Groupe de travail Réseau
Request for Comments : 3453
Catégorie : Information
Traduction Claude Brière de L'Isle

M. Luby, Digital Fountain
L. Vicisano, Cisco
J. Gemmell, Microsoft
L. Rizzo, Univ. Pisa
M. Handley, ICIR
J. Crowcroft, Cambridge Univ.
décembre 2002

Utilisation de la correction d'erreur directe (FEC) dans la diffusion groupée fiable

Résumé
Le présent mémoire décrit l'utilisation des codes de correction d'erreur directe (FEC, *Forward Error Correction*) pour fournir efficacement et/ou augmenter la fiabilité du transport de données fiable de un à plusieurs en utilisant la diffusion groupée IP. Une des propriétés clés des codes de FEC dans ce contexte est la capacité d'utiliser les mêmes paquets contenant les données de FEC pour réparer simultanément différents schémas de perte de paquet chez plusieurs receveurs. Différentes classes de codes de FEC et certaines de leurs propriétés de base sont décrites et la terminologie pertinente pour mettre en œuvre la FEC dans un protocole de diffusion groupée fiable est introduite. Des exemples des formats abstraits possibles pour les paquets portant la FEC sont fournis.

Table des Matières

1. Raisons et vue d'ensemble

Il y a de nombreuses façons de fournir la fiabilité pour les protocoles de transmission. Une méthode courante est d'utiliser la demande automatique de retransmission (ARQ, *Automatic Request for Retransmission*). Avec l'ARQ, les receveurs utilisent un canal de retour vers l'envoyeur pour envoyer les demandes de retransmission des paquets perdus. L'ARQ fonctionne bien pour les protocoles fiables de un à un, comme le montre le succès universel de TCP/IP. L'ARQ a aussi été un outil de fiabilité efficace pour les protocoles de fiabilité de un à plusieurs, et en particulier pour certains protocoles de diffusion groupée IP fiable. Cependant, pour les protocoles de fiabilité de un à beaucoup, l'ARQ a des limitations, incluant le problème de l'explosion de rétroactions parce que de nombreux receveurs retransmettent en retour à l'envoyeur, et à cause du besoin d'un canal de retour pour envoyer ces demandes du receveur. Une autre limitation est que les receveurs peuvent rencontrer différents schémas de perte de paquets, et donc les receveurs peuvent être retardés par la retransmission

de paquets que d'autres receveurs ont perdu mais qu'ils ont déjà reçu. Cela peut aussi causer un gaspillage de la bande passante utilisée pour retransmettre des paquets qui ont déjà été reçus par beaucoup des receveurs.

Dans les environnements où l'ARQ est trop coûteux ou impossible parce qu'il y a une capacité très limitée de canal de retour ou pas de canal de retour du tout, comme dans la transmission par satellite, une approche de la fiabilité par un carrousel de données est parfois utilisée [1]. Avec un carrousel de données, l'envoyeur partage l'objet en parties de données d'égale longueur, qu'on va appeler des symboles de source, les place dans des paquets, et ensuite circule continuellement à travers ces paquets et les envoie. Les receveurs reçoivent des paquets en continu jusqu'à ce qu'ils aient reçu une copie de chaque paquet. Le carrousel de données a l'avantage de ne pas exiger de canal de retour parce qu'il n'y a pas de données qui s'écoulent des receveurs à l'envoyeur. Cependant, le carrousel de données a aussi des limitations. Par exemple, si un receveur perd un paquet dans un tour de transmission, il doit attendre un tour entier avant d'avoir une chance de recevoir de nouveau ce paquet. Cela peut aussi causer un gaspillage de bande passante, car l'envoyeur circule continuellement parmi les paquets et les transmet jusqu'à ce qu'aucun paquet ne manque à un receveur.

Les codes de correction d'erreur directe (FEC, *Forward Error Correction*) fournissent une méthode de fiabilité qui peut être utilisée pour augmenter ou remplacer d'autres méthodes de fiabilité, en particulier pour des protocoles de fiabilité de un à beaucoup comme la diffusion groupée IP fiable. On passe rapidement en revue certaines des propriétés de base et les types de codes de FEC avant de voir leur utilisation dans le contexte de la diffusion groupée IP fiable. On fournira ensuite une description plus détaillée de certains des codes de FEC.

Dans la littérature générale, la FEC se réfère à la capacité de surmonter à la fois les écrasements (pertes) et la corruption au niveau du bit. Cependant, dans le cas d'un protocole de diffusion groupée IP, les couches de réseau vont détecter les paquets corrompus et les éliminer, ou les couches de transport peuvent utiliser l'authentification de paquet pour éliminer les paquets corrompus. Donc la principale application des codes de FEC aux protocoles de diffusion groupée IP est comme code d'écrasement. Les charges utiles sont générées et traitées en utilisant un codeur d'écrasement de FEC et les objets sont réassemblés à partir de la réception des paquets contenant le codage généré en utilisant le décodeur d'écrasement de FEC correspondant.

L'entrée à un codeur de FEC est un nombre k de symboles de source de longueur égale. Le codeur de FEC génère un nombre de symboles de codage qui sont de la même longueur que les symboles de source. La longueur choisie pour les symboles peut varier à chaque application du codeur de FEC, ou elle peut être fixe. Ces symboles de codage sont placés dans des paquets pour la transmission. Le nombre de symboles de codage placés dans chaque paquet peut varier paquet par paquet, ou un nombre fixe de symboles (souvent un) peut être placé dans chaque paquet. Aussi, dans chaque paquet est placé assez d'information pour identifier les symboles de codage particuliers portés dans ce paquet. À réception des paquets contenant les symboles de codage, le receveur fournit ces symboles de codage au décodeur de FEC correspondant pour recréer une copie exacte des k symboles de source. Idéalement, le décodeur de FEC peut recréer une copie exacte à partir de tous les k symboles de codage.

On décrit plus loin une technique pour utiliser les codes de FEC comme décrit ci-dessus à traiter les blocs avec des symboles de source de longueur variable.

Le bloc de codes de FEC fonctionne comme suit. L'entrée à un bloc codeur de FEC est k symboles de source et un nombre n. Le codeur génère un total de n symboles de codage. Le codeur est systématique si il génère n-k symboles redondants donnant un bloc de codage de n symboles de codage au total composé des k symboles de source et des n-k symboles redondants.

Un bloc décodeur de FEC a la propriété que tout k des n symboles de codage dans le bloc de codage est suffisant pour reconstruire les k symboles de source originaux.

Les codes de FEC expansibles fonctionnent comme suit. Un codeur de FEC expansible prend en entrée k symboles de source et génère autant de symboles de codage uniques que demandé, où le temps pour générer chaque symbole de codage est le même indépendamment du nombre de symboles de codage générés. Un décodeur de FEC expansible a la propriété que tout k des symboles de codage uniques est suffisant pour reconstruire les k symboles de source originaux.

Les définitions ci-dessus expliquent la situation idéale quand la réception d'un des k symboles de codage est suffisante pour recupérer les k symboles de source, et dans ce cas les frais généraux de réception sont de 0 %. Pour certains codes de FEC, légèrement plus de k symboles de codage sont nécessaires pour récupérer les k symboles de source. Si k*(1+ep) symboles de codage sont nécessaires, on dit que les frais généraux de réception sont ep*100 %, par exemple, si k*1,05 symboles de codage sont nécessaires, alors les frais généraux de réception sont de 5 %.

Dans une dimension différente, on classe les codes de FEC comme étant petits ou grands. Un petit code de FEC est efficace

en termes d'exigences de temps de traitement pour le codage et décodage de petites valeurs de k, et un grand code de FEC est efficace pour coder et décoder des valeurs beaucoup plus grandes de k. Il y a des mises en œuvre de blocs de codes de FEC qui ont des temps de codage proportionnels à n-k fois la longueur des k symboles de source, et des temps de décodage proportionnels à l fois la longueur des k symboles de source, où l est le nombre de symboles de source manquants parmi les k symboles de codage reçus et l peut être aussi grand que k. À cause du taux de croissance des temps de codage et de décodage comme produit de k et n-k, ils sont normalement considérés comme étant des petits blocs de codes de FEC. Il y a des blocs de codes de FEC avec de faibles frais généraux de réception qui peuvent générer n symboles de codage et peuvent décoder les k symboles de source dans un temps proportionnel à la longueur des n symboles de codage. Ces codes sont considérés comme étant des grands blocs de codes de FEC. Il y a des codes de FEC expansibles avec de faibles frais généraux de réception qui peuvent générer chaque symbole de codage dans un temps en gros proportionnel à sa longueur, et peuvent décoder tous les k symboles de source dans un temps en gros proportionnel à leur longueur. Ils sont considérés comme étant de grands codes de FEC expansibles. On décrit plus loin des exemples de tous ces types de codes.

Idéalement, les codes de FEC dans le contexte de la diffusion groupée IP peuvent être utilisés pour générer des symboles de codage qui sont transmis en paquets de telle façon que chaque paquet reçu soit pleinement utile à un receveur pour réassembler l'objet sans considération des schémas de réception des paquets précédents. Donc, si certains paquets sont perdus dans le transit entre l'envoyeur et le receveur, au lieu de demander une retransmission spécifique des paquets perdus ou d'attendre que les paquets soient envoyés à nouveau en utilisant le carrousel de données, le receveur peut utiliser tout autre nombre égal de paquets suivant qui arrivent pour réassembler l'objet. Ces paquets peuvent être proactivement transmis ou peuvent être explicitement demandés par les receveurs. Cela implique que le même paquet est pleinement utile à tous les receveurs pour réassembler l'objet, même quand les receveurs peuvent avoir rencontré précédemment des schémas de perte de paquet différents. Cette propriéte peut réduire ou même éliminer les problèmes mentionnés ci-dessus associés à l'ARQ et au carrousel de données et augmente considérablement l'adaptabilité du protocole aux ordres de grandeur du nombre de receveurs.

1.1. Application des codes de FEC

Pour certains protocoles de diffusion groupée IP fiable, les codes de FEC sont utilisés en conjonction avec l'ARQ pour fournir la fiabilité. Par exemple, un grand objet pourrait être partitionné en un certain nombre de blocs de source consistant chacun en un petit nombre de symboles de source, et dans un premier tour de transmission, tous les symboles de source pour tous les blocs de source pourraient être transmis. Ensuite, les receveurs pourraient rapporter à l'envoyeur le nombre de symboles de source qu'il leur manque dans chaque bloc de source. L'envoyeur pourrait alors calculer le nombre maximum de symboles de source manquants dans chaque bloc de source parmi tous les receveurs. Sur cette base, un petit bloc de codeur de FEC pourrait être utilisé pour générer pour chaque bloc de source un nombre de symboles redondants égal au nombre maximum calculé de symboles de source manquants dans le bloc parmi tous les receveurs, pour autant que ce maximum pour chaque bloc n'excède pas le nombre de symboles redondants qui peuvent être générés efficacement. Dans un second tour de transmission, le serveur va alors envoyer tous ces symboles redondants pour tous les blocs. Dans cet exemple, si il n'y a pas de perte dans le second tour de transmission, alors tous les receveurs vont être capables de recréer une copie exacte de chaque bloc original. Dans ce cas, même si différents receveurs manquent de différents symboles dans des blocs différents, les symboles redondants transmis pour un bloc donné sont utiles pour tous les receveurs à qui il manque des symboles dans ce bloc dans le second tour.

Pour les autres protocoles de diffusion groupée IP fiable, les codes de FEC sont utilisés à la façon d'un carrousel de données pour assurer la fiabilité, qu'on appelle un carrousel de données de FEC. Par exemple, un carrousel de données de FEC qui utilise un grand bloc de code de FEC pourrait fonctionner comme suit. Le grand bloc de codeur de FEC produit n symboles de codage en considérant tous les k symboles de source d'un objet comme un bloc. L'envoyeur les parcourt et transmet les n symboles de codage dans des paquets dans le même ordre à chaque tour. Un carrousel de données de FEC peut avoir une bien meilleure protection contre la perte de paquet qu'un carrousel de données. Par exemple, un receveur peut se joindre à la transmission à tout moment, et tant que le receveur reçoit au moins k symboles, il peut complètement récupérer l'objet. Ceci est vrai même si le receveur commence à recevoir des paquets au milieu d'un passage à travers les symboles de codage. Cette méthode peut aussi être utilisée quand l'objet est partagé en blocs et qu'un court bloc de code de FEC est appliqué séparément à chaque bloc. Dans ce cas, comme on l'explique plus en détails ci-dessous, il est utile d'entrelacer les symboles provenant des différents blocs quand ils sont transmis.

Comme tout nombre de symboles de codage peut être généré en utilisant un codeur de FEC expansible, les protocoles de diffusion groupée IP fiable qui utilisent des codes de FEC expansibles s'appuient généralement seulement sur ces codes pour la fiabilité. Par exemple, quand un code de FEC expansible est utilisé dans une application de carrousel de données de FEC, les paquets de codage ne se répètent jamais, et donc, tout k des symboles de codage dans le nombre potentiellement illimité de symboles de codage est suffisant pour récupérer les k symboles de source originaux.

Pour des protocoles de diffusion groupée IP fiable supplémentaires, la méthode pour obtenir la fiabilité est de générer assez de symboles de codage pour que chaque symbole de codage soit transmis une seule fois (au plus). Par exemple, l'envoyeur peut décider a priori combien de symboles de codage il va transmettre, utiliser un code de FEC pour générer ce nombre de symboles de codage à partir de l'objet, et ensuite transmettre les symboles de codage à tous les receveurs. Cette méthode est applicable aux protocoles de flux directs, par exemple, lorsque le flux est partagé en objets, les symboles de source pour chaque objet sont codés en symboles de codage en utilisant un code de FEC, et ensuite les ensembles de symboles de codage pour chaque objet sont transmis l'un après l'autre en utilisant la diffusion groupée IP.

2. Codes de FEC

2.1 Codes simples

Il y a des codes très simples qui sont efficaces pour réparer la perte de paquet dans des conditions de pertes très faibles. Par exemple, pour fournir la protection contre une seule perte, on partitionne l'objet en symboles de source de taille fixe et on ajoute ensuite un symbole redondant qui est la parité (OUX) de tous les symboles de source. La taille d'un symbole de source est choisie de façon à tenir parfaitement dans la charge utile d'un paquet, c'est-à-dire si la charge utile de paquet est de 512 octets, alors chaque symbole de source est de 512 octets. L'en-tête de chaque paquet contient assez d'informations pour identifier la charge utile. C'est un exemple d'identifiant de symbole de codage. Les identifiants de symbole de codage peuvent consister en deux parties dans cet exemple. La première partie est un fanion de codage qui est égal à 1 si le symbole de codage est un symbole de source et est égal à 0 si le symbole de codage est un symbole redondant. La seconde partie de l'identifiant de symbole de codage est un identifiant de symbole de source si le fanion de codage est 1 et un identifiant de symbole redondant si le fanion de codage est 0. Les identifiants de symbole de source peuvent être numérotés de 0 à k-1 et l'identifiant de symbole redondant peut être 0. Par exemple, si l'objet consiste en quatre symboles de source qui ont les valeurs a, b, c et d, alors la valeur du symbole redondant est e = a OUX b OUX c OUX d. Ensuite, les paquets qui portent ces symboles ressemblent à : (1, 0: a), (1, 1: b), (1, 2: c), (1, 3: d), (0, 0: e).

Dans cet exemple, l'identifiant de symbole de codage consiste en les deux premières valeurs, où la première valeur est le fanion de codage et la seconde valeur est soit un identifiant de symbole de source, soit l'identifiant de symbole redondant. La portion du paquet après les deux-points est la valeur du symbole de codage. Tout symbole de source seul de l'objet peut être récupéré comme la parité de tous les autres symboles. Par exemple, si les paquets (1, 0: a), (1, 1: b), (1, 3: d), (0, 0: e) sont reçus, alors la valeur manquante de symbole de source avec l'identifiant de symbole de source = 2 peut être récupérée en calculant un OUX b OUX d OUX e = c.

Une autre façon de former l'identifiant de symbole de codage est faire correspondre les valeurs de 0 à k-1 aux k symboles de source et la valeur k correspond au symbole redondant qui est le OUX des k symboles de source.

Quand le nombre de symboles de source dans l'objet est grand, une variante du simple bloc de code ci-dessus peut être utilisée. Dans ce cas, les symboles de source sont groupés en blocs de source d'un certain nombre k de symboles consécutifs chacun, où k peut être différent pour les différents blocs. Si un bloc consiste en k symboles de source, alors un symbole redondant est ajouté pour former un bloc de codage consistant en k+1 symboles de codage. Ensuite, un bloc de source consistant en k symboles de source peut être récupéré de tout k des k+1 symboles de codage à partir du bloc de codage associé.

On peut aussi utiliser des façons un peu plus sophistiquées d'ajouter des symboles redondants en utilisant la parité. Par exemple, on peut grouper un bloc consistant en k symboles de source dans un objet dans une matrice carrée de p x p, où p = sqrt(k). Ensuite, pour chaque rangée un symbole redondant est ajouté qui est la parité de tous les symboles de source dans la rangée. De même, pour chaque colonne, un symbole redondant est ajouté qui est la parité de tous les symboles de source dans la colonne. Ensuite, toute rangée de la matrice peut être récupérée à partir de tout p des p+1 symboles dans la rangée, et la même chose pour toute colonne. Des codes de produit dimensionnel supérieur utilisant cette technique peuvent aussi être utilisés. Cependant, on doit être prudent en utilisant ces constructions et penser aux possibles schémas de perte de symboles. Idéalement, la propriété qu'on voudrait obtenir est que si k symboles de source sont codés dans n symboles de codage (les symboles de codage consistent en les symboles de source et les symboles redondants) alors les k symboles de source peuvent être récupérés de tout k des n symboles de codage. Avec les constructions simples décrites ci-dessus, on n'obtient pas des codes qui soient proches d'obtenir ce comportement idéal.

2.2 Codes de FEC de petit bloc

Les protocoles de diffusion groupée IP fiable peuvent utiliser un bloc (n, k) de codes de FEC [2]. Pour ces codes, k symboles de source sont codés dans n > k symboles de codage, de telle façon que tout k des symboles de codage puisse être

utilisé pour réassembler les k symboles de source originaux. Donc, ces codes n'ont pas de frais généraux de réception quand ils sont utilisés pour coder directement l'objet entier. Les codes de bloc sont généralement systématiques, ce qui signifie que les n symboles de codage consistent en les k symboles de source et les n-k symboles redondants générés à partir de ces k symboles de source, où la taille d'un symbole redondant est la même que celle pour un symbole de source. Par exemple, le premier code simple (OUX) décrit au paragraphe précédent est un code (k+1, k). En général, la liberté de choisir n plus grand que k+1 est désirable, car cela peut fournir une bien meilleure protection contre les pertes. Un exemple courant de ces types de codes est une classe de codes Reed-Solomon, qui se fondent sur une méthode algébrique utilisant des champs finis. Les mises en œuvre des codes d'écrasement de FEC (n, k) sont assez efficaces pour être utilisés par des ordinateurs personnels [16]. Par exemple, [15] décrit une mise en œuvre où les vitesses de codage et décodage diminuent avec C/j, où la constante C est de l'ordre de 10 à 80 M octets/s pour les machines de classe Pentium de diverses marques et j a une limite supérieure de min(k, n-k).

En pratique, les valeurs de k et n doivent être petites (par exemple, en dessous de 256) pour de tels codes de FEC car les grandes valeurs rendent le codage et décodage d'un coût prohibitif. Comme de nombreux objets sont plus longs que k symboles pour des valeurs raisonnables de k et de la longueur de symbole (par exemple plus de 16 k octets pour k = 16 en utilisant des symboles de 1 k octets) ils peuvent être divisés en un certain nombre de blocs de source. Chaque bloc de source consiste en un certain nombre k de symboles de source, où k peut varier entre les différents blocs de source. Le codeur de FEC est utilisé pour coder un bloc de source de k symboles de source en un bloc de codage de n symboles de codage, où le nombre n de symboles de codage dans le bloc de codage peut varier pour chaque bloc de source. Pour qu'un receveur récupère complètement l'objet, pour chaque bloc de source consistant en k symboles de source, k distincts symboles de codage (c'est-à-dire, avec des identifiants différents de symbole de codage) doivent être reçus du bloc de codage correspondant. Pour certains blocs de codage, plus de symboles de codage peuvent être reçus qu'il n'y a de symboles de source dans le bloc de source correspondant, et dans ce cas, les symboles de codage en excès sont éliminés. Un exemple de structure de codage est montré à la Figure 1.

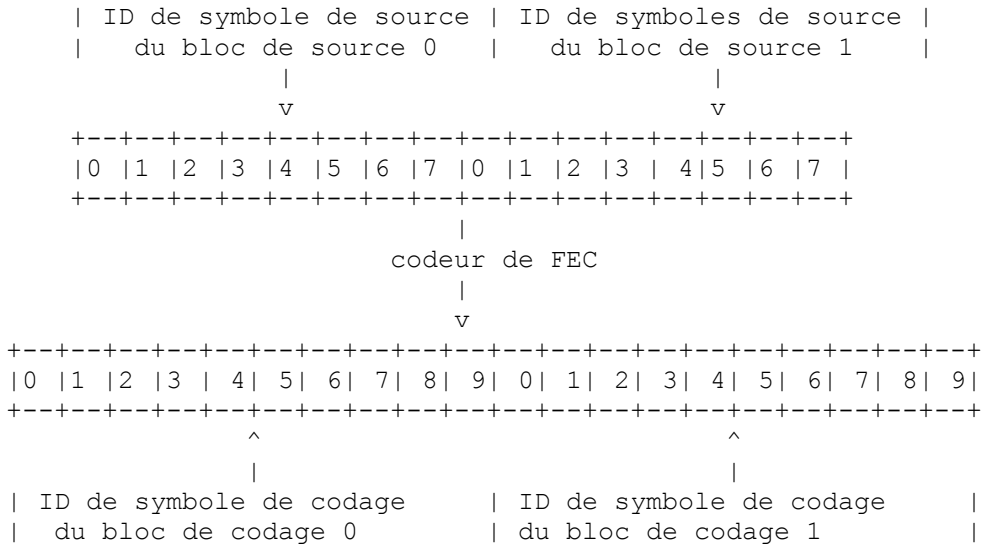

Figure 1. Structure de codage pour un objet divisé en deux blocs de source consistant en 8 symboles de source chacun, et le codeur de FEC est utilisé pour générer 2 symboles redondants supplémentaires (10 symboles de codage au total) pour chacun des deux blocs de source.

Dans de nombreux cas, un objet est partagé en blocs de source de longueur égale chacun consistant en k symboles de source contigus de l'objet, c'est-à-dire, le bloc c consiste en la gamme de symboles de source [ck, (c+1)k-1]. Cela assure que le codeur de FEC peut être optimisé à traiter un nombre particulier k de symboles de source. Cela assure aussi que les références de mémoire sont locales quand l'envoyeur lit les symboles de source à coder, et quand le receveur lit les symboles de codage à décoder. La localisation de référence est particulièrement importante quand l'objet est mémorisé sur un disque, car cela réduit les recherches nécessaires sur le disque. Le numéro de bloc et l'identifiant de symbole de source au sein de ce bloc peuvent être utilisés pour spécifier de façon univoque un symbole de source au sein de l'objet. Si la taille de l'objet n'est pas un multiple de k symboles de source, alors le dernier bloc de source va contenir moins de k symboles.

Les numéros de bloc peuvent être consécutifs en commençant à zéro. Les symboles de codage au sein d'un bloc peuvent être identifiés de façon univoque par un identifiant de symbole de codage. Une façon d'identifier les symboles de codage au sein d'un bloc est d'utiliser la combinaison d'un fanion de codage qui identifie le symbole comme symbole de source ou symbole redondant ainsi que comme identifiant de symbole de source ou identifiant de symbole redondant. Par exemple,

une valeur de fanion de codage de 1 peut indiquer que le symbole de codage est un symbole de source et 0 peut indiquer qu'il est un symbole redondant. Les identifiants de symbole de source peuvent être numérotés de 0 à k-1 et les identifiants de symbole redondants peuvent être numérotés de 0 à n-k-1.

Par exemple, si l'objet consiste en 10 symboles de source avec les valeurs a, b, c, d, e, f, g, h, i, et j, et k = 5 et n = 8, alors il y a deux blocs de source consistant en 5 symboles chacun, et il y a deux blocs de codage consistant en 8 symboles chacun. Soit p, q et r les valeurs des symboles redondants pour le premier bloc de codage, et soit x, y et z les valeurs des symboles redondants pour le second bloc de codage. Alors, les symboles de codage avec leurs identifiants sont :

(0, 1, 0: a), (0, 1, 1: b), (0, 1, 2: c), (0, 1, 3: d), (0, 1, 4: e),
(0, 0, 0: p), (0, 0, 1: q), (0, 0, 2: r),
(1, 1, 0: f), (1, 1, 1: g), (1, 1, 2: h), (1, 1, 3: i), (1, 1, 4: j),
(1, 0, 0: x), (1, 0, 1: y), (1, 0, 2: z).

Dans cet exemple, la première valeur identifie le numéro de bloc et les deux secondes valeurs ensemble identifient le symbole de codage au sein du bloc, c'est-à-dire, l'identifiant de symbole de codage consiste en le fanion de codage avec soit l'identifiant de symbole de source, soit l'identifiant de symbole redondant selon la valeur du fanion de codage. La valeur du symbole de codage est écrite après les deux-points. Chaque bloc peut être récupéré de tous 5 des 8 symboles de codage associés à ce bloc. Par exemple, la réception de

(0, 1, 1: b), (0, 1, 2: c), (0, 1, 3: d), (0, 0, 0: p), (0, 0, 1: q)

est suffisante pour récupérer le premier bloc de source, et la réception de

(1, 1, 0: f), (1, 1, 1: g), (1, 0, 0: x), (1, 0, 1: y), (1, 0, 2: z)

est suffisante pour récupérer le second bloc de source.

Une autre façon d'identifier de façon univoque les symboles de codage au sein d'un bloc est de mettre les identifiants de symbole de codage pour les symboles de source à 0,...,k-1 et de mettre les identifiants de symbole de codage pour les symboles redondants à k,...,n-1.

2.3 Codes de FEC de grand bloc

Les codes Tornado [9], [10], [11], [12], [13], sont des codes de FEC de grands blocs qui fournissent une solution de remplacement aux codes de FEC de petit bloc. Un code Tornado de (n, k) exige un peu plus de k parmi n symboles de codage pour récupérer k symboles de source, c'est-à-dire, il y a de petits frais généraux de réception. L'avantage du faible coût de frais généraux de réception non zéro est que la valeur de k peut être de l'ordre des dizaines de milliers sans nuire à l'efficacité du codage et du décodage. À cause de considérations de mémoire, en pratique la valeur de n doit être restreinte à être un petit mutiple de k, par exemple, n = 2k. Par exemple, [4] décrit une mise en œuvre de codes Tornado où les vitesses de codage et de décodage sont de dizaines de méga octets par seconde pour des machines de classe Pentium de divers fabricants quand k est dans les dizaines de milliers et n = 2k. Les frais généraux de réception pour de telles valeurs de k et n sont dans la gamme de 5 à 10 %. Les codes Tornado exigent qu'une grande quantité d'informations hors bande soient communiquées à tous les envoyeurs et receveurs pour chaque différente longueur d'objet, et exigent une quantité de mémoire sur le codeur et le décodeur qui est proportionnelle à la longueur de l'objet fois 2n/k.

Les codes Tornado sont conçus pour avoir en moyenne de faibles frais généraux de réception par rapport à la réception d'une portion aléatoire des paquets de codage. Donc, pour assurer qu'un receveur peut réassembler l'objet avec de faibles frais généraux de réception, les paquets sont permutés en ordre aléatoire avant la transmission.

2.4 Codes de FEC expansibles

Tous les codes de FEC décrits jusqu'à ce point sont des codes de bloc. Il y a un type différent de codes de FEC qu'on appelle des codes de FEC expansibles. Comme les codes de bloc, un codeur de FEC expansible opère sur un objet de taille connue qui est partagé en symboles de source de longueur égale. À la différence des codes de bloc, il n'y a pas de nombre prédéterminé de symboles de codage qui peuvent être générés pour les codes de FEC expansibles. À la place, un codeur de FEC expansible peut générer autant de symboles de codage uniques que requis à la demande.

Les codes LT [5], [6], [7], [8], sont un exemple de grands codes de FEC expansibles avec peu de frais généraux de

réception. Un codeur LT utilise l'aléation pour générer chaque symbole de codage au hasard et indépendamment de tous les autres symboles de codage. Comme les codes Tornado, le nombre de symboles de source k peut être très grand pour les codes LT, c'est-à-dire, de l'ordre des diraines à des centaines de milliers, et le logiciel de codeur et de décodeur fonctionne efficacement. Par exemple les vitesses de codage et de décodage pour les codes LT sont dans la gamme de 3 à 20 M octets par seconde pour les machines de classe Pentium de divers fabricants quand k est dans les dizaines de milliers. Un codeur LT peut générer autant de symboles de codage que requis à la demande. Quand un nouveau symbole de codage est à générer par un codeur LT, il se fonde sur un identifiant de symbole de codage choisi au hasard qui décrit de façon univoque comment le symbole de codage est à générer à partir des symboles de source. En général, chaque valeur d'identifiant de symbole de codage correspond à un unique symbole de codage, et donc, l'espace des symboles de codage possibles est approximativement de quatre milliards si par exemple l'identifiant de symbole de codage est de 4 octets. Donc, les chances qu'un symbole de codage particulier soit le même qu'un autre symbole de codage particulier sont inversement proportionnelles au nombre d'identifiants de symbole de codage possibles. Un décodeur LT a la propriété que, avec une très forte probabilité, la réception de tout ensemble de symboles de codage de légèrement plus que k générés au hasard et indépendamment, est suffisante pour réassembler les k symboles de source. Par exemple, quand k est de l'ordre des dizaines à des centaines de milliers, les frais généraux de réception sont de moins de 5 % sans échec sur des centaines de millions d'essais dans toutes les conditions de pertes.

Parce que les symboles de codage sont générés au hasard et indépendamment en choisissant des identifiants aléatoires de symbole de codage, les codes LT ont la propriété que les symboles de codage pour les mêmes k symboles de source peuvent être générés et transmis à partir de plusieurs envoyeurs et reçus par un receveur, et les frais généraux de réception et le temps de décodage sont les mêmes que si tous les symboles de codage avaient été générés par un seul envoyeur. La seule exigence est que les envoyeurs choisissent leurs identifiants de symbole de codage au hasard et indépendamment les uns des autres.

Il y a un petit compromis entre le nombre de symboles de source et les frais généraux de réception pour les codes LT, et plus est grand le nombre de symboles de source, plus petits sont les frais généraux de réception. Donc, pour les plus courts objets, il est parfois avantageux de partager l'objet en de nombreux symboles de source courts et d'inclure plusieurs symboles de codage dans chaque paquet. Dans ce cas, un seul identifiant de symbole de codage est utilisé pour identifier les multiples symboles de codage contenus dans un seul paquet.

Il y a un couple de facteurs pour choisir le compromis approprié entre longueur de symbole et nombre de symboles de source. La principale considération est qu'il y a des frais généraux fixes par symbole dans les exigences de traitement globales du codage et du décodage, indépendamment du nombre de symboles de source. Donc, utiliser de plus courts symboles signifie que ces frais généraux fixes de traitement par symbole vont être un plus grand composant des exigences de traitement globales, conduisant à de plus grandes exigences de traitement global. Une seconde considération beaucoup moins importante est qu'il y a un composant du traitement par symbole qui dépend logarithmiquement du nombre de symboles de source, et donc pour cette raison, il y a une légère préférence pour moins de symboles de source.

Comme pour les petits codes de bloc, il y a un point où l'objet est assez grand pour qu'il y ait du sens à le partager en blocs quand on utilise des codes LT. Généralement l'objet est partagé en blocs chaque fois que le nombre de symboles de source multiplié par la longueur de charge utile de paquet est inférieur à la taille de l'objet. Donc, si la charge utile de paquet est de 1024 octets et si le nombre maximum de symboles de source est 128 000, alors tout objet de plus de 128 M octets va être partagé en plus d'un bloc. On peut choisir le nombre maximum de symboles de source à utiliser, selon le compromis entre la vitesse désirée de codage et décodage et les frais généraux de réception désirés. Les symboles de codage peuvent être identifiés de façon univoque par un numéro de bloc (quand l'objet est assez grand pour être partagé en plus d'un bloc) et un identifiant de symbole de codage. Les numéros de bloc, si ils sont utilisés, sont généralement consécutifs à partir de zéro au sein de l'objet. Le numéro de bloc et l'identifiant de symbole de codage sont tous deux choisis uniformément et au hasard dans leur gamme quand un symbole de codage doit être transmis. Par exemple, supposons que le nombre de symboles de source soit 128 000 et que le nombre de blocs soit 2. Chaque paquet généré par le codeur LT pourrait alors être de la forme (b, x: y). Dans cet exemple, la première valeur identifie le numéro de bloc et la seconde valeur identifie le symbole de codage au sein du bloc. Dans cet exemple, le numéro de bloc b est 0 ou 1, et l'identifiant de symbole de codage x pourrait être une valeur de 32 bits. La valeur y après les deux-points est la valeur du symbole de codage.

2.5 Blocs de source avec symboles de source de longueur variable

Pour tous les codes de FEC décrits ci-dessus, tous les symboles de source dans le même bloc de source sont de la même longueur. Dans ce paragraphe, on décrit une technique générale pour traiter le cas où il est désirable d'utiliser des symboles de source de longueur variable dans un seul bloc de source. Cette technique est applicable aux codes de bloc de FEC.

Soit l_1, l_2, ... , l_k les longueurs en octets de k longueurs variables de symboles de source à considérer comme faisant

partie du même bloc de source. Soit lmax le maximum sur i = 1, ... , k de l_i. Pour préparer le bloc de source pour le codeur de FEC, on bourre chaque symbole de source i jusqu'à la longueur lmax avec un suffixe de lmax-l_i zéros, et on ajoute au début de cela la valeur l_i. Donc, chaque symbole de source bourré est de longueur x+lmax, en supposant qu'il faut x octets pour mémoriser un entier avec les valeurs 0,...,lmax possibles, où x est une constante du protocole connue du codeur et du décodeur. Ces symboles de source bourrés, chacun de longueur x+lmax, sont l'entrée au codeur, avec la valeur n. Le codeur génère alors n-k symboles redondants, chacun de longueur x+lmax.

Les symboles de codage qui sont placés dans les paquets consistent en les k symboles de source d'origine de longueur variable et n-k symboles redondants, chacun de longueur x+lmax. À partir de tout k des symboles de codage reçus, le décodeur de FEC recrée les k symboles de source d'origine comme suit. Si tous les k symboles de source d'origine sont reçus, aucun décodage n'est alors nécessaire. Autrement, au moins un symbole redondant est reçu, à partir duquel le receveur peut facilement déterminer si le bloc est composé de symboles de source de longueur variable : si le ou les symboles redondants sont plus longs que les symboles de source, alors les symboles de source sont de longueur variable. Noter que dans un bloc de longueur variable les symboles redondants sont toujours plus longs que le plus long symbole de source, du fait de la présence de la longueur de symbole ajoutée devant. Le receveur peut déterminer la valeur de lmax en soustrayant x de la longueur d'un symbole redondant reçu. Pour chacun des symboles de source originaux reçus, le receveur peut générer le symbole de source bourré correspondant comme décrit ci-dessus. Ensuite, l'entrée au décodeur de FEC est l'ensemble des symboles redondants reçus, avec l'ensemble des symboles de source bourrés construit à partir des symboles originaux reçus. Le décodeur de FEC produit alors l'ensemble de k symboles de source bourrés. Une fois que les k symboles de source bourrés ont été récupérés, la longueur l_i du symbole de source original i peut être récupérée des x premiers octets du ième symbole de source bourré, et ensuite le symbole de source i original est obtenu en prenant les prochains l_i octets qui suivent les x octets du champ de longueur.

3. Considérations sur la sécurité

L'utilisation de la FEC, par elle-même, n'impose pas de considérations de sécurité supplémentaires par rapport à l'envoi des mêmes informations sans FEC. Cependant, tout comme pour les autres systèmes de transmission, un envoyeur malveillant peut essayer d'injecter des paquets portant des symboles de codage corrompus. Si un receveur accepte un ou plusieurs symbole de codage corrompus, au lieu d'authentiques, ce receveur peut reconstruire un objet corrompu.

L'authentification d'objet de transmission au niveau application peut détecter le transfert corrompu, mais le receveur doit éliminer l'objet transféré. En injectant des symboles de codage corrompus, ils sont acceptés comme des symboles de codage valides par un receveur, ce qui au minimum, est une attaque de déni de service effective.

En face de cette possibilité, les receveurs de FEC peuvent confronter l'adresse de source d'un symbole reçu à une liste d'adresses de transmetteurs authentiques. Comme les adresses de source peuvent être usurpées, les protocoles de transport qui utilisent la FEC peuvent fournir des mécanismes pour une robuste authentification de source de chaque symbole de codage. Les routeurs de diffusion groupée le long du chemin d'un transfert de FEC peuvent fournir la capacité d'éliminer les paquets de diffusion groupée qui ont leur origine sur ce sous-réseau, et dont l'adresse IP de source ne correspond pas à celle du sous-réseau.

Il est recommandé qu'un schéma d'authentification de paquet comme TESLA [14] soit utilisé en conjonction avec les codes de FEC. Alors, les paquets qui ne peuvent pas être authentifiés peuvent être éliminés et l'objet peut être fiablement récupéré des paquets authentifiés reçus.

4. Divulgation de droits de propriété intellectuelle

Des droits de propriété intellectuelle ont été notifiés à l'IETF, revendiqués sur certaines ou toutes les spécifications contenues dans le présent document. Pour plus d'informations, consulter la liste en ligne des droits revendiqués.

5. Remerciements

Merci à Vincent Roca et Hayder Radha de leurs commentaires détaillés sur ce document.

6. Références

[1] Acharya, S., Franklin, M. and S. Zdonik, "Dissemination - Based Data Delivery Using Broadcast Disks", IEEE Personal Communications, pp.50-60, décembre 1995.

[2] Blahut, R.E., "Theory and Practice of Error Control Codes", Addison Wesley, MA, 1984.

[3] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, octobre 1996.

[4] Byers, J.W., Luby, M., Mitzenmacher, M. et A. Rege, "A Digital Fountain Approach to Reliable Distribution of Bulk Data", Proceedings ACM SIGCOMM '98, Vancouver, Canada, septembre 1998.

[5] Haken, A., Luby, M., Horn, G., Hernek, D., Byers, J. and M. Mitzenmacher, "Generating high weight coding symbols using a basis", U.S. Patent No. 6,411,223, 25 juin 2002.

[6] Luby, M., "Information Additive Code Generator and Decoder for Communication Systems", U.S. Patent No. 6,307,487, 23 octobre 2001.

[7] Luby, M., "Information Additive Group Code Generator and Decoder for Communication Systems", U.S. Patent No. 6,320,520, 20 novembre 2001.

[8] Luby, M., "Information Additive Code Generator and Decoder for Communication Systems", U.S. Patent No. 6,373,406, 16 avril 2002.

[9] Luby, M. and M. Mitzenmacher, "Loss resilient code with double heavy tailed series of redundant layers", U.S. Patent No. 6,195,777, 27 février 2001.

[10] Luby, M., Mitzenmacher, M., Shokrollahi, A., Spielman, D. and V. Stemann, "Message encoding with irregular graphing", U.S. Patent No. 6,163,870, 19 décembre 2000.

[11] Luby, M., Mitzenmacher, M., Shokrollahi, A. and D. Spielman, "Efficient Erasure Correcting Codes", IEEE Transactions on Information Theory, Special Issue: Codes on Graphs and Iterative Algorithms, pp. 569-584, Vol. 47, No. 2, février 2001.

[12] Luby, M., Shokrollahi, A., Stemann, V., Mitzenmacher, M. and D. Spielman, "Loss resilient decoding technique", U.S. Patent No. 6,073,250, 6 juin 2000.

[13] Luby, M., Shokrollahi, A., Stemann, V., Mitzenmacher, M. and D. Spielman, "Irregularly graphed encoding technique", U.S. Patent No. 6,081,909, 27 juin 2000.

[14] Perrig, A., Canetti, R., Song, D. and J.D. Tygar, "Efficient and Secure Source Authentication for Multicast", Network et Distributed System Security Symposium, NDSS 2001, pp. 35-46, février 2001.

[15] Rizzo, L., "Effective Erasure Codes for Reliable Computer Communication Protocols", ACM SIGCOMM Computer Communication Review, Vol.27, No.2, pp.24-36, avril 1997.

[16] Rizzo, L., "On the Feasibility of Software FEC", DEIT Tech Report, http://www.iet.unipi.it/~luigi/softfec.ps, janvier 1997.

7. Adresse des auteurs

Michael Luby
Digital Fountain
39141 Civic Center Drive
Suite 300
Fremont, CA 94538
mél : luby@digitalfountain.com

Lorenzo Vicisano
Cisco Systems, Inc.
170 West Tasman Dr.,
San Jose, CA, 95134
mél : lorenzo@cisco.com

Jim Gemmell
Microsoft Research
455 Market St. #1690
San Francisco, CA, 94105
mél : jgemmell@microsoft.com

Luigi Rizzo

Mark Handley

Jon Crowcroft

Dip. di Ing. dell'Informazione
Universita` di Pisa
via Diotisalvi 2, 56126 Pisa, Italy
mél : luigi@iet.unipi.it

ICSI Center for Internet Research
1947 Center St.
Berkeley CA, 94704
mél : mjh@icir.org

University of Cambridge
William Gates Building
J J Thomson Avenue
Cambridge CB3 0FD
mél: Jon.Crowcroft@cl.cam.ac.uk

8. Déclaration complète de droits de reproduction

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