(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4680359号
(P4680359)
(45)発行日　平成23年5月11日(2011.5.11)　(24)登録日　平成23年2月10日(2011.2.10)

(51)Int.Cl.		F I		
H03M 13/15	(2006.01)	H03M	13/15	
G06F 11/10	(2006.01)	G06F	11/10	330M
H03M 13/09	(2006.01)	H03M	13/09	
H04B 14/04	(2006.01)	H04B	14/04	Z
H04L 1/00	(2006.01)	H04L	1/00	B

請求項の数 38　外国語出願　(全 50 頁)

(21)出願番号　特願2000-282732 (P2000-282732)
(22)出願日　平成12年9月18日 (2000.9.18)
(65)公開番号　特開2001-189665 (P2001-189665A)
(43)公開日　平成13年7月10日 (2001.7.10)
審査請求日　平成19年9月18日 (2007.9.18)
(31)優先権主張番号　09/399.201
(32)優先日　平成11年9月17日 (1999.9.17)
(33)優先権主張国　米国(US)

(73)特許権者　501114844
デジタル　ファウンテン，　インコーポレイテッド
アメリカ合衆国、カリフォルニア州　９２１２１，サン・ディエゴ、モアハウス・ドライブ　５７７５
(74)代理人　100108855
弁理士　蔵田　昌俊
(74)代理人　100091351
弁理士　河野　哲
(74)代理人　100088683
弁理士　中村　誠
(74)代理人　100109830
弁理士　福原　淑弘

最終頁に続く

(54)【発明の名称】通信システムのための情報付加的群符号発生器およびデコーダ

(57)【特許請求の範囲】
【請求項１】
出力記号の１群を生成する方法であって、１群は１つ以上の出力記号であり、各出力記号は出力アルファベットから選択され、入力アルファベットからそれぞれ選択される順列の複数の入力記号を含む入力ファイルが、このような群の１セットから回復可能であるような該群であり、該方法は、
該群のキーＩを得るステップであって、該キーがキーアルファベットから選択され、該キーアルファベットにおける選択可能なキーの数が、任意の予想される入力ファイルサイズについて、該入力ファイルにおける入力記号の数に対して、実質的に無限である、ステップと、
Ｉを変数とする所定の関数にしたがって、該群のリストＡＬ（Ｉ）を計算するステップであって、ＡＬ（Ｉ）は、該群に関連付けられた入力記号に関連付けられた重みＷ（Ｉ）を示し、重みＷ（Ｉ）は、少なくとも２つの値の間で変化する正の整数であり、少なくとも１つのＩの値について重みＷ（Ｉ）が、１より大きい、ステップと、
ＡＬ（Ｉ）によって示される該関連付けられた入力記号を変数とする所定の関数から、該群における各出力記号の出力記号値を生成するステップと、
を含む方法。
【請求項２】
前記キーＩを得るステップが、ランダム関数または擬似ランダム関数にしたがってキーＩを計算するステップを含む、請求項１に記載の方法。

【請求項３】
前記計算するステップが、Ｉのランダム関数または擬似ランダム関数にしたがってＷ（Ｉ）を計算するステップを含む、請求項１に記載の方法。
【請求項４】
前記計算するステップが、Ｉのランダム関数または擬似ランダム関数にしたがってＡＬ（Ｉ）を計算するステップを含む、請求項１に記載の方法。
【請求項５】
前記計算するステップが、前記群における出力記号の数Ｇ（Ｉ）を計算するステップを含み、該数Ｇ（Ｉ）が正の整数であり、該数Ｇ（Ｉ）がＩのランダム関数または擬似ランダム関数にしたがって計算される、請求項１に記載の方法。
【請求項６】
前記計算するステップが、
Ｉを変数とする所定の関数および確率分布にしたがって、重みＷ（Ｉ）を計算するステップであって、該確率分布が少なくとも２つの正の整数上にあり、そのうちの少なくとも１つが１より大きい、ステップと、
リストＡＬ（Ｉ）のリストエントリを計算するステップと、
Ｗ（Ｉ）個のリストエントリが計算されるまで、リストＡＬ（Ｉ）のリストエントリを計算するステップを繰り返すステップと、
を含む、請求項１に記載の方法。
【請求項７】
前記Ｗ（Ｉ）を計算するステップが、重みＷ（Ｉ）が前記キーアルファベット上の所定の分布を近似するような重みＷ（Ｉ）を決定するステップを含む、請求項６に記載の方法。
【請求項８】
前記所定の分布が一様分布である、請求項７に記載の方法。
【請求項９】
前記所定の分布がつり鐘曲線分布である、請求項７に記載の方法。
【請求項１０】
前記所定の分布は、Ｗ（Ｉ）＝１が１／Ｋの確率を有し、ここで、Ｋが前記入力ファイルにおける入力記号の数であり、そしてＷ（Ｉ）＝ｉがｉ＝２，．．．，Ｋに対して１／ｉ（ｉ－１）の確率を有する、請求項７に記載の方法。
【請求項１１】
前記ＡＬ（Ｉ）によって示される関連付けられた入力記号を変数とする前記所定の関数が、リードーソロモン符号によって決定される、請求項１に記載の方法。
【請求項１２】
前記入力アルファベットおよび前記出力アルファベットが同じアルファベットである、請求項１に記載の方法。
【請求項１３】
前記入力アルファベットが2^{Mi}個の記号を含み、各入力記号がＭｉビットを符号化し、前記出力アルファベットが2^{Mo}個の記号を含み、各出力記号の群がＭｏビットを符号化する、請求項１に記載の方法。
【請求項１４】
各後続キーＩが前記キーアルファベットのシーケンスにおける次のキーである、請求項１に記載の方法。
【請求項１５】
出力記号の複数の群を符号化する方法であって、出力記号の各群は請求項１に記載され、該方法は、
生成されるべき該出力記号の群の各々に対するキーＩを生成するステップと、
および
該生成された出力記号の群の各々をデータイレーズチャネルを介して送信されるべき出

力シーケンスとして出力するステップと、
をさらに含む方法。
【請求項１６】
各キーＩが他の選択されるキーとは独立に選択される、請求項１５に記載の方法。
【請求項１７】
前記ＡＬ（Ｉ）を計算するステップが、
前記入力ファイルにおける入力記号の数Ｋを少なくとも近似的に識別し、かつ重みＷ（Ｉ）を識別するステップと、
Ｋより大きいかまたはＫに等しい最小素数Ｐを決定するステップと、
ＰがＫより大きい場合、少なくとも該入力ファイルにＰ－Ｋ個のブランク入力記号を付加するステップと、
１≦Ｘ＜Ｐである第１の整数Ｘ、および０≦Ｙ＜Ｐである第２の整数Ｙを生成するステップと、
１～Ｗ（Ｉ）の各Ｊに対して、ＡＬ（Ｉ）における第Ｊ番目のエントリを（（Ｙ＋（Ｊ－１）・Ｘ）　ｍｏｄ　Ｐ）に設定するステップと、
を含む、請求項１に記載の方法。
【請求項１８】
前記各Ｊに対してＡＬ（Ｉ）における第Ｊ番目のエントリを設定するステップが、
配列Ｖにおける第１のエントリＶ［Ｊ＝０］をＹに設定するステップと、
１～Ｗ（Ｊ）－１の各Ｊに対して、該配列Ｖにおける第Ｊ番目のエントリＶ［Ｊ］を（（Ｖ［Ｊ－１］＋Ｘ）　ｍｏｄ　Ｐ）に設定するステップと、
該配列Ｖを該リストＡＬ（Ｉ）として使用するステップと、
を含む、請求項１７に記載の方法。
【請求項１９】
ソースから目的地にパケット通信チャネルを介してデータを送信する方法であって、該方法が、
ａ）送信されるべき該データを入力記号の順列セットとして構成するステップであって、各入力記号は、入力アルファベットから選択され、該データにおける位置を有する、ステップと、
ｂ）出力記号の複数の群を生成するステップであって、各出力記号が出力アルファベットから選択され、該複数の群の各群が、以下のステップによって生成され、該ステップは、
キーアルファベットから、生成される該群のキーＩを選択するステップと、
Ｉの関数として重みＷ（Ｉ）を決定するステップであって、ここで、重みＷ（Ｉ）は、少なくとも２つの値の間および該キーアルファベット上で変化する正の整数であり、該キーアルファベットにおける少なくともいくつかのキーに対して１より大きい、ステップと、
Ｉの関数にしたがってＷ（Ｉ）個の該入力記号を選択し、したがって、該群に関連付けられるＷ（Ｉ）個の入力記号のリストＡＬ（Ｉ）を形成するステップと、
該群における出力記号の数Ｇ（Ｉ）を決定するステップと、
該関連付けられるＷ（Ｉ）個の入力記号を変数とする所定の値関数から該群における各出力記号の値Ｂ（Ｉ）を計算するステップとを含む、ステップと、
ｃ）該複数の群のうちの少なくとも１つの出力記号の群のうちの少なくとも１つを、複数のパケットの各々にパケット化するステップと、
ｄ）該複数のパケットを、該パケット通信チャネルを介して送信するステップと、
ｅ）該複数のパケットのうちの少なくともいくつかを該目的地で受信するステップと、
ｆ）該複数の受信されたパケットから該データを復号化するステップと、
を含む方法。
【請求項２０】

前記データを復号化するステップが、
１）受信された出力記号の各群を処理するステップであって、
ａ）該群の前記キーＩを決定するステップと、
ｂ）該群の前記重みW（Ｉ）を決定するステップと、
ｃ）該群の前記W（Ｉ）個の関連付けられた入力記号を決定するステップとを含む、ステップと、
２）任意の入力記号を復号化するのに十分な情報が受信されたかどうかを決定するステップと、
３）該受信された情報から復号化され得る入力記号を復号化するステップと、
を含む、請求項１９に記載の方法。

【請求項２１】
前記キーＩを決定するステップが、前記パケット通信チャネルを介して受信されたパケットにおいて供給されたデータから該キーＩを少なくとも部分的に決定するステップを含む、請求項２０に記載の方法。

【請求項２２】
前記データを復号化するステップが、
１）受信された出力記号の各群を処理するステップであって、
ａ）該受信された群の前記重みW（Ｉ）を決定するステップと、
ｂ）該受信された群の前記W（Ｉ）個の関連付けられた入力記号を決定するステップと、
ｃ）出力記号の群のテーブルにおける群の出力記号の値Ｂ（Ｉ）を、該受信された群に対する該重みW（Ｉ）および前記W（Ｉ）個の関連付けの位置とともに格納するステップとを含む、ステップと、
２）出力記号のさらなる群を受信し、それらをステップ１）およびそのサブステップにしたがって処理するステップと、
３）多くとも、群サイズの重みを有し、使い古された出力記号の群として表記されていない出力記号の各群、ＧＯＳ１、に対して、以下のステップ、
ａ）ＧＯＳ１に対応する入力記号の位置に対して入力記号を計算するステップと、
ｂ）該出力記号の群のテーブルにおける接続された出力記号の群を識別するステップであって、接続された出力記号の群は、ステップ３）ａ）において処理された少なくとも１つの入力記号の関数である出力記号の群である、ステップと、
ｃ）ステップ３）ａ）において処理された該入力記号に独立になるように該接続された出力記号の群を再計算するステップと、
ｄ）ステップ３）ａ）において処理された該入力記号の独立性を反映するようにステップ３）ｃ）において再計算された該出力記号の群の重みを決定するステップと、
ｅ）ＧＯＳ１を使い古した出力記号として表記するステップと、を実行するステップと、
４）前記入力記号の順列セットが前記目的地で回復されるまで、該ステップ１）からステップ３）を繰り返すステップと、
を含む、請求項１９に記載の方法。

【請求項２３】
前記表記するステップが、前記使い古した出力記号にゼロの重みを割り当てるステップである、請求項２２に記載の方法。

【請求項２４】
前記表記するステップが、前記出力記号の群のテーブルから前記使い古した出力記号の群を除去するステップを含む、請求項２２に記載の方法。

【請求項２５】
前記パケット化するステップが、複数の出力記号の群のうちの少なくとも１つの群から少なくとも１つの出力記号の群を各パケットにパケット化するステップであり、該方法が

、パケット内の出力記号の１群の位置を該出力記号の群のキーの一部として使用するステップをさらに含む、請求項１９に記載の方法。

【請求項２６】

前記計算するステップが、前記群における出力記号の数Ｇ（Ｉ）を計算するステップを含み、該Ｇ（Ｉ）は、正の整数であり、Ｉのすべての値に対して同じ正の整数である、請求項１に記載の方法。

【請求項２７】

出力記号のグループを生成する方法であって、グループは１個以上の出力記号であり、出力記号の各々は出力アルファベットから選択され、該グループは、各々が入力アルファベットから選択される順列の複数の入力記号を含む入力ファイルがこのようなグループのセットから回復可能であるようなものであり、該方法は、

所与のグループに対して、該グループに関連付けられるＷ個の関連付け入力記号を指示するリストＡＬを決定するステップであって、Ｗは正の整数であり、少なくとも２つのグループは各々のＷ個の関連付けに対して異なる値を有し、少なくとも１つのグループに対してＷは２以上であり、生成可能なグループの個数は、任意の予想される入力ファイルサイズについて、該入力ファイル内の入力記号の数に対して実質的に無限である、ステップと、

ＡＬにより指示されるＷ個の関連付け入力記号を変数とする所定の関数から、前記グループ内の各々の出力記号の出力記号値を生成するステップと、

を含むことを特徴とする方法。

【請求項２８】

さらに、

所与のグループに関連付けられたキーＩからＷを計算するステップと、

キーＩからＡＬを計算するステップと、

を含むことを特徴とする請求項２７に記載の方法。

【請求項２９】

前記Ｗを計算するステップと前記ＡＬを計算するステップとは、確率分布を使用することを特徴とする請求項２８に記載の方法。

【請求項３０】

入力ファイルの情報コンテンツを送信する方法であって、該入力ファイルは、各々が入力アルファベットから選択される順列の複数の入力記号であり、該入力ファイルは、各々が出力アルファベットから選択される出力記号のセットから回復可能なものであり、該方法は、

出力記号の複数のグループを生成するステップであって、グループは１以上の出力記号であり、前記複数のグループの各々のグループは、

ａ）前記グループに関連付けられたＷ個の関連付け入力記号を指示するリストＡＬを決定するステップであって、Ｗは正の整数であり、少なくとも２つのグループは各々のＷ個の関連付けに対して異なる値を有し、少なくとも１つのグループに対してＷは２以上であり、生成可能なグループの個数は、任意の予想される入力ファイルサイズについて、入力ファイル内の入力記号の数に対して、実質的に無限である、ステップと、

ｂ）ＡＬにより指示されるＷ個の関連付け入力記号を変数とする所定の関数から、前記グループ内の各々の出力記号の出力記号値を生成するステップと、

により生成される、ステップと、

生成された複数のグループを送信するステップと、

を含むことを特徴とする方法。

【請求項３１】

前記複数のグループを生成するステップは、複数のソースの各々において、独立した複数個のグループを生成するステップであり、該複数個のグループは、前記複数のソースの各々のソースが、該複数のソースの他のソースで生成されたリストの参照を必要とすること無く、ある独立ソースが他のソースの情報出力と完全に重複すること無く、関連付け入

力シンボルのリストを決定するという点で独立であることを特徴とする請求項３０に記載の方法。
【請求項３２】
さらに、
あるチャネルにより提供される情報コンテンツが、他のチャネルの情報コンテンツと完全に重複すること無く、複数のチャネルにわたって複数のグループを配信するステップを含むことを特徴とする請求項３０に記載の方法。
【請求項３３】
前記決定するステップは、
キーアルファベットから、生成されるグループに対するキーＩを選択するステップであって、該キーアルファベットは入力記号の順列のセットに含まれる入力記号の個数よりも多い個数を含む、ステップと、
Ｉの関数として重みＷ（Ｉ）を決定するステップと、
Ｉの関数に従って入力記号のＷ（Ｉ）を選択ステップであって、それにより前記グループに関連付けられたＷ（Ｉ）個の入力記号のリストＡＬ（Ｉ）を形成する、ステップと、
前記グループ内の出力記号の個数Ｇ（Ｉ）を決定するステップと、
関連付けられたＷ（Ｉ）個の入力記号を変数とする所定の値関数から、前記グループのＧ（Ｉ）個の出力記号の各々の値Ｂ（Ｉ）を計算するステップと、
を含むことを特徴とする請求項３０に記載の方法。
【請求項３４】
前記複数のグループを生成するステップは、複数のソースの各々において、独立した複数個のグループを生成するステップであり、該複数個のグループは、前記複数のソースの各々のソースが、該複数のソースの他のソースで生成されたキーの参照を必要とすること無く、ある独立ソースが他のソースの情報出力と完全に重複すること無く、キーを選択するという点で独立であることを特徴とする請求項３３に記載の方法。
【請求項３５】
さらに、
あるチャネルにより提供される情報コンテンツが、他のチャネルの情報コンテンツと完全に重複すること無く、複数のチャネルにわたって複数のグループを配信するステップを含むことを特徴とする請求項３３に記載の方法。
【請求項３６】
複数のチャネルを介してソースからデータを送信する方法であって、該方法は、
ａ）順列の入力記号のセットとして送信されるデータを編成するステップと、
ｂ）複数のチャネルの各々に対して出力記号の複数のグループを生成するステップであって、該複数のグループの各々のグループは、
１）キーアルファベットから、生成されるグループに対するキーＩを選択するステップであって、該キーアルファベットは入力記号の順列のセットに含まれる入力記号の個数よりも多い個数を含む、ステップと、
２）Ｉの関数として重みＷ（Ｉ）を決定するステップであって、重みＷ（Ｉ）は、少なくとも２つの値の間で変化する正の整数であり、前記キーアルファベットにおける少なくともいくつかのキーに関して重みＷ（Ｉ）の値が、１より大きい、ステップと、
３）Ｉの関数に従って入力記号のＷ（Ｉ）を選択するステップであって、それにより前記グループに関連付けられたＷ（Ｉ）個の入力記号のリストＡＬ（Ｉ）を形成する、ステップと、
４）前記グループの出力記号の個数Ｇ（Ｉ）を決定するステップと、
５）関連付けられたＷ（Ｉ）個の入力記号を変数とする所定の値関数から、前記グループのＧ（Ｉ）個の出力記号の各々の値Ｂ（Ｉ）を計算するステップと、
により生成される、ステップと、
ｃ）前記複数のグループの少なくとも１つを複数のパケットの各々にパケット化するステップと、

ｄ）前記複数のチャネルにわたって前記複数のパケットを送信するステップであって、該複数のチャネルの少なくとも２つは、キーＩに対して互いに異なる値として生成されたグループを含むパケットを運ぶ、ステップと、
を含むことを特徴とする方法。
【請求項３７】
前記ソースは、自ソースのキーＩに対する値を生成する単一のソースを含み、前記複数のチャネルの任意の２つから受信されるグループが完全な重複とはならないように、前記複数のチャネルにわたって結果のグループを配信する、ことを特徴とする請求項３６に記載の方法。
【請求項３８】
前記複数のチャネルの任意の２つから受信されるグループが完全な重複とはならないように、キーＩに対する前記値は、前記複数のチャネルの各々に対して独立に選択されることを特徴とする請求項３６に記載の方法。
【発明の詳細な説明】
（関連出願への相互参照）
本出願は、発明の名称が「ＩＮＦＯＲＭＡＴＩＯＮ ＡＤＤＩＴＩＶＥ ＣＯＤＥ ＧＥＮＥＲＡＴＯＲ ＡＮＤ ＤＥＣＯＤＥＲＦＯＲ ＣＯＭＭＵＮＩＣＡＴＩＯＮ ＳＹＳＴＥＭＳ」で１９９９年２月５日に出願された米国特許出願第０９／２４６，０１５号（以下、「ＬｕｂｙＩ」と称す）の一部継続である米国特許出願第０９／３９９，２０１号に基づく優先権を主張する。米国特許出願第０９／３９９，２０１号はまた、発明の名称が「ＥＮＣＯＤＩＮＧ ＡＮＤ ＤＥＣＯＤＩＮＧ ＤＡＴＡ ＩＮ ＣＯＭＭＵＮＩＣＡＴＩＯＮ ＳＹＳＴＥＭＳ」であり１９９８年９月２３日に出願された同時係属中の米国仮特許出願第６０／１０１，４７３号（以下、「仮Ｌｕｂｙ」と称す）に基づく優先権を主張する。本出願は、出力記号表現のベースとしてのリードソロモン符号などの群符号の使用をＬｕｂｙＩに示し得たものよりも詳細に例示する。
【０００１】
（発明の背景）
本発明は、通信システムにおけるデータの符号化および復号化に関し、より詳細には、通信データにおける誤りおよびギャップを分析し、１つより多くのソースから生じる通信データを効率的に利用するためのデータを符号化および復号化する通信システムに関する。
【０００２】
送信者と受信者との間で通信チャネルを介したファイルの送信は、多くの文献に取り上げられてきた。好ましくは、受信者は、あるレベルの確実度で送信者によってチャネルを介して送信されたデータの正確なコピーを受信することを望む。チャネルが完全な忠実度を有さない場合（ほとんどすべての物理的に実現可能なシステムに当てはまる）の１つの問題点は、送信中に損失したりまたは化けたデータをどのように扱うかである。損失データ（イレーズ（ｅｒａｓｕｒｅ））は、化けたデータ（誤り）よりも扱いやすいことが多い。なぜなら、受信者は、いつ化けたデータが誤ってデータ受信されたのかを必ずしも当てることができないからである。イレーズを補正するため（いわゆる「イレーズ符号」）および／または誤りを補正するため（「誤り補正符号」または「ＥＣＣ」）の多くの誤り補正符合が開発されてきた。一般に、データの送信中のチャネルおよびその送信されるデータの性質の不忠実度についての何らかの情報に基づいて、使用される特定の符号が選択される。例えば、チャネルが長時間の不忠実度を有することが既知の場合、そのようなアプリケーションには、バースト誤り符号が最も適切であり得る。短くまれな誤りだけが予想される場合、簡単なパリティ符号が最善であり得る。
【０００３】
複数の送信者および／または複数の受信器との間の通信チャネルを介したファイル送信もまた、多くの文献に取り上げられてきた。一般に、複数の送信者からのファイル送信は、送信者が作業の重複を最小限にできるように複数の送信者の間で調整が必要である。１つのファイルを受信器に送信する一般の複数送信者システムにおいて、送信者がどのデータ

をいつ送信するかを調整せずにその代わりにファイルのセグメントをただ送るだけの場合は、受信器が無駄な重複セグメントを多く受信する可能性がある。同様に、一人の送信者の送信に異なる受信器が異なる時点で参加する場合の問題点は、どのようにすればその送信者から受信器が受信するすべてのデータが確実に有用となるかである。例えば、送信者が連続して同じファイルについてのデータを送信しているとする。送信者が元のファイルのセグメントを送信し、いくつかのセグメントが損失した場合、受信器は、そのファイルの各セグメントの１コピーを受信する前に多くの無駄な重複セグメントを受信し得る。

【０００４】

符号を選択する際に別に考慮することは、送信に使用するプロトコルである。「インターネット」（大文字の「Ｉ」を使用）として知られるネットワークのグローバルなインターネットワークの場合、パケットプロトコルがデータ伝送のために使用される。そのプロトコルは、インターネットプロトコル、または略して「ＩＰ」と呼ばれる。１ファイルまたはデータの他のブロックがＩＰネットワークを介して送信される場合、そのデータは、等しいサイズの入力記号に分割され、入力記号は連続パケットに配置される。パケットベースであるので、パケット指向符号化スキームが適切であり得る。入力記号の「サイズ」は、入力記号が実際にビットストリームに分割されるどうかに関わらず、ビット単位で測られ得る。この場合、入力記号が2^M記号のアルファベットから選択されるとすると、入力記号の大きさは、Ｍビットのサイズを有する。

【０００５】

伝送制御プロトコル（「ＴＣＰ」）は、確認応答機構を有する常用のポイントツーポイントパケット制御スキームである。ＴＣＰは、一対一通信に適切である。ここで、送信者および受信者の双方は、いつ送信および受信が生じるかについての合意があり、およびどの送信器および受信器が使用されるかの合意がある。しかし、ＴＣＰは、一対多または多対多通信または送信者および受信者が独立してデータの送信または受信の時間と場所を決定する場合に対しては、適切でないことがよくある。

【０００６】

ＴＣＰを使用する場合、送信者は整列したパケットを送信し、受信者は各パケットの受信の確認応答を行う。パケットが損失すると、送信者に確認応答が送られないので、送信者はパケットを再送信する。パケット損失は、多くの原因による。インターネット上では、パケット損失が頻繁に起こる。なぜなら、散発的な混雑状態によってルータにおけるバッファリング機構がその容量の最大に達し、入力パケットの損失を起こすからである。ＴＣＰ／ＩＰなどのプロトコル使用すると、確認応答パラダイムにより、確認応答の欠如に応答して、または受信者からの明示的要求に応答して損失パケットが単に再送信され得るので、パケット送信は完全には失敗しない。いずれの場合も、確認応答プロトコルは、受信者から送信者の逆向きチャネルを必要とする。

【０００７】

確認応答ベースプロトコルは、一般に多くのアプリケーションに適切であり、実際に現在のインターネットにおいて広く使用されているが、あるアプリケーションに対しては、それらは非効率的であり、完全に実行不可能であることもある。特に、確認応答ベースプロトコルは、待ち時間が長く、パケット損失率が高く、調整されていない受信者の接続および切断があり、および／または帯域の非対称性が高いようなネットワーク中では、十分に機能しない。長い待ち時間とは、確認応答が受信者から伝送して送信者に戻るまでに要する時間が長いことである。待ち時間が長いと、再送信前の総時間が過度に長くなり得る。パケット損失率が高くなるとまた、同じパケットのうちのそれぞれの再送信が到達せずに最後の１つまたは最後の数個の不運なパケットを得ることが大きく遅れるという問題を起こす。

【０００８】

「調整されていない受信者の接続および切断」は、各受信者が進行中の送信セッションに接続および切断を自由裁量でできる状況のことである。このような状況は、インターネット、「ビデオ・オン・デマンド」などの次世代サービスおよび将来のネットワークプロバ

イダによって提供される他のサービス上では一般的なことである。一般のシステムにおいて、受信者は、送信者の調整なしに進行中の送信に接続および切断する場合、受信者が多数のパケットの損失を認識する可能性がある。この場合、異なる受信者によって広く異なる損失パターンが認識される。

【０００９】

非対称性帯域は、受信者から送信者への逆向きデータパス（逆向きチャネル）が利用できないか、または前向きパスよりもさらにコストがかかる状況のことである。非対称性帯域の場合、受信者がパケットに頻繁に確認応答するのを過度に遅くさせ、および／または過度にコストがかかる。また、確認応答を頻繁にしなければやはり遅延が生じる。

【００１０】

さらに、確認応答ベースプロトコルは、放送にまで規模を十分に拡大することができない。放送では、１送信者が複数のユーザに同時にファイルを送信する。例えば、１送信者が複数の受信者に衛星チャネルを介してファイルを放送すると想定する。各受信者は、異なるパターンでパケットが損失し得る。ファイルの確実な送達のために確認応答データ（肯定または否定のいずれか）に基づくプロトコルは、各受信者から送信者への逆向きチャネルを必要とし、それを提供するのは過度にコストがかかり得る。さらに、このような場合には、複雑で強力な送信者が受信者から送信された確認応答データのすべてを適切に処理することができる必要がある。別の欠点は、異なる受信者が異なるパケットのセットを損失する場合に、ほんの少数の受信者によって受信し損ねたパケットのみを再放送することにより、他の受信者が無駄な重複パケットを受信してしまうことである。確認応答ベース通信システムにおいて十分に扱えない別の状況は、受信者が非同期的に受信セッションを開始し得る場合であり、つまり、受信者が送信セッションの途中でデータ受信を開始する可能性のある場合である。

【００１１】

マルチキャストおよび放送のためのＴＣＰ／ＩＰなどの確認応答ベーススキームの性能を向上させるための複雑なスキームがいくつか提案されてきた。しかし、現在のところ種々の理由によりいずれも完全には採用されていない。ひとつには、低地球軌道（ＬＥＯ）衛星放送ネットワークなどのような１受信者が複数の送信者から情報を得るような場合にまで確認応答ベースプロトコルの規模を十分に拡大することができないことである。ＬＥＯネットワークにおいては、ＬＥＯ衛星はその軌道のために上空を速い速度で通過するので、受信者は、いずれの特定の衛星の視野にはほんの短時間しか存在しない。これを補うために、ＬＥＯネットワークは多くの衛星を含み、１つの人工衛生が地平線に沈み別の衛星が地平線を昇る際に受信者が衛星間で引き渡される。確認応答ベースプロトコルが確実性を確保するために使用されるとしても、確認応答が適切な衛星から戻るように調整するためには、おそらく複雑な引き渡しプロトコルが必要であろう。なぜなら、受信者が１つの衛星からパケットを受信しつつなお別の衛星へそのパケットの確認応答を送ることがよく起こり得るからである。

【００１２】

確認応答ベースプロトコルを代替する、実用されることのあるプロトコルは、カルーセル（ｃａｒｏｕｓｅｌ）ベースプロトコルである。カルーセルベースプロトコルは、入力ファイルを等しい長さの入力記号に分割し、各入力記号をパケットに配置し、そして連続的にサイクルを繰り返し、すべてのパケットを送信する。カルーセルベースプロトコルの主な欠点は、受信者がたった１つのパケットを受信し損ねても、受信者がさらに完全な１回のサイクルを待たなければその受信し損ねたパケットを受信する機会を得られないことである。別の観点では、カルーセルベースプロトコルのために大量の無駄な重複データの受信を生じ得ることである。例えば、受信者がカルーセルの開始からパケットを受信し、しばらく受信を停止し、カルーセルの開始時に受信を再開する場合、多くの無駄な重複パケットが受信される。

【００１３】

上記問題を解決するために提案されてきた１つの解決法は、確認応答ベースプロトコルの

使用を避け、その代わりに確実性を上げるためにリードソロモン符号などのイレーズ符号を使用することである。いくつかのイレーズ符号の１つの特徴は、ファイルが入力記号にセグメント化され、その入力記号がパケットで受信者に送信される場合、受信者は、十分に多くのパケットを受信したら、一般にどのパケットが到着したかにかかわらず、そのパケットを復号して完全なファイルを再構築する。このような性質は、たとえパケットが損失してもファイルが回復され得るので、パケットレベルでの確認応答の必要性を排除する。しかし、イレーズ符号による多くの解決方法は、確認応答ベースプロトコルの問題を解決しないか、さもなくば新規の問題を生じる。

【００１４】

多数のイレーズ符号に伴う１つの問題は、それらを処理するために過剰な計算能力またはメモリを必要とすることである。計算能力およびメモリを多少効率的に使用する通信アプリケーション用に最近開発された１つの符号化スキームは、トルネード符号化スキームである。トルネード符号は、入力ファイルがＫ個の入力記号によって表されＮ個の出力記号を決定するために使用される（ここでＮは、符号化処理を開始する前に固定される）という点で、リードソロモン符号と同様である。トルネード符号を使用する符号化は、一般にリードソロモン符号を使用する符号化よりもずっと速い。なぜなら、Ｎ個のトルネード出力記号を生成するのに必要な平均算術演算数がＮに比例し（数十アセンブリ符号演算かけるＮのオーダーである）、ファイル全体を復号化するのに必要な算術演算総数がまたＮに比例するからである。

【００１５】

トルネード符号は、リードソロモン符号よりも速度の点で優れているが、いくつかの欠点がある。まず、出力記号数Ｎが符号化処理の前に決定されなければならない。このため、パケットの損失率が過大に見積もられると効率が低下し、パケットの損失率が過小に見積もられると失敗する。これは、トルネードデコーダが、元のファイルを復号して復元するために所定数の出力記号（具体的には、Ｋ＋Ａ個の出力記号、ここでＡはＫに比較して小さい）を必要とし、損失出力記号数がＮ－（Ｋ＋Ａ）よりも大きい場合、元のファイルが復元されないからである。この制限は、一般にＮがＫ＋Ａよりも少なくとも実際のパケット損失分は大きくなるように選択される限り、多くの通信の問題に対して許容されるが、パケット損失をあらかじめ予測する必要がある。

【００１６】

トルネード符号の別の欠点は、トルネード符号がグラフ構造上で何らかの方法で整合するためにエンコーダおよびデコーダを必要とすることである。トルネード符号は、このグラフが構築されるデコーダで前処理段階を必要とする。その処理は、実質的に復号化を遅らす。さらに、グラフはファイルサイズに固有であるので、使用される各ファイルサイズに対する新しいグラフを生成する必要がある。さらに、トルネード符号によって必要なグラフの構築は複雑であり、最高の性能を得るためには、異なるサイズのファイルに対するパラメータの異なるカスタム設定をする必要がある。これらのグラフは、大きなサイズを有し、送信者および受信者の双方の記憶装置に大容量のメモリを要求する。

【００１７】

加えて、トルネード符号は、固定されたグラフおよび入力ファイルに対してちょうど同じ出力記号値を生成する。これらの出力記号は、Ｋ個の元入力記号およびＮ－Ｋ個の冗長記号を含む。さらに実用上、Ｎは単にＫの小さな倍数である（Ｋの１．５または２倍など）。このように、１より多い送信者から、同じグラフを使用して同じ入力ファイルから生成された出力記号を得る受信者が、多数の無駄な重複出力記号を受信する可能性は非常に高い。なぜなら、Ｎ個の出力記号があらかじめ固定されており、そのＮ個の出力記号は、出力記号が送信されるごとの各送信器から送信されるＮ個の出力記号と同じであり、かつ受信者によって受信されるＮ個の記号と同じである。例えば、Ｎ＝１５００、Ｋ＝１０００、および受信者が、１つの衛星が地平線へ沈む前にその衛星から９００個の記号を受信すると想定する。衛星が同期するように調整されなければ、次の衛星から受信器によって受信されたトルネード符号記号は、追加のものではない。なぜなら、その次の衛星は、同じ

Ｎ個の記号を送信しているからである。この場合、受信器は、入力ファイルの復元に必要な１００個の新しい記号を受信する前にすでに受信した９００個の記号のうちの多く記号のコピーを受信してしまう可能性がある。

【００１８】

したがって、必要なことは、送信者または受信者に過剰な計算能力またはメモリを設置する必要のない、かつ必ずしも１以上の送信者と１以上の受信者との間の調整を必要とせずにその送信者および／またはその受信者によってシステム内にファイルを効率的に配布するために使用され得る簡単なイレーズ符号である。

【００１９】

（発明の要旨）

本発明は以下を提供する。

【００２０】

１．出力記号の１群を生成する方法であって、１群は１つ以上の出力記号であり、各出力記号は出力アルファベットから選択され、入力アルファベットからそれぞれ選択される順列の複数の入力記号を含む入力ファイルが、このような群の１セットから回復可能であるような該群であり、該方法は、該群のキーＩを得るステップであって、該キーがキーアルファベットから選択され、該キーアルファベットにおける可能なキーの数が該入力ファイルにおける入力記号の数よりもずっと大きい、ステップと、Ｉの所定の関数にしたがって、該群のリストＡＬ（Ｉ）を計算するステップであって、ＡＬ（Ｉ）は、該群に関連付けられたＷ（Ｉ）個の関連付けられた入力記号を示し、重みＷは、少なくとも２つの値の間で変化する正の整数であり、Ｉの少なくとも１つの値に対して１より大きい、ステップと、ＡＬ（Ｉ）によって示される該関連付けられた入力記号の所定の関数から、該群における各出力記号の出力記号値を生成するステップと、を含む方法。

【００２１】

２．前記キーＩを得るステップが、ランダム関数または擬似ランダム関数にしたがってキーＩを計算するステップを含む、項目１に記載の方法。

【００２２】

３．前記計算するステップが、Ｉのランダム関数または擬似ランダム関数にしたがってＷ（Ｉ）を計算するステップを含む、項目１に記載の方法。

【００２３】

４．前記計算するステップが、Ｉのランダム関数または擬似ランダム関数にしたがってＡＬ（Ｉ）を計算するステップを含む、項目１に記載の方法。

【００２４】

５．前記計算するステップが、前記群における出力記号の数Ｇ（Ｉ）を計算するステップを含み、該数Ｇ（Ｉ）が正の整数であり、該数Ｇ（Ｉ）がＩのランダム関数または擬似ランダム関数にしたがって計算される、項目１に記載の方法。

【００２５】

６．前記計算するステップが、Ｉの所定の関数および確率分布にしたがって、重みＷ（Ｉ）を計算するステップであって、該確率分布が少なくとも２つの正の整数上にあり、そのうちの少なくとも１つが１より大きい、ステップと、リストＡＬ（Ｉ）のリストエントリを計算するステップと、Ｗ（Ｉ）個のリストエントリが計算されるまで、リストＡＬ（Ｉ）のリストエントリを計算するステップを繰り返すステップと、を含む、項目１に記載の方法。

【００２６】

７．前記Ｗ（Ｉ）を計算するステップが、Ｗが前記キーアルファベット上の所定の分布を近似するようなＷ（Ｉ）を決定するステップを含む、項目６に記載の方法。

【００２７】

８．前記所定の分布が一様分布である、項目７に記載の方法。

【００２８】

９．前記所定の分布がつり鐘曲線分布である、項目７に記載の方法。

【００２９】
１０．前記所定の分布は、Ｗ＝１が１／Ｋの確率を有し、ここで、Ｋが前記入力ファイルにおける入力記号の数であり、そしてＷ＝ｉがｉ＝２，．．．，Ｋに対して１／ｉ（ｉ－１）の確率を有する、項目７に記載の方法。
【００３０】
１１．前記ＡＬ（Ｉ）によって示される関連付けられた入力記号の前記所定の関数が、リードーソロモン符号によって決定される、項目１に記載の方法。
【００３１】
１２．前記入力アルファベットおよび前記出力アルファベットが同じアルファベットである、項目１に記載の方法。
【００３２】
１３．前記入力アルファベットが2^{Mi}個の記号を含み、各入力記号がＭｉビットを符号化し、前記出力アルファベットが2^{Mo}個の記号を含み、各出力記号の群がＭをビットを符号化する、項目１に記載の方法。
【００３３】
１４．各後続キーＩがその先行キーよりも１大きい、項目１に記載の方法。
【００３４】
１５．出力記号の複数の群を符号化する方法であって、各群は項目１に記載され、該方法は、生成されるべき該出力記号の群の各々に対するキーＩを生成するステップと、および該生成された出力記号の群の各々をデータイレーズチャネルを介して送信されるべき出力シーケンスとして出力するステップと、をさらに含む方法。
【００３５】
１６．各キーＩが他の選択されるキーとは独立に選択される、項目１５に記載の方法。
【００３６】
１７．前記ＡＬ（Ｉ）を計算するステップが、前記入力ファイルにおける入力記号の数Ｋを少なくとも近似的に識別し、かつ重みＷ（Ｉ）を識別するステップと、Ｋより大きいかまたはＫに等しい最小素数Ｐを決定するステップと、ＰがＫより大きい場合、Ｐ－Ｋ個のパディング入力記号で該入力ファイルを少なくとも論理的にパディングするステップと、１≦Ｘ＜Ｐである第１の整数Ｘ、および０≦Ｙ＜Ｐである第２の整数Ｙを生成するステップと、１～Ｗ（Ｉ）の各Ｊに対して、ＡＬ（Ｉ）における第Ｊ番目のエントリを（（Ｙ＋（Ｊ－１）・Ｘ）　ｍｏｄ　Ｐ）に設定するステップと、
を含む、項目１に記載の方法。
【００３７】
１８．前記各Ｊに対してＡＬ（Ｉ）における第Ｊ番目のエントリを設定するステップが、配列Ｖにおける第１のエントリＶ［Ｊ＝０］をＹに設定するステップと、１～Ｗ（Ｊ）－１の各Ｊに対して、該配列Ｖにおける第Ｊ番目のエントリＶ［Ｊ］を（（Ｖ［Ｊ－１］＋Ｘ）　ｍｏｄ　Ｐ）に設定するステップと、該配列Ｖを該リストＡＬ（Ｉ）として使用するステップと、を含む、項目１７に記載の方法。
【００３８】
１９．ソースから目的地にパケット通信チャネルを介してデータを送信する方法であって、該方法が、ａ）送信されるべき該データを入力記号の順列セットとして構成するステップであって、各入力記号は、入力アルファベットから選択され、該データにおける位置を有する、ステップと、ｂ）出力記号の複数の群を生成するステップであって、各出力記号が出力アルファベットから選択され、該複数の群の各群が、以下のステップによって生成され、該ステップは、キーアルファベットから、生成される該群のキーＩを選択するステップと、Ｉの関数として重みＷ（Ｉ）を決定するステップであって、ここで、重みＷは、少なくとも２つの値の間および該キーアルファベット上で変化する正の整数であり、該キーアルファベットにおける少なくともいくつかのキーに対しては０より大きい、ステップと、Ｉの関数にしたがってＷ（Ｉ）個の該入力記号を選択し、したがって、該群に関連付けられるＷ（Ｉ）個の入力記号のリストＡＬ（Ｉ）を形成するステップと、該群における

出力記号の数Ｇ（Ｉ）を決定するステップと、該関連付けられるＷ（Ｉ）個の入力記号の所定の値関数から該群における各出力記号の値Ｂ（Ｉ）を計算するステップとを含む、ステップと、ｃ）該複数の群のうちの少なくとも１つの出力記号の群のうちの少なくとも１つを、複数のパケットの各々にパケット化するステップと、ｄ）該複数のパケットを該パケット通信チャネルを介して送信するステップと、ｅ）該複数のパケットのうちの少なくともいくつかを該目的地で受信するステップと、ｆ）該複数の受信されたパケットから該データを復号化するステップと、を含む方法。

【００３９】

２０．前記データを復号化するステップが、１）受信された出力記号の各群を処理するステップであって、ａ）該群の前記キーＩを決定するステップと、ｂ）該群の前記重みＷ（Ｉ）を決定するステップと、ｃ）該群の前記Ｗ（Ｉ）個の関連付けられた入力記号を決定するステップとを含む、ステップと、２）任意の入力記号を復号化するのに十分な情報が受信されたかどうかを決定するステップと、３）該受信された情報から復号化され得る入力記号を復号化するステップと、を含む、項目１９に記載の方法。

【００４０】

２１．前記キーＩを決定するステップが、前記パケット通信チャネルを介して受信されたパケットにおいて供給されたデータから該キーＩを少なくとも部分的に決定するステップを含む、項目２０に記載の方法。

【００４１】

２２．前記データを復号化するステップが、１）受信された出力記号の各群を処理するステップであって、ａ）該受信された群の前記重みＷ（Ｉ）を決定するステップと、ｂ）該受信された群の前記Ｗ（Ｉ）個の関連付けられた入力記号を決定するステップと、ｃ）出力記号の群のテーブルにおける群の出力記号の値Ｂ（Ｉ）を、該受信された群に対する該重みＷ（Ｉ）および前記Ｗ（Ｉ）個の関連付けの位置とともに格納するステップとを含む、ステップと、２）出力記号のさらなる群を受信し、それらをステップ１）およびそのサブステップにしたがって処理するステップと、３）たかだか群サイズの重みを有し、使い古された出力記号の群として表記されていない出力記号の各群、ＧＯＳ１、に対して、以下のステップ、ａ）ＧＯＳ１に対応する入力記号の位置に対して入力記号を計算するステップと、ｂ）該出力記号の群のテーブルにおける接続された出力記号の群を識別するステップであって、接続された出力記号の群は、ステップ３）ａ）において処理された少なくとも１つの入力記号の関数である出力記号の群である、ステップと、ｃ）ステップ３）ａ）において処理された該入力記号に独立になるように該接続された出力記号の群を再計算するステップと、ｄ）ステップ３）ａ）において処理された該入力記号の独立性を反映するようにステップ３）ｃ）において再計算された該出力記号の群の重みを決定するステップと、ｅ）ＧＯＳ１を使い古した出力記号として表記するステップと、を実行するステップと、４）前記入力記号の順列セットが前記目的地で回復されるまで、該ステップ１）からステップ３）を繰り返すステップと、を含む、項目１９に記載の方法。

【００４２】

２３．前記表記するステップが、前記使い古した出力記号にゼロの重みを割り当てるステップである、項目２２に記載の方法。

【００４３】

２４．前記表記するステップが、前記出力記号の群のテーブルから前記使い古した出力記号の群を除去するステップを含む、項目２２に記載の方法。

【００４４】

２５．前記パケット化するステップが、複数の出力記号の群のうちの少なくとも１つの群から少なくとも１つの出力記号の群を各パケットにパケット化するステップであり、該方法が、パケット内の出力記号の１群の位置を該出力記号の群のキーの一部として使用するステップをさらに含む、項目１９に記載の方法。

【００４５】

２６．前記計算するステップが、前記群における出力記号の数Ｇ（Ｉ）を計算するステッ

プを含み、該Ｇ（Ｉ）は、正の整数であり、Ｉのすべての値に対して同じ正の整数である、項目１に記載の方法。

【００４６】

本発明による通信システムの１つの実施形態において、エンコーダは、入力ファイルのデータおよびキーを使用して、出力記号の１群を生成する。ここで、入力ファイルは、入力アルファベットからそれぞれ選択される順列の複数の入力記号であり、キーは、キーアルファベットから選択され、群サイズは、正の整数として選択され、各出力記号は、出力アルファベットから選択される。キーＩを有する出力記号の１群は、生成されるべき出力記号の群の群サイズＧ（Ｉ）を決定するステップ（ここで、群サイズＧは、複数のキー上の少なくとも２つの値の間で変化し得る正の整数である）、生成されるべき出力記号の群の重みＷ（Ｉ）を決定するステップ（ここで、重みＷは、複数のキー上の少なくとも２つの値の間で変化する正の整数である）、Ｉの関数にしたがって、出力記号の群に関連付けられるＷ（Ｉ）個の入力記号を選択するステップ、および選択されたＷ（Ｉ）個の入力記号の所定の値関数Ｆ（Ｉ）からＧ（Ｉ）個の出力記号値の群Ｂ（Ｉ）＝Ｂ＿１（Ｉ），．．．，Ｂ＿Ｇ（Ｉ）（Ｉ）を生成するステップによって生成される。エンコーダは、１回以上呼び出され得、各回別のキーを用い、各回出力記号の１群を生成する。出力記号の群は、一般に互いに独立であり、無限の数（Ｉの解像度に従う）が、必要ならば生成され得る。

【００４７】

いくつかの実施形態において、１群における出力記号数Ｇ（Ｉ）は、キーＩに依存して群ごとに異なる。他の実施形態においては、Ｇ（Ｉ）は、すべてのキーＩの出力記号のすべての群に対して、何らかの正の整数ｂに固定される。

【００４８】

本発明によるデコーダにおいて、受信者によって受信された出力記号は、入力ファイルの符号化に基づいてこれらの出力記号を群単位で生成した送信者から送信された出力記号である。出力記号は送信中に損失され得るが、デコーダは、送信された出力記号の任意の部分を受信するだけの場合でさえ適切に動作する。入力ファイルを復号化するために必要な出力記号数は、そのファイルを構成する入力記号の数に等しいか、またはわずかに大きい。ここで、入力記号および出力記号は、同じビット数のデータと仮定する。

【００４９】

本発明による１つの復号化プロセスにおいて、出力記号の各受信された群に対して以下のステップが実行される：１）受信されたｂ個の出力記号からなる群のキーＩを識別するステップ；２）群サイズＧ（Ｉ）を識別するステップ；３）出力記号の群に対して、受信された出力記号値の群Ｂ（Ｉ）＝Ｂ＿１（Ｉ），．．．，Ｂ＿Ｇ（Ｉ）（Ｉ）を識別するステップ；３）出力記号の群の重みＷ（Ｉ）を決定するステップ；４）出力記号の群に関連付けられたＷ（Ｉ）個の関連付けられた入力記号の位置を決定するステップ；および５）出力記号の群を保持するテーブルにＢ（Ｉ）＝Ｂ＿１（Ｉ），．．．，Ｂ＿Ｇ（Ｉ）（Ｉ）を格納し、重みＷ（Ｉ）および関連付けられた入力記号の位置を格納するステップ。次に、以下のプロセスが、群の重みがたかだかＧ（Ｉ）であるように、キーＩを有する出力記号の未使用群がもはや存在しなくなるまで繰り返し適用される：１）ｂ’≦Ｇ（Ｉ）の重みを有し、出力記号の「使い古した」群として表記されない、キーＩを有する出力記号の各格納された群に対して、キーＩに基づいて、出力記号の群に関連付けられる固有の残りｂ’個の未回復入力記号の位置Ｊ＿１，．．．，Ｊ＿ｂ’を計算するステップ；２）出力記号の群から入力記号Ｊ＿１，．．．，Ｊ＿ｂ’に対する未知の値、およびその群に関連付けられた他の入力記号の既知の値を計算するステップ；３）入力記号Ｊ＿１，．．．，Ｊ＿ｂ’のうちの少なくとも１つを関連付けとして有する出力記号の群のテーブルにおいて出力記号の群を識別するステップ；４）Ｊ＿１，．．．，Ｊ＿ｂ’の中にある各関連付けに対して、これら識別された出力記号の群の各々の重みを１だけデクリメントするステップ；および５）入力記号Ｊ＿１，．．．，Ｊ＿ｂ’を回復されたものとして表記し、キーＩを有する出力記号の群を使い古したものとして表記するステップ。このプロセスは

、入力記号の順列セットが回復されるまで、すなわち、入力ファイル全体が完全に回復されるまで繰り返される。

【００５０】

本発明の１つの利点は、送信中に受信者が任意の所定の時点で受信を開始する必要がないこと、および出力記号の群のセット数が生成された後、送信者が停止する必要がないことである。なぜなら、送信者が任意の所定の入力ファイルに対して出力記号の群の有効な無限のセットを送信し得るためである。その代りに、受信者は、用意ができた場合に受信できるところから受信を開始し得、出力記号の群をランダムパターンでまたは任意のパターンでさえ含むパケットを損失し得、そしてなお、受信されるデータの非常に多くが「情報付加的」データ（すなわち、すでに利用可能な情報の重複でなく、回復プロセスを補助するデータ）である良好な確率を有する。独立に生成された（ランダムに無関係であることが多い）データストリームが、情報付加的な方法で符号化されることによって、複数ソース生成および受信、イレーズロバスト性、ならびに未調整の接続および切断を可能にするという多くの利点が生じる。

【００５１】

本明細書中で開示される本発明の性質および利点のさらなる理解は、本明細書の残りの部分および添付の図面を参照して理解され得る。

【００５２】

（好適な実施形態の説明）

本明細書中において記載される実施例において、「群連鎖反応符号化」と呼ばれる符号化スキームが説明される。そのまえに、本記載中に使用される種々の用語の意味および範囲を説明する。

【００５３】

群連鎖反応符号化を用いて、出力記号の群が必要に応じて入力ファイルから送信者によって生成される。出力記号の各群は、出力記号の他の群がどのように生成されるかにかかわらず生成され得る。どの時点においても、送信者は、出力記号の群生成工程を停止し得る。送信者が出力記号の群生成工程をいつ停止または再開するかについては、制約を必要としない。一旦生成された後、出力記号のこれらの群は、パケット内に配置され、送信先へ送信される。この場合、各パケットは、出力記号の群を１つ以上含む。群連鎖反応符号化の具体的な例では、各群は、１つの出力記号を含み、その結果、ＬｕｂｙＩに記載の連鎖反応符号化システムと同様の性能および動作を有するシステムを生じる。

【００５４】

本明細書中で使用されるように、用語「ファイル」は、１つ以上のソースに格納され、１つ以上の送信先へ１単位として送達される任意のデータのことである。したがって、ファイルサーバまたはコンピュータ記憶装置からのドキュメント、イメージ、およびファイルはすべて、送達され得る「ファイル」の例である。ファイルは、既知のサイズを有するか（ハードディスク上に格納された１メガバイトのイメージなど）、または未知のサイズを有する（ストリーミングソースの出力から取り出されるファイルなど）。いずれの場合も、ファイルは入力記号のシーケンスであり、各入力記号はファイル内の位置および値を有する。

【００５５】

送信は、ファイルを送達するためにデータを１以上の送信者から１以上の受信者にチャネルを介して送信する処理のことである。１送信者が完全なチャネルによって任意数の受信者に接続される場合、すべてのデータが正しく受信されるので受信されたデータが入力ファイルの正確なコピーであり得る。ここで、大半の現実のチャネルの場合のようにチャネルが完全でないと仮定するか、またはいくつかシステムの場合のように１より多い送信者からデータが送信されると仮定するか、またはいくつかシステムの場合のように送信されるデータの一部が途中で故意に脱落されると仮定する。多くのチャネルの欠点のうち、ここで関心のある２つの欠点は、データイレーズおよびデータ不完全性（データイレーズの特殊な場合として処理され得る）である。データイレーズが生じるのは、チャネルがデー

タを損失または脱落させる場合である。データ不完全性が生じるのは、データをうちのいくつかが受信者を通り過ぎてしまうまで受信者がデータの受信を開始しないか、受信者が送信終了前にデータの受信を停止するか、または受信者がデータ受信の停止および再開を断続的に行う場合である。

【００５６】

データ不完全性の例として、移動衛星送信者が、入力ファイルを表すデータを送信し、受信者が範囲に入る前に送信を開始する。受信者が一旦範囲に入ると、衛星が範囲を出るまでデータが受信され得る。この場合、受信者は、衛星放送受信アンテナの向きを変更して（この間はデータを受信しない）、範囲内に入った別の衛星によって送信されている同じ入力ファイルについてのデータの受信を開始し得る。この記載を読むことにより明らかなように、データ不完全性は、データイレーズの特殊な場合である。なぜなら、受信者は、あたかも受信者が常に範囲内に存在しているが、チャネルは受信者がデータ受信を開始する時点までのすべてのデータを損失するかのように、データ不完全性を処理し得る（および受信者は同じ問題を有する）。また、通信システム設計において周知のように、検出可能な誤りは、単に検出可能な誤りを有するデータブロックまたは記号のすべてを脱落させることによってイレーズと等価になり得る。

【００５７】

別の例としては、ルータは、そのバッファがフルまたはフルに近い（混雑状態）場合に故意にパケットを脱落させ、かつ競合するパケットに対して公平であるようにおよび／または強制的に速度制限をするように故意にパケットを脱落させる。

【００５８】

いくつかの通信システムにおいて、受信者は、複数の送信者または複数の接続を有する１送信者によって生成されるデータを受信する。例えば、ダウンロードの速度を上げるために、受信者は、同時に１より多い送信者に接続して同じファイルに関するデータを送信する。別の例として、マルチキャスト送信において、複数のマルチキャストデータストリームは、受信者がこれらデータストリームのその１つ以上に接続できるよう送信され、そのストリームが送信者に接続されるチャネルの帯域に総送信速度を整合させる。すべてのそのような場合において重要なのは、たとえ送信速度が異なるストリーム間で大きく異なる場合であっても、かつ任意の損失パターンがある場合でも、すべての送信データが受信者に対して独立に使用されること、すなわち、複数のソースデータがストリーム間で冗長でないことである。

【００５９】

一般に、送信とは、送信者から受信者へ送信者と受信者とを接続するチャネルを介してデータを移動する行為である。そのチャネルは、チャネルがデータを取得した際にチャネルがそのデータを送信者から受信者へ移動させる実時間チャネルであり得る。あるいは、チャネルは、送信者から受信者への送信中にデータの一部またはすべてを格納する記憶チャネルでもよい。後者の例は、ディスク記憶装置または他の記憶装置である。この例では、データを生成するプログラムまたはデバイスは、データを記憶装置に送信する送信者として考えられ得る。受信者は、記憶装置からデータを読み出すプログラムまたはデバイスである。送信者がデータを記憶装置へ書き込むために使用する機構、記憶装置自体、および受信者が記憶装置からデータを読み出すための機構が、集合的にチャネルを形成する。これらの機構または記憶装置がデータを損失する可能性がある場合、そのような損失をチャネル中のデータイレーズとして処理し得る。

【００６０】

送信者および受信者がデータイレーズチャネルによって分離される場合、入力ファイルの正確なコピーを送信するのではなくて、その代わりにイレーズの回復を補助する入力ファイルから生成されたデータを送信するのが好ましい。エンコーダとは、そのようなタスクを処理する回路、デバイス、モジュール、または符号セグメントである。エンコーダの動作の見る１つの方法は、エンコーダが入力記号から出力記号の群を生成することであり、ここで入力記号値のシーケンスが入力ファイルを表す。したがって、各入力記号は、入力

ファイル内の位置、および値を有する。デコーダとは、受信者によって受信された出力記号の群から入力記号を再構築する回路、デバイス、モジュール、または符号セグメントである。

【００６１】

群連鎖反応符号化は、任意の特定のタイプの入力記号に限定されないが、入力記号のタイプはアプリケーションによって決定されることが多い。一般に、入力記号に対する値は、2^M個の記号（ここでMは正の整数）のアルファベットから選択される。そのような場合、入力記号は、入力ファイルからのMビットのデータシーケンスによって表現され得る。Mの値は、アプリケーションの使用、チャネル、および群の最大サイズに基づいて決定されることが多い。例えば、パケットベースインターネットチャネルに対しては、１０２４バイトのサイズのペイロードを有するパケットが適切であり得る（１バイトは８ビット）。この例において、各パケットが出力記号の１群および８バイトの補助情報を含むと仮定し、かつすべての群が４個の出力記号を含むと仮定すると、入力記号サイズのMは、（１０２４－８）／４、すなわち２５４バイトが適切であり得る。別の例として、ＭＰＥＧパケット規格を使用するいくつかの衛星システムがある。この場合、各パケットのペイロードは１８８バイトを含む。この例において、各パケットは出力記号の１群および４バイトの補助情報を含むと仮定し、かつすべての群が２個の出力記号を有すると仮定すると、記号サイズのMは、（１８８－４）／２、すなわち９７バイトが適切であり得る。群連鎖反応符号化を使用する汎用通信システムにおいて、入力記号サイズ（すなわち、M、入力記号によって符号化されたビット数）などのアプリケーション特定パラメータは、アプリケーションによって設定された変数である。

【００６２】

出力記号の各群は、その群における出力記号の各々に対する値を有する。以下を考慮する１つの好ましい実施形態において、出力記号の各群は、「キー」と呼ばれる識別子を有する。好ましくは、出力記号の各群のキーは、受信者によって容易に決定され、受信者が出力記号のある群を出力記号の別の群と区別できるようにする。好ましくは、出力記号の群のキーは、出力記号の他のすべての群のキーと異なる。また好ましくは、受信者が受信された出力記号の群のキーを決定するためには、送信中に含まれるデータは可能な限り少ない方がよい。

【００６３】

キーイングの簡単な形態では、エンコーダによって生成された出力記号の連続した群に対するキーのシーケンスは、連続した整数のシーケンスである。この場合、各キーは、「シーケンス番号」と呼ばれる。各送信パケット中の出力記号値の１群がある場合、シーケンス番号がパケット中に含まれ得る。一般にシーケンス番号が少数のバイト（例えば、４バイト）に収まるので、いくつかのシステムにおいては、出力記号値の群とともにシーケンス番号を含むことは経済的である。例えば、各１０２４バイトのＵＤＰインターネットパケットを使用して、シーケンス番号用に各パケット内に４バイトを割当ててもわずかに０．４％の負担がかかるだけである。

【００６４】

他のシステムでは、１より多くのデータからキーを生成するのが好ましい。例えば、１以上の送信者からの同じ入力ファイルに基づいて生成された１より多くのデータストリームを受信する受信者を含むシステムを考える。ここで、送信データはパケットのストリームであり、各パケットは、出力記号の１群を含む。すべてのそのようなストリームが、キーと同じシーケンス番号のセットを使用する場合、受信者は、同じシーケンス番号を有する出力記号の群を受信する可能性がある。同じキーを有するか、または同じシーケンス番号を有する場合の出力記号の群は、入力ファイルについて同一の情報を含むので、受信者は、無駄な重複データを受信する。したがって、そのような場合、キーがシーケンス番号と対になった固有のストリーム識別子を含むことが好ましい。

【００６５】

例えば、ＵＤＰインターネットパケットの１ストリームに対して、データストリームの固

有の識別子は、送信者のＩＰアドレス、および送信者がパケットを送信するために使用しているポート数を含み得る。送信者のＩＰアドレスおよびストリームのポート数は、各ＵＤＰパケットのヘッダの一部であるので、キーのこれらの部分が受信者に確実に利用され得るように各パケット中に必要なさらなるスペースはない。送信者は、単にシーケンス番号を各パケットに対応の出力記号の群とともに挿入することのみを必要とし、受信者は、シーケンス番号およびパケットヘッダから受信された出力記号の１群のキー全体を再生成し得る。別の例として、ＩＰマルチキャストパケットの１ストリームに対して、データストリームの固有の識別子は、ＩＰマルチキャストアドレスを含み得る。ＩＰマルチキャストアドレスは各ＩＰマルチキャストパケットのヘッダの一部であるので、ＵＤＰパケットについての上記説明は、この場合についても同様に適用される。

【００６６】

出力記号の群の位置によるキーイングは、それが可能な場合は好ましい。位置キーイングは、ＣＤ－ＲＯＭ（コンパクトディスク読み取り専用メモリ）などの記憶装置から出力記号の群を読み出すために十分機能し得る。ここで、出力記号の１群のキーは、ＣＤ－ＲＯＭ上の位置（すなわち、トラック、プラスセクタ、セクタ内のプラス位置など）である。位置キーイングはまた、ＡＴＭ（非同期転送モード）システムなどの回路ベース送信システムのために十分機能し得る。ここで、整列したデータセルは、厳しいタイミング制約下で送信される。この形態のキーイングを使用すると、受信者は、出力記号の１群のキーを、そのキーを明確に送信するのに必要なスペースを有することなく、再生成し得る。当然、位置キーイングは、そのような位置情報が利用でき確実であることを必要とする。

【００６７】

位置によるキーイングはまた、他のキーイング方法と組み合わせ得る。例えば、各パケットが出力記号の１より多くの群を含むパケット送信システムを考える。この場合、出力記号の群のキーは、固有のストリーム識別子、シーケンス番号、および出力記号の群のパケット内の位置から構築され得る。データイレーズは一般に全パケットの損失を生じるので、受信者は一般にパケットを全部受信する。この場合、受信者は、パケットのヘッダ（固有のストリーム識別子を含む）、パケット中のシーケンス番号、および出力記号の群のパケット内の位置を再生成し得る。

【００６８】

いくつかのシステムにおいて好ましいキーイングの別の形態は、ランダムキーイングである。これらのシステムにおいては、乱数（疑似乱数）が、生成され、出力記号の各群に対するキーの少なくとも１部として使用され、および出力記号の群とともに明確に送信される。異なる物理的位置で異なる送信者によって生成されるキーに対してさえ（可能なキーの範囲が十分大きいと仮定する）、ランダムキーイングの１つの性質は、同じ値を有するキーの割合が小さい可能性がある。この形態のキーイングは、インプリメンテーションが簡単であるため、いくつかのシステムにおいては他の形態よりも利点を有し得る。

【００６９】

上記のように、群連鎖反応符号化が有用であるのは、データイレーズの可能性があるか、または受信者が、送信が開始および終了するちょうどその時点で受信を開始および終了しない場合である。後者の条件は、本明細書中で「データ不完全性」と呼ばれる。これらの条件は、群連鎖反応符号化が使用される場合、通信プロセスに悪影響を与えない。なぜなら、受信される群連鎖反応符号化が、情報追加型であるように、極めて独立しているからである。出力記号の群のほとんどの任意集合が十分に独立しているので、大部分が情報追加型である（これは、本明細書中に記載の群連鎖反応符号化システムに対する場合である）ならば、任意の適切な数のパケットを使用して、入力ファイルを回復し得る。１００パケットがデータイレーズを起こすノイズの発生によって損失する場合、その発生の後でさらに１００パケットを取り上げてイレーズされたパケットの損失を置き換え得る。送信器が送信を開始した際に受信器が送信器に同調できず何千のパケットが損失した場合、受信器は、他の送信期間から（あるいは、別の送信器からの場合もある）その何千のパケットを取り上げる。群連鎖反応符号化を使用すると、受信者は、任意の特定のパケットのセッ

トを取り上げることに制約されず、従って１送信器からいくつかのパケット受信し、別の送信器に切り換え、いくつかのパケットを損失し、所定の送信の開始または終了を逸しても、なお入力ファイルを回復し得る。受信器－送信器調整のない送信に接続および切断する機能は、通信プロセスを大きく簡略化する。

【００７０】

（基本的な実施）

群連鎖反応符号化を使用してファイルを送信するステップは、入力ファイルから入力記号を生成、形成、または抽出し、その入力記号を１以上の出力記号の群に符号化するステップ（ここで、出力記号の各群は、出力記号のすべての他の群とは独立なキーに基づいて生成される）、および出力記号の群を１以上の受信者にチャネルを介して送信するステップを含む。群連鎖反応符号化を使用する入力ファイルの１コピーを受信（および再構築）するステップは、１以上のデータストリームから出力記号の群のあるセットまたはサブセットを受信するステップ、および受信された出力記号の群の値およびキーから入力記号を復号化するステップを含む。

【００７１】

以下に説明するように、デコーダは、出力記号の１以上の群の値および可能ならすでに回復された他の入力記号の値についての情報から入力記号を回復し得る。したがって、デコーダは、出力記号のいくつかの群からいくつかの入力記号を回復し得る。これにより、デコーダは、その復号された入力記号およびすでに受信された出力記号の群などからの他の入力記号を復号し得、したがってこれにより、ファイルの入力記号の回復の「連鎖反応」が受信者側で再構築される。

【００７２】

次に、本発明の局面を図面を参照して説明する。

【００７３】

図１は、連鎖反応符号化を使用する通信システム１００のブロック図である。

通信システム１００において、入力ファイル１０１、または入力ストリーム１０５は、入力記号発生器１１０に与えられる。入力記号発生器１１０は、入力ファイルまたは入力ファイルストリームから１以上の入力記号（ＩＳ（０），ＩＳ（１），ＩＳ（２），．．．）のシーケンスを生成し、各入力記号は、値および位置を有する（図１に括弧付きの整数として示される）。上記のように、入力記号に対して可能な値、すなわち、そのアルファベットは、一般に2^M個の記号のアルファベットであるので、各入力記号は、Ｍビットの入力ファイルを符号化する。Ｍの値は、一般に通信システム１００を使用することによって決定されるが、汎用システムは、入力記号発生器１１０に入力される記号サイズを含み得るので、Ｍは使用ごとに異なり得る。入力記号発生器１１０の出力は、エンコーダ１１５に与えられる。

【００７４】

キー発生器１２０は、エンコーダ１１５によって生成される出力記号の各群に対するキーを生成する。各キーは、どのキー発生器によって生成されたかにかかわらず、上記の方法のうちの１つ、または同じ入力ファイルに対して生成されたキーの大きな割合が固有であることを確保する任意の同等の方法によって生成される。例えば、キー発生器１２０は、カウンタ１２５の出力、固有のストリーム識別子１３０、および／またはランダム発生器１３５の出力の組合せを使用して、各キーを生成し得る。キー発生器１２０の出力は、エンコーダ１１５に与えられる。

【００７５】

キー発生器１２０によって与えられた各キーＩに基づいて、エンコーダ１１５は、入力記号発生器によって与えられる入力記号から記号値Ｂ（Ｉ）のセットを有する１群の出力記号を生成する。出力記号の各群のｂ値は、そのキーおよび１以上の入力記号のある関数に基づいて生成される。本明細書中でｂ値を出力記号の群の「関連付けした入力記号」または単にその「関連付け」と呼ぶ。関数（「値関数」）および関連付けの選択は、以下により詳細に記載されるプロセスにしたがってなされる。一般に、常にではないが、Ｍは、入

力記号および出力記号に対して同じである、すなわち、入力記号は、出力記号と同じ長さである。

【００７６】

いくつかの実施形態において、入力記号数Ｋは、関連付けを選択するためにエンコーダによって使用される。Ｋがあらかじめ未知の場合は（入力がストリーミングファイルの場合など）、Ｋは単に推定され得る。値Ｋはまた、入力記号に対して記憶装置を割り当てるためにエンコーダ１１５によって使用され得る。

【００７７】

エンコーダ１１５は、出力記号の群を送信モジュール１４０に与える。送信モジュール１４０にはまた、キー発生器１２０から出力記号の各そのような群のキーが与えられる。送信モジュール１４０は、出力記号の群を送信し、送信モジュール１４０はまた、使用されるキーイング方法に依存して、出力記号の送信される群のキーについてのいくつかのデータをチャネル１４５を介して受信モジュール１５０に送信する。チャネル１４５は、イレーズチャネルと仮定されるが、これは、通信システム１００の適切な動作に必要ではない。モジュール１４０、１４５および１５０は、送信モジュール１４０が出力記号の群およびそれらのキーについて必要な任意のデータをチャネル１４５に送信するように改造され、受信モジュール１５０が出力記号の群および可能ならばそれらのキーについてのいくつかのデータをチャネル１４５から受信するように改造される限り、適切なハードウェアコンポーネント、ソフトウェアコンポーネント、物理的媒体、またはそれらの任意の組合せならいずれのものでもよい。Ｋの値が、関連付けを決定するために使用される場合、チャネル１４５を介して送信され得るか、またはエンコーダ１１５とデコーダ１５５との一致によってあらかじめ設定され得る。

【００７８】

上記のように、チャネル１４５は、インターネット、またはテレビジョン送信器からテレビジョン受信者への放送リンク、または１地点から別の地点への電話接続などの実時間チャネルであり得る。または、チャネル１４５は、ＣＤ－ＲＯＭ、ディスクドライブ、ウェッブサイトなどの記憶チャネルであり得る。チャネル１４５は、実時間チャネルと記憶チャネルとの組合せでもあり得る。例えば、一人がパソコンからインターネットサービスプロバイダ（ＩＳＰ）に電話回線を介して入力ファイルを送信し、その入力ファイルがウェッブサーバに格納されて、後でインターネットを介して受信者に送信される場合に形成されるチャネルである。

【００７９】

チャネル１４５はイレーズチャネルであると仮定されるので、通信システム１００は、受信モジュール１５０を出る出力記号の群と送信モジュール１４０に入る出力記号の群との間に１対１対応を仮定しない。実際には、チャネル１４５がパケットネットワークを含む場合、通信システム１００は、任意の２つ以上のパケットの相対的順序がチャネル１４５を介した送信中に保存されると仮定できないとされ得る。したがって、出力記号の群のキーは、上記キースキームの内１つ以上を使用して決定され、受信モジュール１５０を出る出力記号の群の順序によって必ずしも決定されない。

【００８０】

受信モジュール１５０は出力記号の群をデコーダ１５５に与え、受信モジュール１５０が受信する出力記号のこれらの群のキーについての任意のデータは、キー発生器１６０に与えられる。キー発生器１６０は、受信した出力記号の群に対するキーを再生成し、これらのキーをデコーダ１５５に与える。デコーダ１５５は、キー発生器１６０によって与えられたキーと対応の出力記号の群とを使用して、入力記号（再度、ＩＳ（０），ＩＳ（１），ＩＳ（２），．．．）を回復する。デコーダ１５５は、回復された入力記号を入力ファイルリアセンブラ１６５に与え、リアセンブラ１６５は入力ファイル１０１または入力ストリーム１０５のコピー１７０を生成する。

【００８１】

（基本エンコーダ）

図２は、図１に示されるエンコーダ１１５の１つの実施形態のブロック図である。図２のブロック図を、ここで図３を参照して説明する。図３は、図２に示すエンコーダによって実行される処理の内のいくつかの論理等価物を示す図である。

【００８２】

エンコーダ１１５には、入力記号および生成すべき出力記号の各群に対するキーが与えられる。示されるように、Ｋ個の入力記号が、入力記号バッファ２０５内に格納される。キーＩ（図１で示されるキー発生器１２０によって与えられる）は、値関数セレクタ２１０、群サイズセレクタ２１２、重みセレクタ２１５、および連想装置２２０への入力である。入力記号数Ｋはまた、これら４つのコンポーネント２１０、２１２、２１５および２２０に与えられる。計算器２２５は、値関数セレクタ２１０、群サイズセレクタ２１２、重みセレクタ２１５、連想装置２２０および入力記号バッファ２０５からの出力を受信するよう結合され、出力記号値の群に対する出力を有する。理解すべきことは、図２に示す要素と等価な他の構成が使用され得、これは本発明のエンコーダの１例にすぎないことである。

【００８３】

動作中、Ｋ個の入力記号が入力記号バッファ２０５に受信および格納される。上記のように、各入力記号は、位置（すなわち、入力ファイル内の元の位置）および値を有する。入力記号は、格納される入力記号の位置が決定され得る限り、それぞれの順序で入力記号バッファ２０５内に格納される必要はない。

【００８４】

群サイズセレクタ２１２は、使用される場合、キーＩおよび入力記号数Ｋから群サイズＧ（Ｉ）を決定する。１つの変形において、群サイズＧ（Ｉ）は、すべてのＩに対して同じである（すなわち、Ｇ（Ｉ）＝ｂであり、ここでｂは正の整数である）。この変形において、群サイズセレクタ２１２は必要ではなく、計算器２２５はｂの値に設定され得る。例えば、群サイズは、すべてのＩに対して４に設定され得る、すなわち、Ｇ（Ｉ）＝ｂ＝４であり、出力記号の各群は、４つの出力記号を含む。いくつかの実施形態において、２または３などの小さな値のｂ、またはわずかに異なる値のＧ（Ｉ）を使用することは、エンコーダの効率化に役立つ。ｂ＝１の場合、その結果は、ＬｕｂｙＩに記載のエンコーダと実質的に同じである。

【００８５】

キーＩおよび入力記号数Ｋを使用して、重みセレクタ２１５は、キーＩを有する出力記号の群の「関連付け」となるべき入力記号の数Ｗ（Ｉ）を判定する。キーＩ、重みＷ（Ｉ）および入力記号数Ｋを使用して、連想装置２２０は、出力記号の群に関連付けられた入力記号の位置のリストＡＬ（Ｉ）を判定する。理解すべきことは、Ｗ（Ｉ）は、連想装置２２０があらかじめＷ（Ｉ）を知らずにＡＬ（Ｉ）を生成し得る場合、別個にまたは明確に計算される必要がないことである。ＡＬ（Ｉ）が一旦生成されると、Ｗ（Ｉ）は容易に判定される。なぜなら、それはＡＬ（Ｉ）中の関連付けの数であるからである。

【００８６】

Ｉ、Ｗ（Ｉ）およびＡＬ（Ｉ）が一旦既知となると、出力記号の群のＧ（Ｉ）個の値Ｂ（Ｉ）＝Ｂ＿１（Ｉ），．．．，Ｂ＿Ｇ（Ｉ）（Ｉ）は、値関数Ｆ（Ｉ）２１０に基づいて計算器２２５によって計算される。適切な値関数の１つの性質は、ＡＬ（Ｉ）中のＧ（Ｉ）個までの関連付けに対する未知の値が出力記号値Ｂ（Ｉ）の群およびＡＬ（Ｉ）中の他の関連付けに対する既知の値から判定され得ることである。このステップにおいて使用される１つの好ましい値関数は、リードーソロモン値関数である。なぜなら、それは、この性質を満足し、容易に計算され、および容易に反転されるからである。

【００８７】

リードーソロモン値関数の場合、値Ｂ（Ｉ）は、リードーソロモン符号化スキームにしたがって、ＡＬ（Ｉ）中の関連付けの値から計算される。そのスキームにおいて、リードーソロモン冗長記号はＢ（Ｉ）であり、そしてＡＬ（Ｉ）中の入力記号とともにこれらの記号は、リードーソロモン符号化スキームを使用して符号化される符号化ブロックを形成す

る。
【００８８】
他の適切な値関数を代わりに使用してもよい。例えば、群サイズが１に等しい場合のＸＯＲ値関数の使用、有限体に関する多項式に基づく方法、線形系方程式に基づく方法、有限体に関するコーシー行列に基づく方法、または、他のＭＤＳ符号（そのリードーソロモン符号は１例である）を含む。
【００８９】
出力記号の群の関連付け数に依存する異なる値関数の混合がまた、使用され得る。例えば、関連付け数Ｗ（Ｉ）がＧ（Ｉ）に等しい場合、識別関数が使用され得る、すなわち、ｉ＝１，．．．，Ｇ（Ｉ）に対して、その群におけるｉ番目の出力記号の値はＢ＿ｉ（Ｉ）＝ＩＳ（Ｐ＿ｉ）であり、ここでＰ＿１，．．．，Ｐ＿Ｇ（Ｉ）はＧ（Ｉ）個の関連付けの位置である。関連付け数Ｗ（Ｉ）がＧ（Ｉ）＋１に等しい場合、ｉ＝１，．．．，Ｇ（Ｉ）に対して、その群におけるｉ番目の出力記号の値は、Ｂ＿ｉ（Ｉ）＝ＩＳ（Ｐ＿ｉ）ＸＯＲＩＳ（Ｐ＿ｉ＋１）であり得る、ここで、Ｐ＿１，．．．，Ｐ＿Ｇ（Ｉ）＋１は、Ｇ（Ｉ）＋１個の関連付けの位置である。関連付け数Ｗ（Ｉ）はＧ（Ｉ）＋１よりも大きい場合、リードーソロモン値関数が使用され得る。
【００９０】
値関数セレクタ２１０が、使用される場合、キーＩおよびＫからの値方法または値関数Ｆ（Ｉ）を判定し、ここで、Ｆ（Ｉ）を使用してＩに対する値を計算する。１つの変形において、値関数セレクタ２１０は、すべてのＩに対して同じ方法または関数Ｆを使用する。この変形において、値関数セレクタ２１０は必要でなく、計算器２２５が値関数Ｆを構成し得る。例えば、値関数はすべてのＩに対してリードーソロモン符号化器であり得る。すなわち、出力記号値の群は、すべてのその関連付けの値に基づいてリードーソロモン符号化器によって計算されるＧ（Ｉ）個の冗長記号を含む。
【００９１】
各キーＩに対して、重みセレクタ２１５は、ＩおよびＫから重みＷ（Ｉ）を判定する。１つの変形において、重みセレクタ２１５は、キーＩを使用することによってＷ（Ｉ）を選択して、まずランダム検索数を生成し、次いでこの数を使用して、重みセレクタ２１５内に格納された分布テーブル中のＷ（Ｉ）の値を検索する。そのような分布テーブルがどのように形成およびアクセスされるかを以下により詳細に説明する。一旦重みセレクタ２１５がＷ（Ｉ）を判定すると、この値は、連想装置２２０および計算器２２５に与えられる。
【００９２】
連想装置２２０は、現在の出力記号に関連付けられたＷ（Ｉ）個の入力記号の位置のリストＡＬ（Ｉ）を判定する。この関連付けは、Ｉの値、Ｗ（Ｉ）の値およびＫ（利用可能ならば）に基づく。一旦連想装置２２０がＡＬ（Ｉ）を判定すると、ＡＬ（Ｉ）は、計算器２２５に与えられる。リストＡＬ（Ｉ）、重みＷ（Ｉ）および、値関数セレクタ２１０によって与えられる値関数Ｆ（Ｉ）または予め選択された値関数Ｆのいずれかを使用して、計算器２２５は、入力記号バッファ２０５中のＡＬ（Ｉ）によって参照されるＷ（Ｉ）個の入力記号にアクセスして、出力記号の現在の群に対する値Ｂ（Ｉ）＝Ｂ＿１（Ｉ），．．．，Ｂ＿ｂ（Ｉ）を計算する。ここでｂは、群サイズセレクタ２１２によって与えられる群サイズＧ（Ｉ）または予め選択された設定値のいずれかである。ＡＬ（Ｉ）を計算するための手順の例を以下に与えるが、別の適切な手順を代わりに使用してもよい。好ましくは、その手順は、各入力記号に、出力記号の所定の群に対する関連付けとして選択される確率を略等しく与え、デコーダがそれにとって利用可能なＡＬ（Ｉ）をまだ有していない場合にデコーダが複製し得るように選択する。
【００９３】
次に、エンコーダ１１５は、Ｂ（Ｉ）＝Ｂ＿１（Ｉ），．．．，Ｂ＿ｂ（Ｉ）を出力する。実際は、エンコーダ１１５は、図３に例示する動作を実行して、すなわち、選択された入力記号のある値関数として出力記号値の群、Ｂ（Ｉ）＝Ｂ＿１（Ｉ），．．．，Ｂ＿ｂ

（Ｉ）を生成する。示される例において、その値関数は、群サイズｂが２に設定されたリードーソロモン符号化であり、出力記号の群の重みＷ（Ｉ）は３であり、関連付けられた入力記号（関連付け）は位置０、２および３であり、値ＩＳ（０）、ＩＳ（２）およびＩＳ（３）をそれぞれ有する。したがって、出力記号の群は、Ｉの値に対して以下のように計算される：

Ｂ＿１（Ｉ）＝ＲＳ＿１（ＩＳ（０），ＩＳ（２），ＩＳ（３））

Ｂ＿２（Ｉ）＝ＲＳ＿２（ＩＳ（０），ＩＳ（２），ＩＳ（３））

ここで、ＲＳ＿ｉ（*）は、リードーソロモンエンコーダを引数に適用してｉ番目の冗長記号を生成することによって生成される記号値である。

【００９４】

次に、生成された出力記号の群は、上記のように送信および受信される。ここで、出力記号の群の内いくつかが損失されるかまたはその順序がみだれた、もしくは１つ以上のエンコーダによって生成されたと仮定する。しかし、受信される出力記号の群の受信は、そのキーＩの表示、および値Ｂ（Ｉ）＝Ｂ＿１（Ｉ），．．．，Ｂ＿ｂ（Ｉ）が正確であるという何らかの確実さをもって受信されると仮定する。図１に示すように、それらの出力記号の受信された群は、キー再発生器１６０による表示およびＫの値から再構築される対応のキーとともに、デコーダ１５５に入力される。

【００９５】

入力記号中に符号化されるのビット数Ｍ（すなわち、そのサイズ）は、アプリケーションに依存する。出力記号のサイズおよび１つの群中の出力記号数ｂもまた、アプリケーションに依存し得るが、チャネルにもまた依存し得る。例えば、一般的な入力ファイルが、複数のメガバイトのファイルである場合、その入力ファイルは、数千個、数万個または数十万個の入力記号に分割され、各入力記号は、数バイト、数十バイト、数百バイトまたは数千バイトである。

【００９６】

いくつかの場合において、出力記号値および入力記号値が同じサイズであれば（すなわち、同じ数のビットによって表現可能か、または同じアルファベットから選択される）、符号化プロセスは、簡略化され得る。そのような場合、出力記号の群をパケットに入れるのが望ましく、かつ出力記号の各群が限られたサイズのパケットに収まる必要があり、かつ各群のサイズｂが固定値に設定される場合などの出力記号値サイズが限定される場合、入力記号値サイズが限定される。キーについてのいくつかのデータが、受信器にてキーを回復するために送信される場合、好ましくは、出力記号の群は、値およびキーについてのデータが１つのパケットに収まるように十分に小さいものであり得る。

【００９７】

上記のように、多くの実施において入力記号の位置は一般に連続であるが、キーは連続ではない。例えば、入力ファイルが６０，０００個の入力記号にまで分割されたとすると、その入力記号の位置の範囲は、０から５９，９９９ある。上記実施のうちの１つにおいて、各キーはランダムな３２ビット数として独立に選択され、出力記号の群は、連続的に生成され、送信者が停止するまで送信され得る。ここで示すように、群連鎖反応符号化は、６０，０００記号の入力ファイルが出力記号の任意の十分に大きな集合（６０，０００＋いくらかの増分Ａ）から再構築される。なお、これは、出力記号のこれらの群が出力シーケンスのどこから取り出されるかに依存しない。

【００９８】

（基本デコーダ）

図４は、デコーダ１５５の１つの実施形態を詳細に示し、多くの部分が図２に示すエンコーダ１１５と共通である。デコーダ１５５は、値関数セレクタ２１０、群セレクタ２１２、重みセレクタ２１５、連想装置２２０、出力記号の１群を格納するバッファ４０５、レジューサ（ｒｅｄｕｃｅｒ）４１５、再構築器４２０および再構築バッファ４２５を含む。エンコーダに関して、群セレクタ２１２および群サイズを格納するために割当てられるバッファ４０５中のスペースは、群のサイズが出力記号のすべての群に対して同じである

場合、任意であり、使用しなくてもよい。同様に、値関数セレクタ２１０および値関数の記述を格納するために割当てられるバッファ４０５中のスペースは、値関数が出力記号のすべての群に対して同じである場合、任意であり、使用しなくてもよい。再構築バッファ４２５のいくつかのエントリが図示され、ここでいくつかの入力記号が再構築され、他は依然未知である。例えば、図４において、位置０、２、５および６の入力記号が回復され、位置１、３および４の入力記号はまだ回復されない。

【００９９】

動作中、キーＩおよび値Ｂ（Ｉ）＝Ｂ＿１（Ｉ），．．．，Ｂ＿ｂ（Ｉ）を有する受信された出力記号の各群に対して、デコーダ１５５は以下を行う。キーＩは、値関数セレクタ２１０、群セレクタ２１２、重みセレクタ２１５および連想装置２２０に与えられる。Ｋおよびキーを使用して、重みセレクタ２１５は、重みＷ（Ｉ）を判定する。Ｋ、キーＩおよび重みＷ（Ｉ）を使用して、連想装置２２０は、出力記号の群に関連付けられた入力記号のＷ（Ｉ）個の位置のリストＡＬ（Ｉ）を生成する。必要に応じて、ＫおよびＩを使用して、群セレクタ２１２は、群サイズＧ（Ｉ）を選択する。必要に応じて、ＫおよびＩを使用して、値関数セレクタ２１０は、値関数Ｆ（Ｉ）を選択する。次に、Ｉ、Ｂ(Ｉ)、Ｗ（Ｉ）およびＡＬ（Ｉ）、任意のＧ（Ｉ）および任意のＦ（Ｉ）は、列をなしてバッファ４０５に格納される。群セレクタ２１２、値関数セレクタ２１０、重みセレクタ２１５および連想装置２２０は、エンコーダ１１５について記載したのと同じようにデコーダ１５５の動作を実行する。特に、図５における群セレクタ２１２、値関数セレクタ２１０、重みセレクタ２１５および連想装置２２０によって生成される群サイズＧ（Ｉ）、値関数Ｆ（Ｉ）、重みＷ（Ｉ）およびリストＡＬ（Ｉ）は、図４に示す対応の部分と同じキーＩである。Ｋが入力ファイルごとに異なる場合、Ｋをメッセージヘッダに含むなどの任意の従来方法でエンコーダからデコーダへ通信され得る。

【０１００】

再構築器４２０は、バッファ４０５を走査して、たかだか群サイズの重みを有するバッファ４０５に格納された出力記号の群を探す。これらの記号は、本明細書中で「復号可能なセット」のメンバと呼ぶ。上記の性質を有する値関数に対して、たかだかそれらの群サイズの重みの出力記号の群は、復号可能なセットに含まれる。なぜなら、それらの群サイズまでの入力記号の未知の値は、その出力記号の群および関連付けである他の入力記号の既知の値から判定され得るからである。当然、入力記号が、たかだかそれらの群サイズの重みを有すること以外の条件下で出力記号の１群によって復号されることを可能にし得る値関数が使用されるならば、その条件を使用して、出力記号の１群が復号可能なセットかどうかを判定し得る。明確にするために、ここで記載した例は、復号可能なセットが、たかだかそれらの群サイズの重みを有する出力記号の群であり、これらの例の、他の値関数復号可能条件への拡張は、この記載から明らかである。

【０１０１】

再構築器４２０が復号可能なセットに含まれるキーＩを有する出力記号の１群を見つけた場合、その出力記号値の群は、Ｂ（Ｉ）＝Ｂ＿１（Ｉ），．．．，Ｂ＿Ｇ（Ｉ）（Ｉ）であり、必要に応じて、値関数Ｆ（Ｉ）を使用してＡＬ（Ｉ）にリストされた未知の入力記号のＷ（Ｉ）個の値を再構築し、再構築された入力記号は、それらの入力記号に対する適切な位置で再構築バッファ４２５に配置される。Ｗ（Ｉ）が厳密に群サイズＧ（Ｉ）より小さい場合、再構築器４２０は、独立した方法ですでに再構築された入力記号値を再構築し得る。この場合、再構築器４２０は、独立に再構築された入力記号を脱落させ、既存の再構築された入力記号を上書きし、または独立に得られた入力記号の値を比較し、それらが異なれば誤りを生成する。このように再構築器４２０は入力記号を再構築するが、復号可能なセット中の出力記号の群からのみ入力記号を再構築する。一旦復号可能なセットからの出力記号の１群を使用して入力記号を再構築すると、それは、バッファ４０５のスペースを節約するために削除され得る。「使い古した」群の出力記号を削除することはまた、再構築器４２０が連続してその出力記号の群に再び訪れないことを確実にする。

【０１０２】

まず、再構築器４２０は、復号可能なセットの１メンバである少なくとも１つの出力記号の群が受信されるまで待機する。一旦その出力記号の１群を使用して未知の入力記号を再構築すると、復号可能なセットは再び空となる。ただし、出力記号のいくつかの他の群が、これらのちょうど再構築された入力記号の内いくつかおよびその群サイズまでまだ再構築されない他の入力記号の関数であり得るという事実を除く。このように、復号可能なセットの１メンバから入力記号を再構築することにより、出力記号の他の群が復号可能なセットに付加される。出力記号の群を復号可能なセットに付加して低減するプロセスは、レジューサ４１５によって実行される。

【０１０３】

レジューサ４１５は、バッファ４０５および再構築バッファ４２５を走査して、回復された入力記号の位置をリストするリストＡＬ（Ｉ）を有する出力記号の群を見つける。レジューサ４１５がキーＩを有する出力記号のそのような「低減可能な」群を見つけた場合、レジューサ４１５は、ＡＬ（Ｉ）中の入力記号のうちでまだ再構築されていなかった入力記号値の数に等しくなるようにＷ（Ｉ）を再計算する。

【０１０４】

レジューサ４１５の作用は、バッファ４０５中の出力記号の群の重みを低減する。出力記号の１群の重みは、たかだかその群サイズに低減される場合（または、他の復号可能な条件が他の値関数に対して生じる）、その出力記号の群は、復号可能なセットの１メンバとなり、それは次に再構築器４２０によって作用され得る。実用においては、一旦十分な数の出力記号の群が受信されると、レジューサ４１５および再構築器４２０は、連鎖反応復号化を生成し、入力ファイルからのすべての入力記号が回復されるまで、再構築器４２０は復号可能なセットを復号してより多くの入力記号を回復し、レジューサ４１５はそれらの新しく受信された入力記号を使用してより多くの出力記号の群を低減し、それによりそれらを復号可能なセットに付加するなどする。

【０１０５】

図４に示されるデコーダは、記憶装置、計算サイクルまたは送信時間をあまり考慮せずに、単純な方法で入力記号を再構築する。デコーダメモリ、複合化時間または送信時間（受信される出力記号の群の数を制限する）が限定される場合、デコーダは、それら限られたリソースをよりよく使用するために最適化され得る。

【０１０６】

（より効率的なデコーダ）

図５は、デコーダ５００のより効率的な実施の好ましい実施形態を詳細に示す。ここで、値関数は、リードーソロモン値関数であると仮定する。さらに効率的になる可能性のある同様の実施が、リードーソロモン値関数以外の値関数に対して適用される。図５を参照して、デコーダ５００は、出力記号の群データ構造５０５（以下、ＧＯＳＤＳ５０５と呼ぶ）、入力記号データ構造５１０（以下、ＩＳＤＳ５１０と呼ぶ）、復号可能なセットスタック５１５（以下、ＤＳＳ５１５と呼ぶ）、受信オーガナイザ５２０、および回復プロセッサ５２５を含む。

【０１０７】

ＧＯＳＤＳ５０５は、出力記号の群についての情報を格納するテーブルであり、ここでＧＯＳＤＳ５０５の行Ｒは、受信された出力記号のＲ番目の群についての情報を格納する。変数Ｒは、受信された出力記号の群の数を追跡し、ゼロに初期化される。ＧＯＳＤＳ５０５は、各行に対してフィールドＫＥＹ、ＶＡＬＵＥ、およびＷＥＩＧＨＴを格納し、図示のフィールドは、列に組織化される。ＫＥＹフィールドは、出力記号の群のキーを格納する。ＶＡＬＵＥフィールドは、出力記号値の群を格納する。ＷＥＩＧＨＴフィールドは、出力記号の群の初期重みを格納する。出力記号の１群のＷＥＩＧＨＴは、たかだか群サイズになるまで経時的に低減され、そして入力記号を回復するために使用され得る。

【０１０８】

ＩＳＤＳ５１０は、入力記号について情報を格納するテーブルであり、ここで行Ｐは、位置Ｐでの入力記号についての情報を格納する。各行に対して、ＩＳＤＳ５１０は、最終的

に回復された入力記号の値になるＲＥＣ＿ＶＡＬフィールド、すべての値が「ｎｏ」に初期化され、入力記号が回復されたかどうかを示すＲＥＣ＿ＩＮＤフィールド、およびＲＬフィールド用の記憶装置を含む。入力記号が回復される場合、入力記号のＲＥＣ＿ＩＮＤは、「ｙｅｓ」に変更される。ＲＬ列は、すべてが「空リスト」値に初期化される。関連付けとして入力記号を有する出力記号の群が受信されるので、出力記号の群のＧＯＳＤＳ５０５における行数が、入力記号に対するＲＬリストに付加される。

【０１０９】

ＤＳＳ５１５は、復号可能なセットについての情報を格納するスタックである。変数Ｓは、復号可能なセットのサイズを追跡し、それをゼロに初期化する。ＤＳＳ５１５において、列ＯＵＴ＿ＲＯＷは、出力記号の群のＧＯＳＤＳ５０５における行数を格納する。

【０１１０】

１つの実施形態において、デコーダ５００は、以下のように、そして図６のフローチャートに示すように動作され、図６の対応するステップは、プロセスの説明において括弧で示す。まず、ＩＳＤＳ５１０は、上記のように初期化され、ＲおよびＳの両方がゼロに初期化される（６０５）。出力記号の１つの新しい群が受信されると（６１０）、すなわち、キーＩおよび出力記号値の群Ｂ（Ｉ）＝Ｂ＿１（Ｉ），．．．，Ｂ＿Ｇ（Ｉ）（Ｉ）、ＧＯＳＤＳ５０５においてＫＥＹ（Ｒ）がＩに設定され、ＶＡＬＵＥ（Ｒ）がＢ（Ｉ）＝Ｂ＿１（Ｉ），．．．，Ｂ＿Ｇ（Ｉ）（Ｉ）に設定される（６１５）。次に、受信オーガナイザ５２０が呼び出されてＧＯＳＤＳ５０５の行Ｒに格納されたキーＩを有する受信された出力記号の群が処理される（６２０）。これは、図７のフローチャートに示すように、ＩＳＤＳ５１０に格納された情報を適切に使用して、情報をＧＯＳＤＳ５０５およびＤＳＳ５１５に付加するステップを含む。次に、Ｒを１だけインクリメントし（６２５）、次の出力記号の群をＧＯＳＤＳ５０５の次の行に格納させる。次に、回復プロセッサ５２５が呼び出され、復号可能なセット中の出力記号の群を処理し、出力記号の新しい群を復号可能なセットに付加する（６３０）。これは、図８ａおよび／または８ｂのフローチャートに示すように、ＩＳＤＳ５１０およびＧＯＳＤＳ５０５の一部を適切に使用および改変することによって、ＤＳＳ５１５に格納された復号可能なセットに付加および削除するステップを含む。デコーダ５００は、回復された入力記号の数を追跡し、この数がＫに達した場合、すなわち、すべての入力記号が回復された場合、デコーダ５００は、首尾よく終了し、そうでなければステップ６１０に戻り次の出力記号の群を受信する（６３５および６４０に示す）。

【０１１１】

受信オーガナイザ５２０の動作を説明するフローチャートを図７に示す（図９から１２を参照）。値Ｂ（Ｉ）＝Ｂ＿１（Ｉ），．．．，Ｂ＿Ｇ（Ｉ）（Ｉ）およびキーＩを有する出力記号の１群が到着した場合、受信オーガナイザ５２０は、以下の動作を実行する（図７を参照）。重みＷ（Ｉ）は、ＩおよびＫから計算され（７０５）、関連付けの位置のリストＡＬ（Ｉ）は、Ｉ、Ｗ（Ｉ）、およびＫから計算される（７１０）。図１１～１２は、Ｗ（Ｉ）の１計算の詳細を示し、図９～１０は、ＡＬ（Ｉ）の１計算の詳細を示す。

【０１１２】

再度図７を参照して、リストＡＬ（Ｉ）上の各位置Ｐに対して、入力記号Ｐが回復されない場合、すなわち、ＩＳＤＳ５１０においてＲＥＣ＿ＩＮＤ（Ｐ）＝「ｎｏ」の場合、Ｒは、ＩＳＤＳ５１０中のリストＲＬ（Ｐ）に付加され、そうでなく、入力記号Ｐが回復される場合、すなわち、ＩＳＤＳ５１０においてＲＥＣ＿ＩＮＤ（Ｐ）＝「ｙｅｓ」の場合、Ｗ（Ｉ）は１だけデクリメントされる（７２０）。次に、ＷＥＩＧＨＴ（Ｒ）は、ＧＯＳＤＳ５０５においてＷ（Ｉ）に設定される（７２５）。次に、ＷＥＩＧＨＴ（Ｒ）は、Ｇ（Ｉ）と比較される（７３０）。ＷＥＩＧＨＴ（Ｒ）がたかだかＧ（Ｉ）である場合、出力記号の群は、復号可能なセットに付加される、すなわち、ＯＵＴ＿ＲＯＷ（Ｓ）がＤＳＳ５１５においてＲに設定され、Ｓの値が１だけインクリメントされる（７３５）。最後に、受信オーガナイザ５２０が戻る（７４０）。

【０１１３】

図８ａに、図９～１２を参照して回復プロセッサ５２５の１つの動作を記載するフローチャートを示す。その動作中、回復プロセッサ５２５はまず、復号可能なセットが空かどうか、すなわち、Ｓ＝０かどうかを確かめるためにチェックし、そうであればただちに呼び出し元へ帰る（８０５、８１０）。復号可能なセットが空でなければ、Ｓは１だけデクリメントされ（８１５）、出力記号の群の行番号Ｒ’をＤＳＳ５１５からロードする（８２０）。次に、出力記号の元のキー、ＧＯＳＤＳ５０５からのＫＥＹ（Ｒ’）がＩへロードされ（８３５）、元の重みＷ（Ｉ）および群の関連付けの元のリストＡＬ（Ｉ）を計算する（８４０，８４５）。ＡＬ（Ｉ）におけるのすべての入力記号がすでに回復された場合、すなわち、ＡＬ（Ｉ）におけるすべてのＰに対してＲＥＣ＿ＩＮＤ（Ｐ）＝「ｙｅｓ」ならば（８４７）、回復プロセッサ５２５は、復号可能なセットのその要素の処理を停止し、次のステップを処理し続ける。そうでなければ、ＧＯＳＤＳ５０５における行番号Ｒ’に格納された出力記号の群を使用して、ＩＳＤＳ５１０におけるＡＬ（Ｉ）にある入力記号すべての未知の値を回復する。これは、リードーソロモンデコーダを利用することによって実行され、出力記号値の群およびＡＬ（Ｉ）における入力記号の既知の値を使用してＡＬ（Ｉ）におけるＧ（Ｉ）個までの入力記号の未知の値を回復する（８５０）。次に、入力記号値が回復されるすべての位置Ｐに対して、ＲＥＣ＿ＶＡＬ（Ｐ）は、ＩＳＤＳ５１０における回復された入力記号値に設定され（８５２）、ＲＥＣ＿ＩＮＤ（Ｐ）が「ｙｅｓ」に設定され、入力記号がＩＳＤＳ５１０において回復されたことを示す（８５２）。

【０１１４】

図８ａにおいて示すプロセスの１つの変形例を図８ｂに示す。そこで、出力記号の各群が処理されるステップ８５０、８５２、８５５および８６０を実行する代わりに、図８ｂのステップ８５１、８５３、８５６および８６１に示すように、Ｒ’の値が後の処理のために実行スケジュールに格納される。保留された実行処理の例を、ステップ８７０、８７５、８８０および８８５により示されるステップを含む図８ｃに示す。この変形例において、図６に示すフローチャートは、ステップ６０５においてＥをゼロに初期化することによって改変される。実行スケジュールの保留された処理は、受信した記号がファイル全体を復号するのに十分であるとデコーダが判定した後、例えば、すべての入力記号が回復可能であると分かった後のステップ６４０にて行われ得る。いくつかの場合において、特に入力記号が大きい場合、スケジュールの実行は、入力ファイルまたはその部分が受信器側で必要となるまで保留され得る。

【０１１５】

いずれの変形例においても、すなわち、図８ａまたは図８ｂのいずれかにおいて、ステップ８５５または８５６で、ちょうど回復された出力記号の群または回復されるはずだった出力記号の群に対し、関連付けとしての入力記号は、これらの入力記号が回復されたこと、または回復されるはずだったことを反映するように改変される。ちょうど回復された、または回復されるはずだった各位置Ｐに対して、ＧＯＳＤＳ５０５における出力記号のこれらの群の行番号は、ＲＬ（Ｐ）内に格納される。ＲＬ（Ｐ）における各行番号Ｒ’’に対して、ＷＥＩＧＨＴ（Ｒ’’）が１だけデクリメントして、ＧＯＳＤＳ５０５の行Ｒ’’における出力記号の群の関連付けとして位置Ｐにおける入力記号の回復を反映させる。この改変により、ＧＯＳＤＳ５０５の行Ｒ’’における出力記号の群が群サイズに等しい重み、すなわち、ＷＥＩＧＨＴ（Ｒ’’）＝Ｇ（ＫＥＹ（Ｒ’’））になる場合、出力記号のこの群は、ＯＵＴ＿ＲＯＷ（Ｓ）をＲ’’に設定し、かつＳを１だけインクリメントすることによって復号可能なセットに付加される（８５５または８５６）。最後に、リストＲＬ（Ｐ）上の出力記号の群の行番号を格納するために使用されるスペースをフリースペースに戻し（８６０または８６１）、処理はステップ８０５に続く。

【０１１６】

（連想装置の実施）

１群の出力記号キーから関連する入力記号へのマッピング（すなわち、重みＷ（Ｉ）、およびキーＩに対する関連付けの位置のリストＡＬ（Ｉ）の決定）は、種々の形態をとり得

る。Ｗ（Ｉ）は、同じキーＩに対してエンコーダおよびデコーダの両方によって同じ値（送信者および受信者のそれぞれにおいて）であるように選択されるべきである。同様に、ＡＬ（Ｉ）は、同じキーＩに対してエンコーダおよびデコーダの両方によって同じ位置のリストを含むように選択されるべきである。連想装置は、Ｉならびに通常Ｗ（Ｉ）およびＫからＡＬ（Ｉ）を計算または生成するオブジェクトである。

【０１１７】

１つの実施形態において、Ｗ（Ｉ）およびＡＬ（Ｉ）は、ランダムプロセスを模倣するように設計された方法で判定される。エンコーダおよびデコータが同じキーに対して同じ結果を生成するという要件を満足するために、擬似ランダムシーケンスが、キーをシードにしてエンコーダおよびデコーダの両方によって生成され得る。擬似ランダムシーケンスの代わりに、本当のランダムシーケンスが、Ｗ（Ｉ）および／またはＡＬ（Ｉ）の生成のために使用され得るが、それが有用であるためには、Ｗ（Ｉ）およびＡＬ（Ｉ）を生成するために使用されたランダムシーケンスが受信者に通信される必要があり得る。

【０１１８】

図４において示すデコーダにおいて、バッファ４０５は、関連付けの位置の出力記号リストの各群の記憶装置、すなわち列でラベルされたＡＬ（Ｉ）内に記憶装置を必要とする。図５に示すより効率的なデコーダは、この記憶装置を必要としない。なぜなら、関連付けの１リストは、例えば図９～１０に示すように、必要に応じて再計算されるからである。これらの計算が必要に応じて速やかに行われ得る場合にのみ、記憶装置を節約するために毎回関連付けリストを再計算する際に利点がある。

【０１１９】

連想装置２２０の好ましい実施形態が図９に示され、図１０に示すプロセスにしたがって動作する。この連想装置は、エンコーダおよびデコーダにおいて使用され得る。エンコーダは１回に１より多くのＡＬ（Ｉ）を格納する必要が通常ないので、エンコーダでのＡＬ（Ｉ）のための記憶装置にはあまり関心がないが、同じプロセスをエンコーダおよびデコーダの両方で使用して、ＡＬ（Ｉ）に対する値が両方の場所で確実に同じとなるようにすべきである。

【０１２０】

連想装置２２０への入力は、キーＩ、入力記号数Ｋ、および重みＷ（Ｉ）である。出力は、キーＩを有する出力記号の群の関連付けのＷ（Ｉ）個の位置のリストＡＬ（Ｉ）である。図９において示すように、アソシエータは、ランダムビットのテーブルＡＳＳＯＣ＿ＲＢＩＴＳ９０５および計算器ＡＳＳＯＣ＿ＣＡＬＣ９１０を含む。特定のＡＬ（Ｉ）が生成される前に、入力ファイルのサイズは、入力記号数が素数となるように調節される。このように、入力ファイルがＫ個の入力記号で開始する場合、Ｋより大きいかまたはＫに等しい最小素数Ｐが識別される。ＰがＫよりも大きい場合、Ｐ－Ｋ個のブランク（例えば、ゼロに設定される）入力記号が入力ファイルに付加され、ＫがＰに再設定される。この改変された入力ファイルに対して、関連付けの位置のリストＡＬ（Ｉ）は、図９および１０に示すように計算される。

【０１２１】

この実施形態において、ＡＳＳＯＣ＿ＣＡＬＣ９１０は、以下に記載され、かつ図１０のフローチャートに示すように動作する。第１ステップは、キーＩ、入力記号数Ｋおよびランダムビットのテーブルを使用して、Ｘが少なくとも１かつたかだかＫ－１であり、Ｙが少なくともゼロかつたかだかＫ－１である性質を有する２つの整数値ＸおよびＹを生成することである（１００５）。好ましくは、ＸおよびＹは、独立であり、それぞれの範囲内で一様分布する。次に、Ｗ（Ｉ）個のエントリを有する配列Ｖ［］が、ＡＬ（Ｉ）の記憶のために、そのメンバが計算される際に初期化される（１０１０）。Ｖ［］は、１つのリストに対する一時記憶装置にすぎないので、バッファ４０５（図４を参照）のＡＬ（Ｉ）列よりもずっと少ないメモリを占めるだけであり得る。Ｖ［０］（これは、リストＡＬ（Ｉ）の第１番目の要素）がＹに設定される（１０１５）。次に、１で始まりＷ（Ｉ）－１で終わるＪのすべての値に対して、Ｖ［Ｊ］の値は、ステ

ップ１０２０～１０５０に示すように、（Ｖ［Ｊ－１］＋Ｘ） ｍｏｄ Ｋに設定される。Ｋが素数であり、かつＷ（Ｉ）がたかだかＫであるので、Ｖ［］値のすべてが固有である。図示のように、「ｍｏｄ Ｋ」演算は、簡単な比較および減算演算であり得る、すなわち、ステップ１０３５および１０４０。このように、出力記号の所定群の関連付けの位置のリストを生成するプロセスは、非常に効率的である。

【０１２２】

ＡＬ（Ｉ）を計算する上記アプローチの１つの利点は、関連付けの位置の分布に十分な変化を生成して、復号化アルゴリズムを、Ｗ（Ｉ）を選択するための良好な手順と合わされる場合、高い確率かつ最小の受信オーバーヘッドで動作させることを確実にする（すなわち、入力ファイルは、出力記号のＫ／ｂ個よりわずかに多い群を受信した後、回復される。ここで入力記号および出力記号は同じ長さとする）。

【０１２３】

ＡＬ（Ｉ）が図１０に示すように計算され、ランダムビットＡＳＳＯＣ＿ＲＢＩＴＳ９０５のテーブルが送信者と受信者との間で秘匿される場合、通信システム１００は、安全な通信システムとして使用され得る。なぜなら、ＡＬ（Ｉ）のシーケンスを知らずに、受信したストリームを復号化することは、不可能ではないが、困難であるからである。この特徴はまた、ＡＬ（Ｉ）を計算または生成する方法である限り、ＡＬ（Ｉ）を計算または生成する他の方法を使用する通信システムに適用し得る、また、好ましくは、発生器ＡＬ（Ｉ）を計算する方法において使用されるシードは、秘匿される。

【０１２４】

（重みセレクタ実施例）

エンコーダ／デコーダの性能および効率は、重みの分布に依存し、いくつかの分布は、他の分布より良好である。重み選択の動作の局面を以下に議論し、その後いくつかの重要な重み分布を説明する。図１１のブロック図および図１２のフローチャートを使用してこれらの概念を例示する。

【０１２５】

図１１に示すように、重みセレクタは、２つのプロセスＷＴ＿ＩＮＩＴ１１０５およびＷＴ＿ＣＡＬＣ１１１０、ならびに２つのテーブルＷＴ＿ＲＢＩＴＳ１１１５およびＷＴ＿ＤＩＳＴＲＩＢ１１２０を含む。プロセスＷＴ＿ＩＮＩＴ１１０５は、第１のキーが通過してテーブルＷＴ＿ＤＩＳＴＲＩＢ１１２０を初期化した場合に１度だけ呼び出される。ＷＴ＿ＤＩＳＴＲＩＢ１１２０の設計は、システムの重要な局面であり、後でずっとより詳細に考察される。プロセスＷＴ＿ＣＡＬＣ１１１０は、各呼び出しごとに呼び出されて、キーＩに基づいて重みＷ（Ｉ）を生成する。図１２のフローチャートに示すように、ＷＴ＿ＣＡＬＣ１１１０は、キーおよびテーブルＷＴ＿ＲＢＩＴＳに格納されたランダムビットを使用して、乱数Ｒを生成する（１２０５）。次に、Ｒの値を使用して、テーブルＷＴ＿ＤＩＳＴＲＩＢ１１２０における行番号Ｌを選択する。

【０１２６】

図１１に示すように、ＷＴ＿ＤＩＳＴＲＩＢ１１２０のＲＡＮＧＥ列におけるエントリは、値ＭＡＸ＿ＶＡＬで終了する正の整数の増加シーケンスである。Ｒに対する可能な値のセットは、ゼロおよびＭＡＸ＿ＶＡＬ－１との間の整数である。所望の性質は、Ｒが、可能な値の範囲においてどの値になるのも等しく確からしいことである。Ｌの値は、ＲＡＮＧＥ（Ｌ－１）≦Ｒ＜ＲＡＮＧＥ（Ｌ）を満足するＬを見つけるまでＲＡＮＧＥ列を検索することによって決定される（１２１０）。一旦Ｌが見つかると、Ｗ（Ｉ）の値がＷＴ（Ｌ）に設定され、これは戻された重みである（１２１５、１２２０）。図１１において、図示のテーブル例に対して、Ｒが３８，５００に等しい場合、Ｌが４であることが分かり、したがって、Ｗ（Ｉ）はＷＴ（４）＝８に設定される。

【０１２７】

重みセレクタおよび連想装置を実施する他の変形例は、Ｉを擬似ランダムに生成し、ＩからＷ（Ｉ）およびＡＬ（Ｉ）を直接生成するステップを含む。明らかなように、Ｗ（Ｉ）は、ＡＬ（Ｉ）を調べることによって決定され得る。なぜなら、Ｗ（Ｉ）は、ＡＬ（Ｉ）

における関連付けの数に等しいからである。この説明から、Ｗ（Ｉ）個の値を生成する多くの他の方法は、説明したばかりのシステムと等価であり、ＷＴ＿ＤＩＳＴＲＩＢによって規定される分布を有する１セットのＷ（Ｉ）値を生成するということが明らかである。

【０１２８】

Ｗ（Ｉ）が図１２に示すように計算され、ランダムビットＷＴ＿ＲＢＩＴＳ１１１５のテーブルが送信者と受信者との間で秘匿される場合、通信システム１００は、安全な通信システムとして使用され得る。なぜなら、Ｗ（Ｉ）のシーケンスを知らずに、受信したストリームを復号化することは、不可能ではないが、困難であるからである。この特徴はまた、Ｗ（Ｉ）を計算または生成する方法である限り、Ｗ（Ｉ）を計算または生成する他の方法を使用する通信システムに適用し得る、また、好ましくは、発生器Ｗ（Ｉ）を計算する方法において使用されるシードは、秘匿される。

【０１２９】

（別のデコーダ）

この開示を読むと、受信器は、上記の実施例よりも効率的に動作し得ることが当業者に明らかである。例えば、受信器は、パケットをバッファし、１群のパケットが到着したときだけ回復ルールを適用すればより効率的であり得る。この改変は、後の不必要な動作を行う際にかかる計算時間を低減し、コンテクストスイッチングによるオーバーヘッドを低減する。特に、デコーダは、少なくともＫ個の出力記号（同じサイズの入力記号および出力記号を仮定する）がパケット単位で到着する前に、Ｋ個の入力記号の元のファイルを回復することは望めないので、復号化プロセスを開始する前に少なくともＫ個の出力記号が到着するまで待機することが有益であり得る。

【０１３０】

図１３は、上記の概念を含み、図６のデコーダによって使用されるプロセスの改変例である復号化の異なる方法を示す。２つの方法間の主な違いは、図１３の方法が１３１５に示すように、バッチ単位で出力記号の群を受信することである。第１のバッチのサイズは、到着する出力記号の総数がＫ＋Ａであるように設定され、ここで、Ａは、入力記号数Ｋの小さな部分である（１３１０）。出力記号の群の第１のバッチが受信された後で、出力記号の群は、出力記号の群を処理するために受信オーガナイザ５２０を使用するステップ（１３４０）と、復号可能なセットを処理し、たかだかそれらの群サイズであり低減された重みを有する出力記号の群から入力記号を回復するために回復プロセッサ５２５を使用するステップ（１３５０）とを組み合わせて、前記のように処理される。Ｋ個の入力記号のすべてを回復することを、出力記号の群の第１のバッチを使用して達成できない場合、各バッチがさらなるＧ個の出力記号を含む、出力記号のさらなるバッチが、すべての入力ファイルが回復されるまで受信および処理される。

【０１３１】

デコーダの補助データ構造に必要な記憶装置をできるだけ低減することは、利点である。すでに説明したように、出力記号の各群に対する関連付けのリストのための記憶装置は必要でない、なぜなら、必要に応じて、連想装置２２０を使用して、それらのリストを速やかに計算し得るからである。まだ回復されない入力記号の各々に対して、関連付けとして入力記号を有する出力記号の群のＧＯＳＤＳ５０５における行番号を格納するために別の記憶装置が必要である。すなわち、図５のテーブルＩＳＤＳ５１０におけるＲＬ列に示されるリストのためのスペースが必要である。図８のステップ８５５においてすでに説明したように、この格納を使用すると、所定の入力記号が再構築された場合、どの出力記号の群が低減可能かを速やかに識別し得る。それが効率良くなされなければ、これらのリストに必要な記憶装置は、すべての入力記号を回復するために使用される出力記号のすべての群の関連付けの総数に比例し得る。

【０１３２】

（予めソートするデコーダ）

次に、図１４および１５を参照して、デコーダのより好ましい実施形態を説明する。図１４は、デコーダを構成する部分を示す。それらは、図５に示すものと同じであるが、ただ

し、テーブルＷＥＩＧＨＴ　ＳＯＲＴ１４０５および図８ｂにおいて説明されるように形成された実行スケジュールを格納するために使用されるＥＸＥＣＵＴＩＯＮ　ＬＩＳＴ１４２０が追加される点で異なる。出力記号の群のＧＯＳＤＳ５０５における行番号のバッチを格納するために、テーブルＷＥＩＧＨＴ　ＳＯＲＴが使用され、それらは受信される際に、重みの昇順で格納される。ＷＴ＿ＶＡＬ列を使用して、重みを格納し、ＲＯＷ＿ＬＩＳＴ列を使用して、ＧＯＳＤＳ５０５における出力記号の群の行番号を格納する。一般に、重みＷＴ＿ＶＡＬ（Ｌ）を有する出力記号のすべての群の行番号は、ＲＯＷ＿ＬＩＳＴ（Ｌ）に格納される。このテーブルを使用して、図１５のステップ１５２０に示すように、重みの昇順で出力記号の群のバッチを処理する。出力記号の低い重み群は、入力記号を回復するためにそれほど集中して計算に使用せず、出力記号の群の重みが大きいほど、それらの関連付けのほとんどがただちに回復されるということがあり得る。したがって、それは、リンク記憶装置スペース（デコーダは、回復された入力リンクによって使用されるスペースを回復し得、処理中の出力記号の群はまだ未回復のわずかな関連付けを有する）を実質的に節約する。

【０１３３】

出力記号の群を適切なサイズのバッチで重みの昇順に処理することは、メモリ要件および処理要件を下げる。

【０１３４】

図１５に示すように、出力記号の群においてＫ個よりわずかに多い出力記号（図ではＫ＋Ａ個の出力記号によって示す）は、いずれの処理の開始よりも前に到着することが許される（１５１５）。ここで、入力ファイルにおける同じサイズの入力ファイルおよび出力ファイル、ならびにＫ個の入力記号を仮定する。初めに、デコーダは、群におけるＫ＋Ａ個の出力記号の受信を単に待機する。なぜなら、デコーダは、Ｋ＋Ａ個の出力記号より少ない出力記号から入力ファイルをどうしても回復し得ないと予想でき、おそらくＫ個より少ない出力記号から任意の入力ファイルを回復し得ない。実用上は、５・√ＫがＡに対して良好な値であることが分かった。

【０１３５】

出力記号の受信された群のＧＯＳＤＳ５０５における行番号は、図１４のテーブルＷＥＩＧＨＴ　ＳＯＲＴ１４０５において重みの昇順で格納される（図１５のステップ１５１５）。Ｔが、そのときの１とＴとの間のＬの値に対する出力記号の重みの可能な群の数である場合、リストＲＯＷ＿ＬＩＳＴ（Ｌ）は、重みＷＴ＿ＶＡＬ（Ｌ）の出力記号の受信されたすべての群を含み、ここで１＝ＷＴ＿ＶＡＬ（１）＜ＷＴ＿ＶＡＬ（２）＜ＷＴ＿ＶＡＬ（３）＜．．．＜ＷＴ＿ＶＡＬ（Ｔ）であり、かつＷＴ＿ＶＡＬ（Ｔ）が出力記号の任意の群のうちの最大の重みである。次に、ステップ１５２０に示すように、図１５に示すデコータの動作の残りは、図１３に示すデコーダとまったく同じであるが、ただし、出力記号の群が重みの昇順で処理されることを除く。

【０１３６】

通常、Ｋ＋Ａ個の出力記号は、すべての入力記号を回復するのに十分であり得る。しかし、全部でＫ＋Ａ個の出力記号を含む出力記号のいくつかのセット群は、すべての入力記号を回復するのに十分でないことがある。そのような場合、Ｇ個のさらなる出力記号のバッチは、群単位で受信され、そしてすべての入力記号が回復されるまで処理される。Ｇに対する良好な設定は、√Ｋである。

【０１３７】

図１６～１９は、図１５において記載するプロセスの実行例の１スナップショットを示す。この例において、群サイズは、すべての群に対して２であり、出力記号の４個の群（１６０３０、１６０３５、１６０４０、１６０４５）は、図１６において矢印つきの線で示す関連付け（１６０００、１６００５、１６０１０、１６０１５、１６０２０、１６０２５）とともに受信されている。まず、キー２３および値（Ａ，Ｄ）、キー１５９および値（ＲＳ＿１（Ａ，Ｅ，Ｆ），ＲＳ＿２（Ａ，Ｅ，Ｆ））、キー３３および値（Ｂ，Ｃ）、ならびにキー８３５および値（ＲＳ＿１（Ｃ，Ｆ，Ｇ，Ｈ），ＲＳ＿２（Ｃ，Ｆ，Ｇ，

H））を有する出力記号の群（第１および第３の出力記号の群は、重み２を有し、その値関数が識別関数であり、第２および第４の群は、それぞれ重み３および重み４を有し、その値関数がリードーソロモン値関数である）が、図１７に示すようにＧＯＳＤＳ５０５に受信および格納される。ＧＯＳＤＳ５０５における行番号は、図１８に示すように、出力記号の重みに対応する行においてＲＯＷ＿ＬＩＳＴ中に格納される。重み２の出力記号の群は、ＧＯＳＤＳ５０５の行０および行２にある。このように、ＲＯＷ＿ＬＩＳＴ（０）は、重みＷＴ＿ＶＡＬ（０）＝２を有する出力記号の群に対応し、図１８に示すように、行番号０および２を含む。同様に、ＷＴ＿ＶＡＬ（１）＝３であり、かつＲＯＷ＿ＬＩＳＴ（１）は１を含み、ＷＴ＿ＶＡＬ（２）＝４であり、かつＲＯＷ＿ＬＩＳＴ（２）は３を含む。

【０１３８】

プロセスのこの時点で、重み昇順にした出力記号の最初の２つの群が処理されており、ＧＯＳＤＳ５０５の行１における出力記号の第３の群は、受信オーガナイザ５２０によって処理されており、出力記号のこの第３の群が回復プロセッサ５２５によってちょうど処理されるところである。行０および２における出力記号の群が、スケジュールにすでに付加され、それぞれ位置０、３、１および２で最終的に入力記号を回復する。ＧＯＳＤＳ５０５の行３における出力記号の群は、まだ回復されてない位置５、６および７の３つの関連付けを有し、したがって、ＩＳＤＳ５１０における位置５、６および７からＧＯＳＤＳ５０５における行３に戻るリンクが存在する。ＧＯＳＤＳ５０５の行１における出力記号の群は、位置０、４、および５において３つの関連付けを有する。位置０における関連付けは、回復されたものとしてすでにマークされており、したがって、そこから行１に戻るリンクは存在しない（それにより、出力記号のこの群の重みが３から２に低減され、このことは、一旦回復プロセッサ５２５が実行されると位置４、５、６および７における残りの入力記号の回復をトリガする）。位置４および５における関連付けは回復されず、したがって受信オーガナイザ５２０は、ＩＳＤＳ５１０における位置４および位置５からＧＯＳＤＳ５０５における行１へのリンクを付与する。これはすべて、図１７に示される。このように、プロセスのこの時点で、入力記号から、それらを関連付けとして有する出力記号の群へ戻る全部で５つのリンクだけが使用される。これは、あらゆる入力記号から、それを関連付けとして有するあらゆる出力記号の群へのリンクを使用する単純な実施例に遜色がない。この例において、このようなリンクが１１可能である。

【０１３９】

一般に、リンク用の格納スペースの節約が、図１３に記載したプロセスよりも図１４および１５において記載したプロセスを使用する場合、劇的に低減する。例えば、入力記号数が５０，０００の場合、スペースの節約は、一般にリンクスペースにおいて１０～１５倍である。この低減の理由は、重みのより小さい出力記号の群は、プロセスの終了時より開始時に入力記号を回復する可能性がより大きく、重みのより大きな出力記号の群は、プロセスの開始時より終了時に出力記号の群を回復する可能性がずっとより大きい。したがって、重みの昇順で出力記号の群を処理することは意味のあることである。図１３よりも図１４および１５に記載するプロセスのさらなる利点は、復号化が一般に３０％～４０％速くなることである。これは、重みのより小さい出力記号の群が、重みのより大きい出力記号の群よりも、入力記号を回復するために使用される可能性が大きく（なぜなら、重みのより小さい出力記号の群が先に考慮されるからである）、かつ特定の入力記号を回復するコストは、それを回復するために使用される出力記号の群の重みに直接依存するからである。

【０１４０】

図１９は、この例の完全な実行スケジュールを示し、実行時に、群サイズ２を有する出力記号の４個の群に基づいて８個の入力記号すべてを回復し得る。

【０１４１】

（重み分布の選択）

重要な最適化は、入力ファイルをできるだけ少ない出力記号の群を用いて完全に再構築し

得るように符号化プロセスを設計することである。この最適化は、送信時間および帯域幅がコスト高または制限される場合、または入力ファイルが、さらなる出力記号の群を待つことなく速やかに復号化されなければならない場合、役に立つ。一般に、入力ファイルの再構築に必要な十分な出力記号数は、元の入力ファイルにおける入力記号数よりもわずかに大きい（同じサイズの入力記号および出力記号を仮定する）。入力ファイルよりも少ないビットを受信した場合、任意の入力ファイルは回復され得ないことを示し得る。したがって、完全な符号化スキームは、入力ファイルと同じビット数を符号化する任意の出力記号の群のセットから入力ファイルを回復することが可能であり得、符号化効率の１つの尺度は、予期される条件下で必要な予備のビットがどれくらい少ないかである。

【０１４２】

図５に示すデコーダにおいて、最大効率は、デコーダがちょうどＫ個の出力記号を受信した後で回復プロセッサ５２５が最後の未知の入力記号を回復する場合に得られる。Ｋ個より多い出力記号が、すべての入力記号が回復され得る時間までにデコーダによって受信されたならば、入力ファイルの回復に必要でない、または使用されない出力記号の群が受信されたであろう。最大効率が良好である間、それを目標とすることは、再構築が完全になる前にＤＳＳ５１５が空になり得る危険によって調節されるべきである。言い換えると、最大効率において、復号可能なセットのサイズは、ちょうど再構築が終了する際にゼロとなるが、符号化／復号化は、復号可能なセットのサイズが再構築の終了前にゼロとなるわずかの確率しかないように、Ｋ＋Ａ個の出力記号を使用して調整されるべきであり、そのため群に含まれるＧ個の出力記号のさらなるセットは必要でない。

【０１４３】

この点を図２０に例示する。この図は、復号可能なセットのサイズ対再構築された入力記号数のプロットを示す。ここで、デコーダは、以下に記載の理想の分布に対して、出力記号のＫ／ｂ個の群を用いて動作し、各群はサイズｂを有する。この例において、Ａ＝０、すなわち、Ｋ個の入力記号のすべてを復号するために、群に受信された出力記号数は、可能な数の最小数である（入力記号および出力記号は同じサイズと仮定する）。理解すべきことは、そのプロットが、重みおよび関連付けを決定するための任意の所定関数に対して異なり、また出力記号のどの特定の群を受信したかに依存して異なり得ることである。そのプロットにおいて、復号可能なセットサイズの予想サイズは、最初は１であり、回復プロセスを通して１を維持する。このように、予想される挙動においては、次のｂ個の入力記号を回復するために使用され得る、復号可能なセットにおける出力記号の１群が常に存在する。図２０はまた、理想の分布の実際の挙動の例を示す。この実際の実行において、復号可能なセットは、回復が完了する前は、すなわち、１００，０００個の入力記号のうちの２つのみが回復された後は、空であることに留意されたい。理想の分布の実際の挙動は、一般的である、すなわち、理想の分布に対して、ランダムなゆらぎはほとんど常に、すべての入力記号が回復される前に復号可能なセットを空にし、このため、以下に記載のように、よりロバストな分布が必要である。

【０１４４】

効率は、復号可能なセットにおける出力記号の群が、関連付けの値を再構築するために使用される場合に、その群サイズより厳密に小さい未知の入力記号を関連付けとして有する回数を限定することによって、向上される。これは、Ｗ（Ｉ）を生成するための関数を適切に選択することによって達成され得る。

【０１４５】

このように、出力記号の十分な群を受信することによって、入力ファイルを任意の所望の確実度で完全に回復することが可能である一方で、Ａの何らかの小さい値に対してＫ＋Ａ個程度に少ない出力記号（入力記号および出力記号は同じサイズと仮定する）を用いて完全な入力ファイルを含むＫ個の入力記号を回復する確率が高くなるように群連鎖反応符号化通信システムを設計することが好ましい。Ａの最小値がゼロであり、各出力記号がＫ個の入力記号のすべてに依存する標準的なリードーソロモン符号化を使用する場合などのいくつかの符号化スキームにおいて達成され得るが、これにより、符号化および復号化時間

が遅くなる。Ａとして何らかの小さい非ゼロ値を許容することによって、改善された通信システムが得られ得る。

【０１４６】

小さい値のＡは、群サイズ、出力記号の群に対する重み分布（すなわち、すべてのＩについてのＷ（Ｉ）の分布）、および出力記号の群についての関連付けの分布（すなわち、すべてのＩについてのＡＬ（Ｉ）のメンバーシップ）を決定するための適切な分布を使用することによって、群連鎖反応符号化において達成され得る。強調すべきことは、復号化プロセスが群サイズ、重み分布および関連付けの選択についての分布にかかわらず適用され得るが、好ましい実施形態は、群サイズ、重み分布および特に最適に近い性能として選択された関連付けの選択についての分布を使用し得る。実際、選択した分布の小さな変化によって性能はほんのわずかしか変化し得ないので、多くの分布が良好に機能し得る。

【０１４７】

１つの好ましい実施形態による分布を決定するための方法論をここで説明する。この実施形態において、出力記号のすべての群の群サイズは、固定された正の整数ｂと同じ値ｂである。使用される実際の重み分布は、理想の数学的分布に基づく。理想の重み分布において、重みＷ（Ｉ）は、「理想」の確率分布にしたがって選択される。最小の重みはｂであり、最大の重みはＫである。ここで、Ｋは、入力記号数である。理想の分布において、ｉの値に等しい重みが以下の確率ｐを用いて選択される：

ｉ＝ｂの場合、ｐ＝ｂ／Ｋ；および

ｉ＝ｂ＋１，．．．，Ｋの場合、ｐ＝ｂ／（ｉ（ｉ－１））

一旦重みＷ（Ｉ）が選択されると、Ｗ（Ｉ）個の関連付けられた入力記号のリストＡＬ（Ｉ）がランダムに（必要ならば、擬似ランダムに）独立かつ均一に選択され、選択された関連付けの全てが確実に異なるようにする。このように、第１の関連付けは、Ｋ個の入力記号からランダムに選択される（各入力記号は、選択される確率が１／Ｋである）。次に、第２の関連付け（Ｗ＞１の場合）は、残りのＫ－１個の記号からランダムに選択される。上記の重み確率分布は、システムが予想通り正確に挙動するならば、正確に出力記号のＫ／ｂ個の群がすべての入力記号を復号および回復するのに十分であり得るという性質を有する。理想の分布に対するこの予想される挙動は、図２０において実線で示される。しかし、重みおよび関連付けの選択のランダム性質のために、かつ出力記号の群の任意のセットが復号化プロセスにおいて使用されるために、プロセスは、必ずしもそのように挙動し得ない。理想の分布に対する実際の挙動の例を図２０において点線で示す。このように、理想の重み分布は、実用上はいくらか改変されなければならない。

【０１４８】

一般に、所定の設定に対する最良のパラメータは、コンピュータシミュレーションによって見つけられ得る。しかし、理想の重み分布に関する簡単な変形例は、効果的なオプションである。この簡単な変形例において、理想の重み分布は、重みｂを有する出力記号の群および出力記号の大きな重みを有する群の確率を上げることによってわずかに改変されて、Ｋ個すべての入力記号が回復される前に復号可能なセットが空になる確率を低減する。出力記号の重みｂを有する群を余分に供給すると、回復プロセスが入力記号の回復の終了近くになるまで、そのプロセスが出力記号の重みがたかだかｂである群を使い果たす（すなわち、復号可能なセットを空にする）確率を低減する。出力記号の大きな重みの群を余分に供給すると、回復プロセスの終了近くで、各々まだ回復されない入力記号に対して、関連付けとしてその入力記号およびまだ回復されない関連付けとしてたかだかｂ－１個の他の入力記号を有し得る出力記号の少なくとも１つの群が存在する確率を増加する。

【０１４９】

より詳細には、改変された重み分布は以下の通りである：

ｉ＝ｂの場合、ｐ＝ｎ・ｂ・Ｒ１／Ｋ；

ｉ＝ｂ＋１，．．．，（Ｋ／Ｒ２－１）の場合、ｐ＝ｎ・ｂ／（ｉ（ｉ－１）（１－ｉＲ２／Ｋ））；および

ｉ＝Ｋ／Ｒ２，．．．，Ｋの場合、ｐ＝ｎ・ＨＷ（ｉ）

ここで、Ｋは入力記号数であり、Ｒ１およびＲ２は調節可能なパラメータ、ならびにｎはｐ値のすべての合計が１となるように使用される正規化因子である。

【０１５０】

ＨＷ（ｉ）ならびにＲ１およびＲ２としてサンプル値の計算を、添付書類Ａにおいて詳細に示す。そこで、Ｃ＋＋の記号ｎＳｔａｒｔＲｉｐｐｌｅＳｉｚｅ、ｎＲｉｐｐｌｅＴａｒｇｅｔＳｉｚｅおよびｎＳｙｍｂｏｌｓは、それぞれ上記式のＲ１、Ｒ２およびＫに対応する。

【０１５１】

この改変された分布は、理想の数学的重み分布に同様であり、重みｂおよびさらに大きな重みを有する出力記号のさらに多くの群、ならびにそれにしたがって再設計された分布を有する。改変された分布において示されるように、Ｒ１は、重みｂの記号の複数の増加した確率を決定するのと同様に、出力記号の重みｂの群の初期の割合を決定し、Ｒ２は、「より大きな」重みと「それほど大きくない」重みとの間の境界を決定する。

【０１５２】

Ｒ１およびＲ２に対する良好な選択は、一般にＫ／ｂに依存し、経験的に決定され得る。例えば、Ｒ１を１．４×（Ｋ／ｂの平方根）に等しくし、Ｒ２を２＋２．１×（Ｋ／ｂの四乗根）に等しくすると、実用上うまくいく。したがって、Ｋ＝４０００およびｂ＝５の場合、Ｒ１＝３９およびＲ２＝１３に設定するとうまくいく。Ｋが６４０００およびｂ＝２の場合、Ｒ１＝２５０およびＲ２＝３０に設定するとうまくいく。Ｒ１およびＲ２のより詳細な計算を、添付書類Ａに示す。図２１は、この分布の予想される挙動が、回復プロセス全体にわたって復号可能なセットを適度に大きくしたままにするので、実際の実行下では、予想される挙動からのランダムなゆらぎは、すべての入力記号が回復される前に復号可能なセットを空にする見込みはない。重み分布の適切な設定によって、いかなる損失条件下においても、最小数の出力記号を受信すれば高い確率で入力記号のすべてを回復することを確実なものとする。

【０１５３】

上記の再構築プロセスは、トルネード符号に対して使用されるものと同様であるが、符号を形成するために使用される異なるプロセスによって、極端に異なる効果を生じる。特に、上記のように、群連鎖反応符号化に必要なメモリは、トルネード符号に必要なメモリよりも著しく小さく、種々の状況における群連鎖反応符号化の使用の容易さは、いくらかのスピードを犠牲にする可能性はあるが、トルネード符号のそれをはるかに超える。そのプロセスの基礎をなす数学的詳細は、以下により詳細に説明される。

【０１５４】

（いくつかの群連鎖反応符号の性質）

生成され、そしてチャネルを介して送信される出力記号の群の数は、他の符号化スキームと同様に、群連鎖反応符号化を用いても制限されない。なぜなら、キーは、入力記号に対して１対１対応を有する必要がなく、Ｉの異なる値の数は、入力記号数の何らかの小さな一定の割合に制限されないからである。したがって、復号可能セットが、入力ファイルが再構築される前に空になる場合でさえ、復号プロセスが失敗しない可能性がある。なぜなら、デコーダが、必要とされるより多くの出力記号の群を集めて、重みがその群サイズである出力記号の群を少なくともさらに１つ得ることができる。その群サイズの重みの出力記号からなるその群が受信されると、それは、復号可能なセットに置かれ、連鎖反応効果によって、前に受信された出力記号の群をたかだかそれらの群サイズの重みに低減させるので、それらは入力記号の再構築のために使用され得る。

【０１５５】

上記の例のほとんどにおいて、入力および出力記号の群は、同じ数のビットとして符号化し、出力記号の各群は、１つのパケットに配置される（１パケットは、送信の１単位であり、完全に受信されるか、または完全に損失される）。いくつかの実施形態において、通信システムは、各パケットが出力記号の数個の群を含むように改変される。次に、出力記号値のサイズは、（変化し得る）群サイズを含む多くのファクタに基づいて、最初にファ

イルを入力記号に分割した際の入力記号のサイズによって決定されるサイズに設定される。符号化プロセスは、実質的に変更されないままであり得る。ただし、出力記号の群が、各パケットが受信される際に、ひとまとまりで到着し得ることを除く。

【０１５６】

入力記号および出力記号サイズの設定は、通常、ファイルのサイズおよび出力記号の群が送信される通信システムによって決定される。例えば、通信システムは、データのビットを規定サイズのパケットにグループ化するか、または他の方法でビットをグループ化する場合、記号サイズの設計は、パケットまたはグループ化サイズから開始される。そこから、設計者は、出力記号の何個の群を１つのパケットまたは群で送信するか、かつ（変化し得る）その群サイズとともに出力記号サイズを決定し得る。簡単のために、設計者は、おそらく入力記号サイズを出力記号サイズと等しくなるように設定し得るが、入力データが異なる入力記号サイズをより便利にする場合は、それが使用され得る。

【０１５７】

入力記号サイズを決定する際の別のファクタは、入力記号数Ｋが受信オーバーヘッドを最小に保つために十分大きくなるように入力記号サイズを選択することである。例えば、Ｋ＝１０，０００にすると、平均の受信オーバーヘッドが、５％～１０％の適度なゆらぎを有するようになり、他方Ｋ＝８０，０００にすると、平均の受信オーバーヘッドは１％～２％で、そのゆらぎが非常に小さくなる。例として、Ｋ＝８０，０００で１，０００，０００回の試行を含む１つのテストにおいて、受信オーバーヘッドは、４％を決して超えなかった。

【０１５８】

上記符号化プロセスは、元のファイルに基づき、出力記号の群を含むパケットの１ストリームを生成する。出力記号の群は、ファイルの符号化形態またはより簡潔には符号化ファイルを保持する。ストリーム中の出力記号の各群は、出力記号の他のすべての群と独立して生成され、生成され得る出力記号の群の数には下限および上限がない。キーは、出力記号の各群に関連付けられる。そのキーおよび入力ファイルの内いくつかの内容は、出力記号の群の値を決定する。出力記号の連続生成された群は、連続するキーを有する必要はなく、いくつかのアプリケーションにおいては、キーのシーケンスをランダムに生成するか、またはシーケンスを擬似ランダムに生成することが好ましくあり得る。

【０１５９】

群連鎖反応符号化は、元のファイルがＫ個の等サイズの入力記号に分割され得、かつ各出力記号値が入力記号値と同じ長さを有する場合に、そのファイルが平均でＫ＋Ａ個の出力記号から回復され得る（ここでＡはＫに比較して小さい）という性質を有する。例えば、Ａは、Ｋ＝２０，０００の場合５００であり得る。出力記号の特定の群がランダムまたは擬似ランダムな順序で生成され、かつ送信中の出力記号の特定の群の損失が任意に仮定されるので、入力ファイルを回復するために必要な出力記号の群の実際の数にはいくらかの小さなばらつきが存在する。Ｋ＋Ａ個の出力記号を含む特定の集合パケットが、入力ファイル全体を復号するのに十分でないようないくつかの場合において、受信器が１つ以上のパケットのソースからより多くのパケットを集め得る場合は、入力ファイルは、依然として回復可能である。

【０１６０】

出力記号の群の数は、Ｉの解像度によってのみ制限されるので、Ｋ＋Ａ個より十分に多くの出力記号の群が生成され得るべきである。例えば、Ｉが３２ビット数である場合、四十億個の異なる出力記号の群が生成され得る。他方、そのファイルは、Ｋ＝５０，０００個の入力記号を含み得る。実用上は、これら四十億個の出力記号の群のうちのほんのわずかな個数の群だけが生成および送信され得、そして入力ファイルが、出力記号の可能な群の非常に小さな割合を用いて回復され得ることがほぼ確実であり、そして入力ファイルが、Ｋ個よりわずかに多くの出力記号を用いて回復され得る可能性は非常に高い（入力記号サイズは、出力記号サイズと同じであると仮定する）。

【０１６１】

出力記号の各群を生成するために必要な算術演算の平均数は、ｌｏｇＫに比例し、そのため、入力ファイルを復号および回復するために必要な算術演算の総数は、ＫｌｏｇＫに比例する。上に示すように、入力ファイルを格納するために必要なメモリよりもほんのわずかに多くのメモリを使用する効率的な復号化プロセスが存在する（一般に、約１５％多い）。上記の数は、従来から公知の符号化技術と比較して演算および記憶装置において著しい減少を示す。

【０１６２】

例えば、標準的なリード－ソロモン符号は、通信アプリケーションのための標準的な符号である。標準的なリード－ソロモン符号を用いると、群連鎖反応符号化を用いるように、入力ファイルはＫ個の入力記号に分割されるが、リード－ソロモン符号におけるそのＫ個の入力記号は、Ｎ個の出力記号に符号化される。ここで、Ｎは一般に、符号化プロセスが開始する前に固定される。これは、出力記号の群の不確定数を考慮する本発明と対照的である。

【０１６３】

出力記号の群の不確定数を有する１つの利点は、受信者が予想より多くの出力記号の群を逸する場合、不十分なチャネルによるか、または出力記号のいくつかの群がすでに通過した後で開始する受信者により、受信者は、単に少し長く受信し、出力記号のより多くの群を取り上げ得ることである。別の利点は、受信者が複数のエンコーダから生成される出力記号の群を集め得るので、各エンコーダは、入力記号を復号するために必要な出力記号の群のほんの小さな割合だけを提供する必要があり、１つのエンコーダからの出力記号の群の数は、どれだけのエンコーダが出力記号の群を受信者に供給しているかに依存し得ることである。

【０１６４】

標準的なリード－ソロモン符号はまた、符号化および復号化の両方について、連鎖反応符号より実質的に多くの時間を必要とする。例えば、標準的なリード－ソロモンを用いて各出力記号を生成するために必要な算術演算の数は、Ｋに比例する。標準的なリード－ソロモン符号を復号化するために必要な算術演算の数は、どの出力記号が受信者に到着したかに依存するが、一般に、そのような演算の数は、（Ｎ－Ｋ）・Ｋに比例する。このように、実用上は、ＫおよびＮの許容値は、非常に小さく、数十のオーダー、おそらくわずか数百までのオーダーである。例えば、クロス－インターリーブ（Ｃｒｏｓｓ－Ｉｎｔｅｒｌｅａｖｅｄ）リード－ソロモン符号は、コンパクトディスク（ＣＤ）およびＣＤ－ＲＯＭ上で使用される。ＣＤの場合、１つの標準的な符号はＫ＝２４およびＮ＝２８を使用し、別の標準的な符号はＫ＝２８およびＮ＝３２を使用する。ＣＤ－ＲＯＭの場合、１つの標準的な符号はＫ＝２４およびＮ＝２６を使用し、別の標準的な符号はＫ＝４３およびＮ＝４５を使用する。ＭＰＥＧファイル（ＭＰＥＧは、ビデオおよびオーディオストリームのためのファイルフォーマットである）の衛星送信のために使用される標準的なリード－ソロモン符号は、Ｋ＝１８８およびＮ＝２０４を使用し、一般に、このような大きな値は、専用のハードウェアを必要とする。

【０１６５】

ｃＫｌｏｇＫ時間での符号化およびｃ’Ｋ（ｌｏｇＫ）^{2}時間での復号化を可能にする標準的なリード－ソロモン符号のより高速な実施例が存在することが公知であるが、ｃおよびｃ’が過度に大きな定数であるので、Ｋの非常に大きな値以外全てに対しては、これらの実施例は標準的なリード－ソロモン符号の他の実施例よりも遅くなる。すなわち、数千または数万のＫの値に対して有効なクロスオーバーポイントが存在する。したがって、クロスオーバーポイントより下のＫの値の場合、標準的なリード－ソロモン符号の他の実施例は、より高速である。そのより高速な実施例は、クロスオーバーポイントより高いＫの値の他の標準的なリード－ソロモン符号よりも高速であるが、そのより高速な実施例は、Ｋのそれらの値で群連鎖反応符号よりも数桁遅い。

【０１６６】

その速度制限のために、標準的なリード－ソロモン符号に対しては、一般に、小さい値の

ＫおよびＮだけが実現可能である。したがって、大きなファイルに関してそれらを使用することにより、そのファイルを多くのサブファイルに分割し、各サブファイルを別々に符号化することを必要とする。そのような分割は、符号の有効性を低減し、送信中のパケット損失を防止する。

【０１６７】

標準的なリード－ソロモン符号の１つの特徴は、Ｋ個の異なる出力記号のいずれもが、入力ファイルを復号化するために受信者によって使用され得ることである。おそらく、少なくともＫ個の出力記号が、任意の入力ファイルを復号化するために必要とされ、したがって、標準的なリード－ソロモン符号はこの点で最適であるということが証明できる。なぜなら、Ｋはまた、入力ファイルを復号化するために必要な出力記号の最大数であるからである。対照的に、群連鎖反応符号化は一般に、Ｋ＋Ａ個の出力記号（Ａは、適切に選択されたＫに比較して小さい）または合計でＫ個よりわずかに多い出力記号を必要とする。上記のネットワークアプリケーションにおいては、おそらくＡ個のさらなる記号が必要であるというこの欠点は、速度の利点および途切れなくより大きなファイルを処理する能力によって極めて影を薄くしている。

【０１６８】

（基本通信システムの変形例）

１つのチャネルに対する本発明による通信システムの実施形態は、上記に詳細に説明された。これらの実施形態の要素は、１つより多いチャネルの有利な使用に拡張し得る。

【０１６９】

図２２～２３は、図１に示すような通信システムを組み込んだ２つのコンピュータの間のシステムを示す。第１の例（図２２）は、入力ファイル２２１０を受信者コンピュータ２２２０にネットワーク２２３０を介して送信する送信者コンピュータ２２００を有する。第２の例（図２３）は、入力ファイル２３１０を受信者コンピュータ２３２０（図には１つだけを示す）へ無線チャネル２３３０を介して放送する送信者コンピュータ２３００を有する。ネットワーク２３３０の代わりに、インターネットのワイヤなどの任意の他の物理的通信媒体を使用し得る。無線チャネル２３３０は、無線ラジオチャネル、ページャーリンク、衛星リンク、または赤外線リンクなどであり得る。図２３に示す構成はまた、１送信者が入力ファイルを多くの受信者に送信する場合、受信者が入力ファイルを多くの送信者から得る場合、または多くの受信者が入力ファイルを多くの送信者から得る場合に、有線または無線媒体とともに使用され得る。

【０１７０】

上記説明を読めば明らかなように、上記の群連鎖反応符号化スキームを使用して、コンピュータネットワーク、インターネット、モバイル無線ネットワークまたは衛星ネットワークなどの損失のある送信媒体を介して、ファイルを所望の性質を有するパケット化されたデータのストリームとして送信し得る。そのような所望の性質の１つは、復号化エージェントは、一旦そのストリームから十分に多くのパケットの任意のセットを受信すると、元のファイルを極めて速やかに再構築し得ることである。群連鎖反応符号化はまた、マルチキャストまたは放送設定において１送信エージェントが、そのファイルを複数の受信エージェントに送信する場合などの、多くのエージェントが関与する状況において有用である。

【０１７１】

群連鎖反応符号化スキームはまた、複数の送信エージェントが同じファイルを複数の受信エージェントに送信する状況においても有用である。これにより、ネットワークインフラストラクチャの局所的な失敗に対して向上したロバスト性を可能にし、受信エージェントが受信するパケットの送信エージェントを１送信エージェントから別の送信エージェントに途切れなく変更でき、かつ受信エージェントが同時に１つより多い送信エージェントから受信することによってそれらのダウンロードを高速化できる。

【０１７２】

１つの局面において、上記の群連鎖反応符号化プロセスは、ホログラフィーイメージのデ

ジタル等価物を実行する。ここで、送信の各部分は、送信されるファイルのイメージを含む。ファイルがメガバイトのファイルである場合、ユーザは、ただ送信されるストリームを利用していずれの任意のメガバイトに値するデータ（それに加えていくらかの余分のオーバーヘッド）を得、そのメガバイトから元のメガバイトファイルを復号し得る。

【０１７３】

群連鎖反応符号化は、送信されたストリームからデータをランダムに選択して動作するので、ダウンロードは、スケジュールされる必要はなく、一貫している必要もない。群連鎖反応符号化を用いたビデオ・オン・デマンドの利点を考察する。特定のビデオが、受信器と送信器との間で調整することなく特定のチャネル上を連続するストリームとして放送され得る。受信器は、興味のあるビデオ用の放送チャネルに単に同調し、送信がいつ開始したかとか、または放送の損失部分のコピーをどのように得るかを理解する必要なしに、元のビデオを再構築するために十分なデータをキャプチャする。

【０１７４】

これらの概念を、図２４～２５に例示する。図２４は、１つの受信器２４０２が３つの送信器２４０４（それぞれ、「Ａ」、「Ｂ」および「Ｃ」で示す）から３つのチャネル２４０６を介してデータを受信する構成を例示する。この構成を使用して、受信器が利用可能な帯域幅を３倍にし得るか、またはいずれか１つの送信器からファイル全体を得るための十分な長さで利用可能でない送信器を扱い得る。図示のように、各送信器２４０４は、値のストリームＳ（Ｉ）を送信する。各Ｓ（Ｉ）値は、キーＩおよび出力記号値の１群Ｂ（Ｉ）＝Ｂ＿１（Ｉ），．．．，Ｂ＿ｂ（Ｉ）を表し、その使用については上記している。例えば、値Ｓ（$n_A + i_A n'_A$）は、値$n_A + i_A n'_A$を有するキーであり、それに値Ｂ（$n_A + i_A n'_A$）を有する出力記号の群が続く。ここで、n_Aおよびn'_Aは、送信器２４０４（Ａ）に関連付けられたランダムに選択された数であり、i_Aは、送信器２４０４（Ａ）から送信されるパケットのシーケンス番号である。１つの送信器からのキーのシーケンスは、好ましくは他の送信器からのキーのシーケンスと異なっているので、送信器は誤りを重複させない。これは、各送信器で使用されるキーシーケンスがその送信器に特定の情報の関数である（例えば、送信器２４０４（Ａ）のキーシーケンスはn_Aおよびn'_Aに依存する）という事実によって図２４に例示する。

【０１７５】

送信器２４０４は、作業を重複させないように同期または調整される必要がないことに留意されたい。実際、調整することなく各送信器は、大きく異なるキーのシーケンスを送信する可能性がある（すなわち、$n_A + i_A n'_A \neq n_B + i_B n'_B \neq n_C + i_C n'_C$、ここで、$i_A$、$i_B$、$i_C$は、それぞれ衛星ＳＡＴ　Ａ、ＳＡＴ　Ｂ、ＳＡＴ　Ｃによって送信されたパケットのシーケンス番号であり、n_A、n'_A、n_B、n'_B、n_C、n'_Cは、それぞれＳＡＴ　Ａ、ＳＡＴ　Ｂ、ＳＡＴ　Ｃに関連付けられたランダムに選択された数である）。

【０１７６】

ランダムに選択したＫ／ｂ＋Ａ個の出力記号の群、おそらくそれに伴うことＧ個の余分の出力記号の群の数束を使用して、入力ファイルが回復され得るので、調整されない送信は、重複的である代わりに付加的である。

【０１７７】

この「情報付加性」を図２５に再度例示する。そこで、１つの入力ファイル２５０２のコピーが複数の送信器２５０４に提供される（そのうちの２つを図に示す）。送信器２５０４は、入力ファイル２５０２の内容から生成された出力記号の群をチャネル２５０６を介して受信器２５０８に独立に送信する。各送信器が記号生成のために異なるセットのキーを使用する場合、そのストリームは、重複的ではなく独立かつ付加的（すなわち、それらは入力記号を回復するために使用される情報プールに付加する）である可能性がある。図示の２つの各送信器は、受信器のデコーダが入力ファイル全体を回復する事ができる前に（Ｋ／ｂ＋Ａ）／２個の出力記号の群を送信することだけが必要であり得る。

【０１７８】

２つの受信器および２つの送信器を使用して、受信器部２５１０によって受信される総情報量は、１つのチャネル２５０６を介して利用可能な情報の４倍の大きさであり得る。情報量は、例えば、送信器が両方の受信器に同じデータを放送する場合は、１つのチャネルの情報の４倍より少なくてもよい。その場合、受信器部２５１０での情報量は、特にデータがチャネルにおいて損失される場合、元のファイルを直接送信する事によって達成され得る速度の少なくとも２倍である。送信器が１個の信号しか放送しないが、受信器が異なる時間に見える所にある場合は、各送信器を受信する１つより多い受信器を有する利点がある。図２５において、受信器部２５１０は、図１に示す受信器１５０、デコーダ１５５、キー発生器１６０および入力ファイルリアセンブラ１６５の機能と同様の機能を実行する。

【０１７９】

いくつかの実施形態において、入力ファイル２５０２は、２つのエンコーダを有する１つの計算デバイスにおいて符号化されるので、計算デバイスは、１つの送信器に対して１つの出力を、他方の送信器に対して別の出力を提供し得る。これらの例の他の変形例は、この開示を読む際に明らかとなる。

【０１８０】

本明細書中に記載の符号化装置および方法はまた、他の通信状況において使用され得、インターネットなどの通信ネットワークに制限されない。例えば、コンパクトディスク技術もイレーズおよび誤り訂正符合を使用して、傷の入ったディスクの問題に対応し、そこに情報を格納する際に群連鎖反応符号の使用により利益を受け得る。別の例として、衛星システムは、送信のためのパワー要件をトレードオフするためにイレーズ符号を使用し得、パワーを低減することによってより多くの誤りを意図的に考慮する。このアプリケーションにおいて、群連鎖反応符号化は有用であり得る。また、イレーズ符号は、情報記憶の信頼性のためのＲＡＩＤ（独立ディスクの冗長配列）システムを開発するために使用されてきた。したがって、本発明は、符号を使用して損失または誤りの可能性のあるデータの問題に対応する、上記の例などの他のアプリケーションにおける有用性を証明し得る。

【０１８１】

いくつかの好ましい実施形態において、上記通信プロセスを実行するための命令のセット（またはソフトウェア）が、損失の可能性のある通信媒体を介して通信する２つ以上の多目的計算機に提供される。機械の数は、１送信者および１受信者から任意の数の送信機および／または受信機の範囲であり得る。機械を接続する通信媒体は、有線、光学、または無線などであり得る。上記通信システムは、本記載から明らかとなるべき多くの使用を有する。

【０１８２】

上記は、例示的であって、限定的ではない。本発明の種々の変形例は、当業者にとって、本開示を読むことによって明らかとなり得る。したがって、本発明の範囲は、上記を参照して決定されるのではなく、その代りに添付の特許請求の範囲およびその等価物の完全な範囲を参照して決定されるべきである。

【０１８３】

（結論として本発明は以下を提供する）

エンコーダは、データの入力ファイルおよびキーを使用して、出力記号の１群を生成する。キーＩを有する出力記号の１群は、生成されるべきその出力記号の群の重みＷ（Ｉ）を決定し、Ｉの関数にしたがってその群と関連付けられるＷ（Ｉ）個の入力記号を選択し、選択されたＷ（Ｉ）個の入力記号の所定の値関数Ｆ（Ｉ）から出力記号値Ｂ（Ｉ）を生成することによって生成される。エンコーダは、繰り返し呼び出されて複数の出力記号の群または複数の出力記号を生成し得る。出力記号の群は、一般に互い独立であり、制限のない数（Ｉの解像度には影響される）が、必要に応じて生成され得る。デコーダは、生成された出力記号の一部またはすべてを受信する。入力ファイルの復号化に必要な出力記号の数は、ファイルを構成する入力記号の数に等しいか、またはわずかに大きい。ここで入力記号および出力記号は同じビット数のデータを表すと仮定する。図２６ａ～ｅは、添付書

類Ａの例示であり、重み分布を実施するためのプログラムのソースコードリストである。

【図面の簡単な説明】

【図１】図１は、本発明の１つの実施形態による通信システムのブロック図である。

【図２】図２は、図１のエンコーダをより詳細に示すブロック図である。

【図３】図３は、出力記号の１群が関連付けられた入力記号の１セットからどのように生成され得るかの例示である。

【図４】図４は、図１に示す通信システムにおいて使用され得るような基本デコーダのブロック図である。

【図５】図５は、別のデコーダのブロック図である。

【図６】図６は、出力記号の１セットの群から入力記号を回復するために、図５に示すデコーダなどのデコーダによって使用され得るプロセスのフローチャートである。

【図７】図７は、出力記号の受信された群をオーガナイズするために、図５に示す受信オーガナイザなどの受信オーガナイザによって使用され得るプロセスのフローチャートである。

【図８ａ】図８ａは、出力記号の受信された群を処理するために、図５に示す回復プロセッサなどの回復プロセッサによって使用され得るプロセスのフローチャートである。

【図８ｂ】図８ｂは、図８ａのプロセスの変形例の一部分のフローチャートであり、図８ｂは、保留される処理を含む回復プロセスにおいて実行されるステップを示す。

【図８ｃ】図８ｃは、図８ａのプロセスの変形例の一部分のフローチャートであり、図８ｃは、保留される処理を示す。

【図９】図９は、図２の連想装置をより詳細に示すブロック図である。

【図１０】図１０は、出力記号の群を用いて入力記号の関連付けを速やかに決定するために、図９に示す連想装置などの連想装置によって使用され得る１つのプロセスのフローチャートである。

【図１１】図１１は、図２の重みセレクタをより詳細に示すブロック図である。

【図１２】出力記号の所定の群の重みを決定するために、図１１に示す重みセレクタなどの重みセレクタによって使用され得るプロセスのフローチャートである。

【図１３】図１３は、特に効率的である必要のないデコーダのための復号化のプロセスのフローチャートである。

【図１４】図１４は、より効率的なデコーダのブロック図である。

【図１５】図１５は、図１２～１３を参照して説明される復号化よりも効率的に復号化するための、図１４のデコーダを使用して実施され得るような復号化のためのプロセスのフローチャートである。

【図１６】図１６は、図１５の復号化プロセスのための、ドキュメントおよび受信された出力記号の群の例を例示する図である。

【図１７】図１７は、図１５に示す復号化プロセス中のデコーダにおけるテーブルの内容を例示する。

【図１８】図１８は、図１５に示す復号化プロセス中に使用され得るような重みソートテーブルの内容を例示する図である。

【図１９】図１９は、図１５に示す復号化プロセス中に形成され得る実行リストを例示する。

【図２０】図２０は、理想の分布に対する、復号可能なセットサイズ対回復された入力記号数のプロットの形態の回復プロセスの進行を例示する。

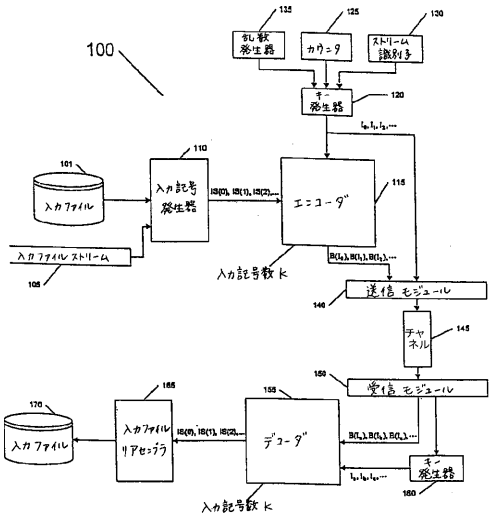
【図２１】図２１は、ロバストな重み分布に対する、復号可能なセットサイズ対回復された入力記号数のプロットの形態の回復プロセスの進行を例示する。

【図２２】図２２は、上記図に例示するようなエンコーダおよびデコーダを使用して、１送信者（送信器）と１受信器との間の１対１通信システムの例示である。

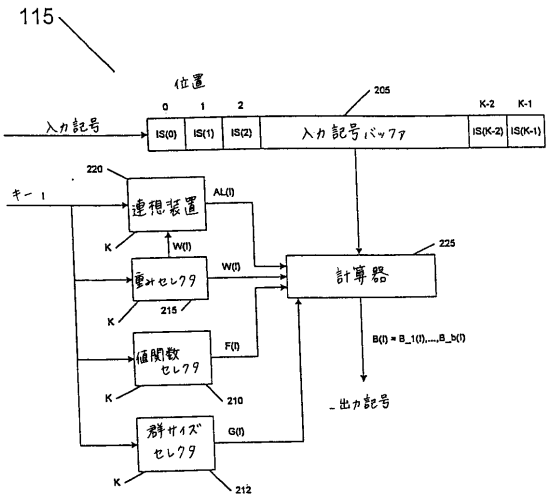
【図２３】図２３は、上記図に例示するようなエンコーダおよびデコーダを使用して、１送信者と複数の受信器（そのうちの１つだけを図示する）との間の放送通信システムの例示である。

【図２４】図２４は、１受信器が、複数の通常独立した送信者から出力記号の群を受信する場合の、本発明の１つの実施形態による通信システムの例示である。
【図２５】図２５は、複数の独立であり得る受信器が、複数の通常独立した送信者から出力記号の群を受信して、１受信器および／または１送信者だけが使用される場合より少ない時間で入力ファイルを受信する場合の、本発明の１つの実施形態による通信システムの例示である。
【図２６ａ】図２６ａは、重み分布を実施するためのプログラムのソースコードリストである添付書類Ａの例示である。
【図２６ｂ】図２６ｂは、重み分布を実施するためのプログラムのソースコードリストである添付書類Ａの例示である。
【図２６ｃ】図２６ｃは、重み分布を実施するためのプログラムのソースコードリストである添付書類Ａの例示である。
【図２６ｄ】図２６ｄは、重み分布を実施するためのプログラムのソースコードリストである添付書類Ａの例示である。
【図２６ｅ】図２６ｅは、重み分布を実施するためのプログラムのソースコードリストである添付書類Ａの例示である。

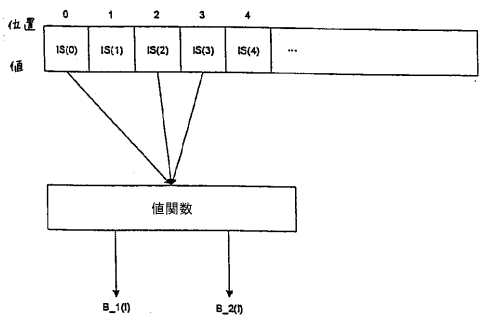
【図１】



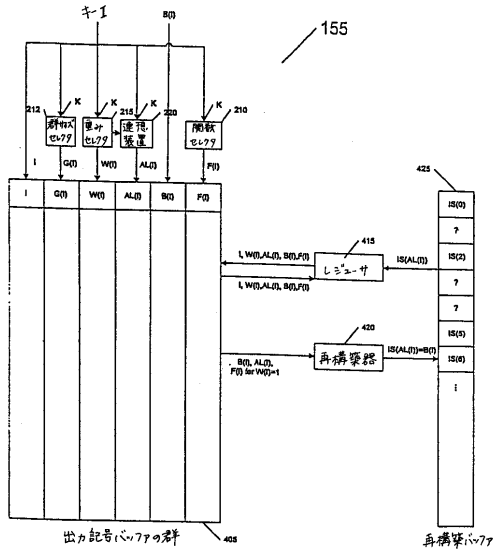
【図２】



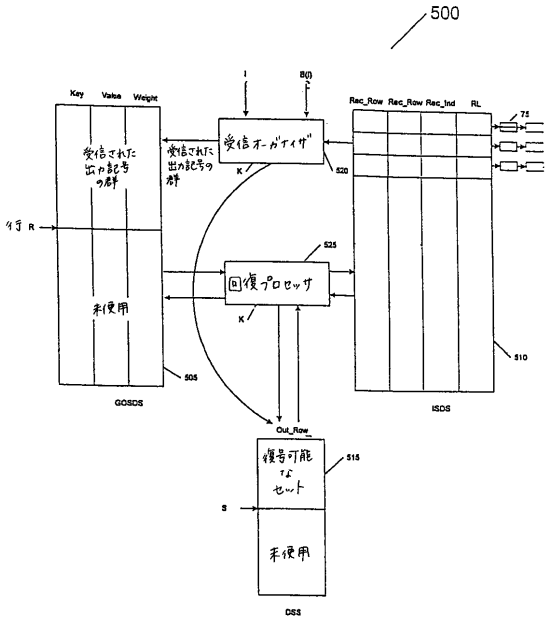
【図３】



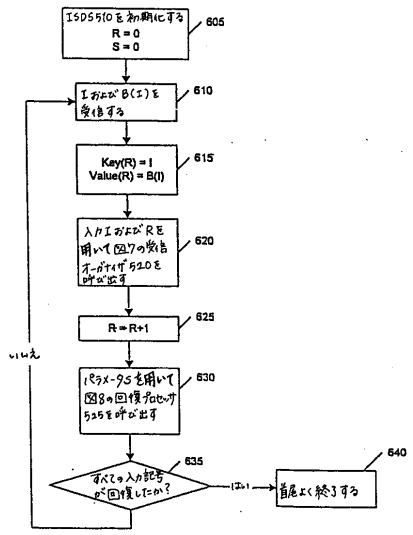
【図４】



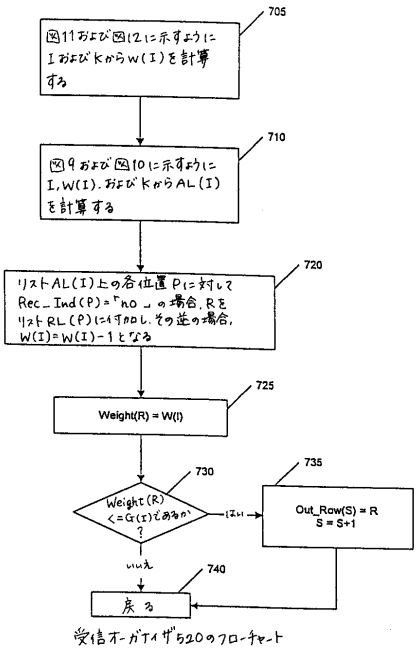
【図５】



【図６】

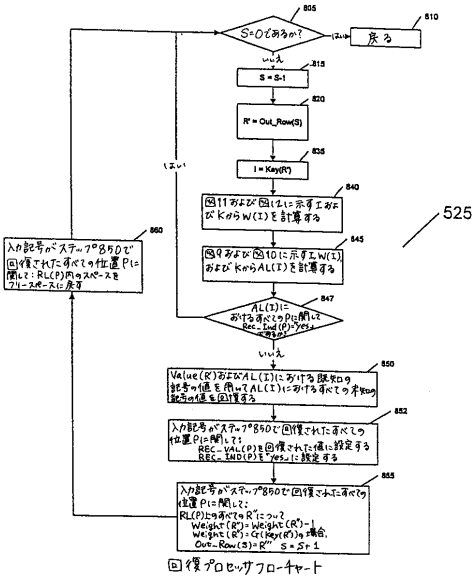


【図７】



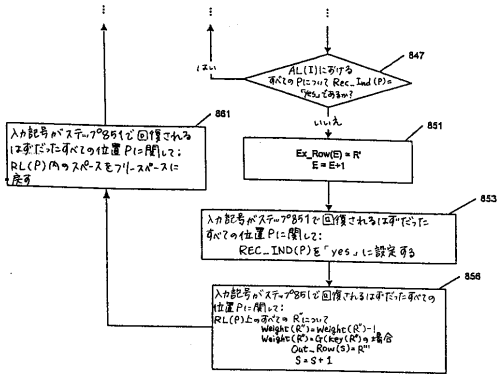
受信オーガナイザ520のフローチャート

【図８ａ】



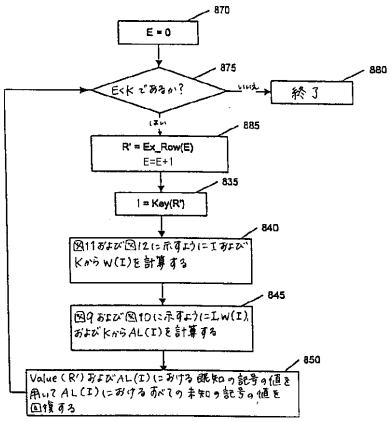
回復プロセッサフローチャート

【図８ｂ】



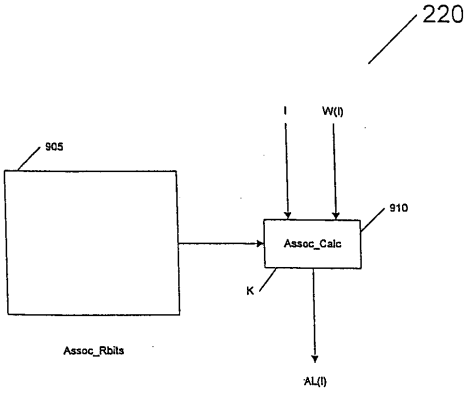
実行スケジュールの準備

【図８ｃ】



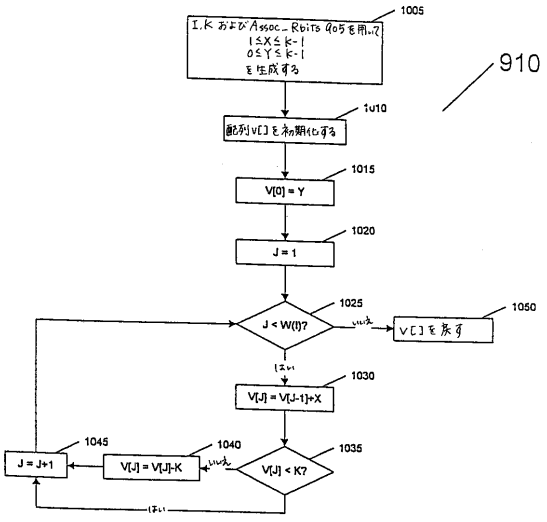
実行スケジュールのプロセス

【図９】



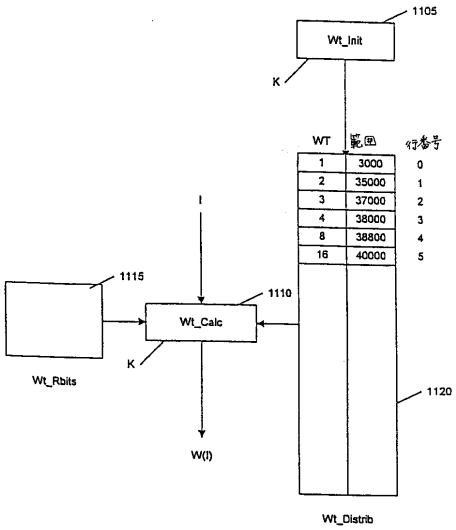
連想装置

【図１０】

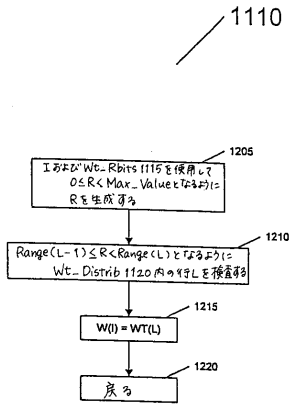


Assoc_Calc

【図１１】

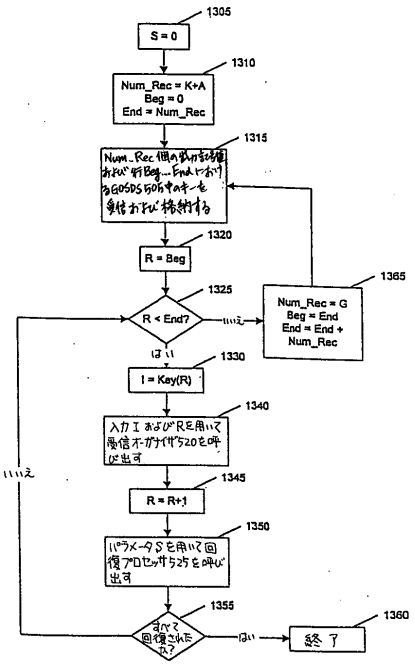


【図１２】



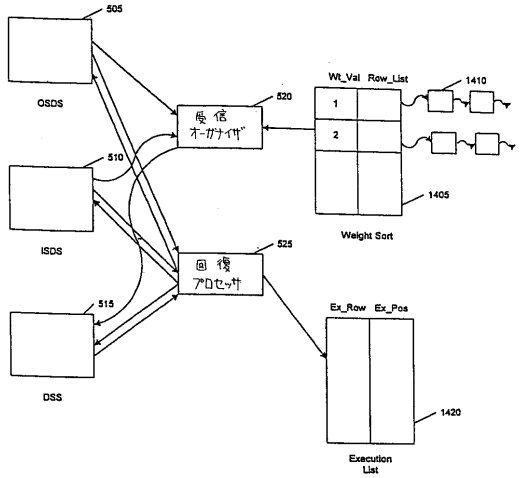
Wt_Calc

【図１３】

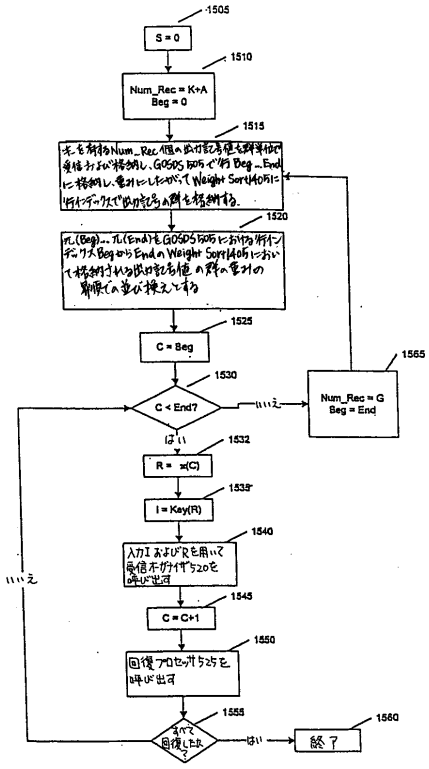


レイジーデコーダ

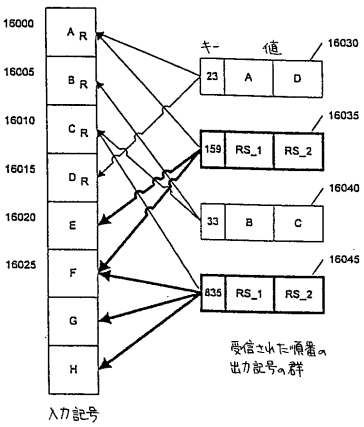
【図１４】



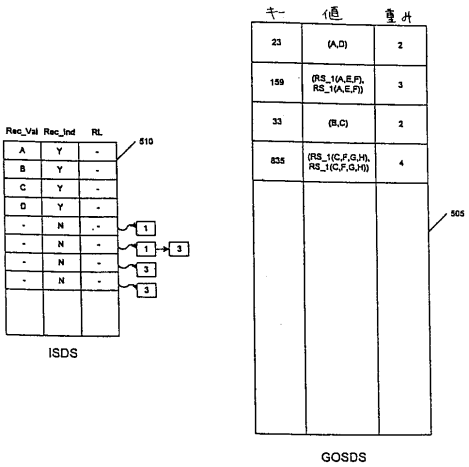
【図１５】



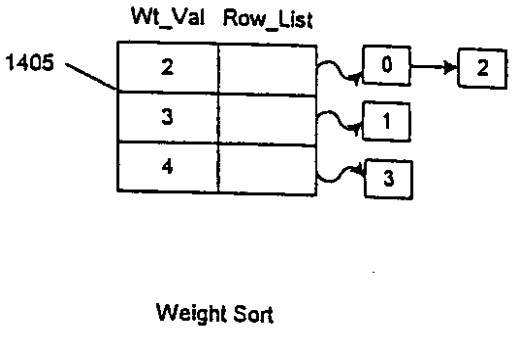
【図１６】



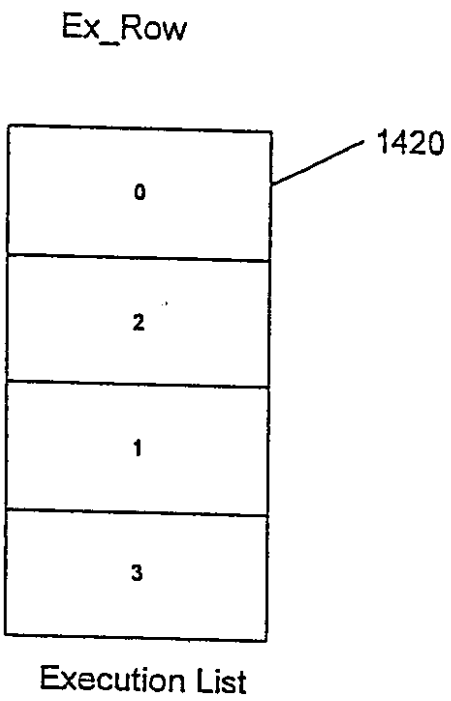
【図１７】



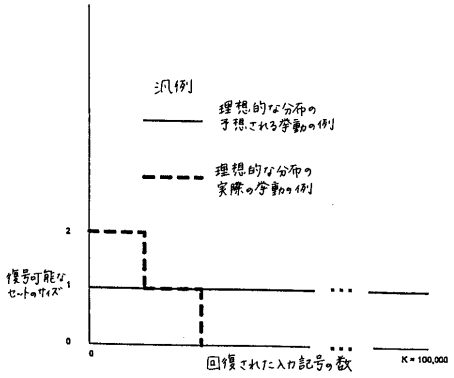
【図１８】



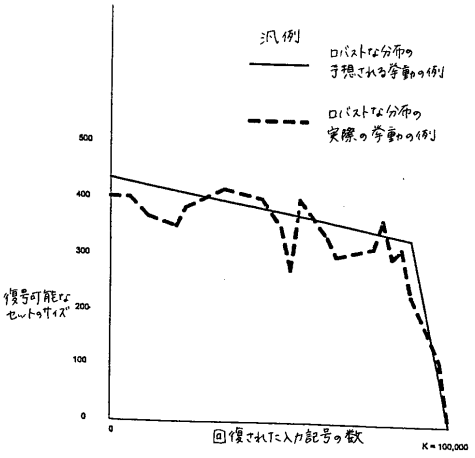
【図１９】



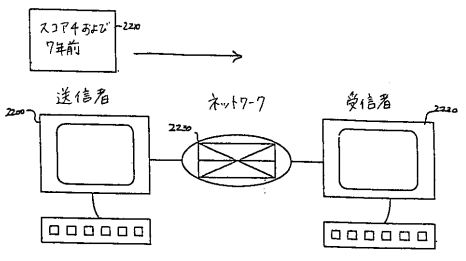
【図２０】



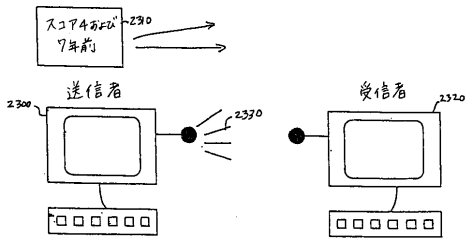
【図２１】



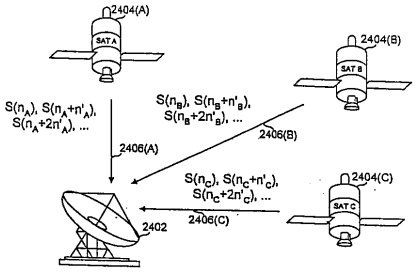
【図２２】



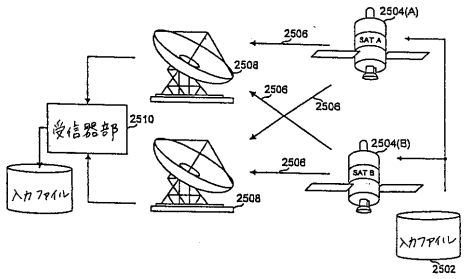
【図２３】



【図２４】



【図２５】



【図２６ａ】

添付書類 A. ソースコードリスト

1. DFSolitonDistribution.cpp

```
//////////////////////////////////////////////////////////////////////
// DFSolitonDistribution.cpp:
//                              CDFSolitonDistribution   クラスの実施例
//
//  出力記号の群の重みの確率分布を構築器において計算し、
//  次に何がキーに関連付けられるかを決定する場合に
//  キーデコーダを参照する
//////////////////////////////////////////////////////////////////////

#include "DFSolitonDistribution.h"
#include "DFRandomBits.h"
#include <math.h>

//////////////////////////////////////////////////////////////////////
//        構築 / 破壊
//////////////////////////////////////////////////////////////////////

CDFSolitonDistribution::CDFSolitonDistribution(
int nSymbols,
int nArity,
int nRippleTargetSize,
int nStartRippleSize)
{
    m_nSegments    = nSegments;

    // RおよびSの値が特定されていなければ、記号数の
    // 四乗根および平方根に基づいた式を有する
    // .hファイルから定数を使用する

    m_nRippleTargetSize  = nRippleTargetSize;

    if (!m_nRippleTargetSize)
        m_nRippleTargetSize =
            int(knRAdd + kdRFactor * sqrt(sqrt(m_nSymbols/nArity)));

    m_nStartRippleSize = nStartRippleSize;

        if (!m_nStartRippleSize)
        m_nStartRippleSize = int(kdSFactor * sqrt(m_nSymbols/nArity));
```

【図２６ｂ】

```
    //////////////////////////////////////////////////////////
    // ソリトン分布のパラメータの計算:
    //
    // これは重みN/R, 2N/R,… Nと先細になった密度を
    // 有する改変されたロバスト分布である
    //////////////////////////////////////////////////////////

    m_nModulus = 0x1 << knPrecision ;

    // ロバスト分布の前回の変形例に対して
    // 値として「-1」を使用する

m_nRobustKinds = m_nSymbols/m_nRippleTargetSize - 1 - m_nArity;
if (m_nRobustKinds < 1)                    m_nRobustKinds = 1;
if (m_nRobustKinds > m_nSymbols)           m_nRobustKinds = m_nSymbols;
// ロバスト分布の前回の変形例に対して
// m_nTailKinds = 0
m_nTailKinds = 1;
for (int d = m_nRippleTargetSize; d > 1; d /= 2)
    m_nTailKinds++;    //          log_2(RippleTargetSize) + 1 になる

m_nKinds = m_nRobustKinds + m_nTailKinds;
if (m_nKinds > m_nSymbols)
{
    m_nTailKinds = m_nSymbols - m_nRobustKinds;
    m_nKinds = m_nSymbols;
}

m_anKindWeight             = new int[m_nKinds];
m_anKindCumulativeDensity  = new int[m_nKinds];

// adKindFraction は正規化されていない分布
double* adKindFraction     = new double[m_nKinds];

// 重み    m_nArity 出力記号          :
adKindFraction[0] =
          double(nArity*m_nStartRippleSize)/double(m_nSymbols);
m_anKindWeight[0] = m_nArity;

// ロバストソリトン分布による密度
for (int i=1; i < m_nRobustKinds; i++)
{
int nWt = i + m_nArity;

adKindFraction[i] = adKindFraction[i-1] +
    double(nArity)/(double(nWt) * double(nWt-1) *
    (1.0 - (double((nWt)*m_nRippleTargetSize)/double(m_nSymbols))));

m_anKindWeight[i] = nWt;
}

int nRPower = 1;
int j;
for (i = 0; i < m_nTailKinds; i++)
    nRPower *= 2; //nRPowerは2を掛けたもの      > m_nRippleTargetSize
```

【図２６ｃ】

```
// 分布の端が先細りの度の密度:
// 重みは合計m_nSegmentsから2倍減り.
// 密度は重みに反比例する

// jは2からnRPowerまで実行する
for (i=m_nRobustKinds, j=2; i < m_nKinds; i++, j*=2)
{
    adKindFraction[i] = adKindFraction[i-1]
        + kdTFactor/(m_nArity*m_nArity) *
            double(nRPower)/double(j*m_nSymbols);
    m_anKindWeight[i] = (j*m_nSymbols)/nRPower;
}

//  m_nModulusに正規化する

    for (i=0; i < m_nKinds; i++)
        m_anKindCumulativeDensity[i] = int(adKindFraction[i] *
                        double(m_nModulus)/adKindFraction[m_nKinds-1]);

    delete[] adKindFraction;

    // 最大重みおよび平均重みを計算する
    m_nMaxWeight = 0;
    for (i=0; i < m_nKinds; i++)
        if (m_anKindWeight[i] > m_nMaxWeight)
                        m_nMaxWeight = m_anKindWeight[i];

    ComputeAverageWeight();
}

CDFSolitonDistribution::~CDFSolitonDistribution()
{
    if (m_anKindWeight)             delete[] m_anKindWeight;
    if (m_anKindCumulativeDensity)  delete[] m_anKindCumulativeDensity;
}

void CDFSolitonDistribution::ComputeAverageWeight()
{
    int i ;
    double dTemp ;
    dTemp = double(m_anKindWeight[0]) *
                double(m_anKindCumulativeDensity[0]) ;
    for (i=1; i < m_nKinds; i++)
        dTemp += double(m_anKindWeight[i]) *
                    double(m_anKindCumulativeDensity[i] -
                        m_anKindCumulativeDensity[i-1]) ;
    m_dAverageWeight = (dTemp/m_nModulus) + 1;
}
```

【図２６ｄ】

2. DFSolitonDistribution.h

```
////////////////////////////////////////////////////////////////////////////
// DFSolitonDistribution.h:
//                                  CDFSolitonDistribution クラスに対するインターフェース
//
// 注意：クラス CDFDominoKeyDecoder は、所与の出力記号キーの
// 近傍の重みおよびクラスを計算する。重みに対する確率分布がこのクラスを
// 参照し、選択するために使用されるランダム整数が
// CDFRandomBits クラスを参照する
////////////////////////////////////////////////////////////////////////////

#include "DFRandomBits.h"
// クラス CDFRandomBits;

////////////////////////////////////////////////////////////////////////////
//
// 分布の計算に使用される定数
//
////////////////////////////////////////////////////////////////////////////

// 分布のビット単位の算術精度
// 分布密度は 2 のこの値の累乗に尺度化される
const int      knPrecision     = 28;

// 「リップルターゲットサイズ」である R の計算に関連する定数
// この値は「ロバストソリトン分布」における予想される「リップル
// サイズ」である。R は値 kdRFactor * N の四捨五入 + knRadd から
// 得られ、ここで N は入力記号数（セグメント）である
const double   kdRFactor = 2.1;
const int      knRAdd    = 2;

// S は復号化開始時どのリップルの予想サイズであり、
// N 個の記号が受信される場合の重み 1 の出力記号の
// 予想される数を意味する
// S = kdSFactor * sqrt(N)
const double   kdSFactor = 1.4;

// 分布の尾は、N, N/2, … N/R 度まで下がった点どの
// より高い次数の密度である。
// 分布密度は、比例定数 kdTFactor で記号重みに反比例する。
// 得られる分布は、なお上記の精度に
// 尺度化される

// 先の値 1.6 を使用したが、それを 1.8 に増加し、最後の 1 つの記号を
// 待つのに長時間を費す「孤立地（outliers）」を切り落とした。
// 次に、2,4 にまで増加し、40000 個の記号に生じる（百万のうち）
// 1 つの孤立地に対応した。
const double   kdTFactor = 2.4;

class CDFSolitonDistribution
{
    friend class CDFDominoKeyDecoder;

public:
```

【図２６ｅ】

```
    ////////////////////////////////////////////////////////////////////////
    // 出力記号キーおよび入力記号の位置
    //
    // このクラスは（前に「インデックス」と呼んだ）入力記号の位置を表すタイプ TKey
    // および TPos を定義する。これらのタイプを公衆とする他のクラスは以下のこれらを
    // 参照すべきである
    ////////////////////////////////////////////////////////////////////////

    // typedef 無符号 int    TKey;
    typedef unsigned _int32 TKey;
    typedef int    TPos;

CDFSolitonDistribution(
                int nSymbols,
                int nArity            = 1,
                int nRippleTargetSize = 0,
                int nStartRippleSize  = 0);

    virtual ~CDFSolitonDistribution();

    inline double dAverageWeight()  { return m_dAverageWeight;    };
    inline int    nMaxWeight()      { return m_nMaxWeight;       };
    inline int    nParameterR()     { return m_nRippleTargetSize; };
    inline int    nParameterS()     { return m_nStartRippleSize;  };

    //確率分布は種類の配列を含み、
    //ここで各種類は、出力記号の重み（前に
    //「度」と呼んだ）およびその重みに関連付け
    //られる密度（確率）に対応する

    inline int    nKinds()          { return m_nKinds;      };

private:

    double m_dAverageWeight;
    void   ComputeAverageWeight();
    int    m_nRippleTargetSize;
    int    m_nStartRippleSize;

    // m_nKinds は確率分布を保持する
    // 配列のサイズである。可能なのは出力記号
    // の異なる重みの数である
    int    m_nKinds;

    // 以下は不完全なロバストソリトン分布
    // からの種類数および生起りの度の
    // 分布による種類数である
    int    m_nRobustKinds;
    int    m_nTailKinds;

    int    m_nModulus; // 精度に対して 2

    int*   m_anKindWeight; // その重みは種類に対応する
    int*   m_anKindCumulativeDensity; // 種類の確率密度

    int    m_nSymbols;
    int    m_nArity;
    int    m_nMaxWeight;
};
```

フロントページの続き

(74)代理人 100075672
弁理士 峰 隆司
(74)代理人 100095441
弁理士 白根 俊郎
(74)代理人 100084618
弁理士 村松 貞男
(74)代理人 100103034
弁理士 野河 信久
(74)代理人 100119976
弁理士 幸長 保次郎
(74)代理人 100153051
弁理士 河野 直樹
(74)代理人 100140176
弁理士 砂川 克
(74)代理人 100100952
弁理士 風間 鉄也
(74)代理人 100101812
弁理士 勝村 紘
(74)代理人 100070437
弁理士 河井 将次
(74)代理人 100124394
弁理士 佐藤 立志
(74)代理人 100112807
弁理士 岡田 貴志
(74)代理人 100111073
弁理士 堀内 美保子
(74)代理人 100134290
弁理士 竹内 将訓
(74)代理人 100127144
弁理士 市原 卓三
(74)代理人 100141933
弁理士 山下 元
(74)代理人 100078282
弁理士 山本 秀策
(72)発明者 マイケル ジー. ルビー
アメリカ合衆国 カリフォルニア ９４７０８, バークレー, ミラー アベニュー １１３３

審査官 渡辺 未央子

(56)参考文献 特開２００１－０３６４１７（ＪＰ，Ａ）
特表２００３－５０７９８５（ＪＰ，Ａ）
特許第３８０９９５７（ＪＰ，Ｂ２）
特許第３９７６１６３（ＪＰ，Ｂ２）

(58)調査した分野(Int.Cl., ＤＢ名)
H03M 13/15
H03M 13/09
H04L 1/00