d'extension de chaîne, ou sous forme d'une séquence après la fin de l'une ou l'autre des procédures.

En mode comprimé, les codes binaires en sortie du codeur (6.3) doivent être transférés à la fonction de commande. Le bit de plus faible poids du préfixe d'un code binaire doit suivre immédiatement le bit de plus fort poids du code binaire précédent. Chaque code est précédé d'un préfixe de code.

Après un passage du mode transparent au mode comprimé, le bit de plus faible poids du préfixe du premier code binaire à transférer doit être le bit 1 du premier octet comprimé.

Lors d'un passage du mode comprimé au mode transparent, il faut transférer suffisamment de bits 0 pour faire en sorte que le caractère suivant qui est transmis présente un alignement des octets.

Après un code de commande FLUSH, il faut transférer suffisamment de bits 0 pour faire en sorte que le code suivant qui est transmis présente un alignement des octets.

6.6.1 Transfert de codes de commande, d'ordinaux et de mots de code

Pour le transfert des codes de commande, on utilise le nombre de bits défini par la valeur courante de C_2.

Pour le transfert des ordinaux, on utilise le nombre de bits défini par la valeur courante de C_5.

Pour le transfert des mots de code, on utilise le nombre de bits défini par la valeur courante de C_2.

6.6.2 Transfert de longueur d'extension de chaîne

Une longueur d'extension de chaîne est un code représentant la longueur par laquelle la chaîne représentée par le mot de code précédent doit être étendue. Elle est constituée d'un ou de plusieurs sous-champs, chacun devant être analysé individuellement par le décodeur. Le nombre de sous-champs dans une longueur d'extension de chaîne dépend du nombre de caractères (c'est-à-dire de la longueur) de l'extension de chaîne. La longueur d'une extension de chaîne varie de 1 à la plus petite des valeurs 253 et (N_{7T} – 2). Le codage des sous-champs pour les extensions de chaîne de longueur comprise entre 1 et 12 est représenté dans le Tableau 3; par souci de clarté, on donne aussi l'ordre de transmission des bits.

Tableau 3/V.44 – Longueur d'extension de chaîne: longueurs comprises entre 1 et 12

Longueur	Sous-champs	Ordre de transmission des bits (de gauche à droite)
1	"1"	1
2	"0" "01"	0 10
3	"0" "10"	0 01
4	"0" "11"	0 11
5	"0" "00" "0" "000"	0 00 0 000
6	"0" "00" "0" "001"	0 00 0 100
7	"0" "00" "0" "010"	0 00 0 010
8	"0" "00" "0" "011"	0 00 0 110
9	"0" "00" "0" "100"	0 00 0 001
10	"0" "00" "0" "101"	0 00 0 101
11	"0" "00" "0" "110"	0 00 0 011
12	"0" "00" "0" "111"	0 00 0 111

Pour les longueurs d'extension de chaîne supérieures à 12, le codage dépend de la valeur de la longueur maximale de chaîne N_{7T}, comme indiqué dans le Tableau 4. La valeur de N_{7T} détermine le nombre de bits du dernier sous-champ, tandis que la longueur d'extension de chaîne détermine la valeur du dernier sous-champ:

$$N = \text{longueur d'extension de chaîne} - 13$$

et:

$$N_x = x^{\text{ième}} \text{ bit de } N$$

Tableau 4/V.44 – Longueur d'extension de chaîne: longueurs comprises entre 13 et 253

Paramètre de longueur maximale de chaîne N_{7T}	Intervalle de longueurs	Sous-champs	Ordre de transmission des bits (de gauche à droite)
$32 \leq N_{7T} \leq 46$	13-44	"0" "00" "1" "$N_4N_3N_2N_1N_0$"	0 00 1 $N_0N_1N_2N_3N_4$
$46 < N_{7T} \leq 78$	13-76	"0" "00" "1" "$N_5N_4N_3N_2N_1N_0$"	0 00 1 $N_0N_1N_2N_3N_4N_5$
$78 < N_{7T} \leq 142$	13-140	"0" "00" "1" "$N_6N_5N_4N_3N_2N_1N_0$"	0 00 1 $N_0N_1N_2N_3N_4N_5N_6$
$142 < N_{7T} \leq 255$	13-253	"0" "00" "1" "$N_7N_6N_5N_4N_3N_2N_1N_0$"	0 00 1 $N_0N_1N_2N_3N_4N_5N_6N_7$
NOTE – La longueur maximale d'une extension de chaîne est de ($N_{7T} – 2$), car la longueur minimale d'une chaîne est de 2 caractères.			

6.6.3 Préfixes de code

Les préfixes de code sont transférés immédiatement avant les codes associés. Ils sont indiqués dans le Tableau 5.

Tableau 5/V.44 – Utilisation des préfixes de code

Type de code	Préfixe: immédiatement après un mot de code		Préfixe: autres cas (Note)	
	Sous-champs	*Ordre de transmission des bits (de gauche à droite)*	*Sous-champs*	*Ordre de transmission des bits (de gauche à droite)*
Code de commande	"1"	1	'1'	1
Ordinal	"0" "0"	0 0	'0'	0
Mot de code	"1"	1	'1'	1
Longueur d'extension de chaîne	"0" "1"	0 1	Non applicable	Non applicable
NOTE – Immédiatement après un code de commande, un ordinal, une longueur d'extension de chaîne; ou après réinitialisation.				

6.6.4 Exemple de transfert

Le Tableau 6 illustre une séquence d'octets comprimés transférés par le codeur. Dans cet exemple, plusieurs codes sont transférés en mode comprimé puis le codeur passe au mode transparent. Dans l'exemple, on suppose que la taille courante de mot de code C_2 vaut 11 et que la taille courante d'ordinal C_5 vaut 8 (voir Tableau 12). Les codes sont les suivants:

- mot de code {A} avec le préfixe "1";
- ordinal {B} avec le préfixe "0" "0" (car il suit immédiatement un mot de code);
- ordinal {C} avec le préfixe "0";
- mot de code {D} avec le préfixe "1";
- longueur d'extension de chaîne 8: préfixe "0" "1", sous-champs "0" "00" "0" "011";
- ordinal {E} avec le préfixe "0";
- code de commande ETM avec le préfixe "1". La valeur de ETM est "00000000" et elle est étendue à 11 bits;

– remplissage: cinq 0 pour établir l'alignement dans l'octet i + 9;

– caractère {F} en mode transparent, alignement des octets.

Tableau 6/V.44 – Mappage des octets pour un exemple de flux de données

Bit n°	8	7	6	5	4	3	2	1	Octet	Mode
	A_5	A_4	A_3	A_2	A_1	***1***	...	...	i	comprimé
	0	***0***	A_{11}	A_{10}	A_9	A_8	A_7	A_6	i + 1	comprimé
	B_8	B_7	B_6	B_5	B_4	B_3	B_2	B_1	i + 2	comprimé
	C_7	C_6	C_5	C_4	C_3	C_2	C_1	***0***	i + 3	comprimé
	D_6	D_5	D_4	D_3	D_2	D_1	***1***	C_8	i + 4	comprimé
	0	***1***	***0***	D_{11}	D_{10}	D_9	D_8	D_7	i + 5	comprimé
	E_1	***0***	0	1	1	0	0	0	i + 6	comprimé
	1	E_8	E_7	E_6	E_5	E_4	E_3	E_2	i + 7	comprimé
	0	0	0	0	0	0	0	0	i + 8	comprimé
	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	0	0	0	i + 9	comprimé
	F_8	F_7	F_6	F_5	F_4	F_3	F_2	F_1	i + 10	transparent
Les bits de préfixe de code sont ***en gras et en italique***, les bits de remplissage sont <u>soulignés</u>.										

7 Fonctionnement de la compression des données

7.1 Communication entre les fonctions de commande et de compression de données

La communication entre la fonction de commande et la fonction de compression de données est modélisée par un ensemble de primitives abstraites de la forme type X-NAME, qui représentent l'échange logique d'informations et de commandes pour exécuter une tâche ou un service. Dans le contexte de la présente Recommandation, la fonction de commande est considérée comme "l'utilisateur de service" tandis que la fonction de compression de données est considérée comme le "fournisseur de services". Les types de primitive sont les suivants: demande, indication, réponse et confirmation. Les services utilisés dans la présente Recommandation sont énumérés dans le Tableau 7.

Tableau 7/V.44 – Services attendus par la fonction de commande

Service	Primitive	Référence
Initialiser la fonction de compression de données	C-INIT	7.5
Transférer des paramètres de négociation vers/depuis la fonction de compression de données	C-PARM	7.4.2
Indiquer une erreur à la fonction de commande	C-ERROR	7.15
Transférer des données non comprimées vers/depuis la fonction de compression de données	C-DATA	7.7
Transférer des données comprimées vers/depuis la fonction de compression de données	C-TRANSFER	7.9
Vider les données restantes non transférées par le codeur	C-FLUSH	7.13

7.2 Communication entre fonctions de compression de données homologues

Les codes de commande et les commandes utilisés pour la communication entre fonctions de compression de données homologues figurent dans les Tableaux 8 et 9.

Tableau 8/V.44 – Codes de commande utilisés en mode comprimé

Code	Nom	Description
0	ETM	Passer au mode transparent
1	FLUSH	Vider les données
2	STEPUP	Augmenter la taille de mot de code ou la taille d'ordinal de un
3	REINIT	Forcer la réinitialisation des dictionnaires

Tableau 9/V.44 – Commandes utilisées en mode transparent

Valeur	Nom	Description
0	ECM	Passer au mode comprimé
1	EID	Echappement dans les données
2	EPM	Passer au mode paramètre
Variable: voir 7.14	ESCAPE	Lancer une séquence de commandes

7.3 Négociation de capacité V.44

Avant le lancement de la fonction de compression de données, les modems homologues doivent déterminer s'ils ont une capacité V.44. Cette détermination doit se faire au moment de l'établissement de la liaison au moyen d'un protocole (par exemple, la procédure XID définie dans l'UIT-T V.42) et la capacité V.44 reste active pendant la durée de la connexion avec correction d'erreur. En particulier, lors de l'utilisation de la présente Recommandation avec le contrôle d'erreur V.42, il faut utiliser la procédure de négociation XID (voir l'ISO/CEI 13239 et 7.7/V.42, 8.10/V.42 et paragraphe 10/V.42). Les paramètres du sous-champ des données d'utilisateur doivent être utilisés à cette fin, en plus des paramètres définis dans l'UIT-T V.42. Le sous-champ des données d'utilisateur apparaît dans la trame XID immédiatement avant la séquence FCS et doit être codé comme dans le Tableau A.1.

La présence du paramètre de "capacité V.44" dans un message XID (ou dans le message d'un autre protocole) indique que l'entité émettrice a la capacité décrite dans la présente Recommandation. Le format du paramètre de capacité V.44 pour un message XID lors de l'utilisation des procédures avec correction d'erreur V.42 est décrit dans l'Annexe A. Il est à noter que certains bits de ce paramètre doivent être ignorés lors de la négociation sur des connexions de modems, car ils spécifient des capacités V.44 à utiliser dans des réseaux de transmission par paquet, comme décrit dans l'Annexe B.

NOTE – Pendant la phase d'établissement de protocole, la présence du type de paramètre 0x40 avec un identificateur d'ensemble de paramètres "V.44" dans le sous-champ des données d'utilisateur de la trame XID indique une demande de compression de données V.44. L'entité répondante doit inclure des paramètres pour au moins un algorithme de compression (UIT-T V.42 *bis* ou UIT-T V.44) dans la trame XID de réponse.

7.4 Négociation des paramètres de compression de données

Une fois que la capacité V.44 a été déterminée, les valeurs des paramètres de compression de données peuvent être négociées entre les homologues de compression de données. Ces paramètres sont définis au Tableau 10 du paragraphe 8: les paramètres "P" sont les valeurs proposées par chaque partie négociatrice et les paramètres "N" sont les valeurs finales adoptées.

Les paramètres de compression de données peuvent être différents pour les deux sens de transmission. Pendant la négociation, chaque extrémité de la connexion peut proposer des valeurs pour l'un quelconque de ces paramètres: l'indice "T" indique que le paramètre considéré est associé au sens émission (codeur) de l'entité envoyant le message; l'indice "R" indique que le paramètre est associé au sens réception (décodeur) de l'entité envoyant le message.

Par conséquent, lors de la négociation des sens dans lesquels la compression de données fonctionnera, la réponse complémentaire à la valeur P_0 de 01 d'une entité (proposant un fonctionnement uniquement dans le sens du codeur de l'entité émettrice et du décodeur de l'entité répondante) est une valeur P_0 de 10 [indiquant l'acceptation d'un fonctionnement uniquement dans le sens du décodeur de l'entité répondante et du codeur de l'entité émettrice (d'origine)]. Une réponse de 01 ou 11 proposerait, de façon inappropriée, la compression de données dans un sens non demandé au départ; une réponse de 00 se traduirait par une compression de données dans aucun des sens.

Lors de la négociation des valeurs des paramètres, on utilise la plus petite valeur parmi les deux proposées. Par exemple, la valeur P_{2T} proposée par une entité est comparée à la valeur P_{2R} proposée par l'autre entité; si les deux valeurs sont valides, on utilisera la plus petite; toute tentative visant à spécifier une valeur inférieure au minimum est une erreur de procédure et doit se traduire par une déconnexion. La valeur finale adoptée est mise dans N_{7T} par l'entité proposant P_{2T} et dans N_{7R} par l'entité proposant P_{2R}.

L'Appendice I donne des indications concernant l'incidence des valeurs de ces paramètres sur la performance de la compression de données.

La négociation de ces valeurs peut se dérouler de deux façons différentes, comme indiqué dans la valeur du paramètre de capacité V.44: soit par le biais de la détermination de capacité XID, soit par le biais de la négociation après établissement de la liaison. La négociation par le biais de la procédure XID est l'option par défaut. Un modem qui reçoit un message XID demandant la négociation de paramètres XID doit répondre par un message XID contenant des paramètres V.44; par conséquent, un modem qui a demandé la négociation du mode paramètres dans la séquence XID, mais qui a reçu une demande de négociation XID, doit répondre lui aussi par un message XID contenant des paramètres V.44.

7.4.1 Négociation par le biais de XID

Dans le sous-champ des données d'utilisateur de XID, le paramètre de capacité V.44 indique comment les paramètres de compression de données doivent être négociés; s'il indique qu'il faut utiliser les procédures XID, les valeurs proposées par l'entité émettrice sont incluses plus tard dans le même sous-champ.

7.4.2 Négociation après établissement de la liaison

Si le paramètre de capacité V.44 indique que la négociation des paramètres doit avoir lieu après l'établissement de la liaison avec correction d'erreur, la fonction de commande l'exécute après l'établissement de la liaison et à tout moment après, selon qu'elle le juge nécessaire. La fonction de commande doit s'assurer qu'aucun transfert de données entre les homologues de compression de données n'est en cours et que le contrôle de flux est actif, avant de lancer une négociation des paramètres.

La fonction de commande utilise des primitives C-PARM pour la transmission des paramètres vers/depuis la fonction de compression de données, comme suit:

- à la réception d'une demande C-PARM provenant de la fonction de commande, le codeur doit:
  - passer du mode comprimé au mode transparent en utilisant les procédures du 6.5.1, ou bien rester dans le mode transparent;
  - passer au mode paramètre et transférer la commande EPM (*enter parameter mode*) pour indiquer cette transition au décodeur homologue;
  - transférer les paramètres transmis avec la demande C-PARM au décodeur homologue;
  - transférer le paramètre End (voir Tableau 10) pour indiquer un passage au mode transparent au décodeur homologue;
  - envoyer une confirmation C-PARM à la fonction de commande;
  - passer au mode transparent;
- à la réception d'une confirmation C-PARM provenant de la fonction de compression de données, la fonction de commande doit:
  - si les paramètres venant d'être transférés au décodeur homologue permettent de faire aboutir la négociation des paramètres, réinitialiser la fonction de compression de données conformément aux procédures définies au 7.5;
  - autrement, continuer à maintenir le contrôle de flux dans l'attente de la réception de paramètres en provenance de l'homologue de compression de données;
- à la réception d'une commande EPM provenant du codeur homologue en mode transparent, le décodeur doit:
  - passer au mode paramètre;
  - recevoir les paramètres;
  - passer au mode transparent lorsque le paramètre End est reçu;
  - envoyer une indication C-PARM à la fonction de commande pour lui transmettre les paramètres reçus;
- à la réception d'une indication C-PARM provenant de la fonction de compression de données, la fonction de commande doit:
  - si les paramètres transmis avec l'indication C-PARM permettent de faire aboutir la négociation des paramètres, réinitialiser la fonction de compression de données conformément aux procédures définies au 7.5;
  - autrement, envoyer une demande C-PARM pour transférer les paramètres de réponse à son homologue de compression de données.

7.5 Initialisation de la fonction de compression de données

Après aboutissement de la négociation de la capacité V.44, la fonction de commande doit envoyer la primitive de demande C-INIT à la fonction de compression de données. Si les paramètres de compression de données V.44 ont été négociés dans l'échange de messages XID, la primitive C-INIT doit indiquer la valeur négociée des paramètres; autrement, elle doit indiquer la valeur par défaut des paramètres définie dans le Tableau 10. La fonction de compression de données doit initialiser le dictionnaire du codeur à l'état défini par le 7.5.1 et le dictionnaire du décodeur à l'état défini par le 7.5.2. La fonction de compression de données doit être placée dans le mode comprimé et la valeur 0 est attribuée à ESCAPE.

La fonction de commande peut invoquer une réinitialisation de la fonction de compression de données. Elle doit envoyer une demande C-INIT à la fonction de compression de données sous réserve de l'une des conditions suivantes:

– réception d'une indication ou d'une confirmation L-ESTABLISH;

– réception d'une indication ou d'une confirmation L-SIGNAL, la primitive indiquant une forme avec destruction. Voir Tableau 4/V.42;

– réception d'une indication C-PARM indiquant un aboutissement de la négociation des paramètres lancée par cette entité. La primitive C-INIT doit indiquer la valeur des paramètres qui vient d'être négociée; ou

– réception d'une confirmation C-PARM indiquant un aboutissement de la négociation des paramètres lancée par l'homologue de compression de données. La primitive C-INIT doit indiquer la valeur des paramètres qui vient d'être négociée.

Il appartient à la fonction de commande de faire en sorte que les primitives de demande C-INIT ne soient envoyées que lorsqu'il n'y a pas de données en transit entre les fonctions de compression de données (par exemple dans les fonctions de contrôle d'erreur), pour garantir la synchronisation entre les codeurs et les décodeurs.

7.5.1 Etat initial du dictionnaire du codeur

Les valeurs initiales des variables du codeur sont les suivantes:

– le "mot de code suivant" $C_1 = N_5$;

– la "taille courante de mot de code" $C_2 = 6$;

– le "seuil de modification de la taille de mot de code" $C_3 = 64$;

– la "position courante dans l'historique" $C_4 = 0$;

– la "taille courante d'ordinal" $C_5 = 7$;

– tous les indices descendants de la rangée de racines sont mis sur des valeurs non valides.

7.5.2 Etat initial du dictionnaire du décodeur

Les valeurs initiales des variables du décodeur sont les suivantes:

– le "mot de code suivant" $C_1 = N_5$;

– la "taille courante de mot de code" $C_2 = 6$;

– la "position courante dans l'historique" $C_4 = 0$;

– la "taille courante d'ordinal" $C_5 = 7$.

7.6 Etablissement de connexion avec contrôle d'erreur

A la réception de la primitive de confirmation C-INIT provenant de la fonction de compression de données, la fonction de commande indique à l'ETTD que le transfert de données peut commencer.

7.7 Transfert de données entre l'interface ETTD/ETCD et la fonction de compression de données

Une fois la connexion établie, la fonction de commande doit demander le codage des données introduites au niveau de l'interface ETTD/ETCD. Pour ce codage, la fonction de commande doit envoyer la primitive de demande C-DATA à la fonction de compression de données. La primitive doit indiquer les données à coder.

A la réception de la primitive d'indication C-DATA provenant de la fonction de compression de données, la fonction de commande remet les données décodées à l'interface ETTD/ETCD.

Des procédures de contrôle de flux seront nécessaires pour éviter toute perte de données éventuelle due à un dépassement de mémoire-tampon. Lorsque les procédures définies dans la présente Recommandation sont utilisées conjointement avec celles définies dans l'UIT-T V.42, les procédures de contrôle de flux définies aux 7.3.1/V.42 et 8.4.2/V.42 s'appliquent.

7.8 Codage

Les caractères d'entrée provenant de la fonction de commande doivent être codés au moyen des procédures décrites au 6.3.

7.9 Transfert de données entre la fonction de compression de données et la fonction de contrôle d'erreur

A la réception de la primitive d'indication C-TRANSFER provenant de la fonction de compression de données, la fonction de commande doit envoyer la primitive de demande L-DATA à la fonction de contrôle d'erreur.

A la réception de la primitive d'indication L-DATA provenant de la fonction de contrôle d'erreur, la fonction de commande doit envoyer la primitive de demande C-TRANSFER à la fonction de compression de données.

La fonction de compression de données utilisera les procédures décrites au 6.6.

7.10 Décodage

Les données codées reçues de la fonction de commande doivent être décodées au moyen des procédures du 6.4.

7.11 Ajustements autonomes

En cours de fonctionnement, certaines variables sont modifiées de façon autonome par la fonction de compression de données. Le mode de fonctionnement peut également passer de façon autonome du mode comprimé au mode transparent.

7.11.1 Taille d'ordinal et STEPUP

Au moment de la réinitialisation de dictionnaire, le codeur et son décodeur homologue procèdent à une initialisation en vue du transfert d'un ordinal de 7 bits ($C_5 = 7$). Lorsque le codeur est sur le point de transférer le premier ordinal avec une valeur numérique supérieure à 127_{10}, il doit mettre C_5 à 8. S'il est en mode comprimé, il doit aussi transférer le code de commande STEPUP immédiatement avant de transférer le préfixe et l'ordinal.

7.11.2 Taille de mot de code et STEPUP

Pour le transfert de mots de code et de codes de commande, on utilise le nombre de bits défini par C_2. Au moment de la réinitialisation de dictionnaire, C_2 est mis à 6 et C_3 est mis à 64. Lorsque le codeur est sur le point de transférer un mot de code avec une valeur numérique supérieure ou égale à C_3:

a) il doit transférer le code de commande STEPUP, en utilisant la taille courante de mot de code, C_2;

b) la taille de mot de code C_2 doit être augmentée de un;

c) C_3 doit être doublé;

d) si le mot de code à transférer est toujours numériquement supérieur ou égal à C_3, il faut répéter les étapes a) à c).

Le préfixe et le mot de code sont ensuite transférés.

7.11.3 Arbre de nœuds plein

Lorsque le codeur n'est pas en mesure de créer un nouveau mot de code car l'arbre de nœuds est plein ($C_1 = N_{2T}$), il doit réinitialiser le dictionnaire, comme décrit au 7.12.

7.11.4 Historique plein

Lorsque l'historique est plein ($C_4 = N_{8T}$), le codeur doit mettre fin à toute activité de mise en correspondance de chaîne ou d'extension de chaîne en cours, transférer le ou les codes qui en résultent et réinitialiser le dictionnaire, comme décrit au 7.12. Le caractère d'entrée suivant doit être placé dans la première position de l'historique.

7.11.5 Surveillance de la compressibilité des données

Le codeur doit périodiquement appliquer un test pour déterminer la compressibilité des données. La nature du test n'est pas spécifiée dans la présente Recommandation; toutefois, il pourrait consister en une comparaison du nombre de bits nécessaires pour représenter un segment du flux de données avant et après compression.

La surveillance du flux de données du point de vue de la compressibilité se fait en continu, aussi bien dans le mode de fonctionnement comprimé que dans le mode de fonctionnement transparent.

En mode comprimé, si le codeur détermine que le flux de données n'est pas compressible, il doit passer au mode transparent, comme décrit au 6.5.1.

En mode transparent, si le codeur détermine que le flux de données est compressible, il doit passer au mode comprimé, comme décrit au 6.5.2.

7.12 Réinitialisation de dictionnaire

Pour réinitialiser son dictionnaire, le codeur le met dans l'état défini au 7.5.1. Si la fonction de compression de données est en mode comprimé, le codeur transfère en outre le code de commande REINIT; à la réception de REINIT, le décodeur met son dictionnaire dans l'état défini au 7.5.2.

7.13 Transfert de données exprès et FLUSH

Dans certaines conditions, il peut être nécessaire de mettre fin à la mise en correspondance de chaîne ou à l'extension de chaîne et de transférer immédiatement les éventuelles données codées partiellement. La spécification de ces conditions sort du cadre de la présente Recommandation, mais on peut donner comme exemple le cas où la fonction de contrôle d'erreur serait inactive. La fonction de commande doit envoyer une primitive de demande C-FLUSH à la fonction de compression de données et doit ensuite transférer les données restantes conformément au 7.9.

Si la fonction de compression de données est en mode comprimé, à la réception d'une demande C-FLUSH provenant de la fonction de commande, le codeur doit:

a) mettre fin à la procédure de mise en correspondance de chaîne ou d'extension de chaîne, comme décrit aux 6.3.1 et 6.3.2;

b) transférer le ou les codes restants conformément au 6.6;

c) mettre à jour le dictionnaire du codeur selon qu'il convient;

d) transférer le code de commande FLUSH;

e) si nécessaire, transférer suffisamment de bits 0 pour établir l'alignement des octets.

Le dictionnaire du codeur n'est pas réinitialisé. Il est mis à jour, selon qu'il convient, en adjoignant le caractère qui vient après le vidage.

A la réception d'un code de commande FLUSH, le décodeur doit établir l'alignement des octets. Le dictionnaire du décodeur n'est pas réinitialisé et la réception du code de commande FLUSH n'a pas d'incidence sur la création de nouvelles chaînes: le décodeur doit traiter le code reçu après le code FLUSH en continuité avec le code reçu juste avant le code FLUSH, conformément au processus de création de chaîne dans le décodeur, défini au 6.4.2.

Si la fonction de compression de données est en mode transparent, à la réception d'une demande C-FLUSH provenant de la fonction de commande, le codeur doit transférer toutes les données d'entrée jusqu'à ce stade.

7.14 Séquence de commandes ESCAPE

Une séquence de commandes en mode transparent doit être constituée de ESCAPE, suivi de l'une des commandes énumérées dans le Tableau 9 donné plus haut, à l'exception de ESCAPE.

La valeur de ESCAPE est variable. Au moment de l'initialisation de la fonction de compression de données, la valeur 0 est attribuée à ESCAPE. En mode transparent, si la valeur courante de ESCAPE est détectée dans le flux de données provenant de l'ETTD, elle doit être transférée et suivie immédiatement de la commande EID. On modifie alors la valeur courante de ESCAPE en lui ajoutant 51_{10}, l'addition étant effectuée modulo 256. Il est à noter qu'en mode comprimé ou en mode paramètre, la valeur courante de ESCAPE N'est PAS modifiée si elle est détectée dans le flux de données provenant de l'ETTD.

7.15 Mesure consécutive à la détection de C-ERROR

L'indication C-ERROR est utilisée pour informer la fonction de commande qu'une anomalie (par exemple, une erreur de procédure ou une perte de synchronisation) a été détectée par la fonction de compression de données. La fonction de commande doit prendre une mesure de récupération appropriée, y compris le rétablissement de la connexion avec correction d'erreur.

Les conditions suivantes reconnues par le décodeur entraînent la génération d'une primitive d'indication C-ERROR:

– réception d'un code de commande STEPUP qui entraînerait le dépassement de N_1 par C_2;

– réception d'un code de commande STEPUP qui entraînerait le dépassement de 8 par C_5;

– réception d'un mot de code supérieur à C_1.

8 Paramètres

Le Tableau 10 définit les paramètres négociés entre les fonctions de compression de données homologues. Les paramètres "P" sont les valeurs proposées par l'une des parties négociatrices et les paramètres "N" sont les valeurs finales adoptées. Le Tableau 11 définit tous les paramètres de fonctionnement pour la compression de données. (MSB désigne l'octet de plus fort poids, LSB l'octet de plus faible poids.)

Tableau 10/V.44 – Paramètres de négociation pour la fonction de compression de données

Paramètre	Identificateur 8……1	Longueur (octets)	Valeur 8……1	Signification	Intervalle	Valeur par défaut	Valeur adoptée
	00000000 to 01000000			Non utilisé			
C_0	01000001	00000001	PM00000N	Capacité V.44: Valeur de N: 0 Négociation de paramètre via un protocole (c'est-à-dire XID) 1 Négociation de paramètre après établissement de liaison Les bits P et M ne sont pas utilisés par les connexions de modems: voir l'Annexe B.			
P_0	01000010	00000001	000000RT	Demande de compression de données: Valeur de RT: 00 aucun sens 01 sens émission uniquement 10 sens réception uniquement 11 les deux sens	00-11	11	
P_{1T}	01000011	00000010	(MSB) (LSB)	Nombre proposé de mots de code (y compris les codes de commande) dans le sens émission	256-65535	1024	N_{2T}
P_{1R}	01000100	00000010	(MSB) (LSB)	Nombre proposé de mots de code (y compris les codes de commande) dans le sens réception	256-65535	1024	N_{2R}

Tableau 10/V.44 – Paramètres de négociation pour la fonction de compression de données *(fin)*

Paramètre	Identificateur 8……1	Longueur (octets)	Valeur 8……1	Signification	Intervalle	Valeur par défaut	Valeur adoptée
P_{2T}	01000101	00000001		Longueur de chaîne maximale proposée dans le sens émission	32-255	255	N_{7T}
P_{2R}	01000110	00000001		Longueur de chaîne maximale proposée dans le sens réception	32-255	255	N_{7R}
P_{3T}	01000111	00000010	(MSB) (LSB)	Longueur d'historique proposée dans le sens émission	≥512	$3 \times P_{1T}$	N_{8T}
P_{3R}	01001000	00000010	(MSB) (LSB)	Longueur d'historique proposée dans le sens réception	≥512	$3 \times P_{1R}$	N_{8R}
	01001001 to 11111110			Réservés pour une utilisation future			
End	11111111			Fin des paramètres: sortie du mode paramètre			

Tableau 11/V.44 – Paramètres de fonctionnement pour la fonction de compression de données

Paramètre	Signification	Valeur
N_{1T}, N_{1R}	Taille maximale de mot de code (en bits)	Déduite de N_{2T}, N_{2R}
N_{2T}, N_{2R}	Nombre maximal de mots de code (y compris les codes de commande)	Voir Tableau 10
N_3	Taille de caractère d'entrée, en bits	8
N_4	Nombre de caractères de l'alphabet	256
N_5	Nombre de codes de commande et premier mot de code disponible	4
N_6	Réservé	
N_{7T}, N_{7R}	Longueur maximale de chaîne	Voir Tableau 10
N_{8T}, N_{8R}	Taille d'historique	Voir Tableau 10

Le Tableau 12 définit des variables de fonctionnement pour la fonction de compression de données. Un ensemble distinct de ces variables doit être tenu à jour par le codeur et par le décodeur.

Tableau 12/V.44 – Variables de fonctionnement pour la fonction de compression de données

Paramètre	Signification
C_1	Valeur de mot de code de l'entrée disponible suivante dans l'arbre de nœuds (codeur) ou dans la collection de chaînes (décodeur).
C_2	Taille courante de mot de code, en bits.
C_3	Seuil de modification de la taille de mot de code; vaut 2 à la puissance C_2. Utilisé uniquement par le codeur.
C_4	Position courante dans l'historique.
C_5	Taille d'ordinal, en bits.

ANNEXE A

Champ d'information XID pour la négociation de la capacité V.44 en cas d'utilisation avec V.42

On se reportera aux 7.3 et 7.4.1 pour la négociation XID.

Tableau A.1/V.44 – Sous-champ des données d'utilisateur XID pour la négociation des paramètres de compression de données V.44

MSB: octet de plus fort poids LSB: octet de plus faible poids	Bit 8.....1	
Identificateur de groupe	1 1 1 1 1 1 1 1	Sous-champ des données d'utilisateur
Identificateur de paramètre	0 1 0 0 0 0 0 0	UIT-T V.44 – identificateur d'ensemble de paramètres
Longueur de paramètre	0 0 0 0 0 0 1 1	Longueur de chaîne, en octets: 3
Valeur de paramètre	0 1 0 1 0 1 1 0 0 0 1 1 0 1 0 0 0 0 1 1 0 1 0 0	V 4 4
Identificateur de paramètre	0 1 0 0 0 0 0 1	Capacité de l'UIT-T V.44 (C_0)
Longueur de paramètre	0 0 0 0 0 0 0 1	Longueur de champ, en octets: 1
Valeur de paramètre	PM 0 0 0 0 0 N	Valeur de PM: 00 ni la méthode paquet ni la méthode multipaquet n'est prise en charge: pour les connexions de modems uniquement 01 non valable 10 méthode paquet prise en charge 11 méthodes paquet et multipaquet prises en charge Le bit N n'est pas utilisé dans des réseaux par paquets Valeur de N: 0 négociation de paramètres en utilisant l'échange XID et les paramètres ci-dessous 1 négociation de paramètres après l'établissement de liaison Les bits P et M ne sont pas pris en compte pour les connexions de modems.

Tableau A.1/V.44 – Sous-champ des données d'utilisateur XID pour la négociation des paramètres de compression de données V.44 (*fin*)

Identificateur de paramètre	0 1 0 0 0 0 1 0	UIT-T V.44 – demande de compression de données (P_0)
Longueur de paramètre	0 0 0 0 0 0 0 1	Longueur de champ, en octets: 1
Valeur de paramètre	0 0 0 0 0 0 RT	Le sens de compression demandé est signalé par la valeur de RT: 00 aucun sens 01 sens émission uniquement 10 sens réception uniquement 11 les deux sens
Identificateur de paramètre	0 1 0 0 0 0 1 1	UIT-T V.44 – nombre de mots de code dans le sens émission (P_{1T})
Longueur de paramètre	0 0 0 0 0 0 1 0	Longueur de champ, en octets: 2
Valeur de paramètre	NNNNNNNN (MSB) NNNNNNNN (LSB)	Valeur du paramètre P_{1T}
Identificateur de paramètre	0 1 0 0 0 1 0 0	UIT-T V.44 – nombre de mots de code dans le sens réception (P_{1R})
Longueur de paramètre	0 0 0 0 0 0 1 0	Longueur de champ, en octets: 2
Valeur de paramètre	NNNNNNNN (MSB) NNNNNNNN (LSB)	Valeur du paramètre P_{1R}
Identificateur de paramètre	0 1 0 0 0 1 0 1	UIT-T V.44 – longueur maximale de chaîne dans le sens émission (P_{2T})
Longueur de paramètre	0 0 0 0 0 0 0 1	Longueur de champ, en octets: 1
Valeur de paramètre	NNNNNNNN	Valeur du paramètre P_{2T}
Identificateur de paramètre	0 1 0 0 0 1 1 0	UIT-T V.44 – longueur maximale de chaîne dans le sens réception (P_{2R})
Longueur de paramètre	0 0 0 0 0 0 0 1	Longueur de champ, en octets: 1
Valeur de paramètre	NNNNNNNN	Valeur du paramètre P_{2R}
Identificateur de paramètre	0 1 0 0 0 1 1 1	UIT-T V.44 – longueur d'historique dans le sens émission (P_{3T})
Longueur de paramètre	0 0 0 0 0 0 1 0	Longueur de champ, en octets: 2
Valeur de paramètre	NNNNNNNN (MSB) NNNNNNNN (LSB)	Valeur du paramètre P_{3T}
Identificateur de paramètre	0 1 0 0 1 0 0 0	UIT-T V.44 – longueur d'historique dans le sens réception (P_{3R})
Longueur de paramètre	0 0 0 0 0 0 1 0	Longueur de champ, en octets: 2
Valeur de paramètre	NNNNNNNN (MSB) NNNNNNNN (LSB)	Valeur du paramètre P_{3R}

ANNEXE B

Fonctionnement de l'algorithme V.44 dans les réseaux de transmission par paquet

Le fonctionnement de l'algorithme V.44 dans les réseaux de transmission par paquet diffère de son fonctionnement dans les modems, principalement parce qu'un terminal de réseau a connaissance des frontières de paquet, y compris de la délimitation entre l'en-tête et les données. Un terminal de réseau peut identifier un paquet complet de couche transport et le traiter en tant qu'unité, alors qu'un modem traite un flux continu de données ne se présentant pas sous la forme de trames.

La présente annexe décrit deux méthodes permettant de compresser des données mises en paquets: la méthode paquet, dans laquelle chaque paquet est traité séparément, et la méthode multipaquets dans laquelle plusieurs paquets ou parties de paquets sont traités les uns à la suite des autres. La méthode multipaquets nécessite une garantie de la remise de tous les paquets. La principale méthode de fonctionnement de l'algorithme V.44, décrite dans le corps de la présente Recommandation, est appelée "méthode flux".

Dans la présente annexe, le terme "paquet" désigne un paquet de couche Transport, un paquet IP, une trame, un bloc, une unité N-PDU, etc. Le terme "réseau de transmission par paquet" sert à décrire tout réseau qui traite des paquets (IP, relais de trames, etc.). Le terme "segment" sert à décrire une partie de paquet qui a été segmenté pour permettre la transmission sur une liaison.

Les caractéristiques importantes de la compression de données dans les réseaux de transmission par paquet sont notamment les suivantes:

– chaque paquet est comprimé et transmis en tant qu'unité. Par conséquent, un vidage est exécuté, au moyen d'un code de commande FLUSH, après la transmission du dernier octet de données d'un paquet au codeur pour terminer la compression et transmettre le paquet dans sa totalité;

– il n'est pas nécessaire de négocier les paramètres de compression de données, étant donné que l'ensemble des paramètres V.44 par défaut pour les réseaux de transmission par paquet est défini au B.1. Si nécessaire, une négociation des paramètres de compression de données et de paramètres connexes peut être effectuée en dehors de l'algorithme V.44, avant le transfert de données. Cette négociation peut utiliser un échange de messages XID ou d'un autre protocole;

– en cas de négociation V.44 entre homologues de compression de données (c'est-à-dire en utilisant XID), les bits P et M du paramètre de capacité V.44 sont utilisés pour indiquer que la méthode paquet V.44 et la méthode multipaquets V.44 sont prises en charge. Un homologue qui prend en charge la méthode multipaquets V.44 doit aussi prendre en charge la méthode paquet. S'il prend en charge ces deux méthodes mais que l'autre homologue ne prenne en charge que la méthode paquet, c'est la méthode paquet qui est retenue. Voir Tableau A.1;

– le mode transparent n'est pris en charge ni dans la méthode paquet ni dans la méthode multipaquets. Un indicateur est utilisé pour signaler à l'autre extrémité si le paquet est comprimé ou non. Par conséquent, on transmet soit le paquet d'origine soit le résultat de l'opération de compression de données, selon celui qui est le plus petit. L'indicateur est soit un bit (ou une autre indication) dans l'en-tête de paquet ou de protocole, soit le code de commande ETM (précédé de son préfixe de code "1") dans le premier octet du paquet, suivi des données d'origine, avec alignement des octets. (Dans le présent contexte, le code ETM est simplement un indicateur et n'implique pas le comportement du 6.5.)

NOTE – Dans les réseaux de transmission par paquet, on peut améliorer le temps d'exécution, lors de l'augmentation de la taille du mot de code (de 8 à 9 bits, par exemple), en vérifiant si la compression de données s'est bien déroulée jusqu'à présent. Si tel n'est pas le cas, le codage peut être interrompu et le paquet initial transmis.

B.1 Méthode paquet V.44

La méthode paquet est la méthode V.44 par défaut dans les réseaux de transmission par paquet. Avec cette méthode, la négociation entre homologues de compression de données n'est pas nécessaire.

B.1.1 Description générale

La méthode paquet doit être utilisée dans les réseaux dans lesquels des paquets ou des segments sont transmis dans le mode sans acquittement, sans garantie de remise. En pareil cas, il est possible qu'un paquet précédent n'ait pas été reçu, mais la méthode paquet ne repose pas sur les informations contenues dans les paquets précédents pour la compression et la décompression de chaque paquet. On peut aussi préférer la méthode paquet même si la remise est garantie, en raison de sa simplicité par rapport à la méthode multipaquet. La méthode paquet permet à une station pivot de réseau ou à un commutateur de paquet prenant en charge de nombreuses liaisons d'utiliser une seule instance de contexte et de dictionnaires de codeur et de décodeur V.44 pour prendre en charge toutes les liaisons, puisque chaque paquet est traité séparément.

En cas de négociation V.44 entre homologues de compression de données, on indique la prise en charge de la méthode paquet dans le paramètre de capacité V.44 en mettant le bit P à "1" et le bit M à "0".

Les différences spécifiques entre la méthode paquet V.44 et la méthode flux V.44 sont notamment les suivantes:

– chaque paquet est comprimé et transmis en tant qu'unité; le codeur transfère un code de commande FLUSH pour terminer le paquet;

– entre deux paquets, les dictionnaires du codeur et du décodeur sont réinitialisés. La réinitialisation est effectuée par la fonction de commande ou est intégrée à l'algorithme proprement dit. Le codeur ne transfère pas de code de commande REINIT;

– les données non comprimées dans le paquet en cours de codage servent d'historique du codeur et les données décomprimées placées dans une mémoire-tampon de sortie par le décodeur servent d'historique du décodeur;

– lorsque le dernier mot de code a été créé (l'arbre de nœuds est plein), les activités de mise en correspondance de chaîne et d'extension de chaîne se poursuivent normalement, mais aucun autre mot de code n'est créé. Par conséquent, les longueurs d'extension de chaîne sont transférées, mais les extensions de chaîne ne sont pas utilisées pour définir d'autres segments de chaîne dans l'arbre de nœuds ou de nouvelles chaînes dans la collection de chaînes;

– les valeurs par défaut des paramètres de compression de données sont énumérées au B.1.2. D'autres valeurs peuvent être déterminées par négociation.

B.1.2 Valeurs par défaut des paramètres de compression de données pour la méthode paquet

En l'absence de négociation de paramètre entre dispositifs d'un réseau de transmission par paquet, l'algorithme V.44 utilise, dans le cas de la méthode paquet, les valeurs par défaut suivantes des paramètres:

– P_0 – 11, les deux sens;

– P_{1T} – 1525 mots de code;

– P_{1R} – 1525 mots de code;

– P_{2T} – 255, longueur maximale de chaîne;

– P_{2R} – 255, longueur maximale de chaîne;

– P_{3T} – non applicable;

– P_{3R} – non applicable.

B.2 Méthode multipaquets V.44

Au prix d'une plus grande complexité et d'une plus grande mémoire, la méthode multipaquets peut offrir une performance de compression meilleure que la méthode paquet. Elle ne peut être utilisée que dans un réseau de transmission par paquet dans lequel les paquets ou les segments sont transmis dans le mode avec acquittement, avec garantie de remise. La méthode multipaquets impose une plus grande complexité et une plus grande mémoire aux stations pivot de réseau et aux commutateurs de paquet qui prennent en charge de nombreuses liaisons point à point avec des terminaux ou avec d'autres dispositifs d'utilisateur, étant donné que chaque liaison point à point utilisant la méthode multipaquets nécessite une instance individuelle de contexte et de dictionnaires de codeur et de décodeur.

La méthode multipaquets nécessite un superensemble des procédures et structures utilisées dans la méthode paquet. Les dispositifs qui prennent en charge la méthode multipaquets doivent aussi prendre en charge la méthode paquet.

B.2.1 Description générale

En cas de négociation V.44 entre homologues de compression de données, on indique la prise en charge de la méthode multipaquets dans le paramètre de capacité V.44. Si l'un des homologues indique la méthode multipaquets et l'autre la méthode paquet, c'est la méthode paquet qui est utilisée.

Les différences spécifiques entre la méthode multipaquets V.44 et la méthode flux V.44 sont mineures et sont notamment les suivantes:

– chaque paquet est comprimé et transmis en tant qu'unité; le codeur transfère un code de commande FLUSH pour terminer le paquet;

– la taille d'historique doit être supérieure à la longueur maximale de paquet. La longueur d'historique est de 3072 par défaut, sauf si une autre valeur a été négociée ou préétablie. Si les tailles des dictionnaires sont modifiées par rapport aux valeurs par défaut par le biais d'une négociation, la taille d'historique doit être modifiée en conséquence;

– le codeur et le décodeur doivent réinitialiser leur dictionnaire après le traitement d'un paquet pour lequel la compression n'a pas abouti (c'est-à-dire un paquet dans lequel les données d'entrée d'origine étaient plus petites que le résultat de l'opération de compression), donnant lieu par conséquent à la transmission du paquet initial;

– avant ou pendant le codage, les données à compresser sont copiées du paquet dans les positions suivantes disponibles de l'historique du codeur, jusqu'à la limite de la taille maximale de l'historique. Si l'historique est plein au milieu d'un paquet, après codage du caractère occupant la dernière position de l'historique, le dictionnaire est réinitialisé, un code de commande REINIT est transféré et le reste des données du paquet est copié dans les premières positions de l'historique et comprimé;

– lorsque le dernier mot de code a été créé (l'arbre de nœuds est plein), le dictionnaire du codeur est réinitialisé, un code de commande REINIT est transféré (même au milieu d'un paquet) et les caractères non traités de l'historique sont transférés dans les premières positions de celui-ci.

B.2.2 Valeurs par défaut des paramètres de compression de données pour la méthode multipaquets

Dans la méthode multipaquets, les valeurs par défaut sont les mêmes que celles utilisées dans la méthode flux. Voir Tableau 10.

A noter que, pour le bon fonctionnement de la méthode multipaquets, le nombre de mots de code utilisables doit être supérieur au paquet de longueur maximale prévu. Ainsi, pour un réseau dont le paquet de longueur maximale est de 1518 octets, le nombre de mots de code doit être d'au moins 1522, compte tenu des 4 mots de code réservés, et considérablement plus élevé pour une compression optimale.

APPENDICE I

Notes relatives à l'implémentation

Les notes qui suivent donnent des informations sur l'implémentation de l'algorithme de compression de données et sur la sélection des paramètres. En faisant varier le nombre de mots de code et la longueur d'historique, on peut faire évoluer l'algorithme V.44 pour tirer parti de la mémoire disponible.

I.1 Sélection de N_2: nombre total de mots de code

La taille de dictionnaire est égale à N_2 (dans l'hypothèse où les entrées sont fournies pour les codes de commande réservés). En général, la sélection d'une grande valeur de N_2 permet une meilleure compression au prix d'une plus grande mémoire pour les dictionnaires. La sélection d'une grande valeur de N_2 signifie que le nombre de chaînes disponibles est augmenté, mais aussi que la valeur de N_1 est augmentée en conséquence. Dans certains cas, le gain de performance obtenu par la sélection d'un dictionnaire plus grand risque d'être réduit à néant par la plus grande taille de mot de code nécessaire. De fait, pour certains types de données, on peut obtenir une meilleure performance en utilisant un dictionnaire plus petit. En général, la valeur 2048 de N_2 offre une bonne performance de compression pour des types de données très divers.

Pour pouvoir prendre en charge des plates-formes matérielles ayant une mémoire limitée, N_2 peut être différent dans les deux sens, ce qui permettrait, par exemple, d'avoir un plus grand nombre de mots de code dans le sens serveur-client que dans le sens client-serveur, afin d'améliorer la compression dans le sens téléchargement depuis le web tout en minimisant la mémoire totale nécessaire pour le codeur et pour le décodeur.

Il n'est pas nécessaire que N_2 soit une puissance de 2. Comme le dictionnaire est réinitialisé dès que l'entrée de l'arbre de nœuds associée au mot de code $(N_2 – 1)$ est créée, il n'y a pas, dans l'algorithme V.44, de pénalité permanente liée à l'augmentation vers une valeur supérieure de N_1 pour obtenir d'autres mots de code. C'est un avantage par rapport à l'algorithme V.42 *bis*, qui réinitialise rarement le dictionnaire et maintient ainsi N_1 à sa valeur maximale, une fois que celle-ci a été atteinte.

I.2 Sélection de N_7: longueur maximale de chaîne

Dans le cas de l'algorithme de compression de données décrit dans la présente Recommandation, contrairement à l'algorithme V.42 *bis*, une plus grande longueur maximale de chaîne permet d'obtenir une compression bien meilleure. Il est donc suggéré d'utiliser la valeur maximale de 255.

I.3 Sélection de N_8: structures de données et longueur d'historique

L'historique contient tous les caractères d'entrée du codeur, ou tous les caractères décodés par le décodeur, depuis la plus récente réinitialisation du dictionnaire. Pour optimiser la performance, le dictionnaire devrait être réinitialisé le moins fréquemment possible. Il est réinitialisé chaque fois que soit l'historique soit l'arbre de nœuds est plein; toutefois, pour les données qui sont davantage compressibles, la performance est généralement meilleure si l'arbre de nœuds est plein avant que

l'historique ne le soit. Par exemple, pour un taux de compression d'environ 10:1, la performance pour un arbre de nœuds de 2044 (correspondant à une valeur de 2048 pour N_{2T}) mots de code est la meilleure avec un historique d'environ 15 000 caractères (même si on peut obtenir une bonne performance avec 4096 caractères).

Pour un fonctionnement satisfaisant en mode V.44, il est proposé que la longueur d'historique (N_8) soit toujours supérieure ou égale à $2 \times N_2$. Cependant, pour pouvoir prendre en charge des plates-formes matérielles ayant une mémoire limitée, N_8 peut être différent dans les deux sens, ce qui permettrait, par exemple, d'avoir un plus grand historique dans le sens serveur-client que dans le sens client-serveur, afin d'améliorer la compression dans le sens descendant (téléchargement de page web) lorsque la mémoire totale pour le codeur et pour le décodeur est fixe. Par exemple, au lieu d'utiliser la valeur par défaut de $3 \times N_2$ pour les deux sens, on pourrait utiliser $4 \times N_2$ dans le sens serveur-client et seulement $2 \times N_2$ dans le sens client-serveur.

Une autre méthode permettant de maximiser l'utilisation des ressources mémoire du codeur consiste à regrouper la mémoire attribuée à l'historique et à l'arbre de nœuds et à permettre à l'historique de croître au-delà de sa limite nominale dans une mémoire non utilisée réservée aux mots de code de l'arbre de nœuds. Alors, si les données sont davantage compressibles que d'habitude et que l'arbre de nœuds croît plus lentement que d'habitude, le codeur pourra fonctionner plus longtemps avant toute réinitialisation, car l'historique peut utiliser le reste de mémoire prévu pour l'arbre de nœuds.

Le codeur ne doit pas utiliser un historique plus grand que ce que le décodeur a négocié, pour éviter tout dépassement. Toutefois, dans certaines applications utiles (par exemple décodeur dans le client, codeur dans le serveur), les ressources mémoire du décodeur peuvent être beaucoup plus grandes que celles du codeur. Si le codeur négocie une taille d'historique de décodeur plus grande que celle qu'il attribue à son propre historique, il a une marge pour croître au-delà de sa propre attribution, tant qu'il procède à des réinitialisations avant que l'historique du décodeur ne soit plein.

A titre d'exemple, supposons que le serveur ait 6 000 octets prévus pour l'historique du codeur (6000 caractères) et 14 308 octets prévus pour l'arbre de nœuds du codeur (2044 mots de code, avec 7 octets par entrée de mot de code: chaque entrée de mot de code utilise un octet pour la longueur de segment de chaîne, deux octets pour l'indice d'historique, deux octets pour l'indice descendant et deux octets pour l'indice latéral); la mémoire cumulée pour le codeur est de 20 308 octets. Toutefois, lors de la négociation avec le client, le codeur du serveur propose une valeur plus grande que 6000 pour l'historique, par exemple 15 000. Le client peut accepter ce nombre ou adopter un nombre plus petit; néanmoins, la valeur adoptée, désignée par N_{8T} par le codeur, sera probablement supérieure à la valeur 6000 prévue par le codeur pour l'historique. Supposons que N_{8T} vaille 15 000.

Dans le codeur du serveur, l'historique et l'arbre de nœuds partageront une matrice de mémoire commune telle que:

- la première position de l'historique est au *premier* octet de la matrice de mémoire;
- l'indice courant d'historique est incrémenté à chaque nouveau caractère placé dans l'historique;
- la structure de chaque entrée de l'arbre de nœuds (tous les champs de données d'un certain mot de code) comprend 7 octets consécutifs dans la mémoire;
- la structure du premier mot de code (4) est située dans les 7 *derniers* octets de la matrice de mémoire;
- la structure du dernier mot de code (2047) est située dans les 7 octets immédiatement *après* le dernier octet de l'historique;
- le pointeur de la structure de mot de code suivante disponible est décrémenté à chaque mot de code créé.

En fonctionnement, l'historique est rempli à partir du premier octet de la matrice puis on procède par incrémentation. Les entrées de l'arbre de nœuds sont créées à partir des 7 derniers octets de la matrice puis on procède par décrémentation. Le codeur réinitialise le dictionnaire chaque fois que l'une des situations suivantes se produit:

a) le dernier mot de code est créé ($C_1 = N_{2T} – 1 = 2047$);

b) la position courante de l'historique atteint N_{8T} ($C_4 = N_{8T} = 15\ 000$);

c) la position courante de l'historique est à moins de N_{7T} de la structure de mot de code suivante disponible.

La condition a) est normale, indiquant que l'arbre de nœuds du codeur est plein. La condition b) permet d'empêcher tout dépassement de l'historique du *décodeur*. La condition c) garantit que l'historique du codeur ne procède pas à une annulation et à un remplacement d'entrées de mot de code, lorsqu'il s'étend dans la mémoire réservée aux structures de mots de code.

Le résultat est que si le taux de compression des données est de l'ordre de 10:1, l'arbre de nœuds croît lentement par rapport à l'historique et l'historique peut s'étendre dans la mémoire attribuée au départ à l'arbre de nœuds, augmentant potentiellement l'historique utilisable d'un nombre pouvant aller jusqu'à 9000 (= 15 000 – 6000) octets.

I.4 Compression efficace de données Unicode

Unicode est un système de codage de caractères à 16 bits normalisé sur le plan international et conçu pour prendre en charge l'échange, le traitement et l'affichage de nombreux langages écrits modernes. Il est constitué de représentations numériques de presque tous les caractères rencontrés dans les principaux langages écrits d'Europe, des Amériques, du Moyen-Orient, d'Inde, d'Afrique, d'Asie et du Pacifique. C'est une implémentation de l'ISO/CEI 10646, restreinte à 2 octets, qui est aussi connue sous le nom de UCS-2.

L'algorithme de compression de données défini dans la présente Recommandation ne porte pas expressément sur les caractères à 16 bits, mais compressera efficacement les données Unicode sous forme de caractères à 8 bits. Pour les applications dans lesquelles les couches de protocole au-dessus de V.42 savent que UCS-2 constitue une partie importante du trafic, il est recommandé d'appliquer la compression spécifiée dans le rapport Unicode Technical Report #6 aux données avant de les transmettre à l'algorithme V.44, car cela peut améliorer la performance globale.

I.5 Applicabilité du mode transparent

Le mode transparent est obligatoire pour l'exploitation par connexion de modem. Dans des réseaux par paquets, le mode transparent n'est pas nécessaire car l'en-tête de la trame est pris en compte. Dans tous les autres cas, on serait fondé à l'utiliser, comme cela a été admis.

I.6 Calcul de la performance de compression

Le calcul de la performance de compression peut être exprimé comme le rapport entre le nombre de caractères reçus par le codeur et le nombre d'octets transférés à la fonction de commande. Le nombre de caractères et le nombre d'octets doivent être mis à zéro au moment de l'initialisation de la fonction de compression de données ainsi qu'entre les tests.

I.7 Différences entre les algorithmes V.44 et V.42 *bis*

On énumère ci-après les principales différences entre l'algorithme de compression de données décrit dans la présente Recommandation et celui qui est décrit dans l'UIT-T V.42 *bis*:

a) pour la plupart des types de fichier rencontrés en navigant sur Internet, l'algorithme V.44 permet d'obtenir des taux de compression supérieurs;

b) le codeur et le décodeur utilisent tous deux une mémoire-tampon d'historique;

c) la structure du dictionnaire du décodeur est différente de la structure du dictionnaire du codeur;

d) les tailles de dictionnaire, etc., pour les sens émission et réception sont négociées de façon indépendante;

e) les nœuds figurant dans le dictionnaire du codeur peuvent définir des segments de chaîne, au lieu de simples caractères;

f) les nœuds de dictionnaire utilisent des indices pointant sur la position d'un caractère dans l'historique, et non sur les caractères proprement dits;

g) on procède à une réinitialisation du dictionnaire lorsqu'il est plein, au lieu de procéder à la récupération de mots de code;

h) on ne réserve pas les 256 premiers mots de code pour tous les caractères à 8 bits possibles;

i) on utilise une procédure d'extension de chaîne pour coder rapidement une chaîne rencontrée pour la deuxième fois;

j) on transfère des codes pour les caractères et pour les longueurs d'extension de chaîne ainsi que pour les chaînes et les codes de commande;

k) on peut utiliser immédiatement tous les mots de code créés par le codeur, même le plus récent, un mot de code que le décodeur peut ne pas avoir encore créé;

l) la valeur de ESCAPE n'est pas modifiée lorsqu'elle est détectée en mode comprimé;

m) le dictionnaire du décodeur n'est pas mis à jour en mode transparent;

n) les dictionnaires sont réinitialisés lors du passage du mode transparent au mode comprimé;

o) la meilleure réponse à une proposition de paramètre P_0 est l'inversion: par exemple, la réponse complémentaire à une proposition, formulée par l'entité A, de compression dans le *sens émission* (de l'entité A) *uniquement*, est un accord, de la part de l'entité B, pour une compression dans le *sens réception* (de l'entité B) *uniquement*.

APPENDICE II

Illustration du fonctionnement de l'algorithme V.44

Dans les exemples donnés dans le présent appendice, on suppose une implémentation de codeur qui adjoint des caractères et met à jour l'arbre de nœuds avant la réception effective du caractère suivant. Par conséquent, à certaines étapes dans ces exemples, le contenu de l'arbre de nœuds peut différer du contenu qu'il aurait pour une implémentation qui adjoindrait des caractères et mettrait à jour l'arbre de nœuds uniquement après la réception du caractère suivant.

II.1 Compression et décompression de "ABCDEXABCDEYABCDE FF_H AC"

Le présent paragraphe vise à illustrer le fonctionnement de l'algorithme V.44 en mode compression, en montrant le traitement d'un exemple spécifique. Pour cela, on montre, à différentes étapes:

i) les nouveaux caractères introduits dans le codeur;

ii) l'analyse du flux d'entrée par l'algorithme, pour le codage;

iii) la structure de dictionnaire révisée du codeur;

iv) les informations transférées par le codeur et la méthode par laquelle elles sont codées;

v) l'analyse des données comprimées par l'algorithme, pour le décodage;

vi) la structure de dictionnaire révisée du décodeur.

Dans cet exemple, on suppose que la structure de dictionnaire vient juste d'être réinitialisée, de sorte qu'il n'y a pas d'historique précédent; C_5 est mis à 7 et C_2 est mis à 6. Les caractères sont introduits un par un, mais on présente ici des étapes, en en regroupant certaines lorsque le traitement est évident. Dans la rangée de racines, les racines avant le "A" ne sont pas montrées.

Les caractères non comprimés sont placés dans l'historique du codeur à mesure qu'ils arrivent; le traitement par le codeur produit des codes binaires, qui sont transférés; après la réception des données comprimées par le décodeur, les caractères décodés sont placés dans l'historique du décodeur.

Etape 0: pas de donnée précédente.

L'état du codeur est le suivant:

Historique du codeur:

Position																			
Caractère																			

Rangée de racines:

Racine	A	B	C	D	E	..	X	Y	..	FF_H
Indice descendant	-	-	-	-	-	-	-	-	-	-

Arbre de nœuds:

Mot de code												
Position du premier caractère												
Longueur de segment												
Indice descendant												
Indice latéral												

L'état du décodeur est le suivant:

Collection de chaînes:

Mot de code												
Position du dernier caractère												
Longueur de chaîne												

Historique du décodeur:

Position																			
Caractère																			

Etape 1:

Nouveaux caractères: A

Analyse par le codeur: dans la rangée de racines, il n'existe pas d'indice descendant pour "A", qui est donc transféré sous forme d'ordinal. La première entrée disponible de l'arbre de nœuds (mot de code 4) est utilisée pour créer un segment de chaîne: celui-ci a comme position du premier caractère 1 (la position suivant "A"), comme longueur de segment 1 (c'est un segment à 1 caractère) et n'a ni indice descendant ni indice latéral valides: il s'agit essentiellement de l'adjonction du caractère suivant, sans examen, pour créer une chaîne à deux caractères. La rangée de racines est également mise à jour avec un indice descendant pour "A", pointant sur le mot de code 4.

Historique du codeur:

Position	0																		
Caractère	A																		

Rangée de racines:

Racine	A	B	C	D	E	..	X	Y	..	FF_H
Indice descendant	4	-	-	-	-	-	-	-	-	-

Arbre de nœuds:

Mot de code	4											
Position du premier caractère	1											
Longueur de segment	1											
Indice descendant	-											
Indice latéral	-											

Transféré: ordinal pour "A": 41_H = "1000001"

Analyse par le décodeur: l'ordinal pour "A" est transformé en caractère à 8 bits et placé dans l'historique. Au moment de la réinitialisation du dictionnaire, le décodeur est initialisé pour ne pas créer de mot de code lorsque le tout premier ordinal est reçu.

Collection de chaînes:

Mot de code												
Position du dernier caractère												
Longueur de chaîne												

Historique du décodeur:

Position	0																		
Caractère	A																		

Etape 2:

Nouveaux caractères: B

Analyse par le codeur: dans la rangée de racines, il n'existe pas d'indice descendant pour "B", qui est donc transféré sous forme d'ordinal. L'entrée suivante disponible de l'arbre de nœuds (mot de code 5) est utilisée pour créer un segment de chaîne: celui-ci a comme position du premier caractère 2 (la position suivant "B"), comme longueur de segment 1 (c'est un segment à 1 caractère) et n'a ni indice descendant ni indice latéral valides. La rangée de racines est également mise à jour avec un indice descendant pour "B", pointant sur le mot de code 5.

Historique du codeur:

Position	0	1																	
Caractère	A	B																	

Rangée de racines:

Racine	A	B	C	D	E	..	X	Y	..	FF_H
Indice descendant	4	5	-	-	-	-	-	-	-	-

Arbre de nœuds:

Mot de code	4	5										
Position du premier caractère	1	2										
Longueur de segment	1	1										
Indice descendant	-	-										
Indice latéral	-	-										

Transféré: ordinal pour "B": 42_H = "1000010"

Analyse par le décodeur: le caractère "B" est placé dans l'historique. On met à jour la collection de chaînes en créant une chaîne se terminant par "B" (en position 1), de longueur 2 (représentant la chaîne "AB") et utilisant le premier mot de code disponible, 4.

Collection de chaînes:

Mot de code	4											
Position du dernier caractère	1											
Longueur de chaîne	2											

Historique du décodeur:

Position	0	1																	
Caractère	A	B																	

Etapes 3, 4, 5, 6:

nouveaux caractères: C D E X

Analyse par le codeur: pour ces caractères, le traitement est le même que pour "B': chacun est transféré sous forme d'ordinal et l'arbre de nœuds et la rangée de racines sont mis à jour en conséquence.

Nouvelles données: C D E X

Historique du codeur:

Position	0	1	2	3	4	5													
Caractère	A	B	C	D	E	X													

Rangée de racines:

Racine	A	B	C	D	E	..	X	Y	..	FF_H
Indice descendant	4	5	6	7	8	-	9	-	-	-

Arbre de nœuds:

Mot de code	4	5	6	7	8	9						
Position du premier caractère	1	2	3	4	5	6						
Longueur de segment	1	1	1	1	1	1						
Indice descendant	-	-	-	-	-							
Indice latéral	-	-	-	-	-							

Transférés: ordinaux pour "C", "D", "E", "X": 43_H 44_H 45_H 58_H = "1000011" "1000100" "1000101" "1011000"

Analyse par le décodeur: les caractères "C", "D", "E", "X" sont placés individuellement dans l'historique. Des chaînes à 2 caractères se terminant par C, D, E et X sont créées dans la collection de chaînes.

Collection de chaînes:

Mot de code	4	5	6	7	8							
Position du dernier caractère	1	2	3	4	5							
Longueur de chaîne	2	2	2	2	2							

Historique du décodeur:

Position	0	1	2	3	4	5													
Caractère	A	B	C	D	E	X													

Etape 7:

nouveaux caractères: A B

Analyse par le codeur: lorsque le "A" est introduit, le codeur suit l'indice descendant de la racine "A" pointant sur le mot de code 4 dans l'arbre de nœuds, correspondant au segment de chaîne "B", ce qui correspond au caractère suivant "B"; étant donné que le mot de code 4 n'a pas d'indice descendant valide, il s'agit de la plus longue chaîne correspondante. Ainsi, le mot de code 4 est transféré et le codeur lance la procédure d'extension de chaîne. Aucun nouveau segment de chaîne n'est créé à ce stade.

Historique du codeur:

Position	0	1	2	3	4	5	6	7											
Caractère	A	B	C	D	E	X	A	B											

Rangée de racines:

Racine	A	B	C	D	E	..	X	Y	..	FF_H
Indice descendant	4	5	6	7	8	-	9	-	-	-

Arbre de nœuds:

Mot de code	4	5	6	7	8	9						
Position du premier caractère	1	2	3	4	5	6						
Longueur de segment	1	1	1	1	1	1						
Indice descendant	-	-	-	-	-	-						
Indice latéral	-	-	-	-	-	-						

Transféré: mot de code 4: "000100"

Analyse par le décodeur: le décodeur accède au mot de code 4 dans la collection de chaînes. Il copie la chaîne "AB" de la position 0 de l'historique aux positions suivantes disponibles de l'historique. Il crée aussi la chaîne à 2 caractères se terminant par le premier caractère de la chaîne représentée par le mot de code 4 (à savoir le "A"), en utilisant le mot de code 9.

Collection de chaînes:

Mot de code	4	5	6	7	8	9						
Position du dernier caractère	1	2	3	4	5	6						
Longueur de chaîne	2	2	2	2	2	2						

Historique du décodeur:

Position	0	1	2	3	4	5	6	7											
Caractère	A	B	C	D	E	X	A	B											

Etape 8a:

nouveaux caractères: C

Analyse par le codeur: la procédure d'extension de chaîne détermine que le nouveau caractère "C" correspond au caractère "C" de l'historique qui suit immédiatement le segment de chaîne "B" du mot de code 4. Comme la longueur totale de la chaîne pour le mot de code 4 et l'extension à 1 caractère est inférieure à la longueur maximale de chaîne, la procédure d'extension de chaîne se poursuit avec le caractère suivant.

Historique du codeur:

Position	0	1	2	3	4	5	6	7	8										
Caractère	A	B	C	D	E	X	A	B	C										

Rangée de racines:

Racine	A	B	C	D	E	..	X	Y	..	FF_H
Indice descendant	4	5	6	7	8	-	9	-	-	-

Arbre de nœuds:

Mot de code	4	5	6	7	8	9						
Position du premier caractère	1	2	3	4	5	6						
Longueur de segment	1	1	1	1	1	1						
Indice descendant	-	-	-	-	-	-						
Indice latéral	-	-	-	-	-	-						

Transféré: rien n'est transféré dans cette sous-étape.

Analyse par le décodeur: rien n'est reçu dans cette sous-étape.

Collection de chaînes:

Mot de code	4	5	6	7	8	9						
Position du dernier caractère	1	2	3	4	5	6						
Longueur de chaîne	2	2	2	2	2	2						

Historique du décodeur:

Position	0	1	2	3	4	5	6	7												
Caractère	A	B	C	D	E	X	A	B												

Etape 8b:

nouveaux caractères: D E Y

Analyse par le codeur: dans la procédure d'extension de chaîne, le codeur détermine que, pour les caractères suivants, la correspondance est maintenue avec l'historique jusqu'au "Y". Une longueur d'extension de chaîne de 3 est transférée, qui représente le "CDE" suivant le "B" d'origine du mot de code 4. Le codeur crée un nouveau segment de chaîne, pour étendre la chaîne correspondante la plus longue, avec une longueur de 3 et une position dans l'historique de 8, la position du premier caractère de l'extension; il utilise le mot de code suivant disponible, 10. Cela termine la procédure d'extension de chaîne: le codeur N'adjoint PAS en plus un segment de chaîne à 1 caractère.

Le caractère sans correspondant "Y" devient la racine d'une nouvelle chaîne possible. Toutefois, comme il n'a pas d'indice descendant valide, il est transféré sous forme d'ordinal; le mot de code 11 est créé pour adjoindre le segment de chaîne à 1 caractère pour le caractère suivant le "Y" et l'indice descendant pour l'entrée racine de "Y" est mis à jour avec le mot de code 11.

Historique du codeur:

Position	0	1	2	3	4	5	6	7	8	9	10	11								
Caractère	A	B	C	D	E	X	A	B	C	D	E	Y								

Rangée de racines:

Racine	A	B	C	D	E	..	X	Y	..	FF_H
Indice descendant	4	5	6	7	8	-	9	11	-	-

Arbre de nœuds:

Mot de code	4	5	6	7	8	9	10	11				
Position du premier caractère	1	2	3	4	5	6	8	12				
Longueur de segment	1	1	1	1	1	1	3	1				
Indice descendant	10	-	-	-	-	-	-	-				
Indice latéral	-	-	-	-	-	-	-	-				

Transférés: longueur d'extension de chaîne 3: "0" "10"; ordinal pour "Y": 59_H = "1011001"

Analyse par le décodeur: le décodeur étend la chaîne du mot de code précédent (mot de code 4) par 3: il copie les 3 caractères de l'historique qui suivent immédiatement la chaîne représentée par le mot de code 4 (en utilisant la position du dernier caractère plus 1 comme point de départ) aux positions suivantes disponibles de l'historique. Il crée ensuite la chaîne se terminant par "E" et lui assigne le mot de code 10: sa longueur est 5 ((la longueur de la chaîne pour le mot de code 4) + (la longueur de l'extension de chaîne)) = (2 + 3) = 5) et la position du dernier caractère est mise à 10, la position du dernier caractère étant copiée dans l'historique.

Le caractère Y est également placé dans l'historique. AUCUNE chaîne à 2 caractères N'est créée.

Collection de chaînes:

Mot de code	4	5	6	7	8	9	10					
Position du dernier caractère	1	2	3	4	5	6	10					
Longueur de chaîne	2	2	2	2	2	2	5					

Historique du décodeur:

Position	0	1	2	3	4	5	6	7	8	9	10	11								
Caractère	A	B	C	D	E	X	A	B	C	D	E	Y								

Etape 9:

nouveaux caractères: A B C D E

Analyse par le codeur: lorsque le "A" est introduit, le codeur suit l'indice descendant de la racine "A" pointant sur le mot de code 4 dans l'arbre de nœuds, correspondant au segment de chaîne "B". Cela correspond au caractère suivant, le codeur continue donc la procédure de mise en correspondance de chaîne: Il suit le pointeur descendant vers le mot de code 10, correspondant au segment de chaîne "CDE". Les caractères suivants correspondent en totalité à ce segment de chaîne. Comme le mot de code 10 n'a pas d'indice descendant valide, il s'agit de la plus longue chaîne correspondante. Le mot de code 10 est donc transféré et le codeur lance la procédure d'extension de chaîne.

Historique du codeur:

Position	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16			
Caractère	A	B	C	D	E	X	A	B	C	D	E	Y	A	B	C	D	E			

Rangée de racines:

Racine	A	B	C	D	E	..	X	Y	..	FF_H
Indice descendant	4	5	6	7	8	-	9	11	-	-

Arbre de nœuds:

Mot de code	4	5	6	7	8	9	10	11				
Position du premier caractère	1	2	3	4	5	6	8	12				
Longueur de segment	1	1	1	1	1	1	3	1				
Indice descendant	10	-	-	-	-	-	-	-				
Indice latéral	-	-	-	-	-	-	-	-				

Transféré: mot de code 10: "001010"

Analyse par le décodeur: le décodeur accède au mot de code 10 dans la collection de chaînes. Il copie cette chaîne de 5 caractères aux positions suivantes disponibles de l'historique. Il adjoint aussi "A" à "Y" et crée une chaîne à 2 caractères, en utilisant le mot de code 11.

Collection de chaînes:

Mot de code	4	5	6	7	8	9	10	11				
Position du dernier caractère	1	2	3	4	5	6	10	12				
Longueur de chaîne	2	2	2	2	2	2	5	2				

Historique du décodeur:

Position	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16			
Caractère	A	B	C	D	E	X	A	B	C	D	E	Y	A	B	C	D	E			

Etape 10:

nouveaux caractères: FF_H

Analyse par le codeur: dans la procédure d'extension de chaîne, le codeur détermine que le caractère suivant "FF_H" ne correspond pas à l'historique et la procédure d'extension de chaîne est terminée. Un nouveau segment de chaîne de longueur 1 (le "FF_H") est adjoint à la fin de "ABCDE" et le mot de code 12 lui est assigné. L'indice descendant pour le mot de code 10 est mis au mot de code 12.

On utilise ensuite le "FF_H" comme racine pour une éventuelle nouvelle chaîne correspondante: comme il n'a pas d'indice descendant valide, il est transféré sous forme d'ordinal. Un segment de chaîne à 1 caractère est créé, avec le mot de code 13, pour adjoindre le nouveau caractère au "FF_H" et l'indice descendant de l'entrée racine de "FF_H" est mis au mot de code 13.

(Comme $FF_H = 255_{10} > 127_{10}$, l'ordinal nécessite 8 bits pour le transfert; C_5 est mis à 8 et le code de commande STEPUP est transféré juste avant l'ordinal.)

Historique du codeur:

Position	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17		
Caractère	A	B	C	D	E	X	A	B	C	D	E	Y	A	B	C	D	E	FF_H		

Rangée de racines:

Racine	A	B	C	D	E	..	X	Y	..	FF_H
Indice descendant	4	5	6	7	8	-	9	11	-	13

Arbre de nœuds:

Mot de code	4	5	6	7	8	9	10	11	12	13		
Position du premier caractère	1	2	3	4	5	6	8	12	17	18		
Longueur de segment	1	1	1	1	1	1	3	1	1	1		
Indice descendant	10	-	-	-	-	-	12	-	-	-		
Indice latéral	-	-	-	-	-	-	-	-	-	-		

Transférés: STEPUP ("000010"), ordinal pour "FF_H": FF_H = "11111111"

Analyse par le décodeur: à la réception du code de commande STEPUP, le décodeur met C_5 à 8. Le caractère "FF_H" est placé dans l'historique. Le décodeur assigne le mot de code 12 à la chaîne de longueur 6 se terminant par "FF_H".

Collection de chaînes:

Mot de code	4	5	6	7	8	9	10	11	12			
Position du dernier caractère	1	2	3	4	5	6	10	12	17			
Longueur de chaîne	2	2	2	2	2	2	5	2	6			

Historique du décodeur:

Position	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17		
Caractère	A	B	C	D	E	X	A	B	C	D	E	Y	A	B	C	D	E	FF_H		

Etape 11:

nouveaux caractères: A C et *demande C-FLUSH*

Analyse par le codeur: lorsque le "A" est introduit, le codeur suit l'indice descendant de la racine "A" pointant sur le mot de code 4 dans l'arbre de nœuds, correspondant au segment de chaîne "B". Cela NE correspond PAS au caractère suivant "C'; comme il n'y a pas d'indice latéral pour une autre option pour une correspondance avec le "C", le "A" est transféré sous forme d'ordinal. Le codeur crée un segment de chaîne à 1 caractère (mot de code 14) pour adjoindre le "C" au "A"; le mot de code 14 est établi comme l'indice latéral pour le mot de code 4 (l'ancien segment de chaîne "AB").

Le "C" devient la racine pour une éventuelle nouvelle chaîne correspondante: toutefois, comme le codeur reçoit une commande FLUSH provenant de la fonction de commande, il doit terminer toute mise en correspondance de chaîne en cours. Dans ce cas, il transfère le "C" sous forme d'ordinal, suivi du code de commande FLUSH. Comme d'habitude, le codeur crée un segment de chaîne à 1 caractère pour adjoindre le caractère suivant au "C", le mot de code 15 lui étant assigné.

Historique du codeur:

Position	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Caractère	A	B	C	D	E	X	A	B	C	D	E	Y	A	B	C	D	E	FF_H	A	C

Rangée de racines:

Racine	A	B	C	D	E	..	X	Y	..	FF_H
Indice descendant	4	5	6	7	8	-	9	11	-	13

Arbre de nœuds:

Mot de code	4	5	6	7	8	9	10	11	12	13	14	15
Position du premier caractère	1	2	3	4	5	6	8	12	17	18	19	20
Longueur de segment	1	1	1	1	1	1	3	1	1	1	1	1
Indice descendant	10	-	-	-	-	-	12	-	-	-	-	
Indice latéral	14	-	-	-	-	-	-	-	-	-	-	

Transférés: ordinaux pour "A" "C", code de commande FLUSH, remplissage de bits 0: "01000001" "01000011" "000001" *0000*

Analyse par le décodeur: le caractère "A" est placé dans l'historique et la chaîne à 2 caractères se terminant par "A" est créée et le mot de code 13 lui est assigné. Le caractère "C" est placé dans l'historique et la chaîne à 2 caractères se terminant par "C" est créée et le mot de code 14 lui est assigné. La réception du code de commande FLUSH indique la fin des données codées pour le moment. Le bit significatif suivant sera à la première position binaire de l'octet suivant. Le décodeur ne crée pas de mot de code 15 tant que le code après le FLUSH n'est pas reçu.

Collection de chaînes:

Mot de code	4	5	6	7	8	9	10	11	12	13	14	
Position du dernier caractère	1	2	3	4	5	6	10	12	17	18	19	
Longueur de chaîne	2	2	2	2	2	2	5	2	6	2	2	

Historique du décodeur:

Position	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Caractère	A	B	C	D	E	X	A	B	C	D	E	Y	A	B	C	D	E	FF_H	A	C

A ce stade, les dictionnaires du codeur et du décodeur sont synchronisés, sauf pour le mot de code 15.

La Figure II.1 montre l'arbre de nœuds après l'étape 11.

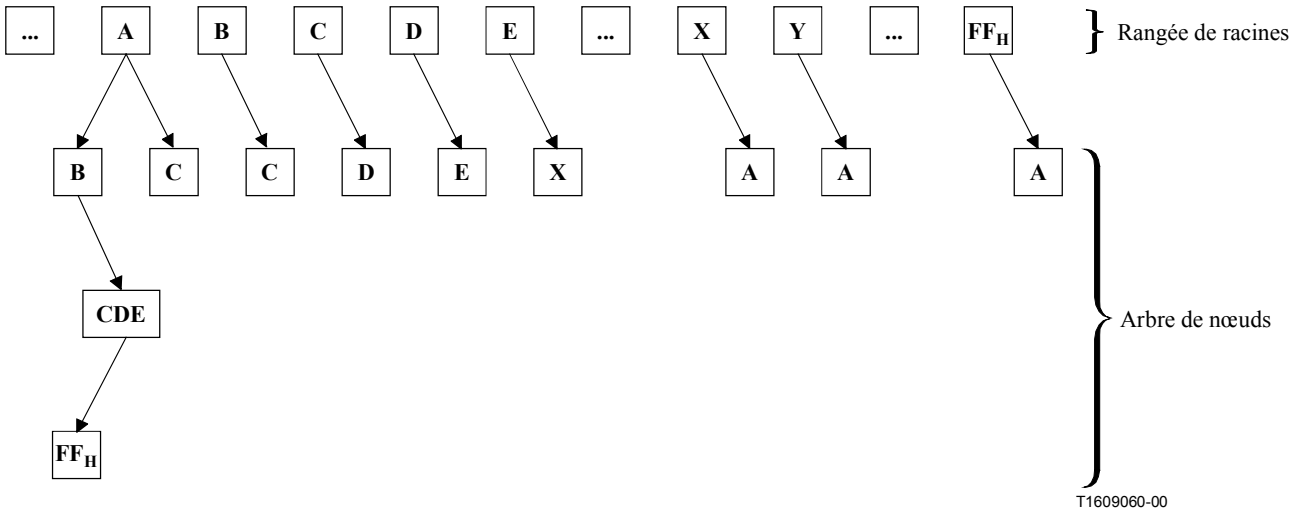

Figure II.1/V.44 – Structures arborescentes après l'étape 11

Le Tableau II.1 montre le mappage des octets pour l'exemple du II.1. On se reportera aux Tableaux 3 et 5 pour le codage du préfixe de code et de la longueur d'extension de chaîne.

Tableau II.1/V.44 – Mappage d'octets pour l'exemple du II.1

Bit n°	8	7	6	5	4	3	2	1	Octet
	1	0	0	0	0	0	1	***0***	1
	1	0	0	0	0	1	0	***0***	2
	1	0	0	0	0	1	1	***0***	3
	1	0	0	0	1	0	0	***0***	4
	1	0	0	0	1	0	1	***0***	5
	1	0	1	1	0	0	0	***0***	6
	0	0	0	0	1	0	0	***1***	7
	0	0	1	***0***	1	0	0	***1***	8
	0	1	0	***1***	1	0	1	1	9
	0	0	1	0	***1***	0	0	1	10
	1	1	1	1	1	***0***	0	0	11
	0	0	0	1	***0***	1	1	1	12
	0	1	1	***0***	0	1	0	0	13
	0	1	***1***	0	1	0	0	0	14
	0	0	0	0	0	0	0	0	15
	Les bits de préfixe de code sont ***en gras et en italique***, les bits de remplissage sont soulignés.								

II.2 Compression et décompression de "CCCCCCCCCCX"

L'exemple donné dans le présent paragraphe vise à illustrer un aspect particulier du fonctionnement de l'algorithme V.44, qui lui permet de compresser des séquences de caractères répétés très rapidement. Cet aspect nécessite que le décodeur interprète un mot de code qu'il n'a pas encore créé.

Dans le fonctionnement général de l'algorithme, des mots de code sont créés en continu aux deux extrémités, même si dans le codeur, ils correspondent spécifiquement à des segments de chaîne et que, dans le décodeur, ils correspondent à la chaîne complète qui se termine par le segment de chaîne correspondant du codeur (dans le contexte de la structure arborescente). Comme illustré dans l'exemple du II.1, le codeur peut avoir une longueur d'avance sur le décodeur en ce qui concerne la création de mots de code. Dans certains cas particuliers, le codeur transférera un mot de code qui est, pour le décodeur, exactement le mot de code suivant disponible (mot de code égal à C_1).

Le décodeur traite cette situation conformément aux règles du 6.4.1, en particulier 3) et 4).

Pour illustrer le traitement de l'exemple, on montre, à différentes étapes:

i) les nouveaux caractères introduits dans le codeur;

ii) l'analyse du flux d'entrée par l'algorithme, pour le codage;

iii) la structure de dictionnaire révisée du codeur;

iv) les informations transférées par le codeur et la méthode par laquelle elles sont codées;

v) l'analyse des données comprimées par l'algorithme, pour le décodage;

vi) la structure de dictionnaire révisée du décodeur.

Dans cet exemple, on suppose que la structure de dictionnaire vient juste d'être réinitialisée, de sorte qu'il n'y a pas d'historique précédent. Les caractères sont introduits un par un, mais on présente ici des étapes, en sautant certaines étapes lorsque le traitement est évident.

Les caractères non comprimés sont placés dans l'historique du codeur à mesure qu'ils arrivent; le traitement par le codeur produit des codes binaires, qui sont transférés; après la réception des données comprimées par le décodeur, les caractères décodés sont placés dans l'historique du décodeur.

Etape 0: pas de donnée précédente.

L'état du codeur est le suivant:

historique du codeur:

Position																				
Caractère																				

Rangée de racines:

Racine	A	B	C	D	E	..	X	Y	Z	..
Indice descendant	-	-	-	-	-	-	-	-	-	-

Arbre de nœuds:

Mot de code												
Position du premier caractère												
Longueur de segment												
Indice descendant												
Indice latéral												

L'état du décodeur est le suivant:

collection de chaînes:

Mot de code												
Position du dernier caractère												
Longueur de chaîne												

Historique du décodeur:

Position																			
Caractère																			

Etape 1: nouveaux caractères: C

Analyse par le codeur: dans la rangée de racines, il n'y a pas d'indice descendant pour "C", qui est donc transféré sous forme d'ordinal. La première entrée disponible de l'arbre de nœuds (mot de code 4) est utilisée pour créer un segment de chaîne: celui-ci a comme position du premier caractère 1 (la position suivant "C"), comme longueur de segment 1 (c'est un segment de chaîne à 1 caractère) et n'a ni indice descendant ni indice latéral valides: il s'agit essentiellement de l'adjonction du caractère suivant, sans examen, pour créer une chaîne à 2 caractères. La rangée de racines est aussi mise à jour avec un indice descendant pour "C", pointant sur le mot de code 4.

Historique du codeur:

Position	0																		
Caractère	C																		

Rangée de racines:

Racine	A	B	C	D	E	..	X	Y	Z	..
Indice descendant	-	-	4	-	-	-	-	-	-	-

Arbre de nœuds:

Mot de code	4											
Position du premier caractère	1											
Longueur de segment	1											
Indice descendant	-											
Indice latéral	-											

Transféré: ordinal "C"

Analyse par le décodeur: le caractère "C" est placé dans l'historique. Au moment de la réinitialisation de dictionnaire, le décodeur est initialisé pour ne pas créer de mot de code lorsque le tout premier caractère est reçu.

Collection de chaînes:

Mot de code												
Position du dernier caractère												
Longueur de chaîne												

Historique du décodeur:

Position	0																			
Caractère	C																			

Etape 2: nouveaux caractères: C C

Analyse par le codeur: lorsque le premier nouveau C est reçu, le codeur suit l'indice descendant de la racine "C" pointant sur le mot de code 4 de l'arbre de nœuds, correspondant au segment de chaîne "C" (à savoir le C en position 1). Cela correspond au deuxième nouveau C; et comme le mot de code 4 n'a pas d'indice descendant valide, il s'agit de la plus longue chaîne correspondante. Le mot de code 4 est donc transféré et le codeur lance la procédure d'extension de chaîne. Aucun nouveau segment de chaîne n'est créé à ce stade.

Historique du codeur:

Position	0	1	2																	
Caractère	C	C	C																	

Rangée de racines:

Racine	A	B	C	D	E	..	X	Y	Z	..
Indice descendant	-	-	4	-	-	-	-	-	-	-

Arbre de nœuds:

Mot de code	4											
Position du premier caractère	1											
Longueur de segment	1											
Indice descendant	-											
Indice latéral	-											

Transféré: mot de code 4

Analyse par le décodeur: conformément à la règle 4) du 6.4.1: comme le mot de code 4 est le mot de code suivant à créer ($C_1 = 4$), le décodeur copie la précédente chaîne à 1 caractère (le "C" de la position 0) dans l'historique (en position 1) puis place le premier caractère de la chaîne copiée (le "C" de la position 1) dans l'historique (en position 2).

Conformément au Tableau 2: le premier caractère de la chaîne du mot de code 4 ("CC') est adjoint au caractère précédent (le C en position 0) pour créer le mot de code 4: c'est une chaîne à 2 caractères, le dernier caractère étant en position 1.

Collection de chaînes:

Mot de code	4											
Position du dernier caractère	1											
Longueur de chaîne	2											

Historique du décodeur:

Position	0	1	2																
Caractère	C	C	C																

Etape 3: nouveaux caractères: C

Analyse par le codeur: la procédure d'extension de chaîne détermine que le nouveau caractère C correspond au caractère qui suit le segment de chaîne (pour le mot de code 4) dans l'historique (le C en position 2). La procédure d'extension de chaîne se poursuit donc (étant donné que la longueur totale de la chaîne associée au mot de code 4 plus cette extension à 1 caractère est inférieure à la longueur maximale de chaîne).

Historique du codeur:

Position	0	1	2	3															
Caractère	C	C	C	C															

Rangée de racines:

Racine	A	B	C	D	E	..	X	Y	Z	..
Indice descendant	-	-	4	-	-	-	-	-	-	-

Arbre de nœuds:

Mot de code	4											
Position du premier caractère	1											
Longueur de segment	1											
Indice descendant	-											
Indice latéral	-											

Transféré: rien n'est transféré dans cette sous-étape.

Analyse par le décodeur: rien n'est reçu dans cette sous-étape.

Collection de chaînes:

Mot de code	4											
Position du dernier caractère	1											
Longueur de chaîne	2											

Historique du décodeur:

Position	0	1	2																
Caractère	C	C	C																

Etape 4: nouveaux caractères: C C C C C C

Analyse par le codeur: dans la procédure d'extension de chaîne, chaque nouveau caractère est successivement comparé aux caractères de l'historique: dans ce cas, comme l'extension commence avec le caractère placé en position 3 (qui a été comparé au caractère en position 2), le premier nouveau caractère est celui situé en position 4 (à comparer avec celui situé en position 3) et ainsi de suite. Comme les nouveaux caractères sont constitués de C consécutifs, chacun correspondant à celui qui le précède, l'extension de chaîne se poursuit jusqu'au 6^{e} nouveau C. La longueur totale de chaîne

[(longueur de la chaîne du mot de code 4) + (nombre de caractères d'extension jusque là) = 2 + 7 = 9] est toujours inférieure à la longueur maximale, de sorte que la procédure d'extension de chaîne se poursuit.

Historique du codeur:

Position	0	1	2	3	4	5	6	7	8	9										
Caractère	C	C	C	C	C	C	C	C	C	C										

Rangée de racines:

Racine	A	B	C	D	E	..	X	Y	Z	..
Indice descendant	-	-	4	-	-	-	-	-	-	-

Arbre de nœuds:

Mot de code	4											
Position du premier caractère	1											
Longueur de segment	1											
Indice descendant	-											
Indice latéral	-											

Transféré: rien n'est transféré dans cette sous-étape.

Analyse par le décodeur: rien n'est reçu dans cette sous-étape.

Collection de chaînes:

Mot de code	4											
Position du dernier caractère	1											
Longueur de chaîne	2											

Historique du décodeur:

Position	0	1	2																	
Caractère	C	C	C																	

Etape 5: nouveaux caractères: X

Analyse par le codeur: le X met fin à la procédure d'extension de chaîne: le codeur détermine qu'il ne correspond pas au caractère en position 9 de l'historique (le 10e C). Il crée alors un nouveau segment de chaîne (mot de code 5) pour étendre la plus longue chaîne correspondante avec une longueur d'extension de 7 (représentant le "CCCCCCC" suivant la plus longue chaîne correspondante, mot de code 4), la position du premier caractère étant 3. Cela met fin à la procédure d'extension de chaîne et le codeur n'adjoint pas de segment de chaîne à 1 caractère.

Le caractère sans correspondant "X" devient la racine d'une nouvelle chaîne possible. Toutefois, comme il n'a pas d'indice descendant valide, il est transféré sous forme d'ordinal; le mot de code 6 est créé pour adjoindre le segment de chaîne à 1 caractère pour le caractère suivant le "X" et l'indice descendant pour l'entrée racine de "X" est mis à jour avec le mot de code 6.

Historique du codeur:

Position	0	1	2	3	4	5	6	7	8	9	10									
Caractère	C	C	C	C	C	C	C	C	C	C	X									

Rangée de racines:

Racine	A	B	C	D	E	..	X	Y	Z	..
Indice descendant	-	-	4	-	-	-	6	-	-	-

Arbre de nœuds:

Mot de code	4	5	6									
Position du premier caractère	1	3	11									
Longueur de segment	1	7	1									
Indice descendant	5											
Indice latéral	-											

Transférés: longueur d'extension de chaîne 7, ordinal pour "X'

Analyse par le décodeur: le décodeur étend la chaîne du précédent mot de code par 7: il copie les 7 caractères de l'historique qui suivent immédiatement la chaîne représentée par le mot de code 4, en utilisant la position du dernier caractère (position 1) plus 1 comme point de départ. Pour cela, on procède caractère par caractère: le point de départ pour la copie est la position 2 (à copier dans la première position disponible, qui est la position 3); le caractère suivant à copier est celui situé en position 3 (à copier en position 4) et ainsi de suite. Le décodeur crée alors la chaîne se terminant par le 10e C, en utilisant le mot de code 5: sa longueur est 9 [(longueur de chaîne pour le mot de code 4) + (longueur de l'extension de chaîne) = (2 + 7) = 9]; et la position du dernier caractère de la chaîne est la position du dernier caractère copié dans l'historique, 9.

Le caractère "X" est ensuite placé dans l'historique en position 10. Aucune chaîne à 2 caractères n'est créée.

Collection de chaînes:

Mot de code	4	5										
Position du dernier caractère	1	9										
Longueur de chaîne	2	9										

Historique du décodeur:

Position	0	1	2	3	4	5	6	7	8	9	10									
Caractère	C	C	C	C	C	C	C	C	C	C	X									

Bibliographie

– UIT-T V.14 (1993), *Transmission de caractères arythmiques sur des voies supports synchrones*.

– UIT-T V.76 (1996), *Multiplexeur générique utilisant les procédures basées LAPM de la Recommandation V.42*.

– ISO/CEI 10646-1:2000: *Technologies de l'information – Jeu universel de caractères codés à plusieurs octets – Partie 1: Architecture et table multilingue*.

– Unicode Technical Report N°. 6, *A Standard Compression Scheme for Unicode, Revision 3.1*.

SÉRIES DES RECOMMANDATIONS UIT-T

Série A	Organisation du travail de l'UIT-T
Série B	Moyens d'expression: définitions, symboles, classification
Série C	Statistiques générales des télécommunications
Série D	Principes généraux de tarification
Série E	Exploitation générale du réseau, service téléphonique, exploitation des services et facteurs humains
Série F	Services de télécommunication non téléphoniques
Série G	Systèmes et supports de transmission, systèmes et réseaux numériques
Série H	Systèmes audiovisuels et multimédias
Série I	Réseau numérique à intégration de services
Série J	Transmission des signaux radiophoniques, télévisuels et autres signaux multimédias
Série K	Protection contre les perturbations
Série L	Construction, installation et protection des câbles et autres éléments des installations extérieures
Série M	RGT et maintenance des réseaux: systèmes de transmission, de télégraphie, de télécopie, circuits téléphoniques et circuits loués internationaux
Série N	Maintenance: circuits internationaux de transmission radiophonique et télévisuelle
Série O	Spécifications des appareils de mesure
Série P	Qualité de transmission téléphonique, installations téléphoniques et réseaux locaux
Série Q	Commutation et signalisation
Série R	Transmission télégraphique
Série S	Equipements terminaux de télégraphie
Série T	Terminaux des services télématiques
Série U	Commutation télégraphique
Série V	**Communications de données sur le réseau téléphonique**
Série X	Réseaux de données et communication entre systèmes ouverts
Série Y	Infrastructure mondiale de l'information et protocole Internet
Série Z	Langages et aspects informatiques généraux des systèmes de télécommunication

Genève, 2001