(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2006-246488
(P2006-246488A)
(43)公開日　平成18年9月14日(2006.9.14)

(51)Int.Cl.	F I		テーマコード（参考）
H04L 12/56 (2006.01)	H04L	12/56 100Z	5K030

審査請求 未請求　請求項の数 16　ＯＬ　（全 20 頁）

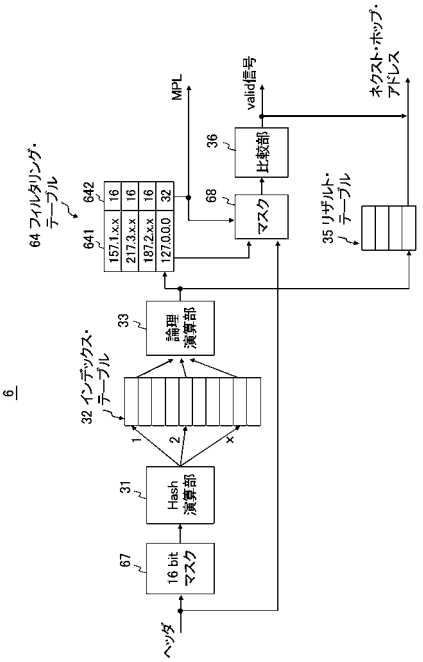
(21)出願番号　特願2006-58509(P2006-58509)
(22)出願日　平成18年3月3日(2006.3.3)
(31)優先権主張番号　60/658,168
(32)優先日　平成17年3月4日(2005.3.4)
(33)優先権主張国　米国(US)
(31)優先権主張番号　11/133,227
(32)優先日　平成17年5月20日(2005.5.20)
(33)優先権主張国　米国(US)

(71)出願人　302062931
ＮＥＣエレクトロニクス株式会社
神奈川県川崎市中原区下沼部１７５３番地
(71)出願人　306008377
エヌイーシーラボラトリーズアメリカインコーポレイテッド
アメリカ合衆国　ニュージャージー州　０８５４０　プリンストン　インディペンデンス・ウェイ　４
(74)代理人　100103894
弁理士　家入　健
(72)発明者　柴田　大彦
神奈川県川崎市中原区下沼部１７５３番地　ＮＥＣエレクトロニクス株式会社内

最終頁に続く

(54)【発明の名称】ネットワーク・ルータ、アドレス処理方法及びコンピュータ・プログラム

(57)【要約】
【課題】ネットワーク検索エンジンのメモリ使用効率を向上する。
【解決手段】インデックス・テーブル３２は、入力アドレスと関連する関数がエンコードされた値を格納するための２以上の格納場所を有する。エンコードされた値は、前記入力アドレスに対するハッシュ値を用いて、全てのエンコードされた値によって関数を復元することができるように生成されたものである。フィルタリング・テーブル６４は、少なくとも２つの異なる長さの複数のプリフィックスを格納し、かつインデックス・テーブル３２に格納されたエントリによって索引付けされる。リザルト・テーブル３５は、複数のネクスト・ホップ・アドレスを格納し、かつインデックス・テーブル３２に格納されたエントリによって索引付けされる。さらに、フィルタリング・テーブル６４は、格納されるプリフィックスのプリフィックス長を格納可能なフィールド６４２を有する。
【選択図】図６



【特許請求の範囲】
【請求項１】
少なくとも１つのインデックス・テーブル、少なくとも１つのフィルタリング・テーブル、及び少なくとも１つのリザルト・テーブルを有するネットワーク・ルータであって、
前記インデックス・テーブルは、入力アドレスに関連する関数がエンコードされた値を格納するための２以上の格納場所を有し、
前記エンコードされた値は、前記入力アドレスに対するハッシュ演算で得られるハッシュ値を用いて、全ての前記エンコードされた値によって前記関数を復元することができるように生成されたものであり、
前記フィルタリング・テーブルは、少なくとも２つの異なる長さの複数のプリフィックスを格納し、かつ前記インデックス・テーブルに格納されたエントリによって索引付けされ、
前記リザルト・テーブルは、複数のネクスト・ホップ・アドレスを格納し、かつ前記インデックス・テーブルに格納されたエントリによって索引付けされ、
前記フィルタリング・テーブルの少なくとも１つのレコードは、当該少なくとも１つのレコードに格納されるプリフィックスのプリフィックス長を格納可能なプリフィックス長フィールドを有する、ネットワーク・ルータ。
【請求項２】
前記プリフィックス長は、前記プリフィックスと前記入力アドレスの一致判定を行う際に比較すべきビット数を示す、請求項１に記載のネットワーク・ルータ。
【請求項３】
前記複数のプリフィックスのうちのプリフィックス長の長いプリフィックスのサブ・プリフィックスは、前記複数のプリフィックスのうちのプリフィックス長の短いプリフィックスと一致せず、
前記プリフィックス長の長いプリフィックスを、前記フィルタリング・テーブルにおける前記短いプリフィック長のプリフィックスに対応する格納場所に格納することが可能である、請求項１に記載のネットワーク・ルータ。
【請求項４】
前記ハッシュ演算に用いられるハッシュ関数は、前記複数のプリフィックスのうちの長さの長いプリフィックスのプリフィックス長より短いビット数を用いるものである、請求項１に記載のネットワーク・ルータ。
【請求項５】
前記フィルタリング・テーブルに格納されるプリフィックスは、プリフィックスの集合に含まれるプリフィックスのうち、同じネクスト・ホップ・アドレスを有するプリフィックスの最大部分集合をまとめたものであり、前記最大部分集合は共通のサブ・プリフィックスを有する、請求項１に記載のネットワーク・ルータ。
【請求項６】
前記最大部分集合に含まれるプリフィックスは、前記フィルタリング・テーブルの１つの格納場所に格納される、請求項５に記載のネットワーク・ルータ。
【請求項７】
ネットワーク検索エンジンにおけるアドレス処理方法であって、
入力アドレスを入力し、
前記入力アドレスに対するハッシュ演算を行ってハッシュ値を生成し、
前記ハッシュ値に基づいて、インデックス・テーブルに格納されたエントリを選択し、
選択された前記エントリに基づいて、少なくとも２つの異なる長さの複数のプリフィックスを格納するフィールドと当該複数のプリフィックスの長さを格納するフィールドとを有するフィルタリング・テーブルから、１のプリフィックス及び当該１のプリフィックスのプリフィックス長を索引付けし、
選択された前記エントリに基づいて、複数のネクスト・ホップ・アドレスが格納されたリザルト・テーブルから、１のネクスト・ホップ・アドレスを索引付けし、

前記索引付けされたプリフィックスと前記入力アドレスの間で、前記索引付けされたプリフィックス長に相当する部分の一致判定を行う、アドレス処理方法。

【請求項８】

前記複数のプリフィックスのうちのプリフィックス長の長いプリフィックスのサブ・プリフィックスは、前記複数のプリフィックスのうちのプリフィックス長の短いプリフィックスと一致せず、

前記プリフィックス長の長いプリフィックスが、前記フィルタリング・テーブルにおける前記短いプリフィック長のプリフィックスに対応する格納場所に格納されている、請求項７に記載の方法。

【請求項９】

前記ハッシュ演算に用いられるハッシュ関数は、前記複数のプリフィックスのうちの長さの長いプリフィックスのプリフィックス長より短いビット数を用いるものである、請求項７に記載の方法。

【請求項１０】

前記フィルタリング・テーブルに格納されるプリフィックスは、同じネクスト・ホップ・アドレスを有するプリフィックスの最大部分集合をまとめたものであり、前記最大部分集合は共通のサブ・プリフィックスを有する、請求項７に記載の方法。

【請求項１１】

前記最大部分集合に含まれるプリフィックスは、前記フィルタリング・テーブルの１つの格納場所に格納されている、請求項１０に記載の方法。

【請求項１２】

コンピュータを、

入力アドレスを受信する手段、

前記入力アドレスに対するハッシュ演算を行ってハッシュ値を生成する手段、

前記ハッシュ値に基づいて、インデックス・テーブルに格納されたエントリを選択する手段、

選択された前記エントリに基づいて、少なくとも２つの異なる長さの複数のプリフィックスを格納するフィールドと当該複数のプリフィックスの長さを格納するフィールドとを有するフィルタリング・テーブルから、１のプリフィックス及び当該１のプリフィックスのプリフィックス長を索引付けする手段、

選択された前記エントリに基づいて、複数のネクスト・ホップ・アドレスが格納されたリザルト・テーブルから、１のネクスト・ホップ・アドレスを索引付けする手段、

及び、

前記索引付けされたプリフィックスと前記入力アドレスの間で、前記索引付けされたプリフィックス長に相当する部分の一致判定を行う手段、

として機能させるためのコンピュータ・プログラム。

【請求項１３】

前記複数のプリフィックスのうちのプリフィックス長の長いプリフィックスのサブ・プリフィックスは、前記複数のプリフィックスのうちのプリフィックス長の短いプリフィックスと一致せず、

前記プリフィックス長の長いプリフィックスが、前記フィルタリング・テーブルにおける前記短いプリフィック長のプリフィックスに対応する格納場所に格納されている、請求項１２に記載のコンピュータ・プログラム。

【請求項１４】

前記ハッシュ演算に用いられるハッシュ関数は、前記複数のプリフィックスのうちの長さの長いプリフィックスのプリフィックス長より短いビット数を用いるものである、請求項１２に記載のコンピュータ・プログラム。

【請求項１５】

前記フィルタリング・テーブルに格納されるプリフィックスは、同じネクスト・ホップ・アドレスを有するプリフィックスの最大部分集合をまとめたものであり、前記最大部分

集合は共通のサブ・プリフィックスを有する、請求項１２に記載のコンピュータ・プログラム。
【請求項１６】
前記最大部分集合に含まれるプリフィックスは、前記フィルタリング・テーブルの１つの格納場所に格納されている、請求項１５に記載のコンピュータ・プログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、ネットワーク検索エンジンにおけるプリフィックスの最適化方法に関する。
【背景技術】
【０００２】
以下に示す文献には、有益な背景技術情報が記載されている。これらの文献は、本明細書内で参照されることによって、本開示に組み込まれる。以下では、これらの文献を角括弧つきの参照符号によって引用することとする。例えば［３］は、Dharmapurikar等の文献を示す。
１．P Gupta, B Prabhakar, S Boyd、「Near-Optimal Routing Lookups with Bounded Worst Case Performance」、in Infocomm. 2000. Tel Aviv、イスラエル
２．P Gupta, N McKeown、「Algorithms for Packet Classification」、 IEEE Network、2001年、第15巻、第2号、p.24-32
３．S. Dharmapurikar, P. Krishnamurthy, D E Taylor、「Longest prefix matching using bloom filters」、in Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications、2003年8月
４．B. Chazelle , J.Kilian, R.Rubinfeld, A. Tal、「The Bloomier Filter: An Efficient Data Structure for Static Support Lookup Tables」、Proceedings, Symposium on Discrete Algorithms (SODA)、2004年
５．V.Srinivasan, G.Varghese、「Fast Address Lookups Using Controlled Prefix Expansion」、ACM Transactions on Computer Systems (TOCS)、1999年、第17巻、第1号、p.1-40
６．V. Srinivasan, G.Varghese、「Method and Apparatus for Fast Hierarchical Address Lookup using Controlled Expansion of Prefixes」、米国特許第6,011,795号
【０００３】
ビデオ会議やリアルタイム映画配信等の広帯域を要求するアプリケーションの普及によって、インターネットに接続されるホスト（端末）数の増加とともに、インターネットを流れるトラフィックが急速に増加している。これらの広帯域アプリケーションの十分な品質を確保するために、通信回線の広帯域化が進められている。しかしながら、端末相互間におけるエンド・ツー・エンドでのパフォーマンスを向上するためには、通信回線速度の向上だけでなく、ＩＰ（Internet Protocol）パケットのフィルタリング及びルーティングを行うネットワーク・ルータの性能向上が不可欠である。
【０００４】
ネットワーク・ルータにおいて重大なボトルネックとなる処理は、ＩＰアドレス・ルックアップである。ここで、ＩＰアドレス・ルックアップとは、ＩＰパケットを宛先に向けて転送する処理を意味しており、一般にＩＰフォワーディングと呼ばれている。アドレス・ルックアップが高速化されれば、パケット・クラシフィケーション等のネットワーク・ルータで実行される他の処理の高速化も可能となる。
【０００５】
ネットワーク・ルータが行うアドレス・ルクアップとは、アドレス・プリフィックスの集合の中から、入力されたＩＰパケットのヘッダに含まれる宛先アドレスに最長一致するアドレス・プリフィックス（以下、最長プリフィックスと呼ぶ）を決定する処理である。なお、１つのアドレス・プリフィックスは、１つのインターネット・アドレス又はその前半部分であるネットワーク・アドレスに対応する。以下では、アドレス・プリフィックス

を単にプリフィックスと呼ぶ。

【０００６】

通常、プリフィックスは長さによって表現される。プリフィックスの長さがＬであることは、アドレス全体のうちの上位のＬビットがプリフィックスとして有効であることを意味する。有効でない他のビットは無視される、言い換えると、"don't-care"とみなされる。本明細書では、プリフィックスを整数（例えば、ネットワーク・アドレス）で表現する。また、プリフィックスに含まれる有効なビットの数をプリフィックス長と呼ぶこととする。図１は、８ビット、１６ビット、２４ビット及び３２ビットの４つのプリフィックス長に分類された３２ビットのＩＰアドレスに対するプリフィックス集合の例を示したものである。なお、図１では、ＩＰアドレスの標準的な表記方法に則って、８ビット毎にピリオドで区切り、ピリオドで区切った８ビットを１０進数の整数値で表記している。例えば、１２７．ｘ．ｘ．ｘ→Ａという表記において、"ｘ"はＩＰアドレスのうち、８ビットのプリフィックス（１０進数表記で１２７、ビット表記で０１１１１１１１）を除く任意部分であることを示している。また、矢印の後の"Ａ"は、プリフィックス"１２７．ｘ．ｘ．ｘ"に含まれる宛先アドレスを有するパケットの転送先を示しており、具体的にはネクスト・ホップ・アドレス又はネットワーク・ルータの出力ポートに相当する。本明細書においてＩＰアドレス及びプリフィック集合を示す際は、図１と同様の表記方法をとるものとする。

【０００７】

図１のプリフィックス集合１は、４つのサブセット（部分集合）１１乃至１４を有する。部分集合１１は、プリフィックス長が８ビットのプリフィックスの集合である。同様に、部分集合１２は１６ビット、部分集合１３は２４ビット、部分集合１４は３２ビットのプリフィックス長のプリフィックスの集合である。つまり、プリフィックス長は、１又は複数のプリフィックスを含む部分集合のコンテナに相当するものである。このため、以下では、プリフィックス長で分類した部分集合をプリフィックスの容器（bin）を示すものと考える。

【０００８】

上述した最長プリフィックスの決定処理は、最長プリフィックス一致判定（ＬＰＭ：Longest Prefix Matching）と呼ばれている。ＬＰＭを行うための主要な解決策は、コンテント・アドレサブル・メモリ（ＣＡＭ：Content-Addressable Memories）、ツリーベースのアルゴリズム［１，２］、及びハッシュベースのアルゴリズム［３］の３つである。

【０００９】

文献［４］では、Ｂｌｏｏｍｉｅｒフィルタのコンセプトに基づいたＣＡＭが、ネットワーク検索エンジン及びネットワーク・ルータを構成するアーキテクチャとして論じられている。このＣＡＭのアーキテクチャはＬＰＭに適用可能なものである。さらに、より複雑なパケット・クラシフィケーションの問題を解決するために、文献［４］に開示されたアーキテクチャを並列化して使用することも考えられている。このため、ＬＰＭを実行する基本的なアーキテクチャにおいて、ＬＰＭ実行時のメモリ使用量を低減することができれば、上記の文献［４］で議論されているようなアーキテクチャにおけるＬＰＭだけでなく、パケット・クラシフィケーションにも有効である。

【００１０】

また、本出願の発明者らによる米国特許出願１１／１３３，２２６には、特定のアーキテクチャに依存せず、ＬＰＭに対して汎用的に使用可能なプリフィックス処理技術が記載されている。この技術は、ツリーベースのアプローチ及びハッシュベースのアプローチの双方に有効である。また、文献［５］には、特定のネットワーク検索エンジンのアーキテクチャに依存のものであるが、２つのＬＰＭにおけるプリフィックス最適化手法が記載されている。

【００１１】

また、本出願の発明者らは、米国特許出願１０／９０９，９０７において、エンベデッドＤＲＡＭ技術に基づいたハッシュベースの検索アーキテクチャを開示している。当該ア

ーキテクチャは、低遅延、低消費電力、低コスト、高パフォーマンスという特徴がある。また、当該アーキテクチャは、汎用の検索エンジンに適用可能であり、ＬＰＭ及びパケット・クラシフィケーションに適用することも容易である。以下では、本発明の背景として、米国特許出願１０／９０９，９０７に開示された検索アーキテクチャを適用したネットワーク検索エンジンを、従来のネットワーク検索エンジンとして説明する。

【００１２】

従来のネットワーク検索エンジンは、Ｂｌｏｏｍｉｅｒフィルタと呼ばれるコンテントの読み出しが可能なデータ構造に基づいており、情報の格納及び検索を迅速に行うことができる。例えば、エレメントｔの関数ｆ（ｔ）を従来のネットワーク検索エンジンに格納する場合は、様々なｔの値とこれに対応するｆ（ｔ）の値を格納することによって行われる。このアイデアによって、与えられたｔに対応するｆ（ｔ）を迅速に読み出すことが可能である。従来のネットワーク検索エンジンでは、このｆ（ｔ）の読み出しを一定の時間で行うことができる。以下では、一定時間での読み出しを行う手法について、いくつかの定義とともに説明する。

【００１３】

関数ｆ（ｔ）の一定時間での読み出しが可能となるように従来のネットワーク検索エンジンのデータ構造中に関数ｆ（ｔ）を格納することを、"ファンクション・エンコーディング"と呼ぶ。また、与えられたエレメントｔに対応する関数ｆ（ｔ）を従来のネットワーク検索エンジンから読み出すことを、"ルックアップ"と呼ぶ。

【００１４】

ファンクション・エンコーディングは、複数のエレメントｔに対応する複数の関数ｆ（ｔ）の値を格納することによって行われる。従来のネットワーク検索エンジンのデータ構造中に格納される複数のエレメントをまとめて"エレメント集合"と呼ぶ。

【００１５】

従来のネットワーク検索エンジンの主要部分は、関数ｆ（ｔ）をエンコードして格納可能な、Ｂｌｏｏｍｉｅｒフィルタに基づくデータ構造によって構成されている。このデータ構造は、Ｋ個のハッシュ関数によって索引付けされるテーブルを有している。このテーブルを、"インデックス・テーブル"と定義する。ある１つのエレメントに対して演算されたＫ個のハッシュ値をまとめて、"ハッシュ・ネイバーフッド"と呼ぶ。また、あるエレメントｔのハッシュ・ネイバーフッドをＨＮ（ｔ）と表す。

【００１６】

あるエレメントに対して得られたハッシュ値が、エレメント集合に含まれるその他のエレメントのハッシュ・ネイバーフッド内に含まれていない場合、そのハッシュ値を"シングルトン"と呼ぶ。

【００１７】

インデックス・テーブルは、エレメント集合に含まれる全てのエレメントｔに対して、関数ｆ（ｔ）のようなエレメントｔに対応付けられた情報がハッシュ・ネイバーフッドに属する固有のアドレスに安全に格納されるように構築されている。この固有のアドレスをτ（ｔ）と表す。また、あるエレメントｔ１と異なるエレメントｔ２に対応付けられた情報は、τ（ｔ１）に格納されてはならない。むしろ、ｔ２はｔ２に対してユニークなアドレスτ（ｔ２）を有する必要がある。したがって、エレメント集合に含まれる個々のエレメントｔに対して固有のアドレスτ（ｔ）を探すことが望まれる。このための手続きの一例を以下に説明する。

【００１８】

あるエレメントｔが与えられた場合、上述のルックアップを行うことにより、アドレスτ（ｔ）に格納された情報が取り出される必要がある。ルックアップで取り出される情報には、例えば、関数ｆ（ｔ）がある。あるエレメントｔに対して、アドレスτ（ｔ）を生成するハッシュ関数をｈτ（ｔ）と記述する。アドレスτ（ｔ）は、エレメントｔのハッシュ・ネイバーフッドに含まれるものであるが、アドレスτ（ｔ）から情報を取り出す際には、ハッシュ関数ｈτ（ｔ）について知ることはできない。ハッシュ関数ｈτ（ｔ）は

、ファンクション・エンコーディングを行う場合にだけ知ることができる。

【００１９】

関数ｈτ（ｔ）を知ることなく確実に関数ｆ（ｔ）を取り出すため、ハッシュ演算によって得られるアドレスτ（ｔ）が示す全ての格納場所に格納される値に対して簡単な論理演算を行えるように、ある情報をエレメントｔのハッシュ・ネイバーフッドに含まれる全ての格納場所に格納する。具体的には、エレメント集合に含まれる、あるエレメントｔに対するハッシュ演算によってτ（ｔ）を得た時に、以下の（１）式に示す値Ｖ（ｔ）を演算してアドレスτ（ｔ）の格納場所に格納する。

【００２０】

【数１】

$$V(t)=\left(\bigwedge_{\substack{i=1\\ i\neq h\tau(t)}}^{i=k} D[H_i(t)]\right)\hat{}\,I(t) \qquad (1)$$

【００２１】

式（１）において、"＾"は、排他的論理和（ＸＯＲ）演算を表している。H_i（ｔ）は、エレメントｔのｉ番目のハッシュ値を表している。Ｄ［H_i（ｔ）］は、インデックス・テーブルのアドレスH_i（ｔ）に格納されたデータである。Ｋは、ハッシュ関数の数を表している。Ｉ（ｔ）は、エレメントｔに対応する情報を示している。つまり、Ｉ（ｔ）は、ネットワーク検索エンジンに格納し、かつ読み出しを行う対象となるものである。最後に、ｈτ（ｔ）は、上述したように、エレメントｔに対してアドレスτ（ｔ）を生成するハッシュ関数を表す。このように、Ｖ（ｔ）は、エレメントｔの関数Ｉ（ｔ）がエンコードされた値である。以下では、Ｖ（ｔ）をエンコード値と呼ぶ。

【００２２】

ルックアップの実行時は、エレメントｔが与えられると、エレメントｔに対して演算した全てのハッシュ値によって示される格納場所（以下、ハッシュ位置と呼ぶ）に格納された値に対してＸＯＲ演算を行うことにより、エレメントｔに対応する情報Ｉ（ｔ）を取り出すことができる。つまり、ルックアップ時には、以下の（２）式に示す演算を行う。

【００２３】

【数２】

$$\bigwedge_{i=1}^{i=k} D[H_i(t)] = I(t) \qquad (2)$$

【００２４】

続いて以下では、全てのエレメントｔに対してτ（ｔ）を見つける手順、及び上記のＥＯＲ演算によってファンクション・エンコーディングを行う手順について説明する。

【００２５】

Ｂｌｏｏｍｉｅｒフィルタリングでは、τ（ｔ）を見つけるために、ｇｒｅｅｄｙアルゴリズムとして知られているアルゴリズムを使用する。このアルゴリズムの詳細は文献［４］に詳しく記載されている。以下では、当該アルゴリズムの概要を説明する。

【００２６】

始めに、順序Γが、従来のネットワーク検索エンジンに格納されるエレメントに対して定義される。順序Γは、エレメントｔのハッシュ演算を行う順序を規定する。より詳しくは、順序Γ中のあるエレメントｔに対応するハッシュ値は、順序Γの中でエレメントｔより前に位置するエレメントに対するハッシュ演算によっては生成されない値となっている。順序Γが決定されると、エレメントｔとそのハッシュ位置に格納されるエンコード値Ｖ

（ｔ）が同じ順序で処理される。これは、得られたハッシュ位置をτ（ｔ）とする際の十分な条件である。その理由は以下の通りである。

【００２７】

まず順序Γで指定された最初のエレメントｔ１に対して、他のエレメントのハッシュ・ネイバーフッドに含まれていないハッシュ位置が決定される。つまり、他のエレメントに対するファンクション・エンコーディングは未だ行われていないため、ｔ１に対応する情報を決定されたハッシュ位置に確実に格納することができる。順序Γで指定される２番目のエレメントｔ２は、エレメントｔ１のハッシュ・ネイバーフッドに含まれないハッシュ位置を有する。エレメントｔ１に対するエンコードは既に行われているため、ｔ２に対応する情報を安全にハッシュ位置に格納することができる。つまり、関数のエンコードを実行している間は、エレメントに対応する１つのハッシュ位置のみがエンコード値の書き込みによって変更される。このため、ｔ２に対するファンクション・エンコーディングによって、エレメントｔ１に対応して書込み又は読み出しが行われる格納場所を上書きされることはない。この処理が、全てのエレメントに対して順序どおりに行われる。

【００２８】

このような順序Γは、以下に示すｇｒｅｅｄｙアルゴリズムによって決定することができる。当該アルゴリズムを図２に示す。エレメント集合２１の中からハッシュ値がシングルトンであるエレメントｔ１を判定し、スタック・メモリ２２の底にエレメントｔ１を格納して、ｔ１をエレメント集合２１から削除する（ステップＳ２０２及びＳ２０３）。後は、この処理を、エレメント集合２１が空になるまで、再帰的に繰り返せばよい（ステップＳ２０１）。最終的に得られるスタック・メモリは、エレメント集合に含まれるエレメントが順序Γで格納された状態となる。なお、図２のステップＳ２０４及びＳ２０５は、上述した順序Γ決定後の関数ｆ（ｔ）のエンコード手順を示している。

【００２９】

Ｂｌｏｏｍフィルタと同様に、基本的なＢｌｏｏｍｅｒフィルタも僅かな確率ではあるが誤検出（false positives）を生じる。このことは、インデックス・テーブルにエンコードされたエレメント集合に含まれていないエレメントｔ′がルックアップされた場合に、上記の式（２）による演算によって、一見したところでは妥当なＩ（ｔ）の値が得られる場合があることを意味する。このため、従来のネットワーク検索エンジンでは、フィルタリング・テーブルと呼ばれる第２のテーブルを用いることによって、このような誤検出（false positives）を防止している。

【００３０】

フィルタリング・テーブルは、インデックス・テーブルにエンコードされたエレメント数と同数のエントリを有している。フィルタリング・テーブルの格納場所ごとに、インデックス・テーブルにエンコードされている実在の１つのエレメントが格納される。ルックアップの際には、ルックアップによって得られたエレメントとフィルタリング・テーブルに格納された実在のエレメントとを比較することによって、誤検出（false positives）を防止することができる。

【００３１】

従来のネットワーク検出エンジンでは、フィルタリング・テーブルの生成は以下のように行われる。ファンクション・エンコーディングの際、エレメントｔに対応する情報Ｉ（ｔ）が、フィルタリング・テーブルのアドレスとして指定され、当該アドレス位置にエレメントｔが格納される。次に、誤検出（false positives）の判定手順を説明する。以下では、エレメント集合に実在しないエレメントｔ′に対するルックアップが行われた場合を考える。インデックス・テーブルからＩ（ｔ′）が読み出されたとき、フィルタリング・テーブルのアドレスＩ（ｔ′）に格納されたエレメントｔ′′が読み出される。そして、従来のネットワーク検索エンジンに入力されたｔ′と、読み出されたエレメントｔ′′が比較される。この比較の結果が不一致であれば、ルックアップが誤検出（false positives）であることが判定できる。このことを実行する有利な方法は、フィルタリング・テーブルのアドレスを順序Γのエレメント順序に従って配置することである。

【００３２】
続いて以下では当該ネットワーク検出エンジンの主要部であるサブ・セルの構成について説明する。従来のサブ・セル３の構成を図３に示す。サブ・セル３は、３つのテーブルを有している。具体的には、上述したインデックス・テーブル３２及びフィルタリング・テーブル３４、並びに第３のテーブルであるリザルト・テーブル３５を有する。上述したように、あるエレメントｔの関数ｆ（ｔ）がインデックス・テーブル３２内にエンコードされており、エレメントｔに対応する値がフィルタリング・テーブル３４に格納されている。
【００３３】
あるエレメントｔ′に対するルックアップを行う際は、入力されたｔ′と比較するために、インデックス・テーブル３２より取り出されたＩ（ｔ′）をアドレスに指定して、フィルタリング・テーブル３４からエレメントｔ′′が読み出される。なお、エレメントｔ′′に加えて、関数ｆ（ｔ′）を、アドレスＩ（ｔ′）で指定されるフィルタリング・テーブル３４の格納場所に保存してもよい。関数ｆ（ｔ）は、ネットワーク検索の結果として出力されるものであり、リザルト・テーブル３５は、関数ｆ（ｔ）を保存するものである。このため、フィルタリング・テーブル３４のうちの関数ｆ（ｔ）を保存した部分をリザルト・テーブル３５と呼ぶ。
【００３４】
なお、リザルト・テーブル３５を実装する際には、フィルタリング・テーブル３４と独立した別個のメモリとして実現することもできる。しかしながら、フィルタリング・テーブル３４とリザルト・テーブル３５は時間的に同時並行的にアクセスされるものであり、同じ数のエントリを有している。
【００３５】
以下、図３のその他の構成要素を説明する。ハッシュ演算部３１は、入力されたヘッダに含まれており、エレメントｔに相当する宛先アドレスに対するハッシュ演算を行ってハッシュ値τ（ｔ）を出力する。論理演算部３３は、τ（ｔ）によってアドレスされたインデックス・テーブル３２の格納場所のデータを読み出し、上述した式（２）の演算を実行して関数Ｉ（ｔ）を算出する。比較部３６は、Ｉ（ｔ）によってアドレスされたフィルタリング・テーブル３４の格納場所に格納されたプリフィックスを読み出し、入力されたアドレス（宛先アドレス）と比較し、プリフィックスの一致の有無を示す信号（ｖａｌｉｄ信号）を出力する。当該比較によって、宛先アドレスとプリフィックスが一致した場合は、"有効"を示すｖａｌｉｄ信号を出力する。他方、宛先アドレスとプリフィックスが一致しない場合は、"無効"を示すｖａｌｉｄ信号を出力する。これと並行して、Ｉ（ｔ）によってアドレスされたリザルト・テーブル３５の格納場所に格納されたネクスト・ホップ・アドレスが出力される。
【００３６】
次に、一般的な最長プリフィックス一致判定（ＬＰＭ）の処理について説明する。上述したようにプリフィックスは、インターネット・アドレス（ＩＰアドレス）又はその前半部分であるネットワーク・アドレスに対応する。例えば、"１００．１０"は、ＩＰアドレス"１００．１０．１．２"のプリフィックスである。プリフィックスのデータベースには、プリフィックス"１００．１０"についてのフォワーディング情報を含まれている。しかしながら、当該データベースは、さらに長いプリフィックス"１００．１０．１．２"に対する詳細なフォワーディング情報を有している可能性がある。したがって、ネットワーク検索エンジンに入力されたＩＰアドレスに対するネクスト・ホップを決定するためには、データベース内の全てのプリフィックスとの比較を行い、入力されたＩＰアドレスと最長一致するプリフィックスに対応するフォワーディング情報を発見しなければならない。
【００３７】
一例として、". ｃｏｍ"を宛先に指定するパケットをポートＡに転送するルータを考える。このルータが、"ｎｅｃ．ｃｏｍ"ドメインにはポートＢを介してより容易にアクセスできるロケーションに配置されていると仮定すると、当該ルータは、"ｎｅｃ．ｃｏｍ"を

宛先に指定するパケットをポートＢにルーティングする必要がある。したがって、"ｎｅｃ．ｃｏｍ"を宛先アドレスとする入力パケットはポートＢに転送され、その他の".ｃｏｍ"ドメインを宛先アドレスとする入力パケットはポートＡに転送される。

【００３８】

図１に例示したプリフィックス集合では、ヘッダに指定された宛先アドレスが"１２７．０．０．０"のパケットは、プリフィックス長３２ビットまでの最長一致判定によって、ポートＭに転送されることを示している。一方、宛先アドレスが"１２７．０．０．１"のパケットは、プリフィックス長２４ビットまでの最長一致判定によって、ポートＨに転送される。

【００３９】

従来のネットワーク検索エンジンでＬＰＭを行うためには、プリフィックス長ごとに図３に示した３つのテーブルを有するサブ・セルを並列化した構成が採用される。このようなネットワーク検索エンジン４の構成を図４に示す。サブ・セル３Ａ乃至３Ｄはそれぞれ、インデックス・テーブル、フィルタリング・テーブル及びリザルト・テーブルの３つのテーブルを有するが、このうちリザルト・テーブルは当該サブ・セルのチップの外部に置くことも可能である。

【００４０】

プリフィックスは、その長さによって分類され、別個のサブ・セルに格納されている。図４では、サブ・セル３Ａにはプリフィックス長８ビット、サブ・セル３Ｂにはプリフィックス長１６ビット、サブ・セル３Ｃにはプリフィックス長２４ビット、サブ・セル３Ｄにはプリフィックス長３２ビットのプリフィックスが格納されるものとする。入力パケットの転送先を決定するネットワーク検索を行う際には、入力パケットのヘッダ情報が全てのサブ・セルに並行して送られる。そして、個々のサブ・セルによる判定結果が、優先度判定部４１に送られる。個々のサブ・セルが出力する判定結果には、プリフィックスの一致の有無を示す情報（ｖａｌｉｄ信号）と、ネクスト・ホップ・アドレスＮＨ１乃至ＮＨ４が含まれる。上述したように、サブ・セル３Ａ乃至３Ｄは、宛先アドレスとプリフィックスが一致した場合に"有効"を示すｖａｌｉｄ信号を出力する。他方、宛先アドレスとプリフィックスが一致しない場合は"無効"を示すｖａｌｉｄ信号を出力する。

【００４１】

優先度判定部４１は、サブ・セル３Ａ乃至３Ｄの出力のうち、宛先アドレスとプリフィックスとの一致を示した出力の中から最も長いプリッフィクスを判定し、最も長いプリッフィクス一致によって決定されたネクスト・ホップ・アドレスを、入力パケットに対するフォワーディング情報として出力する。

【特許文献１】米国特許第６０１１７９５号明細書

【発明の開示】

【発明が解決しようとする課題】

【００４２】

図４に示した従来のネットワーク検索エンジン４では、プリフィックスが格納されるサブ・セル３Ａ乃至３Ｄはプリフィックス長に応じて決定される。つまり、１つのサブ・セルに含まれるプリフィックス長は１通りである。このような構成は、特定のプリフィックス長を持つプリフィックスの数が比較的少ない場合に、無駄の多い構成といえる。つまり、各サブ・セルに含まれる３つのテーブルはメモリ・モジュールによって構成されるものであるが、プリフィックスの数が比較的少ないサブ・セルでは、メモリの使用効率が低下するという課題がある。

【課題を解決するための手段】

【００４３】

本発明にかかるネットワーク・ルータは、少なくとも１つのインデックス・テーブル、少なくとも１つのフィルタリング・テーブル、及び少なくとも１つのリザルト・テーブルを有する。前記インデックス・テーブルは、入力アドレスに関連する関数がエンコードされた値が格納された２以上の格納場所を有する。ここで、前記エンコードされた値は、前

記入力アドレスに対するハッシュ演算で得られるハッシュ値を用いて、全ての前記エンコードされた値によって前記関数を復元することができるように生成される。前記フィルタリング・テーブルは、少なくとも２つの異なる長さの複数のプリフィックスを格納することができる。前記複数のプリフィックスは、ネットワーク・アドレスに対応する。また、前記フィルタリング・テーブルは、前記インデックス・テーブルに格納されたエントリによって索引付けされる。前記リザルト・テーブルは、複数のネクスト・ホップ・アドレスを格納するものであり、前記インデックス・テーブルに格納されたエントリによって索引付けされる。さらに、前記フィルタリング・テーブルの少なくとも１つのレコードは、当該少なくとも１つのレコードに格納されるプリフィックスのプリフィックス長を格納可能なプリフィックス長フィールドを有する。

【００４４】

本発明の別態様である方法は、ネットワーク検索エンジンにおけるアドレス処理方法である。当該方法は以下の（１）～（６）の手順を含む。（１）入力アドレスを入力する。（２）前記入力アドレスに対するハッシュ演算を行ってハッシュ値を生成する。（３）前記ハッシュ値に基づいて、インデックス・テーブルに格納されたエントリを選択する。（４）選択された前記エントリに基づいて、少なくとも２つの異なる長さの複数のプリフィックスを格納するフィールドと当該複数のプリフィックスの長さを格納するフィールドとを有するフィルタリング・テーブルから、１のプリフィックス及び当該１のプリフィックスのプリフィックス長を索引付けする。（５）選択された前記エントリに基づいて、複数のネクスト・ホップ・アドレスが格納されたリザルト・テーブルから、１のネクスト・ホップ・アドレスを索引付けする。（６）前記索引付けされたプリフィックスと前記入力アドレスの間で、前記索引付けされたプリフィックス長に相当する部分の一致判定を行う。

【００４５】

さらに、本発明の別態様であるコンピュータ・プログラムは、コンピュータを上述した本発明にかかるアドレス処理方法を実行する手段として機能させるための命令を含むものである。

【発明の効果】

【００４６】

本発明にかかるネットワーク・ルータは、少なくとも２つの異なる長さの複数のプリフィックス、及び当該複数のプリフィックスの長さを格納するフィルタリング・テーブルを有している。これにより、１つのインデックス・テーブル、１つのフィルタリング・テーブル、及び１つのリザルト・テーブルを含む１つのサブ・セルに、長さの異なるプリフィックスを共存させることが可能となる。このような構成により、例えば、プリフィックス集合に含まれるプリフィックスのうち特定のプリフィックス長のものが比較的少ない場合に、これらの比較的少ないプリフィックスをプリフィックス長が異なる他のプリフィックスを扱うサブ・セルに移動できる。これによって、上記３つのテーブルを構成するメモリの利用効率を向上させることができる。

【発明を実施するための最良の形態】

【００４７】

以下では、本発明を適用した具体的な実施の形態について、図面を参照しながら詳細に説明する。各図面において、同一要素には同一の符号が付されており、説明の明確化のため、必要に応じて重複説明は省略する。

【００４８】

発明の実施の形態１．

本実施の形態にかかるネットワーク検索エンジンは、プリフィックス長が異なる複数のプリフィックスを１つのサブ・セルに共存させるよう改良したものである。以下では、本実施の形態にかかるネットワーク検索エンジンについて、図５乃至図８を用いて説明する。

【００４９】

図５に改良されたプリフィックス集合５を示す。図１においては、プリフィックス長３

２ビットの部分集合１４には比較的少ないプリフィックス、具体的には１つのプリフィックス"１２７．０．０．０"しか存在していない。図５のプリフィックス集合５では、プリフィックス長３２ビットの部分集合に存在していたプリフィックス"１２７．０．０．０"が、プリフィックス長１６ビットの部分集合５２に移動されている。移動前の部分集合５２には、移動されたプリフィックス"１２７．０．０．０"のサブ・プリフィックスが存在していないため、移動されたプリフィックス"１２７．０．０．０"と元の１６ビットのプリフィックスとの間でコンフリクトを生じることがない。ここで、サブ・プリフィックスとは、あるプリフィックスの上位ビットで示されるプリフィックス長の短いプリフィックスを意味しており、例えば、プリフィックス"１２７．０"はプリフィックス"１２７．０．０．０"のサブ・プリフィックスの１つである。これにより、図５に示すプリフィックス集合５では、プリフィックス長３２ビットの部分集合が不要となり、これに割り当てるメモリを削減し、また、プリフィックス長１６ビットの部分集合５２に割り当てるメモリの使用効率を向上することができる。

【００５０】

このように、サブ・プリフィックスのコンフリクトを生じないことを条件に、長さの異なるプリフィックスを１つの集合内、言い換えると１つのフィルタリング・テーブル内、さらい言い換えると１つのサブ・セル内に共存させることにより、メモリ利用効率の向上が可能となる。

【００５１】

１つのサブ・セル内においてプリフィックス長の異なる複数のプリフィックスを扱うように改良された本実施の形態にかかるサブ・セル６の構成を図６に示す。サブ・セル６では、フィルタリング・テーブル６４にプリフィックス長フィールド６４２が追加されている。プリフィックス長フィールド６４２には、プリフィックス・フィールド６４１に格納される個々のエレメント（プリフィックス）に対応付けてプリフィックス長が格納されている。フィルタリング・テーブル６４を生成するためには、例えば、上述したファンクション・エンコーディングの工程において、フィルタリング・テーブル６４のアドレスＩ（ｔ）で示される格納場所にエレメントｔが格納される際には、エレメントｔのプリフィックス長も同様にアドレスＩ（ｔ）の格納場所に格納すればよい。

【００５２】

ハッシュ関数を生成する際には、３２ビットのプリフィックス"１２７．０．０．０"が１６ビットのプリフィックスとして扱われる。つまり、３２ビットのエレメントのうち上位の１６ビット分だけがハッシュ関数に使用される。また、フィルタリング・テーブル６４の適切な格納場所に、３２ビットのプリフィックス"１２７．０．０．０"とそのプリフィックス長（３２ビット）が格納される。これによって、ルックアップの実行時には、入力されたエレメント（宛先アドレス）とフィルタリング・テーブル内の３２ビットのプリフィックス"１２７．０．０．０"の全体とが一致した場合にのみ、ルックアップが有効と判断することができる。

【００５３】

つまり、フィルタリング・テーブル６４のプリフィックス長フィールド６４２から読み出されたビット数によって、入力された宛先アドレスと比較されるべきプリフィックス長を知ることができる。具体的には、図６に示すマスク部６８において、入力ヘッダに含まれる宛先アドレスとフィルタリング・テーブル６４から取り出されたプリフィックスが、同じくフィルタリング・テーブル６４から取り出されたプリフィックス長によってマスクされて、比較部３６に出力される。

【００５４】

なお、図６に示すサブ・セル６は、通常は長さ１６ビットのプリフィクスを格納するサブ・セルを示している。このため、ルックアップの実行時は、はじめに１６ビットマスク部６７において、入力された宛先アドレスのうちハッシュ演算の対象となる先頭の１６ビット以外がマスクされる。

【００５５】

このようなサブ・セル６が備えるフィルタリング・テーブル６４中に、３２ビットのプリフィックス"１２７．０．０．０"が格納されている。ここで、３２ビットのプリフィックス"１２７．０．０．０"のサブ・プリフィックスである"１２７．０．ｘ．ｘ"が、フィルタリング・テーブル６４のエントリに存在しないことに注意する必要がある。もしこのサブ・プリフィックスが存在するのであれば、このサブ・セル６にプリフィックス"１２７．０．０．０"を格納することは許されない処理となる。

【００５６】

また、図６のサブ・セル６は、３種類の出力を行う。プリフィックスの一致の有無を示す情報（ｖａｌｉｄ信号）及びネクスト・ホップ・アドレスを出力する点は図３に示した従来のサブ・セルと同様であるが、これに加えて一致したプリフィックス長（ＭＰＬ）を出力する。ＭＰＬは、後述する優先度判定部７２において最長プリフィックス一致判定（ＬＰＭ）を行うための情報として使用される。

【００５７】

本実施の形態にかかるネットワーク検索エンジン７の構成を図７に示す。ネットワーク検索エンジン７は、宛先アドレス等の入力エレメントに対する最長プリフィックス一致判定（ＬＰＭ）を行ってネクスト・ホップ・アドレス等の入力エレメントに対応付けられた関数を出力するものであり、図５に示した従来のネットワーク検索エンジン５を改良したものである。

【００５８】

ネットワーク検索エンジン７は、図６に示したサブ・セル６が並列に配置されている。ネットワーク検索エンジン７に入力された入力パケットのヘッダ情報が、全てのサブ・セル６Ａ乃至６Ｃに並行して送られる。そして、個々のサブ・セルによる判定結果が、優先度判定部７２に送られる。上述のように、サブ・セルが優先度判定部７２に出力する判定結果には、ｖａｌｉｄ信号、ネクスト・ホップ・アドレス、及び一致したプリフィックス長（ＭＰＬ）が含まれる。

【００５９】

優先度判定部７２は、ｖａｌｉｄ信号が有効であるサブ・セルから入力されたＭＰＬを比較することにより、最長のＭＰＬを判定する。そして、最長のＭＰＬと同じサブ・セルから入力されたネクスト・ホップ・アドレスを、最長プリフィックス一致判定（ＬＰＭ）によって得られたネクスト・ホップ・アドレスとして出力する。

【００６０】

このような構成による利点は、プリフィックス集合に含まれるプリフィックスのうち特定のプリフィックス長のものが比較的少ない場合に、これらの比較的少ないプリフィックスをプリフィックス長が異なる他のプリフィックスを扱うサブ・セルに移動できる点である。これによって、サブ・セル内のフィルタリング・テーブルの利用効率を向上させ、サブ・セルの有効利用が可能となる。

【００６１】

発明の実施の形態２．

従来のネットワーク検索エンジン５が備えるサブ・セル３においては、フィルタリング・テーブル３４の大きさ（深さ）、つまりエントリ数は、サブ・セル内に格納されるプリフィックスの数と同数以上である必要があった。これに対して本実施の形態のネットワーク検索エンジンが備えるサブ・セル９は、フィルタリング・テーブルに格納するエントリ数を、サブ・セル９内に格納されるプリフィックスの数より小さくすることが可能である点が特徴である。

【００６２】

以下では、共通のサブ・プリフィックス長Ｌを有する長さＰビットのプリフィックスの完全集合Ｓを考える。ここで完全集合とは、上記の条件を満足する2^{P-L}通りのプリフィックスが全て含まれているプリフィックス集合を意味する。本実施の形態のネットワーク検索エンジンでは、ファンクション・エンコーディングの工程において、この完全集合Ｓに含まれるプリフィックスのうち、同じネクスト・ホップ・アドレスと対応付けられる

最大部分集合を抽出する。さらに、最大部分集合に含まれるプリフィックスをフィルタリング・テーブルの１つの格納場所に集約して格納する。

【００６３】

一例として、図８に示すプリフィックス集合８を考える。プリフィックス集合８は、プリフィックス長Ｐが４ビットであり、サブ・プリフィックス長Ｌが２ビットである場合の完全集合Ｓである。プリフィックス集合８には、共通のサブ・プリフィックスが"００"である４つのプリフィックス"００００"、"０００１"、"００１０"、及び"００１１"が属している。このうち、３つのプリフィックス"００００"、"０００１"、及び"００１０"のネクスト・ホップは"Ｅ"であり、"００１１"のネクスト・ホップは"Ｆ"であるものとする。上述した従来のネットワーク検索エンジン５が備えるサブ・セル３においては、これらのプリフィックスを、フィルタリング・テーブル３４の個別の格納場所にそれぞれ格納する必要があった。これに対して、本実施の形態のネットワーク検索エンジンが備えるサブ・セル９では、ネクスト・ホップが共通する３つのプリフィックスは、フィルタリング・テーブルの１つの格納場所に集約して格納することができる。

【００６４】

本実施の形態にかかるネットワーク検索エンジンが有するサブ・セル９の構成を図９に示す。なお、サブ・セル９が有する構成要素のうち、フィルタリング・テーブル９４４及びビットマスク部９７を除く他の構成要素は発明の実施の形態１のサブ・セルが有する構成要素と同様であるため、同じ符号を付して詳細な説明を省略する。また、本実施の形態にかかるネットワーク検索エンジンの全体構成は発明の実施の形態１と同様であるため説明を省略する。

【００６５】

本実施の形態のフィルタリング・テーブル９４は、プリフィックスが格納されるプリフィックス・フィールド９４１に加えて、発明の実施の形態１のフィルタリング・テーブル６４と同様にプリフィックス長を格納するプリフィックス長フィールド９４２を有している。プリフィックス長フィールド９４２は、誤検出（false positives）を防止するために入力アドレスと比較すべきサブ・プリフィックスの長さが格納される。図８のプリフィックス集合の場合は、３つのプリフィックス"００００"、"０００１"、及び"００１０"を集約したプリフィックス"００ｘｘ"に対応するプリフィックス長フィールド９４２の値は"２"ビットである。他方、プリフィックス"００１１"に対応するプリフィックス長フィールド９４２の値は"４"ビットである。

【００６６】

図８に示した３つのプリフィックス"００００"、"０００１"、及び"００１０"に対するハッシュ演算部３１でのハッシュ演算結果は、インデックス・テーブル３２の異なる格納場所をそれぞれ指定するものである。このため、本実施の形態では、それぞれのハッシュ演算結果（ハッシュ値）で指定されるインデックス・テーブル３２の格納場所には、同じ値がエンコードされるものとする。これによって、それぞれのハッシュ値は異なるものの、インデックス・テーブル３２にエンコードされた値によって指定されるフィルタリング・テーブル９４のアドレスは同一にできる。

【００６７】

これにより、３つのプリフィックス"００００"、"０００１"、及び"００１０"の場合は、先頭の２ビット"００"だけが入力アドレスと比較され、プリフィックス"００１１"の場合は、４ビット全体が入力アドレスと比較される。

【００６８】

なお、図９に示すサブ・セル９は、通常は長さ４ビットのプリフィクスを格納するサブ・セルを示している。このため、ルックアップの実行時は、はじめに４ビットマスク部９７において、入力された宛先アドレスのうちハッシュ演算の対象となる先頭の４ビット以外がマスクされる。

【００６９】

上述した構成によって、フィルタリング・テーブル９４に格納するエントリ数を、サブ

・セル内に格納されるプリフィックスの数より小さくすることができる。したがって、フィルタリング・テーブル９４に必要とされるメモリ容量の削減が可能となる。

【００７０】

その他の実施の形態.

発明の実施の形態１及び２に示したネットワーク検索エンジンの構成は、文献［５，６］に開示されたプリフィックスの前処理技術や、本出願の発明者らによる米国特許出願１１／１３３，２２６に開示されたプリフィックスの前処理技術と組み合わせると有効である。

【００７１】

なお、文献［５，６］は、ツリーベースの解決策及びハッシュベースの解決策に適したプリフィックス展開技術を提案している。当該展開技術は、与えられたプリフィックスを、予め定められたプリフィックス長の集合に属するように展開するものである。例えば、３２ビットのＩＰｖ４アドレスであれば、８ビットの倍数の４種類のプリフィクス長（８、１６、２４、３２ビット）に展開する。このようなプリフィックス展開によって、与えられたプリフィックス集合は、上記のＩＰｖ４アドレスの例であれば、４つのプリフィックス・テーブルによって構成することができる。このとき、各テーブルは、８ビットのアクセスワードによってアドレス可能であって、８ビットで表される網羅的な2^8個のビット・パターンをテーブル要素に有する。一般的には、アクセスワードがｎビットであれば、2^n個のプリフィックスをテーブル要素とするＬｍａｘ／ｎ個のプリフィックス・テーブルによって構成することができる。ここでＬｍａｘは入力アドレス長である。このような、プリフィックスの前処理を行って、プリフィックス・テーブルを構成することにより、ＬＰＭ探索時のプリフィックス・テーブルへのアクセス回数、つまりメモリアクセス回数を抑制するものである。

【００７２】

このような、予め定められたプリフィックス長に展開するプリフィックスの前処理技術を、上述したＢｌｏｏｍｉｅｒフィルタのデータ構造に基づくネットワーク検索エンジンに適用すする場合を考える。上記のプリフィックスの前処理を行った後のプリフィックスは、限られた数のプリフィックス長しか持たないため、限られた数のプリフィックスを処理するサブ・セルのみを設ければよいことになる。

【００７３】

例えば、３２ビットのアドレスであれば、プリフィックスの前処理を行わなければ３２種類のサブ・セルを設ける必要がある。これに対して、プリフィックスの前処理を行って、例えば８，１６，２４及び３２ビットの４通りのプリフィックス長のみに集約すれば、４つのサブ・セルを設けるのみで良く、回路規模を削減できる。

【００７４】

しかしながら、プリフィックスの前処理によって、サブ・セル数を削減できても、各サブ・セルが処理するプリフィックス長は１通りのみである。従って、上述した発明の実施の形態１のネットワーク検索エンジンによって、フィルタリング・テーブルにプリフィックスと対応付けてプリフィックス長を格納し、１つのサブ・セルに複数のプリフィックス長を共存させることにより、メモリ利用効率の向上が可能となる。

【００７５】

また、さらに、上述したプリフィックスの前処理によって、サブ・セル数を削減できても、各サブ・セルが有するフィルタリング・テーブル及びリザルト・テーブルのエントリ数を削減することができない。つまり、集約後のプリフィックス長に含まれる全てのパターンを網羅したエントリを有するフィルタリング・テーブルを備えて、誤検出（false positives）を除去する必要があるため、テーブルを保持するためのメモリサイズが大きいという問題ある。

【００７６】

これに対して、上述した発明の実施の形態２のネットワーク検索エンジンであれば、同じネクスト・ホップ・アドレスと対応付けられる複数のプリフィックスを、フィルタリン

グ・テーブルの１つのエントリ、つまり格納場所に集約して格納することが可能である。これにより、フィルタリング・テーブルのサイズを小さくでき、同様にリザルト・テーブルのサイズを小さくできる。

【００７７】

上述した発明の実施の形態１及び２によって、プリフィックスの前処理技術を適用したプリフィックス集合を、さらに効率よくサブ・セル内に格納することができる。

【００７８】

なお、上述した各実施の形態にかかるネットワーク検索エンジンは、ＩＰパケット等のルーティングを行うネットワーク・ルータにおいて、入力パケットの転送先を決定するルーティング処理を行うルーティング・エンジンとして使用可能なものである。なお、上述した実施の形態で示したフィルタリング・テーブル及びリザルト・テーブルで示される１つのエントリは、プリフィックス、プリフィックス長、及びネクスト・ホップ・アドレスを示しており、ルーティング・プロトコルによって算出されるルーティング・テーブルと同じ構成である。このため、エントリの追加、削除又は変更を行う際に、集約後のプリフィックス長に含まれる全てのパターンを網羅したエントリを有するフィルタリング・テーブルの有する別体系のアドレスを参照するといった必要性も生じない。また、各サブ・セルは、ネットワーク・ルータとしての必要性に応じて、ルーティング・プロトコルによって算出されるルーティング・テーブルのコピーを各々のサブ・セルで持っても、複数のサブ・セルで１つのテーブルを共有しても良い。

【００７９】

また、発明の実施の形態１及び２にかかるネットワーク検索エンジンは、コンピュータ・システムを用いて実現することが可能である。したがって、発明の実施の形態１及び２にかかるネットワーク検索エンジンの処理をコンピュータに実行させるコンピュータ・プログラムも本発明の範囲に含まれるものである。

【００８０】

さらに、本発明は上述した実施の形態のみに限定されるものではなく、既に述べた本発明の要旨を逸脱しない範囲において種々の変更が可能であることは勿論である。

【図面の簡単な説明】

【００８１】

【図１】従来のプリフィックスの集合の一例を示す図である。

【図２】インデックス・テーブルの生成手法を説明するための図である。

【図３】従来のネットワーク検索エンジンが有するサブ・セルの構成例を示す図である。

【図４】従来のネットワーク検索エンジンの構成例を示す図である。

【図５】プリフィックス長の異なる複数のプリフィックスを含むテーブルの生成を説明するための図である。

【図６】プリフィックス長の異なる複数のプリフィックスを含むテーブルのために改良されたサブ・セルの構成を示す図である。

【図７】本発明にかかるネットワーク検索エンジンの構成例を示す図である。

【図８】完全集合であるプリフィックス集合の一例を示す図である。

【図９】本発明にかかるネットワーク検索エンジンが有するサブ・セルの構成例を示す図である。

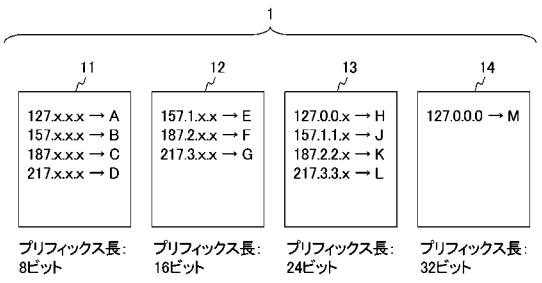
【符号の説明】

【００８２】

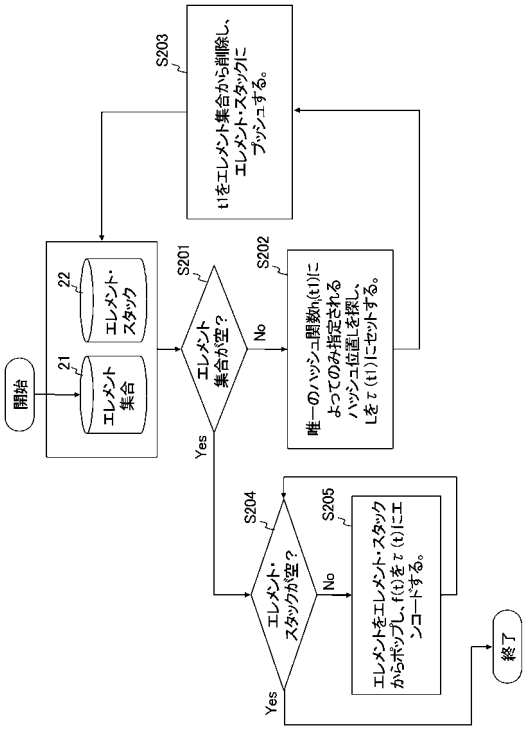
６、６Ａ～６Ｃ　サブ・セル
７　ネットワーク検索エンジン
３１　ハッシュ演算部
３２　インデックス・テーブル
３３　論理演算部
３５　リザルト・テーブル
３６　比較部

６４、９４　フィルタリング・テーブル
６４１、９４１　プリフィックス・フィールド
６４２、９４２　プリフィックス長フィールド
６７　１６ビットマスク部
６８　マスク部
７２　優先度判定部
９７　４ビットマスク部

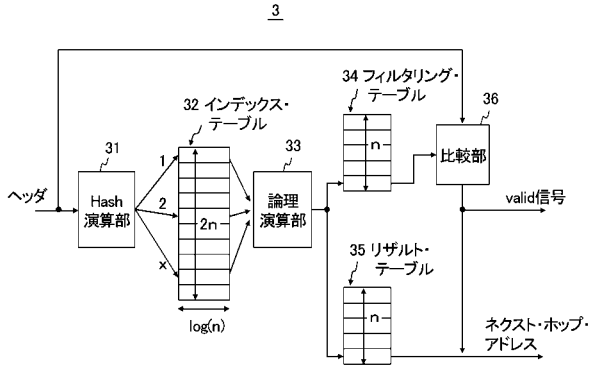
【図１】



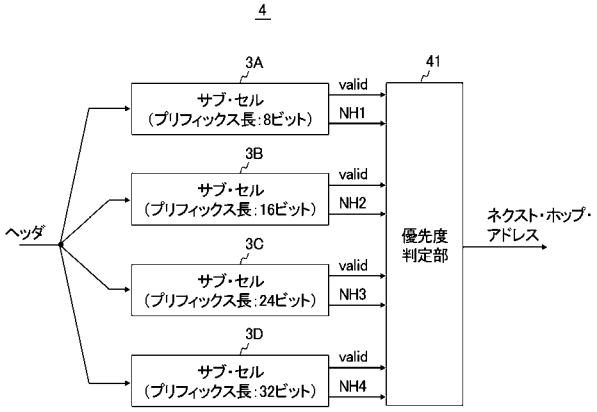
【図２】



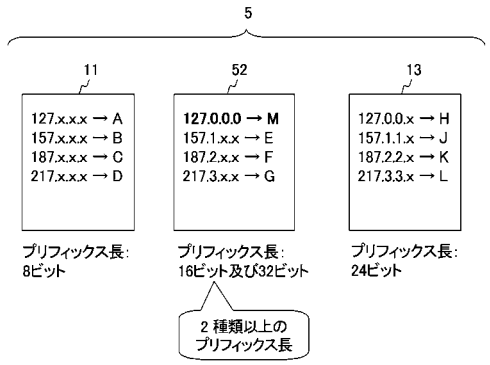
【図３】


3
34 フィルタリング・テーブル
32 インデックス・テーブル
36
31
33
ヘッダ
Hash 演算部
1
2
x
2n
論理 演算部
n
比較部
valid信号
35 リザルト・テーブル
n
log(n)
ネクスト・ホップ・アドレス

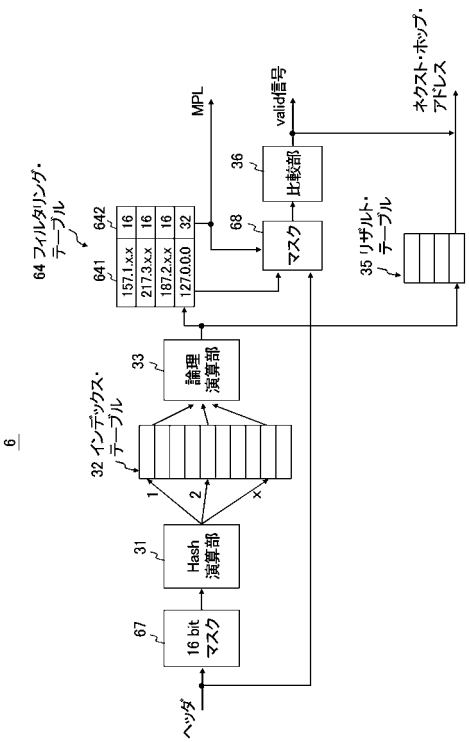
【図４】


4
3A
サブ・セル（プリフィックス長:8ビット）
valid
NH1
41
3B
サブ・セル（プリフィックス長:16ビット）
valid
NH2
ヘッダ
3C
サブ・セル（プリフィックス長:24ビット）
valid
NH3
優先度判定部
ネクスト・ホップ・アドレス
3D
サブ・セル（プリフィックス長:32ビット）
valid
NH4

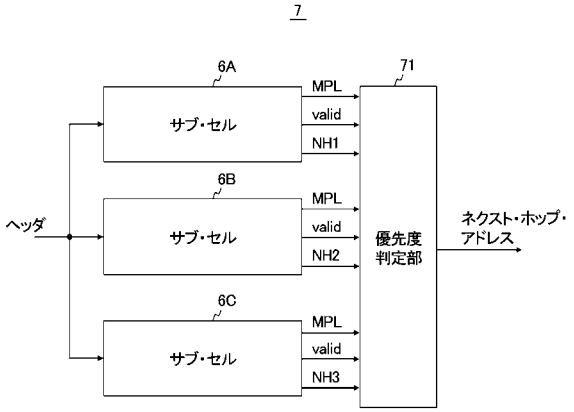
【図５】


5
11
127.x.x.x → A
157.x.x.x → B
187.x.x.x → C
217.x.x.x → D
プリフィックス長: 8ビット
52
127.0.0.0 → M
157.1.x.x → E
187.2.x.x → F
217.3.x.x → G
プリフィックス長: 16ビット及び32ビット
2 種類以上のプリフィックス長
13
127.0.0.x → H
157.1.1.x → J
187.2.2.x → K
217.3.3.x → L
プリフィックス長: 24ビット

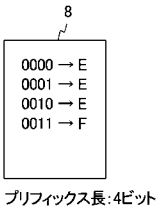
【図６】


6
64 フィルタリング・テーブル
642
641
157.1.x.x
217.3.x.x
187.2.x.x
127.0.0.0
16
16
16
32
MPL
valid信号
ネクスト・ホップ・アドレス
36
比較部
68
マスク
35 リザルト・テーブル
33
論理 演算部
32 インデックス・テーブル
1
2
x
31
Hash 演算部
67
16 bit マスク
ヘッダ

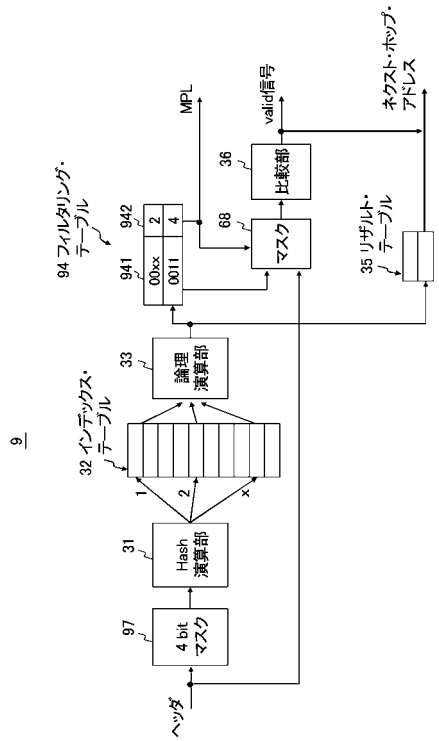
【図７】


7
6A
71
MPL
valid
NH1
サブ・セル
6B
MPL
valid
NH2
サブ・セル
ヘッダ
優先度
判定部
ネクスト・ホップ・
アドレス
6C
MPL
valid
NH3
サブ・セル

【図８】


8
0000 → E
0001 → E
0010 → E
0011 → F
プリフィックス長:4ビット

【図９】


9
94 フィルタリング・
テーブル
942
941
2
4
00xx
0011
MPL
valid信号
ネクスト・ホップ・
アドレス
36
比較部
68
マスク
35 リザルト・
テーブル
33
論理
演算部
32 インデックス・
テーブル
1
2
x
31
Hash
演算部
97
4 bit
マスク
ヘッダ

フロントページの続き

(72)発明者　スリハリ・カダンビ
アメリカ合衆国　ニュージャージー州　０８５４０　プリンストン　インディペンデンス・ウェイ　４　エヌイーシーラボラトリーズアメリカインコーポレイテッド内
(72)発明者　スリマト・チャクラダー
アメリカ合衆国　ニュージャージー州　０８５４０　プリンストン　インディペンデンス・ウェイ　４　エヌイーシーラボラトリーズアメリカインコーポレイテッド内

Ｆターム(参考) 5K030 GA01 HA08 HD03 HD09 JA11 KA01 KA05 MD10