(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4703114号
(P4703114)
(45)発行日　平成23年6月15日(2011.6.15)　(24)登録日　平成23年3月18日(2011.3.18)

(51)Int.Cl.　F I
H04N 7/26 (2006.01)　H04N 7/13 A

請求項の数 10 (全 18 頁)

(21)出願番号	特願2003-563224 (P2003-563224)
(86)(22)出願日	平成15年1月22日 (2003.1.22)
(65)公表番号	特表2006-502605 (P2006-502605A)
(43)公表日	平成18年1月19日 (2006.1.19)
(86)国際出願番号	PCT/US2003/002137
(87)国際公開番号	WO2003/063499
(87)国際公開日	平成15年7月31日 (2003.7.31)
審査請求日	平成18年1月23日 (2006.1.23)
(31)優先権主張番号	60/351,143
(32)優先日	平成14年1月22日 (2002.1.22)
(33)優先権主張国	米国(US)

(73)特許権者　500046438
マイクロソフト　コーポレーション
アメリカ合衆国　ワシントン州　９８０５２－６３９９　レッドモンド　ワン　マイクロソフト　ウェイ
(74)代理人　100077481
弁理士　谷　義一
(74)代理人　100088915
弁理士　阿部　和夫
(72)発明者　ゲアリー　ジェイ．サリバン
アメリカ合衆国　９８０５３　ワシントン州　レドモンド　ノースイースト　２２７　アベニュー　６２３９

最終頁に続く

(54)【発明の名称】開始符号エミュレーションの防止およびデータ充填のための方法およびシステム

(57)【特許請求の範囲】
【請求項１】
データストリームとして符号化されたデータを受信し、前記データストリームにおいて、第１の開始符号は前記受信した符号化されたデータの前に位置し、前記データストリームにおいて、第２の開始符号又は前記データストリームの終端は前記受信した符号化されたデータの後に位置し、前記第１の開始符号を前記データストリーム内でロケーティングする方法であって、
前記第１の開始符号と前記第２の開始符号又は前記データストリームの終端との間の前記受信した符号化されたデータを復号する前に、前記第１の開始符号と前記第２の開始符号又は前記データストリームの終端との間の前記受信した符号化されたデータにおける開始符号エミュレーション防止バイトは除去され、前記受信した符号化されたデータ内の複数のバイトのパターンが探索され、前記パターンが、少なくとも１つの複数のバイトの開始符号プレフィックスと、開始符号エミュレーションが発生するのを防止する開始符号エミュレーション防止バイトとを含み、前記受信した符号化されたデータ内に前記パターンが発見されことに応じて、前記受信した符号化されたデータから前記パターンの前記開始符号エミュレーション防止バイトが除去されることを特徴とする方法。
【請求項２】
前記探索は、前記受信した符号化されたデータのバイトと前記パターンの複数のバイトとをバイト比較することを含むことを特徴とする請求項１に記載の方法。
【請求項３】

前記第１の開始符号をロケーティングすることは、前記第１の開始符号の開始符号プレフィックスをロケーティングすることを含むことを特徴とする請求項１に記載の方法。
【請求項４】
前記第１の開始符号の前記開始符号プレフィックスをロケーティングすることは、複数の連続する０のストリングを探索することを含むことを特徴とする請求項３に記載の方法。
【請求項５】
前記第１の開始符号の前記開始符号プレフィックスをロケーティングすることは、直後に１が続く複数の連続する０のストリングを探索することを含むことを特徴とする請求項３に記載の方法。
【請求項６】
前記複数の連続する０のストリングは、３１バイトの０を含むことを特徴とする請求項５に記載の方法。
【請求項７】
前記データストリームは、音声データを含むことを特徴とする請求項１に記載の方法。
【請求項８】
前記少なくとも１つの前記複数のバイトの開始符号プレフィックスは、それぞれ０に等しい２バイトを含むことを特徴とする請求項１に記載の方法。
【請求項９】
前記データストリームは、ビデオデータを含むことを特徴とする請求項１に記載の方法。
【請求項１０】
前記開始符号エミュレーション防止バイトは０ｘ０３であることを特徴とする請求項１に記載の方法。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、開始符号エミュレーションを防止するため、ならびにデータ充填のための方法およびシステムに関する。
【背景技術】
【０００２】
本出願は、参照により開示が本明細書に組み込まれている２００２年１月２２日出願の米国特許仮出願第６０／３５１，１４２号明細書を端緒とし、該仮出願の優先権を主張する。
【０００３】
デジタルデータは、通常、何らかのタイプの送信機から何らかのタイプの受信機に伝送される。送信機は、通常、伝送のためにデータを符号化する符号器を含み、受信機は、通常、自らが受信するデータを復号する復号器を含む。ビデオデータ、オーディオデータ、オーディオ／ビデオデータ、テキストデータ、コンピュータ実行可能なプログラムデータ、アーカイブデータ、データベース情報など様々なタイプのデジタルデータが存在する。デジタルデータは、伝送される際、通常、何らかのタイプのチャネルで伝送される。同様な意味合いで、コンピュータメモリまたは任意の記憶装置または記憶媒体も、本明細書では、伝送チャネルと考えることができる。
【０００４】
デジタルデータが伝送される際、チャネルにおけるデータ内の特定のポイントを見つけることができることが重要である。これは、チャネルを介するデータの伝送の誤りまたは損失からの回復を可能にするポイントを探し出すため、ストリーム全体の先頭以外の場所で復号プロセスを開始することを可能にするポイントを探し出すため、または異なる目的で利用される、異なるタイプのデータを探索することを可能にするポイントを探し出すためなど様々な目的から行われる。したがって、例えば、復号器側で、復号器、ならびにデ

ジタルデータを処理するその他のコンポーネントは、しばしば、データを適切に処理することができるように、データのコンテキストを知る必要がある。これは、送信された最初のビットから開始することができ、復号器が全く誤りなしに動作することが可能であるとすれば、それほど重要ではないであろう。その状況では、理想的には、復号器は、データのフォーマットがどのようなものであるかを知っていることに応じて、送信されている情報を単に追跡することが可能である。残念ながら、この理想的な状況はあまり生じない。デジタルデータを送受信するシステムを設計する人々、および使用する人々に、困難な問題をもたらす誤り、およびその他の不測の事態が実際に生じる。進行中のブロードキャストデータストリームに同調する場合など一部のケースでは、復号器は、データ伝送の始まりで開始することができない。また、データフォーマット解析によってポイントを探し出すことも、復号器において相当な量の複雑な処理を要する可能性がある。

【０００５】

多くのタイプのチャネル環境では、そのような問題は、データの中で、いわゆる再同期マーカを提供することによって対処される。再同期マーカは、システムが復号プロセスを開始することができる、または誤りから回復することができる機構を提供する。例えば、デジタルデータが一続きのビットまたはバイトとしてストリーミングされる場合、そのストリームの中に再同期マーカを有することにより、伝送に誤りが生じた場合に回復を開始する基準点を復号器に提供することが可能である。

【０００６】

再同期マーカを使用することができる１つの形は、開始符号の文脈においてである。開始符号は、特定の値を有するビットまたはバイトのストリングである。一般に、多くのシステムは、バイトを伝送し（例えば、Ｈ．２２２．０／ＭＰＥＧ－２システム）、したがって、開始符号は、固有値を有するバイトストリングとして定義することができる。固有のバイトストリングは、あるパターンを提供し、そのパターンの存在により再同期ポイントが示される。再同期ポイントは、通常、いくらかの独立に復号可能な量のデータの先頭または境界を示す。例えば、Ｈ．２６２／ＭＰＥＧ－２ビデオデータでは、再同期ポイントは、スライス（すなわち、ピクチャの独立に復号可能な領域）の先頭、ピクチャの先頭、ＧＯＰ（すなわち、「ピクチャ群（Ｇｒｏｕｐ　ｏｆ　Ｐｅｃｔｕｒｅｓ）」）、つまり独立に復号可能なピクチャシーケンス）、または新たなビデオシーケンスの先頭を示すことが可能である。デジタルビデオストリームは、開始符号を前に置くことが可能な、いわゆる補助的なデータまたは補足的なデータも含むことができる。

【０００７】

時として、開始符号は、ビデオストリームなどのデータストリーム内だけでなく、システムの多重レベルによっても使用される。Ｈ．２２２．０／ＭＰＥＧ－２システム規格は、開始符号を使用し、システムレベル情報とオーディオ情報がインターリーブされたビデオデータのストリームを伝送するシステムの例である。

【０００８】

開始符号は、データストリーム内で再同期ポイントを提供する限りにおいて重要であり得るので、データストリームの中で、実際には、開始符号を表すことが意図されていない箇所で開始符号をエミュレートするのを回避することが賢明である。

【０００９】

例えば、次のことを考察してみる。開始符号が、新たなデータ単位の先頭を明らかにすることができる特定のビットパターンまたはバイトパターンを定義する。開始符号と開始符号の間で任意のデータを送信している場合、その任意のデータが、それ自体、開始符号として使用しているパターンと同じパターンを含む可能性がある。例えば、伝送されているデータが完全にランダムであるものと想定した場合には、開始符号がＫビット長である場合、何らかの特定のビット位置で開始するビット群において開始符号が偶然にエミュレートされる確率は、$1/2^k$である。

【００１０】

いくつかのケースでは、開始符号のビット数が多い場合、開始符号が偶然にエミュレー

トされる尤度がかなり低い可能性があると判断することができる。これは、一部のオーディオデータフォーマットに関して該当する。通常、それらのフォーマットは、ビット／秒で測定する非常に高いビットレートは利用せず、したがって、任意の特定の時間間隔中に開始符号が偶然にエミュレートされる尤度はあまり高くない。ビデオデータに関しては、これは、一般に、該当しない。というのは、ビデオは、しばしば、はるかに高いビットレートを要するからである。

【００１１】

過去の主要なビデオ符号化標準（おそらく、１つを例外として）では、データペイロード内のビデオ構文フォーマットは、開始符号エミュレーションを回避するように設計されていた。つまり、どのような種類のデータ要素がビデオ構文を構成するか分かっている場合、偶然の開始符号が全く生じる可能性がないように、その構文を注意深く設計することができる。例えば、従来のビデオ符号化標準における開始符号は、０のビットの長いストリングで開始して、その後に１のビットが続く。この長いストリングは、２３個の０のビットに続いて１つの１のビットを含むことが可能である。送信されるデータのほとんどが、可変長符号（しばしば、非公式にハフマン符号と呼ばれる）を使用してエントロピー符号化されているものと想定する。可変長符号（ＶＬＣ）は、本明細書では、１組の代表される記号のなかから選択を行うのに利用される、可変深度のツリー構造を有する符号として定義される。バイナリツリーＶＬＣを使用する１つの技術は、ルートから、有効な記号を表すすべてのリーフへのツリーにおけるパスが、そのどこかに「１」を常に有すること、およびツリー構造が過度に深くないことを保証することである。

【００１２】

したがって、例えば、すべての可変長符号ストリングが１０ビット長を超える長さではなく、すべてのそのようなストリングが、その中に少なくとも１つの１の値のビットを有することが分かっている場合、ＶＬＣからの符号化されたデータのシーケンスが、１８を超える連続するゼロの値を有するビットを含む可能性は全くないことが分かる。つまり、最悪のシナリオは、１０００００００００の後に０００００００００１が続くことである。したがって、構文を注意深く設計し、すべての０の値のビットおよびすべての１の値のビットの位置を点検して、いくつの０が連続して生じる可能性があるかを確かめた場合、その構文において生じる可能性があるものよりも長い０のストリングを含む開始符号を使用することができる。例えば、構文は、有効な構文が、開始符号ではない場所で決して２３個の０を含む可能性がないように設計することができる。したがって、２３個の０が出現する場合はすべて、開始符号であるはずであり、復号器は、開始符号を正確に検出することができるはずである。

【００１３】

【非特許文献１】ＩＴＵ－Ｔ勧告Ｈ．２６３の付録Ｅ

【発明の開示】

【発明が解決しようとする課題】

【００１４】

前述した操作は単純明快に見えるが、この操作は、開始符号パターンが偶然に送信される可能性がないことを保証するため、可能なデータのすべてを（ビットレベルで）、データが送信される可能なすべての順序で点検しなければならないため、かなり困難な目論見になる可能性がある。これは、誤りが生じやすい困難な構文設計方法である。

【００１５】

このビットレベル点検の設計プロセスは、一般に、多くのビデオ符号化規格が過去に設計されてきたやり方を述べている（すなわち、Ｈ．２６１、ＭＰＥＧ－１、Ｈ．２６２／ＭＰＥＧ－２、Ｈ．２６３のほとんど、およびＭＰＥＧ－４）。それの唯一の例外が、数学的な仕様からアルゴリズム的手法で圧縮ビットを生成する算術符号化と呼ばれる技術を使用する（非特許文献１）。この場合、生成されたビットを点検するエントロピー符号器の終りに追加のプロセスが存在し、符号器側において、過度に多くの連続する０が存在する場合、所定の数の０が見つかる前に「マーカ」ビット（１のビット）が挿入される。復

号器側で、復号器は、ゼロを数え上げ、臨界数のゼロを見つけた場合、本当の開始符号を見つけたことが分かる。復号器が、臨界数より少ないゼロを見つけた場合、復号器は、後続の１のビットが、開始符号エミュレーションを回避するために挿入されたマーカビットであることが分かり、そのビットを破棄し、後続のビットを実データの続きとして受け取る。

【００１６】

この解決策の問題は、この解決策が、符号器および復号器にビットレベルで着信データを点検させ、処理させることである。単一のビット位置で処理されているデータの場所を分析し、移動させることは、困難になり、復号器に望ましくない形で負担をかける可能性がある。ビットに関して移動させることも、プロセッサを集中的に使用する操作である。

【００１７】

したがって、本発明は、開始符号エミュレーションを防止するための改良された方法およびシステムを提供することに関連する関心から生じている。

【課題を解決するための手段】

【００１８】

ビットレベルより粗い細分性で実行される操作による、開始符号エミュレーション防止へのアプローチを提供する方法およびシステムを説明する。ビットレベル以外のレベルで操作を行うことにより、処理効率を高めることができる。１つまたは複数の実施形態によれば、開始符号エミュレーション防止方法は、単一のビットより大きい固定サイズのデータ部分に関するデータパターンを探す。特定のパターンが見つかった場合、開始符号エミュレーション防止データが、開始符号エミュレーションを防止するように挿入される。挿入されるデータは、単一のビットより大きく、一部の実施形態では、１バイトを含む。復号器は、開始符号エミュレーション防止データが挿入されているデータを復号した際、正規の開始符号を容易に特定することができ、次に、開始符号エミュレーション防止データを除去して、伝えられることが意図されていた元のデータを提供することができる。

【００１９】

さらに、ペイロードデータのサイズを切り上げて整数のバイトサイズにすることを可能にし、次に、復号器が容易に検出できる形で充填データを追加することを可能にするデータ充填方法を説明する。

【発明を実施するための最良の形態】

【００２０】

（概要）

ビットレベルよりも粗い細分性で開始符号エミュレーション防止へのアプローチを提供する方法およびシステムを以下に説明する。ビットレベルよりも高いレベルで操作を行うことにより、処理効率を高めることができる。本明細書の文脈では、ビットレベルより高いレベルで操作を行うとは、単一のビットより大きい固定サイズのデータ部分に関するデータパターンを探すプロセスを指すものとする。例えば、固定サイズのデータ部分には、バイト（すなわち、８ビット）、「ワード」（すなわち、１６ビット）、「ダブルワード」（３２ビット）などが含まれる可能性がある。したがって、本発明の技術は、バイト、ワードなどの範囲内で、またそれらの間でパターンを探すことができる。

【００２１】

さらに、ペイロードデータのサイズを切り上げてバイト量などの整数のデータ単位サイズにすることを可能にし、復号器が容易に検出できる形で充填データを追加することを可能にするデータ充填方法を説明する。

【００２２】

さらに、以下に提供する例は、ビデオデータの文脈において説明しているが、本発明の技術は、通常、符号化および復号が行われ、開始符号エミュレーション防止が望ましい、または必要であるあらゆるタイプのデータに関連して使用できることを認識して、理解されたい。そのようなデータの例には、オーディオデータ、オーディオ／ビデオデータなどが含まれる。

【００２３】
図１は、一実施形態による方法におけるステップを説明する流れ図である。方法は、任意の適切なハードウェア、ソフトウェア、ファームウェア、または以上の組み合わせで実装することができる。図示し、説明する実施形態では、方法は、少なくとも部分的にソフトウェアで実装される。さらに、方法は、２つの異なる分岐、すなわち、「符号器」として示された分岐と「復号器」として示された分岐を有するものとして図示されていることに、読者は留意されたい。「符号器」分岐は、符号器により、または符号器に関連して実行されるステップを示している。同様に、「復号器」分岐は、復号器により、または復号器に関連して実行されるステップを示している。

【００２４】
ステップ１００で、開始符号の後に伝送が予定される量のデータを取得するか、または生成する。データは、任意の適切なデータを含むことが可能である。データのタイプの例には、限定としてではなく、様々な量のビデオデータ、オーディオデータ、オーディオ／ビデオデータなどが含まれる。ステップ１０１で、データの前に開始符号を付ける。このステップは、実質的に、ステップ１００で取得された、または生成されたデータに開始符号を追加する。ステップ１０２で、固定サイズのデータ部分の１つまたは複数のパターンについて着信データを調べるか、または探索する。図示し、説明する実施形態では、探索されるパターンは、少なくとも２つの固定サイズのデータ部分を含み、それぞれの個別データ部分は、少なくとも２ビットを含む。ステップ１０４は、パターンが見つかったかどうかを判定する。パターンが見つからなかった場合、方法は、ステップ１０８に分岐し、データを伝送することが可能である。

【００２５】
反対に、パターンが見つかった場合、ステップ１０６で、パターンを含むデータに関して開始符号エミュレーション防止データを挿入する。図示し、説明する実施形態では、開始符号エミュレーション防止データの個々のインスタンスは、複数のビットを含む。好ましくは、開始符号エミュレーション防止データは、個別の固定サイズのデータ部分とビット数が等しいデータ量を含む。したがって、固定サイズのデータ部分が８ビット（バイトと呼ばれるデータ量）を含む場合、開始符号エミュレーション防止データは８ビットを含む。開始符号エミュレーション防止データが挿入された後、ステップ１０８で、データを伝送する。ステップ１１０で、次の開始符号より前に伝送されるべき追加のデータが存在するかどうかを判定する。存在する場合、方法は、ステップ１０２に戻り、前述したとおりに行われる。存在しない場合、方法は、ステップ１１１で、伝送されるべき追加のデータが存在するかどうかを判定することができる。存在する場合、方法は、分岐してステップ１００に戻る。存在しない場合、方法は、ステップ１１２で終了することが可能である。

【００２６】
話はそれるが、次のことを考察してみる。この特定の技術の用法の一例が、開始符号を「開始符号プレフィックス」と「開始符号タイプ」サフィックスに分離することであり、ただし、プレフィックスは、単一の固有なストリングの値であり、サフィックスは、開始符号の後に続くデータのタイプを示すことに留意されたい。詳細には、これは、ＭＰＥＧ２およびＨ．２６４／ＡＶＣの開始符号の構造であり、以下の「第１の例示的な方法」という題名のセクションで使用する構造である。プレフィックス／サフィックス構造を特例として包含するさらに一般的な形態の用法は、１つまたは複数の開始符号パターンを有することの一般的な概念である。その場合、１つまたは複数のエミュレーション防止パターンも有することが可能である。様々な開始符号パターンが様々なエミュレーション防止パターンとは全く異なっており、それらのパターンのすべてがペイロードデータの処理において回避される限り、スキームは、相応に機能する。さらに、開始符号パターンがすべて同じ長さであると必ずしも想定すべきではない。これは、特定の場合において重要であり得る。というのは、例えば、Ｈ．２６４／ＡＶＣの使用は、複数の長さの開始符号を使用しているものと解釈することができるからである。

【００２７】
復号器側で、次のことを考察してみる。開始符号エミュレーション防止データは、符号器によって挿入された後、その他のデータの適切な解釈のために何らかの時点で特定し、および除去し、または無視することが可能である。復号器は伝送されたデータを受け取ると、正規の開始符号を探すことができることも考察してみる。復号器は、正規の開始符号を見つけると、どこに開始符号が画定するデータ境界が位置しているかが分かる。次に、復号器は、開始符号エミュレーション防止データを探して、除去することに取りかかり、実データをさらに処理することができる。
【００２８】
具体的には、ステップ１１４で、開始符号のエミュレーションを防止するように符号器によって処理された伝送データを受け取る。ステップ１１８で、データを処理して開始符号を見つける。開始符号が見つかり、適切に処理される（例えば、読み取られ、破棄される）と、ステップ１２０で、データを探索して開始符号エミュレーション防止データを特定する。開始符号エミュレーション防止データが見つかると、ステップ１２２で、その開始符号エミュレーション防止データを除去する。開始符号エミュレーション防止データが除去されると、受け取られたデータのタイプにとって典型的な方法でデータを処理することができる。例えば、データは、ステップ１２４におけるように、コンシューマ装置によって消費される可能性がある。
【００２９】
（第１の例示的な方法）
以下に説明する方法は、図１に示し、および説明する方法の一具体例だけを示している。以下に説明する方法では、ペイロードデータのＮ＋１バイトのストリングが、開始符号プレフィックス全体に一致した場合、または開始符号プレフィックスの最初のＮ個のバイトにエミュレーション防止バイトの値を加えたものに一致した場合にはいつでも、エミュレーション防止データのバイトは挿入される。この方法は、「第２の例示的な方法」という題名のセクションで説明する方法より低い頻度でデータを追加し、したがって、ペイロードデータを送信する伝送能力要件を低くする。
【００３０】
ＭＰＥＧ－２開始符号プレフィックス構造は、バイト整合の位置で開始し、２３個の０の後に１個の１が続く。この開始符号プレフィックスは、以下のとおり示される。すなわち、
０００００００００　０００００００００　０００００００１
【００３１】
この構造は、同じ値を有する何らかの複数バイトＮに続いて、異なる値を有する他の何らかの単一バイトを含むパターンとして一般化することができる。ＭＰＥＧ－２において、Ｎ＝２であり、最初の２つのバイトは０であり（以下に「Ｗ」と呼ぶ）、最後のバイトは１である（以下に「Ｘ」と呼ぶ）と述べることができる。したがって、開始符号プレフィックスは、以下のパターンを有する。すなわち、
ＷＷＸ
【００３２】
以上３つのバイトの後、ＭＰＥＧ－２では、別のバイトが後に続き、それがどのような種類の開始符号であるかを明らかにする。この後続のバイトを「Ｙ」と呼ぶ。したがって、基本的に、開始符号は、開始符号プレフィックスＷＷＸと、開始符号のタイプを明らかにする後続する単一バイトＹから成る。ＭＰＥＧ－２開始符号全体は、以下のとおり表現することができる。すなわち、
ＷＷＸＹ
【００３３】
開始符号プレフィックス（ＷＷＸ）は固定値を有するが、Ｙは、開始符号のタイプ（例えば、スライス、ピクチャ、ＧＯＰ、シーケンス、システムなど）を示すいくつかの異なる値を有する。

【００３４】
一実施形態によれば、データは、パターンＷＷＸを探して処理される。ＷＷＸパターンが見つかった場合、開始符号エミュレーションを防止するように開始符号エミュレーション防止データが挿入される。この場合、開始符号エミュレーション防止データは、Ｗバイトおよびバイトの値とは異なる値を有するバイトＺを含む。したがって、符号器がデータバイトを点検しており、パターンＷＷＸに気付くものと想定されたい。データの中でこのパターンを見つけたことに応答して、符号器は、値Ｚを有するバイトを挿入して以下のパターンを提供する。すなわち、
ＷＷＺＸ
【００３５】
この時点で、符号器は、復号器によって伝送され、処理されるべきペイロードデータが、開始符号または開始符号プレフィックスを偶然にエミュレートしないことを保証している。ここで、次のことを考察してみる。ペイロードデータは、ＷＷＸパターンを任意に含むことによって開始符号プレフィックスをエミュレートする機会性を有するのと全く同様に、開始符号エミュレーション防止データを含むデータを任意にエミュレートする機会性も有する。つまり、ペイロードデータは、本質的にパターンＷＷＺＸを含む可能性がある。これが該当し、符号器が何もしないことになった場合、復号器が開始符号エミュレーション防止データを除去しようと試みる際、復号器は、このケースでは実データであるＺバイトを除去する。
【００３６】
したがって、説明する実施形態では、符号器は、ペイロードデータが開始符号または開始符号プレフィックスをエミュレートすることを防止するように構成されるだけでなく、データが開始符号エミュレーション防止データの使用からもたらされるデータパターンをエミュレートすることを防止するようにも構成される。具体的には、この例では、符号器は、パターンＷＷＺを特定した場合、値Ｚを有するバイトを第２のＷとＺの中間に挿入して以下のパターン（挿入されるバイトＺは、以下で表れる最初のＺである）を提供する。すなわち、
ＷＷＺＺ
【００３７】
次に、処理されるデータを復号器の観点から考察してみる。復号器は、ＷＷＺに続いてＺまたはＸを含む何らかのバイトパターンを見つけた場合、最初のＺが、符号器によって挿入されたエミュレーション防止バイトであることが分かる。したがって、復号器は、最初のＺを破棄することができる。したがって、この例では、エミュレーション防止バイトを挿入することが可能な２つの状況が存在する。第１の状況は、データが、開始符号または開始符号プレフィックスを偶然にエミュレートする場合である。第２の状況は、データが、エミュレーション防止バイトが挿入されているデータを偶然にエミュレートする場合である。
【００３８】
いずれの場合も、復号器は、単に適切なパターンを探し、エミュレーション防止バイトを破棄して、通常どおりデータを処理することができる。
【００３９】
よりプログラム的な形で以上の処理を説明するため、次のことを考察してみる。符号器側で、同じ値ＷのＮ個またはそれより多くのバイトと、異なる値Ｘの最後のバイトから成る開始符号プレフィックスで始まり、その後に値Ｙを有する１バイトの識別用の開始符号タイプサフィックスが続くＢバイトのパケットＰ［］を送信するため、値Ｚ（ただし、Ｗ、Ｘ、Ｙ、およびＺは互いに異なる値を有し、Ｐ［Ｂ－１］はＷに等しくない）を有するエミュレーション防止バイトを挿入する以下の擬似コードプロセスを実行し、ただし、チャネルを充填するために送信される追加データの量は、Ｅで規定される。すなわち、
【００４０】

【表１】

```
int  B, N, E, i, j;
byte *P, W, X, Y, Z;
for(j=0; j<N+E; j++) /* start code prefix (SCP) */
  send_byte( W );    /* jth byte of SCP  */
send_byte( X );      /* last byte of SCP */
send_byte( Y );      /* start code type suffix */
for(i=j=0; i<B; i++) {
  if(j >= N && (P[i] == X || P[i] == Z)) {
    send_byte( Z );
    j = 0;
  }
  send_byte( P[i] ); /* a byte of data payload */
  if (P[i] == W) j++;
  else           j = 0;
}
```

【００４１】

以上の擬似コードでは、関数「ｓｅｎｄ＿ｂｙｔｅ（）」が、データ単位の伝送（図１のプロセス１０８）を実行するものと想定される。

【００４２】

復号器側で、パケットを受信するため、復号器が、同じ値ＷのＮ個またはそれより多くのバイトと、異なる値Ｘの最後のバイトから成る既知の開始符号プレフィックスを既に見つけ、読み取り、破棄しているものと想定する。また、未知の単一バイトの開始符号タイプサフィックスを変数Ｙに読み込み、ペイロードデータのパケットを配列Ｐ［］に読み込み、ペイロードデータの量を特定し、その量の指示を変数Ｂに入れる一方で、値Ｚを有するエミュレーション防止バイトを除去することを所望しているものと想定する（ただし、Ｗ、Ｘ、Ｙ、およびＺは互いに異なる値を有し、Ｐ［Ｂ－１］はＷに等しくない）。すなわち、

【００４３】

【表２】

```
int  B, N, j, next;
byte *P, W, X, Y, Z;
/* assume start code prefix was already read */
Y = receive_byte( ); /* start code type suffix */
for(B=j=0, next=0; more_data() && !next; B++) {
  P[B] = receive_byte( );
  if(j >= N) {
    if(P[B] == W)
      j++;
    else{
      j = 0;
      next = (P[B] == X);
      if(P[B] == Z)
        B--;
    }
  }else
    if(P[B] == W) j++;
    else          j=0;
}
if(next) /* another start code found */
  B -= j+1;
```

【００４４】
以上の擬似コードでは、関数「ｒｅｃｅｉｖｅ＿ｂｙｔｅ（）」がデータ単位の受け取りを実行するものと想定し、関数「ｍｏｒｅ＿ｄａｔａ（）」が、受け取られるべきさらなるデータ単位が存在するかどうかを判定するものと想定する（以上２つの関数が図１のプロセス１１４を構成している）。
【００４５】
前述した方法により、開始符号に先行する任意の量のW値の充填が可能になる。W値プレフィックスの数をN個に固定する定式化も同じように可能である。
【００４６】
（第２の例示的な方法）
以下に説明する方法は、図１に示し、説明した方法のもう１つの具体例を示すに過ぎない。この場合、方法は、ペイロードの中のデータのNバイトストリングが開始符号プレフィックスの最初のNバイトに一致する場合には、いつでも、後続のペイロードデータの値に関わりなく、エミュレーション防止データのバイトを挿入する。上記の例の表記法を使用すると、データが、後続するのが何であれパターン「WW」を含む場合、この方法は、エミュレーション防止バイトを挿入する。したがって、符号器は、パターンWWを識別した場合、エミュレーション防止バイトを挿入して以下のパターンを提供する。すなわち、
ＷＷＺ
【００４７】
最初に説明した方法とここで上記に説明する方法との違いは、最初の方法が、最初のＮ＋１個のバイトを調べて、どこにエミュレーション防止バイトを挿入するかを確かめるのに対して、第２に説明する方法は、最初のＮ個のバイトを調べることである。
【００４８】
最初の方法は、伝送されるべき追加データの量を減らし、他方、第２に説明する方法は、より単純な規則を使用して動作する。したがって、これらの方法の両者が一括で、伝送されるデータ量を減らすことと、規則を簡略化することとの間の選択を提供する。第１に説明した方法を使用すると、第２に説明した方法と比べてデータ量が少なくなる。第２に説明した方法を使用すると、より単純な規則が利用される。
【００４９】
上記の処理をよりプログラム的な形で説明するため、以下のことを考察してみる。符号器側で、同じ値Wの正確にN個のバイトと、異なる値Xの最後のバイトから成る開始符号プレフィックスで始まり、その後に値Yを有する１バイトの識別用の開始符号タイプサフィックスが続くBバイトのパケットＰ［］を送信するため、値Ｚを有するエミュレーション防止バイトを挿入する以下の擬似コードプロセスを実行する（ただし、W、X、Y、およびＺは互いに異なる値を有し、Ｐ［Ｂ－１］はWに等しくない）。すなわち、
【００５０】

【表３】

```
int  B, N, i, j;
byte *P, W, X, Y, Z;
for(j=0; j<N; j++)    /* start code prefix (SCP) */
  send_byte( W );     /* jth byte of SCP */
send_byte( X );       /* last byte of SCP */
send_byte( Y );       /* start code type suffix */
for(i=j=0; i<B; i++) {
  send_byte( P[i] ); /* a byte of data payload */
  if( P[i] != W )
    j=0;
  else
    if(++j == N) {
      send_byte( Z );
      j = 0;
        }
}
```

【００５１】

上記の擬似コードでは、関数「ｓｅｎｄ＿ｂｙｔｅ（）」が、データ単位の伝送（図１のプロセス１０８）の動作と想定される。

【００５２】

復号器側で、パケットを受け取るため、復号器が、同じ値Ｗの正確にＮ個のバイトと、異なる値Ｘの最後のバイトから成る既知の開始符号プレフィックスを既に見つけ、読み取り、破棄しているものと想定し、また、未知の単一バイトの開始符号タイプサフィックスを変数Ｙに読み込み、ペイロードデータのパケットを配列Ｐ［］に読み込み、ペイロードデータの量を特定し、その量の指示を変数Ｂに入れる一方で、値Ｚを有するエミュレーション防止バイトを除去することを所望しているものと想定する（ただし、Ｗ、Ｘ、Ｙ、およびＺは互いに異なる値を有し、Ｐ［Ｂ－１］はＷに等しくない）。すなわち、

【００５３】

【表４】

```
int  B, N, j, k;
byte *P, W, X, Y, Z;
/* assume start code prefix was already read */
Y = receive_byte( ); /* start code type suffix */
for(B=j=0, k=Z; more_data() && k != X; B++) {
  P[B] = receive_byte( );
  if(P[B] == W) {
    if(++j == N) {
      k = receive_byte( ); /* more_data() always */
      if(k != Z && k != X)
        declare_error( );
      j = 0;
    }
  }else
    j = 0;
}
if(k == X) /* another start code found */
  B -= N;
```

【００５４】

上記の擬似コードでは、関数「ｒｅｃｅｉｖｅ＿ｂｙｔｅ（）」がデータ単位の受け取

りを実行するものと想定し、関数「ｍｏｒｅ＿ｄａｔａ（）」が、受け取られるべきさらなるデータ単位が存在するかどうかを判定するものと想定する（以上２つの関数が、図１のプロセス１１４を構成している）。

【００５５】

前述した方法は、大量の理想的なランダムな入力ペイロードデータの量を第２に説明した方法の場合、およそ１／２５６^{N}だけ、第１に説明した方法の場合、１／２５６$^{(N+1)}$だけ拡張すると考えられる。以上の量は、Ｎが大きい（例えば、ＭＰＥＧ－２開始符号の場合、Ｎ＝２であることに留意して、２以上）場合、小さい。ペイロードに関する最悪の場合の拡張要因は、第２に説明した方法の場合、１／Ｎであると考えられ、第１に説明した方法の場合、１／（Ｎ＋１）であると考えられる。Ｎが増大した場合、開始符号自体によって使用されるデータの量は増大するが、ペイロード拡張要因は、統計分析においても、最悪の場合の分析においても小さくなる。

【００５６】

前述したエミュレーション防止プロセスは、パケットを送信することを始める前にパケットの中にどれだけのデータがあるかを知っていることに依存しないことを認識されたい。したがって、有意な遅延は追加されない。

【００５７】

第２に説明した方法の定式化は、挿入されるエミュレーション防止バイトが、値Ｚを有する単一のバイトであるものと想定している。挿入されるデータの最初のバイトが、有効な開始符号をエミュレートすることになるか、またはプレフィックスの先頭の続きであるように見えることになるＷまたはＸに等しくない限り、エミュレーション防止データとして任意の値または複数の値、あるいは１つまたは複数の値のストリングを代わりに使用することも可能である。

【００５８】

これらのエミュレーション防止バイトの中で情報を伝送することさえできる（例えば、Ｈ．２６３式ＧＯＢフレームＩＤ／ピクチャシーケンス番号、または仮に、ＭＳＢだけを「１」に設定し、その他の７つのＡＳＣＩＩ文字を送るのに使用するためなど）。

【００５９】

復号器側でパケットの終端において生じることを考慮した場合、データパケットペイロードの最後のバイトがＷではない場合、動作を制御することがより容易であることに気付く。これは、開始符号の前に送信される最後のバイトが、決してエミュレーション防止バイトである必要はなく、検出可能な境界が、復号器によってペイロードデータの終端と次の開始符号のためのＷに等しいバイトシーケンスの先頭の間に見出される可能性があることを意味する。この状況を課すことにより、ペイロードの終端の後、次の開始符号の前に、ペイロードの終端がどこにあるかを見失うことなしに任意の量のＷバイト（例えば、ゼロのバイト）を詰め込むこともできるようになる。

【００６０】

（データ充填）

通常、ビデオデータでは、データペイロードとして送信されるデータは、整数のバイトではない可能性がある。例えば、２つの開始符号の間で送信されるべき６２７ビットを有することが可能である。しかし、システムの多重レベルは、バイトで動作する可能性がある。ＭＰＥＧ－２規格の場合、これが該当する。伝送の誤りによって生成された、いくつかの偽の開始符号パターンの検出を可能にすること、またはペイロードの始まりのデータ内容に関する単純な復号プロセスを可能にすることなどの他の理由によっても、パケットがバイトなどの整数のデータ単位を含むのを所望することを正当化することができる。したがって、６２７ビットのデータを伝送するためには、もう少し多くのデータを送信しなければならない可能性がある。すると、問題は、どのようにデータに埋め込みを行ってデータを整数のバイトにするかということになる。

【００６１】

単に追加の充填データを送信することが役立つ他の状況も存在する。例えば、チャネル

が、毎秒１メガビットの容量を有し、送信されるべきペイロードデータの量が９００キロビット／秒だけである場合、チャネルを充填データで充填することを必要とする、または所望する可能性がある。

【００６２】

一実施形態によれば、データ充填技術により、余分なデータをチャネルに追加して、基本的に、ペイロードデータをパッディングすることが可能になる。

【００６３】

図２は、開始符号が、ゼロに等しいビットストリングで始まるものと想定される一実施形態によるデータ充填方法におけるステップを説明する流れ図である。ステップ２００で、送信されることが予定されるデータの終端の位置を確立する。ステップ２０２で、ペイロードデータの最後のビットの後に「１」のビットを挿入する。ステップ２０４で、整数のバイトを送信するのに要する追加のビットの数を算出する。ステップ２０６で、要求される数の「０」ビットを挿入済みの「１」のビットの後に挿入する。ステップ２０８で、任意の所望のバイト数の充填データを追加する。充填データは、真のペイロードデータの場所、および意図的な開始符号の場所についての混乱を回避するように設計された任意のデータパターンから成ることが可能である。これは、通常、値「０」のバイトを挿入することによって実施される。

【００６４】

例として、以下を考察してみる。６２７ビットが送信されるものと想定する。この場合、ステップ２０２は、「１」のビットを第６２７番のビットの後に挿入する。次に、ステップ２０４で、整数のバイト、この場合は、７９バイトを提供するのにさらに４ビットを要すると算出する。これに応じて、ステップ２０６で、４つの「０」のビット、つまり００００を挿入済みの「１」のビットの後に挿入する。この時点で、整数のバイトが確立され、ステップ２０８で、所望される場合、任意の所望のバイト数の充填データを追加することができる。この特定の例では、０の値を有するバイトを挿入することができる。充填データは、単に充填データとして、あるいは復号器が何らかの目的で使用することができる情報を含めるなどの何らかの他の目的で使用することができる。

【００６５】

次に、復号器における状況を考察してみる。復号器は、充填されたデータを受信し、そのデータの終端から始めて、データを逆方向に調べることができる。復号器が最初に見るバイトのすべては、復号器が「１」のビットを有するバイトに達するまで、０のバイトである。この「１」のビットは、ペイロード、つまり実データの終端の位置を復号器に告げる。したがって、復号器は、挿入された「１」のビットを見つけると、実データが正確にどこで終わるかを特定することができる。

【００６６】

したがって、前述した技術を使用して送信されるビットの数を「切り上げて」、送信されるビット数が整数のデータ単位を含むようにすることができる。さらに、以上の技術を使用して、ペイロードデータの先頭を示す開始符号と開始符号の間に充填データを埋め込むことができる。

【００６７】

（例示的なコンピューティング環境）

図３は、以下に説明するシステムおよび関連する方法を実施することができる適切なコンピューティング環境３００の例を示している。

【００６８】

コンピューティング環境３００は、適切なコンピューティング環境の一例に過ぎず、前述した符号化／復号システムの用途または機能の範囲に関して何ら限定を示唆するものではないことを理解されたい。また、コンピューティング環境３００が、例示的なコンピューティング環境３００に示したコンポーネントのいずれか１つ、またはいずれかの組み合わせに関連する依存関係または要件を有するものと解釈すべきでない。

【００６９】

様々な説明する実施形態は、多数の他の汎用または専用のコンピュータシステム環境またはコンピュータシステム構成で動作することが可能である。メディア処理システムで使用するのに適している可能性がある周知のコンピューティングシステム、コンピューティング環境、および／またはコンピューティング構成の例には、限定ではないが、パーソナルコンピュータ、サーバコンピュータ、シン（ｔｈｉｎ）クライアント、シック（ｔｈｉｃｋ）クライアント、ハンドヘルド装置またはラップトップ装置、マルチプロセッサシステム、マイクロプロセッサベースのシステム、セットトップボックス、プログラマブル家庭用電化製品、ネットワークＰＣ、ミニコンピュータ、メインフレームコンピュータ、以上のシステムまたは装置のいずれかを含む分散コンピューティング環境などが含まれる。

【００７０】

一部の実装では、システムおよび関連する方法は、コンピュータによって実行されているプログラムモジュールなどのコンピュータ実行可能命令の一般的な文脈において説明することができる。一般に、プログラムモジュールには、特定のタスクを実行する、または特定の抽象データ型を実装するルーチン、プログラム、オブジェクト、コンポーネント、データ構造などが含まれる。実施形態は、通信ネットワークを介してリンクされた遠隔処理装置によってタスクが実行される分散コンピューティング環境において実施することもできる。分散コンピューティング環境では、プログラムモジュールは、メモリ記憶装置を含むローカルのコンピュータ記憶媒体と遠隔のコンピュータ記憶媒体の両方の中に配置することが可能である。説明するコンピューティングシステムのコンポーネントを使用して、前述したとおり機能する符号器および復号器を実装することができる。

【００７１】

図３の図示する例示的な実施形態によれば、コンピューティングシステム３００が、１つまたは複数のプロセッサまたは処理装置３０２、システムメモリ３０４、ならびににシステムメモリ３０４からプロセッサ３０２までを含む様々なシステムコンポーネントを結合するバス３０６を含んでいるのを示している。

【００７２】

バス３０６は、様々なバスアーキテクチャのいずれかを使用するメモリバスまたはメモリコントローラ、周辺バス、アクセラレーテッドグラフィックスポート（ａｃｃｅｌｅｒａｔｅｄ　ｇｒａｐｈｉｃｓ　ｐｏｒｔ）、ならびにプロセッサバスまたはローカルバスを含め、いくつかのタイプのバス構造のいずれかの１つまたは複数を表すものとする。例として、限定としてではなく、そのようなアーキテクチャには、ＩＳＡ（Ｉｎｄｕｓｔｒｙ　Ｓｔａｎｄａｒｄ　Ａｒｃｈｉｔｅｃｔｕｒｅ）バス、ＭＣＡ（Ｍｉｃｒｏ　Ｃｈａｎｎｅｌ　Ａｒｃｈｉｔｅｃｔｕｒｅ）バス、ＥＩＳＡ（Ｅｎｈａｎｃｅｄ　ＩＳＡ）バス、ＶＥＳＡ（Ｖｉｄｅｏ　Ｅｌｅｃｔｒｏｎｉｃｓ　Ｓｔａｎｄａｒｄｓ　Ａｓｓｏｃｉａｔｉｏｎ）ローカルバス、およびメザニン（Ｍｅｚｚａｎｉｎｅ）バスとしても知られるＰＣＩ（Ｐｅｒｉｐｈｅｒａｌ　Ｃｏｍｐｏｎｅｎｔ　Ｉｎｔｅｒｃｏｎｎｅｃｔｓ）バスが含まれる。

【００７３】

コンピュータ３００は、通常、様々なコンピュータ読取り可能な媒体を含む。そのような媒体は、コンピュータ３００がローカルおよび／または遠隔でアクセス可能な任意の利用可能な媒体とすることが可能であり、揮発性媒体および不揮発性媒体、ならびに取外し可能な媒体および取外し不可能な媒体がともに含まれる。

【００７４】

図３では、システムメモリ３０４が、ランダムアクセスメモリ（ＲＡＭ）３１０などの揮発性の形態、および／または読み取り専用メモリ（ＲＯＭ）３０８などの不揮発性メモリの形態でコンピュータ読取り可能な媒体を含んでいる。始動中などにコンピュータ３００内部の要素間で情報を転送するのを助ける基本ルーチンを含む基本入出力システム（ＢＩＯＳ）３１２が、ＲＯＭ３０８の中に格納されている。ＲＡＭ３１０は、通常、処理装置３０２が即時にアクセス可能であり、および／または現在、操作しているデータおよび／またはプログラムモジュールを含む。

【００７５】
コンピュータ３００は、その他の取外し可能／取外し不可能、揮発性／不揮発性のコンピュータ記憶媒体をさらに含むことが可能である。単に例として、図３は、取外し不可能な不揮発性の磁気媒体（図示せず、通常、「ハードドライブ」と呼ばれる）から読み取りおよび書き込みを行うためのハードディスクドライブ３２８、取外し可能な不揮発性の磁気ディスク３３２（例えば、「フロッピー（登録商標）ディスク」）から読み取りおよび書き込みを行うための磁気ディスクドライブ３３０、ならびにＣＤ－ＲＯＭ、ＤＶＤ－ＲＯＭ、または他の光媒体などの取外し可能な不揮発性の光ディスク３３６から読み取りまたは書き込みを行うための光ディスクドライブ３３４を示している。ハードディスクドライブ３２８、磁気ディスクドライブ３３０、および光ディスクドライブ３３４はそれぞれ、１つまたは複数のインターフェース３２６でバス３０６に接続される。

【００７６】
以上のドライブおよび関連するコンピュータ読取り可能な媒体により、コンピュータ読取り可能な命令、データ構造、プログラムモジュール、およびその他のデータの不揮発性ストレージがコンピュータ３００に提供される。本明細書で説明する例示的な環境は、ハードディスク３２８、取外し可能な磁気ディスク３３２、および取外し可能な光ディスク３３６を使用しているが、磁気カセット、フラッシュメモリカード、デジタルビデオディスク、ランダムアクセスメモリ（ＲＡＭ）、読み取り専用メモリ（ＲＯＭ）などの、コンピュータがアクセス可能なデータを格納することができる他のタイプのコンピュータ読取り可能な媒体も、例示的な動作環境において使用できることが、当分野の技術者には理解されよう。

【００７７】
例として、限定としてではなく、オペレーティングシステム３１４、１つまたは複数のアプリケーションプログラム３１６（例えば、マルチメディアアプリケーションプログラム３２４）、その他のプログラムモジュール３１８、およびプログラムデータ３２０を含むいくつかのプログラムモジュールをハードディスク３２８、磁気ディスク３３２、光ディスク３３６、ＲＯＭ３０８、またはＲＡＭ３１０に記憶することができる。ユーザは、キーボード３３８やポインティングデバイス３４０（「マウス」などの）などの入力装置を介して、コマンドおよび情報をコンピュータ３００に入力することができる。他の入力装置には、オーディオ／ビデオ入力装置３５３、マイク、ジョイスティック、ゲームパッド、衛星パラボラアンテナ、シリアルポート、スキャナなど（図示せず）が含まれる可能性がある。以上の入力装置、およびその他の入力装置は、バス３０６に結合された入力インターフェース３４２を介して処理装置３０２に接続されるが、パラレルポート、ゲームポート、またはユニバーサルシリアルバス（ＵＳＢ）などのその他のインターフェースおよびバス構造で接続してもよい。

【００７８】
モニタ３５６または他のタイプのディスプレイ装置も、ビデオアダプタまたはビデオ／グラフィックスカード３４４などのインターフェースを介してバス３０６に接続される。モニタに加えて、パーソナルコンピュータは、通常、出力周辺インターフェース３４６を介して接続することができるスピーカやプリンタなどのその他の周辺出力装置（図示せず）も含む。

【００７９】
コンピュータ３００は、遠隔コンピュータ３５０のような１つまたは複数の遠隔コンピュータに対する論理接続を使用してネットワーク化された環境において動作することができる。遠隔コンピュータ３５０は、コンピュータに関連して本明細書で説明した要素および特徴の多く、またはすべてを含むことが可能である。

【００８０】
図３に示すとおり、コンピューティングシステム３００は、ローカルエリアネットワーク（ＬＡＮ）３５１および汎用ワイドエリアネットワーク（ＷＡＮ）３５２を介して遠隔装置（例えば、遠隔コンピュータ３５０）と通信するように結合されている。そのような

ネットワーキング環境は、オフィス、企業全体のコンピュータネットワーク、イントラネット、およびインターネットで一般的である。

【００８１】

ＬＡＮネットワーキング環境で使用される場合、コンピュータ３００は、適切なネットワークインターフェースまたはネットワークアダプタ３４８を介してＬＡＮ３５１に接続される。ＷＡＮネットワーキング環境で使用される場合、コンピュータ３００は、通常、ＷＡＮ３５２を介して通信を確立するためのモデム３５４またはその他の手段を含む。内部にあることも、外部にあることも可能なモデム３５４は、ユーザ入力インターフェース３４２、またはその他の適切な機構を介してシステムバス３０６に接続することができる。

【００８２】

ネットワーク化された環境では、パーソナルコンピュータ３００に関連して示すプログラムモジュール、またはプログラムモジュールの諸部分は、遠隔メモリ記憶装置の中に格納することができる。例として、限定ではなく、図３は、遠隔アプリケーションプログラム３１６が、遠隔コンピュータ３５０のメモリ装置上に常駐しているのを示している。図示し、説明するネットワーク接続は、例示的であり、コンピュータ間で通信リンクを確立するその他の手段も使用できることが理解されよう。

【００８３】

（結論）

上述した方法およびシステムの一部は、ビットレベル以外の処理レベルにおいて開始符号エミュレーション防止を提供することができる。これは、処理の複雑さを軽減することができるため、有利である。本明細書で説明した技術は、任意の適切な文脈で、例えば、可変長符号、ハフマン符号、および算術符号化によって生成された内容を含むペイロードに関連して使用することができる。さらに、一部の実施形態は、所望される場合に整数のバイトが送信されることを保証することができ、開始符号、エミュレーション防止パターン、および基本的なペイロードデータのためのデータに加えて追加の充填データの送信を可能にすることができるデータ充填のための単純明快な方法を提供する。

【００８４】

本発明は、構造上の特徴および／または方法上のステップに特有の言葉で説明してきたが、特許請求の範囲で定義する本発明は、説明した特定の特徴またはステップに必ずしも限定されないことを理解されたい。むしろ、特定の特徴およびステップは、請求する本発明を実施する好ましい形態として開示している。

【図面の簡単な説明】

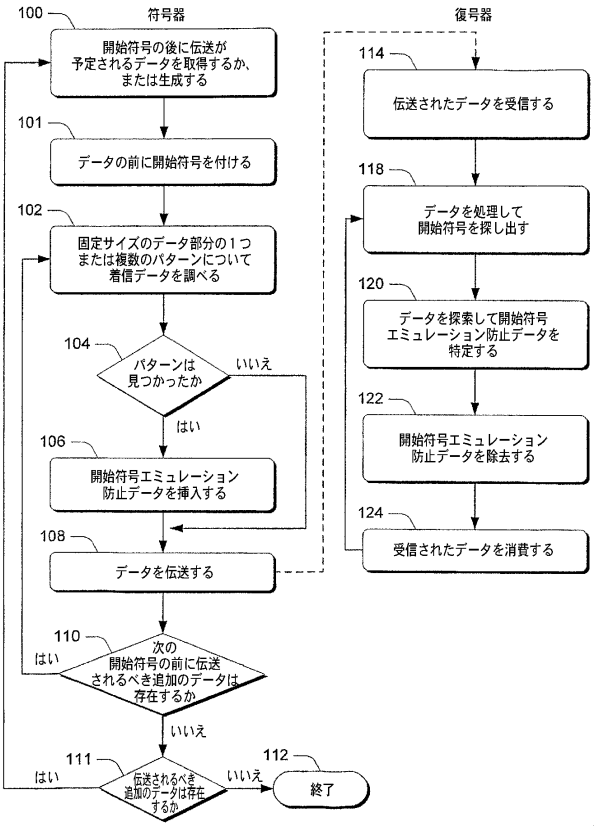
【００８５】

【図１】一実施形態による方法におけるステップを説明する流れ図である。

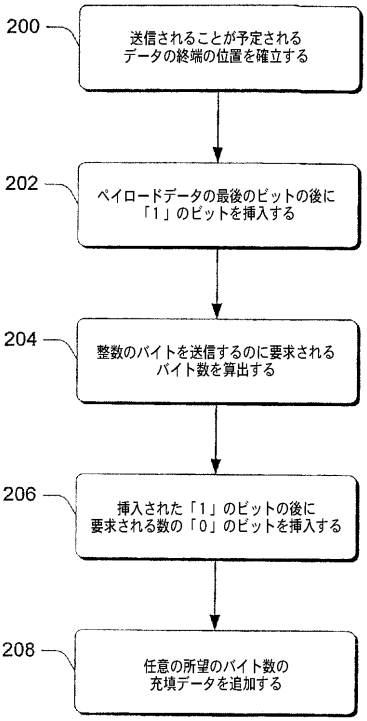
【図２】一実施形態による方法におけるステップを説明する流れ図である。

【図３】１つまたは複数の実施形態を関連して実施することができるコンピューティング環境を示す高レベルの図である。

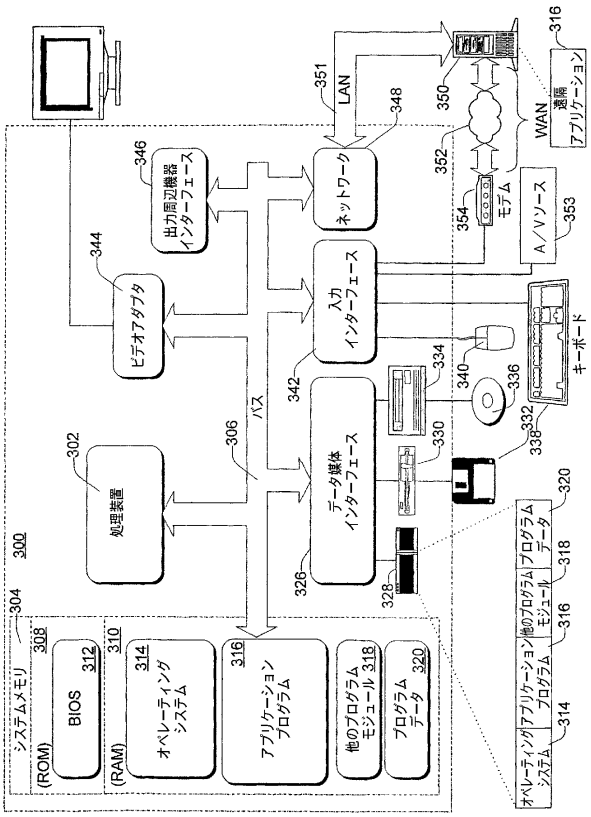
【図１】


符号器
復号器
100
開始符号の後に伝送が予定されるデータを取得するか、または生成する
101
データの前に開始符号を付ける
102
固定サイズのデータ部分の１つまたは複数のパターンについて着信データを調べる
104
パターンは見つかったか
いいえ
はい
106
開始符号エミュレーション防止データを挿入する
108
データを伝送する
110
次の開始符号の前に伝送されるべき追加のデータは存在するか
はい
いいえ
111
伝送されるべき追加のデータは存在するか
はい
いいえ
112
終了
114
伝送されたデータを受信する
118
データを処理して開始符号を探し出す
120
データを探索して開始符号エミュレーション防止データを特定する
122
開始符号エミュレーション防止データを除去する
124
受信されたデータを消費する

【図２】


200
送信されることが予定されるデータの終端の位置を確立する
202
ペイロードデータの最後のビットの後に「１」のビットを挿入する
204
整数のバイトを送信するのに要求されるバイト数を算出する
206
挿入された「１」のビットの後に要求される数の「０」のビットを挿入する
208
任意の所望のバイト数の充填データを追加する

【図３】


300
304
システムメモリ
(ROM)
308
BIOS
312
(RAM)
310
オペレーティングシステム
314
アプリケーションプログラム
316
他のプログラムモジュール
318
プログラムデータ
320
処理装置
302
バス
306
ビデオアダプタ
344
出力周辺機器インターフェース
346
ネットワーク
348
LAN
351
モデム
354
WAN
352
遠隔アプリケーション
350
316
A/Vソース
353
入力インターフェース
342
キーボード
334
340
336
332
338
データ媒体インターフェース
330
326
328
320
プログラムデータ
318
他のプログラムモジュール
316
アプリケーションプログラム
314
オペレーティングシステム

フロントページの続き

(72)発明者　ステファン　ジェイ. エストロップ
アメリカ合衆国　９８０１４　ワシントン州　カーネーション　ノースイースト　６３　ウェイ　２８２２３

審査官　長谷川　素直

(56)参考文献　特開平０８－０５６３５６（ＪＰ，Ａ）
特開平１１－１３６２２５（ＪＰ，Ａ）
特開２０００－０３２３９３（ＪＰ，Ａ）
特開２００１－０７８１４６（ＪＰ，Ａ）
特開２００１－１５５４３７（ＪＰ，Ａ）
特開平０６－００６３３５（ＪＰ，Ａ）
特開２００１－１６９２９２（ＪＰ，Ａ）
特開２０００－２３６５２２（ＪＰ，Ａ）
特開２０００－２９９８５６（ＪＰ，Ａ）
特開２００１－３５９１０７（ＪＰ，Ａ）
特開２０００－９２０３６（ＪＰ，Ａ）
特開２０００－１７５１１８（ＪＰ，Ａ）
特開２０００－１７５１５５（ＪＰ，Ａ）
特開２０００－２２４５８１（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
H04N　7/24-7/68