(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第4387519号
(P4387519)

(45)発行日　平成21年12月16日(2009.12.16)　(24)登録日　平成21年10月9日(2009.10.9)

(51)Int.Cl.　　F I
H04L 12/18 (2006.01)　　H04L 12/18
H04L 12/56 (2006.01)　　H04L 12/56　260Z

請求項の数 6　(全 11 頁)

(21)出願番号　特願平11-299464
(22)出願日　平成11年10月21日(1999.10.21)
(65)公開番号　特開2000-134199(P2000-134199A)
(43)公開日　平成12年5月12日(2000.5.12)
審査請求日　平成18年9月26日(2006.9.26)
(31)優先権主張番号　09/176703
(32)優先日　平成10年10月21日(1998.10.21)
(33)優先権主張国　米国(US)

(73)特許権者　506355419
エリクソン　アーベー
Ｅｒｉｃｓｓｏｎ　ＡＢ
スウェーデン，ストックホルム，トーシャ
マンズガーテン　２３
(74)代理人　100066728
弁理士　丸山　敏之
(74)代理人　100100099
弁理士　宮野　孝雄
(74)代理人　100111017
弁理士　北住　公一
(74)代理人　100119596
弁理士　長塚　俊也
(74)代理人　100141841
弁理士　久徳　高寛

最終頁に続く

(54)【発明の名称】マルチキャスト樹を蓄積するための効果的な手段

(57)【特許請求の範囲】
【請求項１】
ノード及び該ノードを繋ぐリンクを含んでいるネットワークと、ノード間に重複したブランチを有さないマルチキャストコネクションを構築する構築メカニズムとを含んでおり、
構築メカニズムは、葉がマルチキャストコネクションに追加或いは削除されるときに、重複するブランチを有さないマルチキャストコネクションを構築し、
マルチキャストコネクションは、ネットワークのノードの中に根ノードを持ち、該根ノードが構築メカニズムを含んでおり、
構築メカニズムは、重複するブランチを有さないマルチキャストコネクションを構成するのに必要なマルチキャストコネクションに関するすべての情報を、根ノードへ蓄積し、管理し、
マルチキャストコネクションは、ノード、リンク及び葉の経路を決定し、
構築メカニズムは、マルチキャストコネクションのための経路を選択する経路選択メカニズムを含んでおり、
マルチキャストコネクションの各ノードは、メモリーを持っており、樹形図で表される、経路のリンク、ノード及び葉を含めたマルチキャストコネクションに関する情報が、電子化された形で、メモリー内に蓄積され、
単一のＰＮＮＩピアグループにわたった、マルチキャストコネクションの経路の葉及びノードがＤＴＬを規定し、葉がマルチキャストコネクションに追加される場合に、樹形図

に未だ現れていなかった、ＤＴＬに使用される全てのリンクが樹形図に追加され、葉は、樹形図に追加された最後のリンクに追加される、マルチキャストのためのシステム。

【請求項２】

経路の各ノードは、ノードから、該ノードより下位の全ての葉に至る経路のために、対応する樹の経路を管理する、請求項１に記載のシステム。

【請求項３】

各ノードは、対応する樹の経路を管理する構築メカニズムを持ち、経路のリンクはブランチを含んでおり、葉が削除されたときに、そのブランチよりも下位に葉を持たないあらゆるブランチも同様に削除される、請求項２に記載のシステム。

【請求項４】

根ノードは、経路の各ＰＮＮＩブランチに対するリファレンス数を管理し、リファレンス数が０になった場合には、それに対応するＰＮＮＩブランチは削除されるものであり、ここに於いて、ＰＮＮＩブランチのリファレンス数は、子ブランチの数に等しいものであり、葉ブランチは、それに対応する葉がマルチキャストコネクションから取り除かれたときに削除される、請求項３に記載のシステム。

【請求項５】

葉がマルチキャストコネクションに追加或いは削除されるときに、重複するブランチを持たないマルチキャストコネクションを、ネットワークに構築する構築メカニズムと、ネットワークに繋がったポートメカニズムとを含んでおり、ポートメカニズムを通じてコネクションがマルチキャストネットワークに送られ、構築メカニズムは、ポートメカニズムに接続されているＡＴＭネットワークのためのノードであって、

構築メカニズムに接続されたメモリーを持っており、該メモリーは、重複したブランチを持たないマルチキャストコネクションを構築メカニズムが構築するのに必要な、マルチキャストコネクションに関する情報を有しており、

マルチキャストコネクションに関する情報は、マルチキャストコネクションを規定しているノード、リンク、及び葉の経路を含んでおり、

経路のリンク、ノード、及び葉は、メモリー内に電子化された形態で樹形図として表されており、

構築メカニズムは、経路の各ＰＮＮＩブランチのためのリファレンス数を管理し、リファレンス数が０になった場合には、それに対応するＰＮＮＩブランチは削除され、ここに於いて、ＰＮＮＩブランチのリファレンス数とは子ブランチの数に等しいものであり、葉ブランチは、それに対応する葉がマルチキャストコネクションから取り除かれたときに削除されるノード。

【請求項６】

葉がマルチキャストコネクションに追加或いは削除されるときに、重複するブランチを持たないマルチキャストコネクションを、ネットワークに構築する工程を含んでおり、

マルチキャストコネクションの各ノードはメモリーを持っており、マルチキャストコネクションは、ノード、リンク、及び葉の経路を決め、

マルチキャストコネクションを構築する工程の前に、樹形図で表されるような、マルチキャストコネクションの経路のリンク、ノード、及び葉を、電子化された形でメモリー内に蓄積する工程があり、

マルチキャストコネクションは、ＤＴＬを規定する単一のＰＮＮＩにわたった経路の葉、リンク、及びノードを含み、マルチキャストコネクションを構築する工程は、葉を経路に追加し、樹形図に未だ現れていないＤＴＬに用いられるリンクを樹形図に追加し、葉が、樹形図に追加された最後のリンクに追加される工程を含んでいる、マルチキャストコネクションを構築するための方法。

【発明の詳細な説明】

【０００１】

【発明の分野】

本発明は、マルチキャストコネクションに関係するものである。特に、本発明は、重複す

るブランチを持たない、マルチキャストコネクションに関係するものである。
【０００２】
【発明の背景】
ＡＴＭネットワーク内のマルチキャストコネクション（一点から複数点へ、あるいは、複数点から一点へ）が正しく動作するためには、必ずしも、完全な樹形図(tree graph)を形作る必要はない。所定のマルチキャストコネクションのブランチは、重複し、「ループ」を構築しても、データフローの正確さを損なうことはない。しかしながら、ブランチが重複していると、著しい非効率を引き起こすことになる。コネクションの所定のノードが、同じデータセルの複数のコピーを不必要に受け取ることになり、従って、帯域幅と処理能力を浪費してしまうのである。
【０００３】
幾つかのマルチキャストコネクション（例えば、ＬＡＮエミュレーションコネクション）は、長時間、回線を占有してしまう。ブランチの重複によって引き起こされる非効率は、ネットワーク全体のスループットを、長時間にわたって極度に低下させる恐れがある。
本発明は、所定のマルチキャストコネクションの根(root)ノードにおいて、参加者がコネクションに追加或いは削除されるときに、該コネクションは、重複するブランチを持たない事を保障するために、コネクションに関する必要な情報の全てを蓄積し、管理するためのメカニズムを提供する。蓄積された情報は、経路選択アルゴリズムと連結して使用される。
【０００４】
【発明の要約】
本発明は、マルチキャストのためのシステムに関係する。該システムは、ノードと、該ノードを接続するリンクからなるネットワークを含んでいる。該システムは、重複するブランチを持たないマルチキャストコネクションを構築するための、メカニズムを含んでいる。
本発明は、ＡＴＭネットワークのノードに関係する。該ノードは、構築メカニズムであり、参加者がコネクションに追加或いは削除される時に、重複するブランチを持たないマルチキャストコネクションを、ネットワーク内に構築するためのものを含んでいる。該ノードは、ネットワークに接続するポートメカニズムを含み、それを通じて、コネクションは、ネットワークへと送られる。構築メカニズムは、ポートメカニズムに接続している。
【０００５】
本発明は、マルチキャストコネクションを構築するための手段に関係する。該手段は、第一のノードと第二のノードの間に、第一のコネクションを構築するステップを含んでいる。続いて、第一のノードと第三のノードの間に、第一のコネクションを延長することによって、第二のマルチキャストコネクションを構築する工程を含んでいる。
添付した図には、発明の望ましい具体例と、発明の実施の望ましい手段が描かれている。
【０００６】
【詳細な説明】
図を参照しながら説明する。図面中の同じ数字は、数個のビューを通じて、同様或いは同一の部分に対応している。特に、図１と図３は、マルチキャストのためのシステム(10)を示している。システム(10)は、ノード(14)と、ノード(14)を接続しているリンク(16)を含む、ネットワーク(12)を含んでいる。システム(10)は、ノード(14)の間に重複するブランチ(30)を持たない、マルチキャストコネクションを構築するためのメカニズム(18)を含んでいる。
【０００７】
望ましくは、構築メカニズム(18)は、参加者がコネクションに追加或いは削除されるときに、重複するブランチ(30)を持たないマルチキャストコネクションを構築する。各コネクションは、望ましくは、ネットワーク(12)のノード(14)に繋がった根ノード(root node)(14r)を持っており、根ノード(14r)は、構築メカニズム(18)を含んでいる。望ましくは、構築メカニズム(18)は、根ノード(14r)にあって、参加者がコネクションに追加或いは削

除されるときに、重複するブランチ(30)を持たないマルチキャストコネクションをノードが構築することを保証するために必要なコネクションに関する全ての情報を蓄積し、保管する。

【０００８】

コネクションは、望ましくは、ノード(14)、リンク(16)、葉(24)の経路(23)を確定する。このとき、構築メカニズム(18)は、コネクションのために経路(23)を選択するために、当該技術分野でよく知られている、経路選択メカニズム(21)を含んでいる。望ましくは、各ノード(14)はメモリ(26)を備えている。そしてコネクションに関する情報には、メモリ(26)中に、電子化された形で記憶されている樹形図(28)によって表される様な、経路(23)のリンク(16)、ノード(14)、葉(24)が含まれている。単一のＰＮＮＩピアグループにわたった、コネクションのための経路の葉(24)及びノード(14)は、望ましくは、ＤＴＬを規定し、また、葉(24)が追加された場合には、樹形図(28)に未だ含まれていなかった、ＤＴＬで用いられている全てのリンク(16)が、樹形図(28)に追加され、葉(24)は、樹形図(28)へ最後に追加されたリンク(16)へ加えられることになる。

【０００９】

望ましくは、経路(23)の各ノード(14)は、ノード(14)から、該ノード(14)よりも下位にあたる全ての葉(24)に至る、経路(23)に対応する樹の経路(23)を管理している。各ノード(14)は、望ましくは、対応する樹の経路(23)を管理する、結合された構築メカニズム(18)を持っており、ここに於いては、経路(23)のリンク(16)はブランチ(30)を含んでおり、葉(24)が削除された場合には、ブランチ(30)より下位にいかなる葉(24)も持たないブランチ(30)も、同様に削除される。望ましくは、根ノード(14r)は、経路(23)の各PNNIブランチ(30)のためのリファレンス数を管理しており、リファレンス数が０になった場合には、対応するPNNIブランチ(30)は削除される。ここに於いて、PNNIブランチ(30)のリファレンス数は、このブランチ(30)の数に等しいものであり、葉のブランチ(30)は、それに対応する接続先(24)がコネクションから削除された場合に、削除される。

【００１０】

本発明は、ＡＴＭネットワーク(12)のノード(14)に関する。ノード(14)は、構築メカニズム(18)であり、参加者がコネクションに追加或いは削除されたときに、重複するブランチ(30)を持たないようなマルチキャストコネクションをネットワーク(12)内に構築するためのものを含んでいる。ノード(14)は、ポートメカニズム(32)であり、ネットワーク(12)に繋がっており、それを通じてコネクションがネットワーク(12)に送られるものに繋がっている。構築メカニズムは(18)は、ポートメカニズム(32)に繋がっている。

【００１１】

望ましくは、各ノード(14)は、メモリ(26)であり、参加者がコネクションに追加或いは削除されるときに、重複するブランチ(30)を有しないマルチキャストコネクションを、構築メカニズム(18)が構築することを確実にするのに必要なコネクションに関する情報を含んでいる。コネクションに関する情報は、コネクションを形作る、ノード(14)、リンク(16)、及び葉(24)の経路(23)を含んでいることが望ましい。各経路(23)のリンク(16)、ノード(14)、及び葉(24)は、樹形図(28)によって表される様な、電子化された形でメモリ(26)内に記憶されることが望ましい。構築メカニズム(18)は、望ましくは、各経路(23)の各PNNIブランチ(30)のためのリファレンス数を管理しており、リファレンス数がゼロになった場合には、対応するＰＮＮＩブランチ(30)は削除される。ここに於いて、ＰＮＮＩブランチ(30)のリファレンス数は、このブランチ(30)の数に等しいものであり、葉であるブランチ(30)は、対応する葉(24)がコネクションから取り除かれた場合には、削除される。

【００１２】

本発明は、マルチキャストコネクションを構築する方法に関係する。該方法は、第一のノード(14a)と第二のノード(14b)の間に、第一のコネクション(50)を構築する工程を含んでいる。次に、第一のノード(14a)と第三のノード(14c)の間に、第一のコネクション(50)を延長することで、第二のマルチキャストコネクション(52)を構築する工程を含んでいる。望ましくは、第一のコネクションを構築する工程の後に、第一のノード(14a)から追加さ

れたノード(14)とを繋ぐマルチキャストコネクションを、重複したブランチ(30)を持つマルチキャストコネクションとならないように構築する工程がある。マルチキャストコネクションを構築する工程は、望ましくは、参加者がコネクションに追加或いは削除されたときに、重複するブランチ(30)を持たないマルチキャストコネクションを構築する工程を含んでいる。

【００１３】

望ましくは、各ノード(14)はメモリ(26)を有し、各コネクションは、ノード(14)の経路(23)、リンク(16)、及び葉(24)によって造形られるのだが、マルチキャストを構築する工程の前に、樹形図(28)によって表される経路(23)のリンク(16)、ノード(14)、及び葉(24)を、電子化した形で、メモリ(26)内に記憶する工程がある。少なくとも一つのコネクションは、望ましくは、一つのＰＮＮＩピアグループにわたって、ＤＴＬを決めており、経路(23)の葉(24)、リンク(16)、及びノード(14)を含んでおり、樹形図に未だ表されていないＤＴＬに用いられているリンク(16)を樹形図(28)に追加し、葉は、樹形図(28)の最後のリンク(16)へ加えられる。

【００１４】

以下の定義は、望ましい実施例の操作の、より良い理解に役立つものである。

ADD PARTY：既存のマルチキャストコネクションに対して、新しい葉(24)（端末）を追加するための要求が為されていることを示す信号。

ADD PARTY REJECT：新しい参加者を、マルチキャストコネクションに追加する作業に、失敗したことを示す信号。

帯域幅：ネットワーク(12)内のノード(14)或いはリンク(16)のデータ通信のための、理論的な容量。

ブランチ(30)：図の一部分；樹形図の一部。

コネクション：ネットワーク(12)内の、二つ（以上）のノード(14)を繋ぐ通信回路。

ＤＴＬ：指定された経路のリスト(Designated Transit List)。一つのPNNIピアグループにわたって、経路(23)を完全に特定する、ノード(14)とリンク(16)のリスト。

DROP PARTY：既存のマルチキャストコネクションから、葉(24)（端末）を削除するための要求が為されていることを示す信号。

葉(24)：示された樹形図(28)上の、一つのブランチ(30)のみと繋がっている頂点。

葉ユーザー(leaf user)：データの流れの一つの目的地とされたノード(14)。

ＬＧＮ：ノード(14)の論理上のグループ(Logical Group Node)。ＰＮＮＩルーティングの階層構造上の一つのレベルの上で操作を行う目的で、一つのポイントとして仮定された、低位のピアグループの略称。

リンク(16)：二つの論理上のノード(14)の接続の略称。

マルチキャスト：単一のソースから、数個の目的先に対して、ネットワークを介して同時に行われるデータ通信。

ネットワーク(12)：リンク(16)によって接続されており、それを介してデータの通信が可能とされた、ノード(14)のグループ。

ノード(14)：ＰＮＮＩルーティングプロトコルの、一つの機材を示す略称。

経路(23)：ネットワークを介した、特定のソースノード(14)から特定の目的先ノード(14)への通路。

ピアグループ：ルーティングの階層を作り出す目的でまとめられた、一連の論理上のノード(14)。

ＰＮＮＩ：個人的なネットワーク(12)からネットワーク(12)へ、或いは、ピアグループからネットワーク(12)へのインターフェース(Private Network to Network Interface / Peer Network to Network Interface)。同時に、ＡＴＭフォーラムによって作られた規格に準拠するプロトコルを表している。バージョン1.0の仕様は、１９９６年に承認され、ここに引用によって組み込まれている。

ポート：リンク(16)をノード(14)に接続する部分。

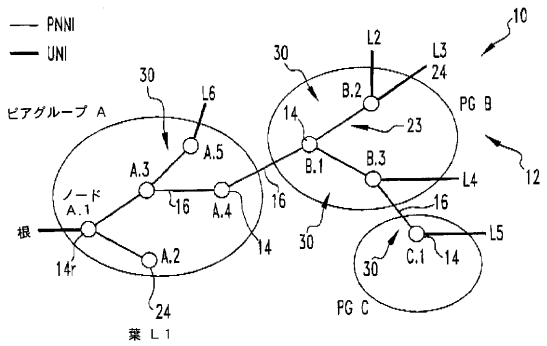
RELEASE：コネクションの切断を要求する信号。

根ユーザー：データの流れの源泉とされたノード(14)。
SET UP：新しいコネクションを確立するための要求が為されたことを示す信号。
樹（樹形）：ネットワーク(12)のグラフ。
ＵＮＩ：使用者とネットワーク(12)のインターフェース(User to Network Interface)。同時に、ＡＴＭフォーラムによって作製された規格に準拠したプロトコルを示す。
【００１５】
望ましい実施例の操作に於いて、マルチキャストコネクションによって網羅される各ノード(14)は、重複する二つのブランチ(30)を持たないようにＤＴＬを割り出すことを保証するために、コネクションに関する情報を管理している。後述するマルチキャストコネクションのビューは、この目的のために各ノード(14)が管理する情報を表している。
ネットワーク(12)内の不安定によって、ブランチ(30)の重複を回避できない場合がおこりうる（引用を以て本願への記載加入とされている、ＰＮＮＩ Errata item #28を参照）。これらの場合、マルチキャストコネクションによって網羅されている特定のノード(14)上に、複数の特定のマルチキャストコネクションの可能例が存在しうる。これらのそれぞれの可能例は、固有の組み合わせ（受信したインターフェース、呼び出しリファレンス）によって識別される。一つのコネクションのビューは、マルチキャストコネクションのそれぞれの可能な例のために管理されている。
【００１６】
マルチキャストコネクションのノード(14)からのビューは、ノード(14)に根を置く樹形図(28)であり、ＰＮＮＩノード(14)を頂点とし、末端ユーザーを葉(24)とし、ＰＮＮＩ及びＵＮＩリンク(16)はブランチ(30)となる。（樹形図(28)の葉(24)は、一つのブランチ(30)のみと接続する頂点である）。
ブランチ(30)は、それに接続される対象によって、以下の２種の位相に分類することが出来る：
・ＰＮＮＩブランチ(30)：二つのＰＮＮＩノード(14)の接続するブランチ(30)。
・葉ブランチ(30)：参加者をＰＮＮＩノード(14)に接続するブランチ(30)。
【００１７】
以下の語彙は、樹形図(28)内での相対的な位置について、ブランチ(30)間の関係性を示すために用いられる；
・親ブランチ(30)：所定のブランチ(30)よりも上位（即ち、樹形図の根に向かう方向）に直接に繋がれたブランチ(30)。
・子ブランチ(30)：所定のブランチ(30)よりも下位（即ち、樹形図の根と反対の方向）に直接に繋がれたブランチ(30)。
根ノード(14r)が直接に取り付けられているノード(14)、即ち、入り口の境界にあたるノード(14)にとっては、樹形図(28)を構成するＰＮＮＩリンク(16)は、マルチキャストコネクションのためにノード(14)によって作られたＤＴＬに含まれる、全てのポートである。最低位のノード(14)にとっては、樹形図(28)を構成するＰＮＮＩリンク(16)は、ノード(14)が所定のマルチキャストコネクションに対してSETUP或いはADD PARTYのメッセージを送った、全てのポートである。
【００１８】
新しい葉(24)への経路(23)を計算するときは、ＤＴＬ作製装置は、マルチキャストコネクションのビューを考慮に入れ、新しい経路(23)が樹形図(28)にループを引き起こさないように保障する。同様に、入り口との境界にあたるノード(14)も、低位のＤＴＬを計算するときに、そのビューを考慮に入れる。最低位のノード(14)は、複数の並列なリンク(16)があり、最低位のＤＴＬがＩＤゼロのポートを特定している場合には、次のノード(14)のためのADD PARTYを送るべき特定のポートを決定するために、そのビューを用いる。一般的に、エリアの入り口の境界にあたるノード(14)、例えば図１のＢ.１は、そのエリアの全ての情報を管理しているので、根ノード(14r)は、エリアＢとＣを、単純にノードＢ.１として把握する。しかし、図２(a)に示されるような根ノードＡ.１のビューから語ると、ノードＢ.1はその樹の下位のものに関する全ての情報を持っているので、根ノードＡ.１は

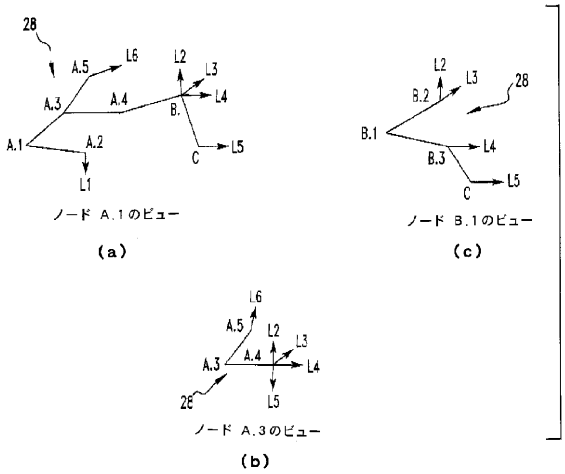
、エリアＢとＣのマルチスキャンコネクション全体を把握している。
図１及び図２(a)，２(b)，２(c)は、マルチキャストコネクションと、コネクションの様々なノード(14)に対応したビューの例を表している。
【００１９】
ノード(14)は、葉(24)が追加或いは削除されたときに、そのマルチキャストコネクションのビューを管理する。葉(24)が追加されたとき、樹形図(28)にはまだ現れていないＤＴＬに用いられる、全てのリンク(16)はグラフに追加され、葉(24)は、直前のリンク(16)に取り付けられる。葉(24)が削除された場合、その葉と、最も近い枝分かれ地点から葉(24)へ連なる全てのブランチ(30)は、樹形図(28)から削除される。例えば、葉Ｌ.５がコネクションから削除された場合には、ノード(14)Ｂ.１は葉Ｌ.５とブランチＢ.３->Ｃをそのビューから取り去り、ノード(14)Ａ.３はＬ.５を取り去り、そして、ノード(14)Ａ.１はＬ.５とブランチＢ->Ｃを取り去る。
【００２０】
コネクションビューの管理は、リファレンス数を用いることで、簡単に実施することができる。リファレンス数は、コネクションビューの各ブランチ(30)ごとに保持される。ＰＮＮＩブランチ(30)は、そのリファレンス数がゼロになった場合には、樹形図(28)から削除される。ＰＮＮＩブランチ(30)のリファレンス数は、その子ブランチ(30)の全体の数に等しい。これらの子ブランチ(30)は、葉又はＰＮＮＩブランチ(30)のいずれかであろう。葉ブランチ(30)に対しては、リファレンス数は存在しない。葉ブランチ(30)は、対応する葉がコネクションから削除された場合には、樹形図(28)から削除される。
図１の例を用いると、ブランチＢ.１ ->Ｂ.３のリファレンス数は２であり、同じくブランチＢ.３->Ｃは１である。Ｌ５がＢ.１のビューから削除された場合には、ブランチＢ.３->Ｃのリファレンス数はゼロとなり、従って、該ブランチは削除される。ブランチＢ.１ ->Ｂ.３のリファレンス数は１に低下し、従って、このブランチは存続する。
【００２１】
マルチキャストコネクションのノードからのビューは、葉(24)を追加或いは削除することによって管理される。リファレンス数は各ＰＮＮＩブランチに関係しており、その子ブランチ(30)の数によって定義される。子ブランチ(30)は、ＰＮＮＩでも葉ブランチ(30)でもあり得る。ＰＮＮＩブランチ(30)は、そのリファレンス数がゼロになった場合に、ビューから削除される。葉ブランチ(30)は、葉(24)が削除されたときに、ビューから削除される。
葉ブランチ(30)は、終端リファレンスによって識別される。この終端リファレンスは、対応する葉を識別するために、出力のためのインターフェースで局所的に用いられる。終端リファレンスが（DROP PARTY又はRELEASEのメッセージを使って）消去される場合には、対応するブランチ(30)は、コネクションビューから削除される。
【００２２】
葉(24)は、ノード(14)が、葉をコネクションに参加させるためのSETUP又はADD PARTYのメッセージを送った場合に、ノードのコネクションビューに追加される。葉を追加する場合には、以下の過程が踏まえられる：
・ノード(14)がSETUP又はADD PARTYに対してＤＴＬを生起させた場合には、ノード(14)は、樹形図(28)に未だ現れていない、生起されたＤＴＬに含まれる全てのＰＮＮＩリンク(16)を、ビューに追加する。続いて、葉が、生起されたＤＴＬによって特定される目的先ノード(14)に取り付けられる。ノード(14)が、根ノード(14r)に対して直接に取り付けられているノード(14)である場合、目的先ノード(14)は、ノード(14)或いは、葉を含む論理上のノード(14)のグループであることを記しておく。ノード(14)が入り口の境界にあたるノード(14)である場合、目的先ノード(14)は、ＬＧＮであって、境界ノード(14)が受け取ったＤＴＬの次の通路として特定されるものである。
・ノード(14)が、SETUP又はADD PARTYに対していかなるＤＴＬ（即ち、それが通過点にあたるノード(14)である）も生起せず、メッセージがＰＮＮＩリンク(16)に送られた場合、ノード(14)は、ＰＮＮＩリンク(16)をビューに加え、葉をリンク(16)の遠い側の終端に

取り付ける。
・ノード(14)が、SETUP又はADD PARTYに対していかなるＤＴＬも生起せず、メッセージがＵＮＩリンク(16)に送られた場合、ノード(14)は、葉をノード(14)自体へ取り付ける。ブランチ(30)（葉ブランチ(30)或いはその他）がビューに加えられる場合、その親ブランチ(30)のリファレンス数は、如何なるものであれ、一つづつ増加する。
【００２３】
葉は、信号がコネクションから関連する参加者を削除した場合即ち、下記の何れかの出来事が起こった場合に、ノードのコネクションビューから削除される：
・ノード(14)が、葉(24)に対するDROP PARTYを受け取る場合。
・ノード(14)が、葉(24)に対するADD PARTY REJECTを受け取る場合。
・ノード(14)が、葉(24)が加えられたインターフェースでRELEACEを受け取る場合。
・葉が取り付けられたインターフェースが故障した場合。
葉(24)がビューから削除される場合、葉(24)ブランチ(30)も削除される。葉(24)ブランチ(30)が親ブランチ(30)を持っていた場合、親ブランチ(30)のリファレンス数はゼロとなり、親ブランチ(30)も削除される。リファレンス数がゼロになった場合、親ブランチ(30)もまた削除され、その親ブランチ(30)（存在すれば）のリファレンス数も１減少し、以下同様に続く。
【００２４】
ノード(14)が、マルチキャストコネクションへの新しい葉ユーザーにSETUP又はADDPARTYを配送するために、ＤＴＬを計算することが必要となった場合には、そのコネクションのビューを考慮に入れて、新しいＤＴＬがブランチ(30)の重複制限にかかってしまうことが無いように保障する。同様に、最低位のノード(14)は、ADD PARTYを送るべき特定のＰＮＮＩリンク(16)を決定することが必要である場合には、そのビューを考慮して、一つ以上のリンク(16)が次のノード(14)に到達するために用いられないように保障する。
附表(図４～図６)は、システム(10)の可能な履行を記載している。
【００２５】
本発明は、簡素化の目的で、上記実施例の詳細にわたって説明されてきたが、そのような詳細は単にその目的のためであり、当該分野の専門家によって、後述の特許請求の範囲に記載されるようなもののほかにも、発明の精神と範囲から逸脱することなく、変化を加えることが出来ることが理解されるであろう。
【図面の簡単な説明】
【図１】マルチキャストコネクションの概略図である。
【図２】 (a), (b), (c)は、図１のマルチキャストコネクションの、ノードＡ.１，Ａ.３，Ｂ.１のそれぞれから見た概略図である。
【図３】本発明のノードの概略図である。
【図４】システム(10)の可能な履行を記載している。
【図５】図４の続きを示す図である。
【図６】図５の続きを示す図である。
【符号の説明】
(10) システム
(12) ネットワーク
(14) ノード
(16) リンク
(23) 経路
(24) 葉
(28) 樹形図
(30) ブランチ

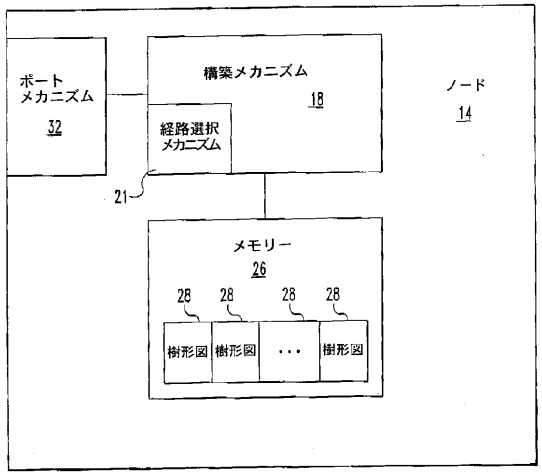
【図１】



【図２】



【図３】



【図４】

Data Structures

Multicast Connection identifier
Type : mcon_id_t
Fields :
portid (unsigned integer) port ID of the root-side interface of the connection
c_ref (unsigned integer) call reference for the connection at the root-side interface

DTL Transit
Type : dtl_transit_t
Fields :
nodeid (22 byte node ID) ID of the transit node
portid (unsigned integer) ID of the transit port

Edge
Type : mcon_edge_t
Fields :
dtl (dtl_transit_t) DTL transit represented by this edge, also used as a key to identify the edge
parent (mcon_edge_t pointer) pointer to the parent edge
refcnt (integer) number of child edges

Leaf
Type : mcon_leaf_t
Fields :
parent (mcon_leaf_t pointer) pointer to parent edge

Multicast Connection
Type : mcon_t
Fields :
id (mcon_id_t) ID of the multicast connection
leaves binary tree of mcon_leaf's indexed by end point reference
edges binary tree of mcon_edge's indexed by edges' dtl field

(図５へ続く)

【図５】

(図４より)
Routines

Adding a Leaf
Arguments :
mcon (mcon_t) the affected multicast connection
leaf (mcon_leaf_t) the leaf to be added
dtl (array of dtl_transit_t's) the DTL computed for the given leaf; the DTL transits in dtl are in reverse order, i.e., from destination to source
Local variables :
hop (dtl_transit_t) DTL transit
prev_edge (mcon_edge_t) previous edge
is_merged (boolean_t) true when the DTL to be added merges with the tree
edge (mcon_edge_t) edge being examined currently

```
BEGIN :
    set hop to first transit in dtl;
    prev_edge = 0;
    is_merged = false;

    tree = mcon->edges;

    while (hop is a valid DTL transit) {
      find an edge in tree matching hop
      if not found {
        if is_merged is true {
          /*
           * Try to branch out after merging with tree, ignore rest of
           * dtl and return.
           */
          return;
        }
        /*
         * Else, create edge and add to tree
         */
        create edge with hop as id and add to tree;
        if (prev_edge) {
          prev_edge->parent = edge;
        }
        else {
          leaf->parent = edge;
        }
        edge->refcnt ++;
```

(図６へ続く)

【図６】

```
(図５より)
        }
        else { /* edge found */
          if is_merged is false { /* first time hitting an old branch */
            is_merged = true;
            if (prev_edge) {
               prev_edge->parent = edge;
            }
            else {
              leaf->parent = edge;
            }
             edge->refcnt ++;
          } /* else do nothing */
        }
        prev_edge = edge;
        set hop to next transit in dtl
    } /* while */
END

Deleting a Leaf
Arguments :
mcon            (mcon_t)  the affected multicast connection
leaf            (mcon_leaf_t)  the leaf to be deleted
Local Variables :
parent_edge     (mcon_edge_t)  the parent edge being processed
child_edge      (mcon_edge_t)  the child edge from parent edge leading to leaf
BEGIN:
    parent_edge = leaf->parent;
    while (parent_edge is a valid edge ) {
        /*
         * Decrement reference count to account for the leaf drop
         */
        parent_edge->refcnt--;
        if (parent_edge->refcnt != 0) {
          /*
           * There are other leaves still reachable from this edge, just return
           */
          return;
        }
        child_edge = parent_edge;
        /*
         * Last leaf reachable from edge to be removed, drop the edge
         */
        Remove child_edge from mcon->edges;
    }
END
```

フロントページの続き

(72)発明者 セオドア イー. テディジャント
アメリカ合衆国 ９５１２０ カリフォルニア, サンホゼ, レンウッド ウェイ ６９２４
(72)発明者 ラビ タンガラジャ
アメリカ合衆国 １５０９０ ペンシルベニア, ウェックスフォード, ディアフィールド サークル ７１３

審査官 齋藤 浩兵

(56)参考文献 特開平０９－２８４２７５（ＪＰ, Ａ）
特開平１０－０６３５９８（ＪＰ, Ａ）
特開平０７－３２７０４２（ＪＰ, Ａ）
特開平１１－１２７２００（ＪＰ, Ａ）
特開平１０－０３２５９４（ＪＰ, Ａ）
特開平０７－２７３７９８（ＪＰ, Ａ）

(58)調査した分野(Int.Cl., ＤＢ名)
H04L 12/18
H04L 12/56