(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6018217号
(P6018217)
(45)発行日　平成28年11月2日(2016.11.2)　(24)登録日　平成28年10月7日(2016.10.7)

(51)Int.Cl.　F I
H04L 12/753 (2013.01)　H04L 12/753

請求項の数 11　(全 18 頁)

(21)出願番号	特願2014-541332(P2014-541332)	(73)特許権者	502303739
(86)(22)出願日	平成24年11月9日(2012.11.9)		オラクル・インターナショナル・コーポレイション
(65)公表番号	特表2014-533071(P2014-533071A)		アメリカ合衆国カリフォルニア州９４０６５レッドウッド・シティー，オラクル・パークウェイ５００
(43)公表日	平成26年12月8日(2014.12.8)	(74)代理人	110001195
(86)国際出願番号	PCT/US2012/064479		特許業務法人深見特許事務所
(87)国際公開番号	WO2013/071130	(72)発明者	ボグダンスキー，バルトシュ
(87)国際公開日	平成25年5月16日(2013.5.16)		ノルウェー、エヌ－０２７５　オスロ、エイチ・０２０３、ホフ・テラス、１５
審査請求日	平成27年9月10日(2015.9.10)		
(31)優先権主張番号	61/557,722	審査官	安藤　一道
(32)優先日	平成23年11月9日(2011.11.9)		
(33)優先権主張国	米国(US)		
(31)優先権主張番号	13/653,303		
(32)優先日	平成24年10月16日(2012.10.16)		
(33)優先権主張国	米国(US)		

最終頁に続く

(54)【発明の名称】ファットツリートポロジーにおけるスイッチ間のデッドロックフリールーティングを提供するためのシステムおよび方法

(57)【特許請求の範囲】

【請求項１】

１つ以上のマイクロプロセッサに対して機能するミドルウェアマシン環境における複数のスイッチ間のルーティングをサポートする方法であって、

ルーティングアルゴリズムを用いて、前記ミドルウェアマシン環境におけるスイッチ間トラフィックのためのルーティングを実行すること、を含み、前記ルーティングアルゴリズムは、前記ミドルウェアマシン環境におけるファットツリートポロジーにおいてソーススイッチから宛先スイッチへのスイッチ間トラフィックをルーティングするアップ／ダウンターンモデルを含み、前記ファットツリーは複数のリーフスイッチとその上方のレベルの複数のスイッチとを含み、各リーフスイッチは終端ノードに接続されて、

前記複数のリーフスイッチから、特定のリーフスイッチを、ハブスイッチとして選択することと、

前記ハブスイッチの兄弟スイッチに関連付けられているルーティングテーブルが、前記宛先スイッチへの経路を含むことを判断することと、

前記ソーススイッチと前記ハブスイッチのそれぞれに、前記兄弟スイッチを通して、前記宛先スイッチへの経路を設定することと、

前記ソーススイッチと宛先スイッチと前記ハブスイッチを含む経路で、前記ハブスイッチと関連付けられたルーティングテーブルを更新することとを含む、方法。

【請求項２】

前記ミドルウェアマシン環境内の複数のスイッチをファットツリートポロジー内に存在

させることをさらに含む、請求項１に記載の方法。
【請求項３】
前記特定のリーフスイッチは前記ファットツリートポロジー内の複数のスイッチ宛先に到達することができる、請求項１または２に記載の方法。
【請求項４】
前記アップ／ダウンターンモデルが機能しないとき、スイッチ間トラフィックのダウン／アップターンを行ない、前記ファットツリートポロジーにおけるデッドロックを防止するために、すべてのダウン／アップターンを、前記ハブスイッチをサブツリールートノードとするデッドロックフリーの単一のサブツリーにローカライズすることをさらに含む、請求項１から３のいずれか１項に記載の方法。
【請求項５】
前記ファットツリートポロジーにおける複数のスイッチ宛先に到達可能な１つ以上のリーフスイッチを検知するために、前記ファットツリートポロジーにおける複数のリーフスイッチを通して繰返すことをさらに含む、請求項１～４のいずれか１項に記載の方法。
【請求項６】
前記ハブスイッチに前記兄弟スイッチがあるか否かを確認するとともに、宛先スイッチへの経路が存在しないときに前記兄弟スイッチに関連付けられたルーティングテーブルが前記経路を含むか否かを確認することをさらに含む、請求項１～５のいずれか１項に記載の方法。
【請求項７】
前記設定することは、
前記兄弟スイッチが存在し前記兄弟スイッチに関連付けられたルーティングテーブルが前記宛先スイッチへの経路を含むとき、前記兄弟スイッチを通して前記宛先スイッチへの経路を前記ソーススイッチおよび前記サブツリールートスイッチに設定することを含む、請求項１～６のいずれか１項に記載の方法。
【請求項８】
前記宛先スイッチに対する出力ポートを前記ハブスイッチに対する出力ポートと同一にすることをさらに含む、請求項１～７のいずれか１項に記載の方法。
【請求項９】
スイッチ間ルーティングにおけるホップ計算のために再帰関数を用いることをさらに含み、前記再帰関数は、前記経路を構成する１つ以上のスイッチに対して繰返される、請求項１～８のいずれか１項に記載の方法。
【請求項１０】
１つ以上のコンピュータシステムによって実行されると前記１つ以上のコンピュータシステムに請求項１～９のいずれか１項に記載の方法を実行させるプログラム命令を含むコンピュータプログラム。
【請求項１１】
ミドルウェアマシン環境におけるスイッチ間トラフィックルーティングをサポートするためのシステムであって、
１つ以上のマイクロプロセッサ上で実行する複数のスイッチを備え、前記複数のスイッチは、
ルーティングアルゴリズムを用いて、前記ミドルウェアマシン環境におけるスイッチ間トラフィックのためのルーティングを実行するステップを実行するように動作し、前記ルーティングアルゴリズムは、前記ミドルウェアマシン環境におけるファットツリートポロジーにおいてソーススイッチから宛先スイッチへのスイッチ間トラフィックをルーティングするアップ／ダウンターンモデルを含み、前記ファットツリーは複数のリーフスイッチとその上方のレベルの複数のスイッチとを含み、各リーフスイッチは終端ノードに接続されて、
さらに、
前記複数のリーフスイッチから、特定のリーフスイッチを、ハブスイッチとして選択す

るステップと、
前記ハブスイッチの兄弟スイッチに関連付けられているルーティングテーブルが、前記宛先スイッチへの経路を含むことを判断するステップ、
前記ソーススイッチと前記ハブスイッチのそれぞれに、前記兄弟スイッチを通して、前記宛先スイッチへの経路を設定するステップと、
前記ソーススイッチと宛先スイッチと前記ハブスイッチを含む経路で、前記ハブスイッチと関連付けられたルーティングテーブルを更新するステップとを実行するように動作する、システム。
【発明の詳細な説明】
【技術分野】
【０００１】
著作権に関する注意
本特許文献の開示の一部には、著作権保護の対象となるものが含まれている。著作権者は、この特許文献または特許開示の何者かによる複製が、特許商標庁の特許ファイルまたは記録にある限り、それに対して異議を唱えないが、そうでなければ、いかなる場合もすべての著作権を留保する。
【０００２】
発明の分野
本発明は、概してコンピュータシステムに関し、具体的にはミドルウェアマシン環境におけるスイッチ間通信のサポートに関する。
【背景技術】
【０００３】
背景
ファットツリートポロジーは、現代の高性能計算（high performance computing）（ＨＰＣ）クラスタ、および、インフィニバンド（Infiniband）（ＩＢ）技術に基づいたクラスタに使用される。ファットツリーにとっては、その他多くのトポロジーと同様、ネットワークリソースを効率的に利用するためにはルーティングアルゴリズムが有益である。しかしながら、既存のルーティングアルゴリズムは、スイッチ間通信においては限界がある。既存のルーティングアルゴリズムの中で、効率的なシステム管理にとって有益な、デッドロックフリーで完全接続のスイッチ間通信をサポートするものはない。これらが、本発明の実施の形態が取組もうとしている一般的な分野である。
【発明の概要】
【課題を解決するための手段】
【０００４】
概要
本明細書に記載のシステムおよび方法は、ミドルウェアマシン環境における複数のスイッチ間のパケットのルーティングをサポートすることができる。ミドルウェアマシン環境は、インターネットプロトコル（ＩＰ）に基づく管理トラフィックを、ミドルウェアマシン環境におけるインフィニバンド上でのＩＰ（IP over Infiniband）（ＩＰｏＩＢ）通信を可能にすることで、サポートすることができる。複数のスイッチは、ルーティングアルゴリズムを用いて、ミドルウェアマシン環境におけるスイッチ間トラフィックのためのルーティングを実行することができる。次に、このルーティングアルゴリズムを用いて、ミドルウェアマシン環境内のスイッチを、宛先に到達できないスイッチ間トラフィックのためのハブスイッチとして選択することができる。さらに、ソーススイッチと宛先スイッチとの間にハブスイッチを介した経路が存在するとき、ハブスイッチと関連付けられたルーティングテーブルを更新することができる。
【０００５】
本発明のある代表的な実施の形態において、ミドルウェアマシン環境におけるスイッチ間トラフィックルーティングをサポートするためのシステムが提供される。このシステムは、複数のスイッチを含み得る。これら複数のスイッチのうちの少なくとも１つは、ルー

ティング部と、選択部と、更新部とを含み得る。ルーティング部は、第１のルーティングアルゴリズムを用いて、ミドルウェアマシン環境におけるスイッチ間トラフィックのためのルーティングを実行し得る。選択部は、第１のルーティングアルゴリズムを用いて、ミドルウェアマシン環境内のスイッチを、宛先に到達できないスイッチ間トラフィックのためのハブスイッチとして選択し得る。更新部は、ソーススイッチと宛先スイッチとの間にハブスイッチを介した経路が存在するとき、ハブスイッチと関連付けられたルーティングテーブルを更新する機能を果たし得る。

【０００６】

ある実施の形態において、複数のスイッチはファットツリートポロジーを形成し得る。選択部によって、ファットツリートポロジー内のリーフスイッチを、ハブスイッチとして選択してもよく、リーフスイッチは、ファットツリートポロジー内の複数のスイッチ宛先に到達することができる。ファットツリートポロジーにおいて、第１のルーティングアルゴリズムは、アップ／ダウンターンモデルを用いてスイッチ間トラフィックをルーティングしてもよい。アップ／ダウンターンモデルが機能しないとき、スイッチ間トラフィックはダウン／アップターンを行ない、ファットツリートポロジーにおけるデッドロックを防止するために、すべてのダウン／アップターンは、ハブスイッチをサブツリールートノードとするデッドロックフリーの単一のサブツリーにローカライズされる。

【０００７】

別の実施の形態において、上記複数のスイッチのうちの少なくとも１つのスイッチは繰返し部をさらに含み得る。繰返し部は、ファットツリートポロジーにおける複数のスイッチ宛先に到達可能な１つ以上のリーフスイッチを検知するために、ファットツリートポロジーにおける複数のリーフスイッチを通して繰返す。一例では、上記複数のスイッチのうちの少なくとも１つのスイッチは確認部をさらに含み得る。確認部は、ハブスイッチに兄弟スイッチがあるか否かを確認するとともに、宛先スイッチへの経路が存在しないときに上記兄弟スイッチに関連付けられたルーティングテーブルがこの経路を含むか否かを確認してもよい。一例では、上記複数のスイッチのうちの少なくとも１つのスイッチは経路設定部をさらに含み得る。経路設定部は、上記兄弟スイッチが存在しこの兄弟スイッチに関連付けられたルーティングテーブルが宛先スイッチへの経路を含むとき、上記兄弟スイッチを通して宛先スイッチへの経路をソーススイッチおよびサブツリールートスイッチに設定する。別の例では、上記複数のスイッチのうちの少なくとも１つのスイッチは、宛先スイッチに対する出力ポートがハブスイッチに対する出力ポートと同一になるように設定するためのポート設定部をさらに含み得る。

【図面の簡単な説明】

【０００８】

【図１】ミドルウェアマシン環境においてデッドロックが生じるときの代表的なシナリオを説明する図である。

【図２】本発明のある実施の形態に従い、シングルコアのファットツリートポロジーにおいてスイッチ間のデッドロックフリールーティングを提供することを説明する図である。

【図３】本発明のある実施の形態に従い、マルチコアのファットツリートポロジーにおいてスイッチ間のデッドロックフリールーティングを提供することを説明する図である。

【図４】本発明のある実施の形態に従い、ファットツリートポロジーにおいてスイッチ間のデッドロックフリールーティングを提供するための代表的なフローチャートを示す。

【図５】本発明に従うスイッチの代表的な実施の形態を示す。

【発明を実施するための形態】

【０００９】

詳細な説明

本明細書に記載のシステムおよび方法は、ミドルウェアマシン環境における複数のスイッチ間のパケットのルーティングをサポートすることができる。ミドルウェアマシン環境は、インターネットプロトコル（ＩＰ）に基づく管理トラフィックを、ミドルウェアマシン環境におけるインフィニバンド上でのＩＰ（ＩＰｏＩＢ）通信を可能にすることで、サ

ポートすることができる。
【００１０】
インフィニバンド（ＩＢ）ネットワークおよびサブネット管理
本発明のある実施の形態に従うと、ＩＢネットワークはサブネットと呼ぶことができる。１つのサブネットは、スイッチおよびポイント間リンクを用いて相互接続された一組のホストからなる。ＩＢファブリックは１つ以上のサブネットを構成することができ、各サブネットはルータを用いて相互接続することができる。１つのサブネット内のホストおよびスイッチは、ローカル識別子（local identifier）（ＬＩＤ）を用いてアドレス指定され、単一のサブネットは４９１５１個のＬＩＤに限定される。
【００１１】
１つのＩＢサブネットは少なくとも１つのサブネットマネージャ（subnet manager）（ＳＭ）を有することができる。ＳＭは、サブネット内の、スイッチ、ルータ、およびホストチャネルアダプタ（host channel adapter）（ＨＣＡ）上にあるすべてのＩＢポートの構成を含むネットワークを、初期化し起動する役割を果たす。初期化時、ＳＭは検出状態で始動する。検出状態において、ＳＭは、すべてのスイッチおよびホストを検出するためにネットワークのスイープを行なう。検出状態中に、ＳＭは、他のＳＭを検出することができ、どのＳＭがマスタＳＭになるべきかについて交渉することができる。検出状態が終了すると、ＳＭはマスタ状態になる。マスタ状態において、ＳＭは、ＬＩＤの割当、スイッチの構成、ルーティングテーブルの計算および導入、ならびにポートの構成を進める。マスタ状態が終了した時点で、サブネットは立ち上げられて使用できる状態となり、ＳＭは、サブネットが構成された後の変化についてネットワークをモニタリングする役割を果たす。
【００１２】
加えて、ＳＭは、完全接続性、デッドロックフリー状態、およびソースと宛先のすべての対の間での適切な負荷分散を維持するルーティングテーブルを計算する役割を果たすことができる。ルーティングテーブルはネットワークの初期化時に計算することができ、このプロセスを、トポロジーが変化する度に、ルーティングテーブルを更新し最適性能を保証するために繰返すことができる。
【００１３】
通常動作中、ＳＭは、リンクが機能停止したとき、デバイスが追加されたとき、またはリンクが削除されたとき等の、トポロジー変化事象について確認するために、定期的にネットワークを軽くスイープする。ＳＭが、軽くスイープを行なっている間に変化事象を検出した場合またはネットワークの変化を知らせるメッセージ（トラップ）を受けた場合、ＳＭは、検出された変化に従ってネットワークを再構成することができる。この再構成も、初期化中に用いられるステップを含むことができる。
【００１４】
ＩＢは、フロー制御をバーチャルレーン（virtual lane）（ＶＬ）毎に行なうことができる、ロスのないネットワーキング技術である。ＶＬは、別々のバッファリング、フロー制御、および輻輳管理リソースを備えた、同一の物理リンク上の論理チャネルである。ＶＬの概念によって、物理トポロジーの上にバーチャルネットワークを構築することが可能になる。これらバーチャルネットワークまたは層を、効率的なルーティング、デッドロックの回避、フォールトトレランス、およびサービス差別化等のさまざまな目的に使用することができる。
【００１５】
ファットツリールーティング
本発明のある実施の形態に従うと、ファットツリートポロジーは階層ネットワークトポロジーであり、たとえば、平衡のとれたファットツリーでは、すべての層のリンク容量を等しくすることができる。さらに、ファットツリートポロジーは、複数のルートを有する１つのツリーを構築することによって実現でき、たとえば、ＸＧＦＴ表記を用いて記載できる、ｍポートｎツリー定義またはｋ分木ｎツリー定義によって実現できる。

【００１６】
ファットツリールーティングアルゴリズムは、利用可能なネットワークリソースを活用することができる。ファットツリールーティングアルゴリズムは、２つの段階、すなわちパケットをソースから送る上昇段階と、パケットを宛先に向けて送るときの下降段階とを含むことができる。これら２つの段階の間の移行期は、最下部にある共通の祖先で発生する。この祖先は、ソースおよび宛先双方に、その下向きのポートを通して到達できるスイッチである。このようなルーティングの実装はデッドロックフリーを保証し、この実装はまた、同じ宛先に向かうすべての経路が同一のルート（トップ）ノードで収束することで、この宛先に向かうすべてのパケットが下向きの１つの専用経路を辿ることを保証する。すべての宛先に対して専用下降経路を設けることによって、下降段階における競合を効果的になくす（上昇段に移す）ことができ、そのため、宛先が異なるパケットは、その経路上のスイッチのうちの２分の１における出力ポートについてしか競合しない。加えて、オーバーサブスクリプション状態のファットツリーにおける下降経路は、専用経路ではなく、数個の宛先が共有できるものである。

【００１７】
スイッチ間通信
ファットツリートポロジーでは、終端ノード間の通信に加えて、スイッチ間通信が可能である。スイッチ間通信から生じるトラフィックがほとんどないとき、スイッチ間通信は無視してもよい。言い換えると、ＩＢスイッチは、ルーティングアルゴリズムの観点から安全に無視できる透明なデバイスとみなし得る。

【００１８】
一方、スイッチ間通信トラフィックは常に無視できる訳ではない。こういった場合、先のセクションで説明したファットツリールーティングを用いて、制限のあるスイッチ間通信のルーティングを行なうことができる。なぜなら、いずれか２つのスイッチの間の経路を検知して利用できる最初のポートからトラフィックを送るためには、最下部にある共通の先祖を検知する必要があるからである。さらに、ファットツリールーティング方式は、最下部にある共通の先祖を持たないスイッチの間は接続しない、すなわち、先のセクションで説明したファットツリールーティング方式は、ファットツリートポロジーにおいて完全接続を構築しないことがある。

【００１９】
終端ノードおよびスイッチ双方を含むファットツリートポロジーにおける完全接続は、さまざまな理由から有益である可能性がある。たとえば、ＩＢ診断ツールは、ＬＩＤルーティングに依拠しており、基本的なファブリック管理およびモニタリングのために完全接続を必要とする。さらに、ベンチマーキングに使用されるｐｅｒｆｔｅｓｔ等の進化したツールは、基礎をなすＬＩＤルーティングに依拠し完全接続を必要とするＩＰｏＩＢを採用することがある。ＩＰｏＩＢは、ＩＢネットワーク上でＴＣＰ／ＩＰトラフィックを実行できるようにするカプセル化方法である。加えて、ＩＰｏＩＢは、非インフィニバンドを意識した管理およびモニタリングプロトコルならびにＳＮＭＰまたはＳＳＨのようなアプリケーションが必要とする場合がある。このように、ファブリック内のすべてのスイッチ間の完全接続は有益であり得る。なぜなら、スイッチが、任意のトラフィックを生成することができ、他の終端ノードのようにアクセスされることができ、埋込まれたＳＭを含むことが多いからである。

【００２０】
ファットツリートポロジーにおいて、デッドロックは、バッファまたはチャネル等のネットワークリソースが共有されていることが原因で生じ得る。

【００２１】
図１は、ミドルウェアマシン環境においてデッドロックが生じるときの代表的なシナリオを説明する図である。図１に示されるように、ミドルウェアマシン環境におけるファットツリートポロジー１００では、スイッチ間通信の対が２つあり（スイッチ０～スイッチ３およびスイッチ３～スイッチ６）ノード間通信の対が２つある（ノードＢ～ノードＤお

よびノードＤ～ノードＡ）、混合通信パターンが可能である。ここで、スイッチ０からスイッチ３へのトラフィックはスイッチ４、８、および５を介して導かれ、スイッチ３からスイッチ６へのトラフィックはスイッチ７および９を介して導かれる。また、ノードＢからノードＤへのトラフィックはスイッチ８、５、３、７、および９を介して導かれ、ノードＤからノードＡへのトラフィックはスイッチ９、６、０、４、および８を介して導かれる。

【００２２】

図１の点線で示されるように、上記混合通信パターンがファットツリートポロジー１００内に存在するとき、循環クレジット従属性またはクレジットループ１０１が作成される。さらに、デッドロックは、循環クレジット従属性が作成されたときに発生し得る。これは、循環クレジット従属性が存在するときには常にデッドロックがあるということを意味しているのではない。言い換えると、循環クレジット従属性を作成することで、デッドロックが発生する可能性が生じる。たとえば、ｍｉｎｈｏｐ等のルーティングアルゴリズムを使用するとき、デッドロックは、ノード間およびスイッチ間トラフィックがある３段のファットツリーにおいて生じ得る。その理由として、ｍｉｎｈｏｐが５ノード以上のリングをデッドロックフリーで解決することができないことが挙げられ、３段のファットツリーは６ノードのリングを含み得る。このように、すべてのスイッチの間でスイッチ間通信が行なわれるときにファットツリートポロジーにおいてデッドロックフリールーティングを提供できるアルゴリズムが必要である。

【００２３】

デッドロックは、利用可能な物理リソースをセグメントに分けるＶＬを用いることによって回避できる。しかしながら、ファットツリーにおいてデッドロックを回避するためにＶＬを使用することは、効率的でない可能性がある。なぜなら、サービス品質（Quality of Service）（ＱｏＳ）のようなその他の目的のために追加のＶＬを使用することがあるからである。

【００２４】

たとえば、ＩＢのためのＬＡＳＨアルゴリズムは、完全接続とデッドロックフリーをサポートできる。ＬＡＳＨアルゴリズムは、ＶＬを用いることによってクレジットループ１０１を切断し、ファブリック内のすべてのノード間の完全接続を提供する。ＬＡＳＨは、経路を割当てるときにファットツリーの特性を活用しないので、通常のファットツリールーティングよりもネットワーク性能が悪く、加えて、ＬＡＳＨはルーティング時間が長い。ＤＦＳＳＳＰは、ファットツリーにおける全対全のスイッチ間通信をサポートするのに使用し得るもう１つのルーティングプロトコルである。しかしながら、ＤＦＳＳＳＰは非ＨＣＡノード間のクレジットループを切断しない。このことは、スイッチ間通信がデッドロックフリーではない場合があることを意味する。

【００２５】

このように、相互接続ネットワーク内でデッドロックが生じると、このデッドロックは、ネットワークの一部またはすべてにおける通信を妨げる可能性がある。したがって、相互接続ネットワークのシステム全体の観点から、ファットツリートポロジーにおいてサポートされる完全接続は、デッドロックフリーであることが好ましい。

【００２６】

シングルコアのファットツリートポロジーにおけるデッドロックフリールーティング

本発明のある実施の形態に従い、ミドルウェアマシン環境におけるノード間の完全接続をサポートすることができ、１つのＶＬを用いるだけでファブリック全体をデッドロックフリーにすることができる。

【００２７】

図２は、本発明のある実施の形態に従い、シングルコアのファットツリートポロジーにおいてスイッチ間のデッドロックフリールーティングを提供することを説明する図である。図２に示されるように、ミドルウェアマシン環境におけるシングルコアのファットツリートポロジー２００は、複数のスイッチを含むことができる。さらに、ミドルウェアマシ

ン環境は、ファットツリートポロジー２００内のリーフスイッチ２０２に接続するサーバノード等の複数の終端ノード２２０を含むことができる。

【００２８】

アップ／ダウンターンモデルを用いることにより、ミドルウェアマシン環境におけるファットツリールーティングを提供することができる。たとえば、スイッチＡ１およびＡ２はＸ１またはＸ２を経由してアップすることによって接続性を得ることができる。一方、アップ／ダウンターンモデルは、最下部の共通の先祖がないスイッチたとえばＡ１とＢ１との間の通信には有用でないことがある。

【００２９】

本発明のある実施の形態に従い、代替の手法を採用することによって、アップ／ダウンターンモデルを用いて通信を確立することができないスイッチ間の接続性を得ることができる。図２に示されるように、上下逆のツリー、すなわちサブツリー２１０（影を付けた領域に示される）を、ファットツリートポロジー２００内に形成することができる。サブツリー２１０は、ファットツリートポロジー２００内のリーフスイッチであるサブツリールートノードＳｎすなわちハブスイッチ２０１を有することができる。完全接続は、ファットツリートポロジー２００内のハブスイッチ２０１を通して確立することができる。

【００３０】

サブツリー２１０内において、スイッチＸ１とＹ１はサブツリールートＳｎ２０１を経由して通信することができ、Ｘ１とＸ２はスイッチＡ１を経由して通信できる。さらに、サブツリー２１０の外にあるスイッチは、ファットツリートポロジー２００内の他のスイッチと、サブツリー２１０を通して通信することができる。これは、最短でない（または最適に及ばない）経路の使用につながる。ファブリック内の、アップ／ダウンターンモデルを用いて別のスイッチに到達することが不可能なスイッチは、先ずパケットをサブツリー２１０に送ればよい。次に、これらのパケットを、宛先スイッチへの経路を有するスイッチに到達するまで、サブツリー２１０内で送ればよい。言い換えると、これらのパケットは、ハブスイッチ２０１でＵターンする、すなわち、下降し折返して上昇し続いて上昇してから宛先に向かって下降する。ミドルウェアマシン環境におけるサブツリー２１０は、ファブリック内のすべてのダウン／アップターンを、１つのデッドロックフリーツリーにローカライズすることができる。

【００３１】

たとえば、ミドルウェアマシン環境における第１のスイッチたとえばＡ１から第２のスイッチたとえばＢ２まで１つ以上のパケットをルーティングすることを要求されたとき、システムは先ず、サブツリールートスイッチすなわちハブスイッチ２０１に関連付けられたルーティングテーブルが第２のスイッチＢ２への経路を含むか否か判断することができる。図２において点線で示されるように、ハブスイッチ２０１は、Ｂ１、Ｙ１またはＹ２を通る経路を介して第２のスイッチＢ２に接続できる。次に、第２のスイッチＢ２への経路を、第１のスイッチＡ１に関連付けられたルーティングテーブルに挿入することができる。加えて、Ａ１に関連付けられたルーティングテーブルにおいて、スイッチＡ２に対する出力ポートをハブスイッチ２０１に対する出力ポートと同一になるように設定することができる。

【００３２】

上下逆のサブツリー２１０のルートは、リーフスイッチのうちのいずれでもよい。さらに、ミドルウェアマシン環境におけるデッドロックフリーの完全接続を得るために、ファットツリートポロジー２００内のすべてのスイッチがともに１つのリーフスイッチを通信用に選択することができる。

【００３３】

本発明のある実施の形態に従い、すべてのＵターンがサブツリー２１０内でのみ行なわれ上昇（アップ）から下降（ダウン）へのいずれかのターンによってトラフィックがサブツリー２１０から出るようにすることによって、ファットツリートポロジー２００内をデッドロックフリーにすることができる。また、サブツリー２１０の外部ではＵターンがで

きないので、サブツリー２１０から出たトラフィックは循環してサブツリー２１０の中に戻ることができないかもしれない。

【００３４】

このように、シングルコアのファットツリー２００内では、ソーススイッチをハブスイッチ２０１に接続しハブスイッチ２０１をすべての宛先スイッチに接続する経路が常に存在し得る。さらに、このトポロジー内のリーフスイッチを通して接続性を得るための通信方式は、ファットツリールーティングを用いて互いに到達することができるスイッチ間の上方向から下方向への経路を変更しない。

【００３５】

添付の付録Ａは、ミドルウェアマシン環境におけるスイッチ間のルーティングのサポートおよび本開示を通して記載されているプラットフォームのさまざまな側面に関するさらに他の情報を提供する。付録Ａにおける情報は、例示を目的として提供されており、本発明の実施の形態すべてを限定するものであると解釈されてはならない。

【００３６】

マルチコアのファットツリートポロジーにおけるデッドロックフリールーティング

本発明のある実施の形態に従うと、複雑なマルチコアのファットツリーまたは不規則なファットツリーでは、２つのスイッチ間に常に接続性があるとは限らないであろう。こういった場合、ベストエフォートの手法を用いることによってスイッチ間通信をルーティングすることができる。

【００３７】

図３は、本発明のある実施の形態に従い、マルチコアのファットツリートポロジーにおいてスイッチ間のデッドロックフリールーティングを提供することを説明する図である。図３に示されるように、ミドルウェアマシン環境におけるマルチコアのファットツリートポロジー３００は、複数のスイッチを含むことができる。さらに、このミドルウェアマシン環境は、ファットツリートポロジー３００内のリーフスイッチに接続するサーバノード等の複数の終端ノード３２０を含むことができる。

【００３８】

同様に、上下逆のツリーすなわちサブツリー３１０（影を付けた領域に示される）をファットツリートポロジー３００内に形成することができる。サブツリー３１０は、ファットツリートポロジー３００内のリーフスイッチであるサブツリールートノードＳｎすなわちハブスイッチ３０１を有することができる。図２とは異なり、このサブツリールートスイッチすなわちハブスイッチ３０１はすべての宛先への経路を有さないかもしれない。

【００３９】

加えて、このミドルウェアマシン環境には兄弟スイッチ３０２が存在し得る。この兄弟スイッチは、水平リンク３３０を通してサブツリールートスイッチＳｎすなわちハブスイッチ３０１に接続できる。

【００４０】

一例において、第１のスイッチたとえばＢ２は、ミドルウェアマシン環境において１つ以上のパケットを第２のスイッチたとえばＢ３にルーティングすることを要求できる。サブツリールートスイッチ３０１には、Ｂ３に接続する経路がない。なぜなら、Ｂ３は、他のノードと水平リンクのみを介して接続する別のファットツリーの中にあるからである。

【００４１】

システムは先ず、サブツリールートスイッチ３０１が、そのルーティングテーブルがＢ３への経路を含んでいる兄弟スイッチ３０２を有するか否か確認することができる。システムは次に、上記兄弟スイッチ３０２を通して、第２のスイッチＢ３への経路を、第１のスイッチＢ２およびサブツリールートスイッチ３０１に設定することができる。結果として、このシステムを、スイッチＢ２からのパケットをスイッチＢ３までたとえばＹ１またはＹ２、Ｂ１、サブツリールートノード３０１、およびその兄弟ノード３０２を介してルーティングする（点線で示される）ように構成できる。

【００４２】

ＳＦＴＲＥＥアルゴリズム

本発明のある実施の形態に従い、ルーティングアルゴリズム、すなわちＳＦＴＲＥＥアルゴリズムを用いて、スイッチ間のデッドロックフリーの完全接続を得ることができる。ＳＦＴＲＥＥアルゴリズムは、２つのステップ、すなわち、リーフスイッチのうちの１つであってもよいサブツリールートＳｎを検知する第１のステップと、スイッチ間ルーティングを実行する第２のステップとを含むことができる。

【００４３】

以下のアルゴリズム１は、サブツリールートを検知するための疑似コードを含み、サブツリーを確立する。

【００４４】

【表１】

アルゴリズム1　サブツリールート関数を選択

Require: ルーティングテーブルが生成されている。
Ensure: サブツリールート（sw_{root}）が選択される。

```
1: found = false
2: for sw_leaf = 0 to max_leaf_sw do
3:   if found = = true then
4:     break
5:   end if
6:   sw_root = sw_leaf
7:   found = true
8:   for dst = 1 to max_dst_addr do
9:     if sw_root.routing_table[dst] = = no_path then
10:      found = false
11:      break
12:    end if
13:  end for
14: end for
15: if found = false then
16:  sw_root = get_leaf(0)
17: end if
```

【００４５】

アルゴリズム１は、トポロジー内のすべてのリーフスイッチをトラバースし、各リーフスイッチについて、そのルーティングテーブルにおいてスイッチ宛先アドレスのうちのいずれかが到達不能と記されているか否か確認する。ルーティングテーブル内のすべてのスイッチ宛先が到達可能と記されている場合は、最初に遭遇したリーフスイッチをサブツリールートスイッチとして選択する。さらに、ファットツリーにおける潜在的なサブツリーの数は、完全接続のリーフスイッチの数と同一である可能性がある。これらリーフスイッチのうちの１つのみをサブツリールートとして選択すればよく、この特定のリーフスイッチにおいてルートがあるサブツリーは１つのみであってもよい。

【００４６】

以下のアルゴリズム２は、ファットツリートポロジーにおけるスイッチ間ルーティングを実行するための疑似コードを含む。

【００４７】

【表２】

アルゴリズム2 スイッチ間ルーティング関数

```
Require: サブツリールート(sw_root)
Ensure: 最悪の場合でもsw_rootによって各sw_srcが各sw_dstに到達する。
 1: if found or sw_root.routing_table[dst] ≠ no_path then
 2:   port = sw_src.routing_table[sw_root.addr]
 3:   sw_src.routing_table[dst] = port
 4:   get_path_length (sw_src, null, dst, sw_root , hops)
 5:   set_hops (sw_src, hops)
 6:   return true
 7: else if (sw_sib = get_sibling_sw(sw_root)) ≠ null then
 8:   if sw_src.routing_table[sw_sib.addr] ≠ no_path then
 9:     if sw_sib.routing_table[dst] ≠ no_path then
10:       port = get_port_to_sibling (sw_sib)
11:       sw_root.routing_table[dst] = port
12:       hops = get_hops (sw_sib,dst)
13:       set_hops(sw_root, hops + 1)
14:       hops = 0
15:       port = sw_src.routing_table[sw_sib.addr]
16:       sw_src.routing_table[dst] = port
17:       get_path_length(sw_src, null, dst, sw_root, hops)
18:       set_hops(sw_src, hops)
19:       return true
20:     end if
21:   end if
22: end if
23: print 'SW2SW failed for sw_src and sw_dst.'
24: return false
```

【００４８】

スイッチ間ルーティング関数がコールされたとき、第１のステップは、アルゴリズム２のライン１に示されているように、サブツリールート（ｓｗ$_{root}$）のルーティングテーブルが宛先スイッチ（ｓｗ$_{dst}$）への経路を含むか否か判断することである。含む場合、宛先スイッチへの経路をソーススイッチ（ｓｗ$_{src}$）のルーティングテーブルに挿入し、宛先スイッチに対する出力ポートは、サブツリールートへの経路に対する出力ポートと同一である（ライン２～６）。すべてのスイッチに対してスイッチ間ルーティング関数がコールされるので、最終的に、到達不能な各宛先スイッチに対するすべての経路はサブツリーに収束する。言い換えると、選択されたサブツリールートは、到達不能な宛先スイッチすべてにとって新たなターゲットである。ダウン／アップターンは、ソーススイッチおよび宛先スイッチ双方への上向きの経路を有するスイッチであってもよい、サブツリー内に位置する可能な第１のスイッチで、生じるであろう。

【００４９】

アルゴリズム２の第２の部分（ライン７～２４）は、ベストエフォートの手法が用いられサブツリールートがすべての宛先への経路を有していない複雑なマルチコアまたは不規則なファットツリーの場合に対応する。一方、このアルゴリズム２の第２の部分は、シングルコアのファットツリーではスキップしてもよい。なぜなら、ソーススイッチからサブツリールートへの経路は常に存在し得るものでありサブツリールートはすべての宛先スイッチへの経路を有し得るからである。

【００５０】

アルゴリズム２は、サブツリールートに、このサブツリールートが図３に示されるように水平リンクを通して接続される真隣の「隣人」があるか否か確認する（ライン７）。次

に、アルゴリズム２は、兄弟とも呼ばれるこの隣人が、宛先スイッチへの経路を有するか否か確認する（ライン８）。兄弟が存在しこの兄弟が宛先スイッチへの経路を有していれば、兄弟スイッチを通して、宛先スイッチへの経路を、サブツリールートおよび起点となるソーススイッチ双方に設定する（ライン９～１９）。言い換えると、到達不能な宛先スイッチにとってのターゲットは依然としてサブツリールートであってもよく、サブツリールートはパケットをその兄弟スイッチを介して宛先に送ることができる。

【００５１】

上記２つのステップがいずれも真値とともに関数からリターンしない場合、ＳＦＴＲＥＥアルゴリズムは２つのスイッチ間の経路を検知していないかもしれない。たとえば、この事態が生じ得るのは、リンク障害が複数あるためまたはノード間の接続が非常に不規則であるためにファットツリールーティングを実行することが推奨されないトポロジーの場合である。このようなトポロジーの一例は、大きさが異なる２つのファットツリーが非対称に相互接続されているトポロジーである可能性があり、たとえば大きい方のツリー上の２、３の中間段スイッチが小さい方のツリー上のルートに接続されている。ＯｐｅｎＳＭにおけるさまざまなコマンドラインパラメータを用いて、ファットツリールーティングをこういったトポロジー上で強制的に実行することができる。ＳＦＴＲＥＥアルゴリズムは、性能だけでなく接続に関しても最適に及ばないルーティングを与えるかもしれない。

【００５２】

本発明のある実施の形態に従い、ＯｐｅｎＳＭは、宛先へのすべての経路にホップカウントを記してもよい。ファットツリールーティングアルゴリズムは、ツリー内のスイッチランキングを用いてホップの数を計算することができる。また、カウントは、主ルーティング関数の単純なカウンタを用いて行なうことができる。これらのカウント方式は、スイッチ間ルーティングが、ファットツリーアルゴリズムによって実施される主ルーティングから完全に独立しているときは、ジグザグ経路（すなわち経路が最短ホップ経路に従っていない）が存在し得るので、信頼できないことがある。

【００５３】

スイッチ間ルーティングの間、ホップ計算、たとえばget_path_length（ライン１７）に対して再帰関数を用いることができる。この関数は、経路を構成する一連のスイッチに対して繰返して用いられることができ、この関数は、宛先への適切なホップカウントを有するノードに達したとき、ホップカウントを、経路上のスイッチのルーティングテーブルに書込むことができる。

【００５４】

加えて、サブツリールートを、サブネットマネージャノード（エンドノードであってその上でサブネットマネージャが実行している）がサブツリールートと記されたスイッチに接続されないように、選択することができる。このようにスイッチ間トラフィックをサブネットマネージャから切離すことによって、重要な場所でボトルネックが生じないようにすることができる。

【００５５】

図４は、本発明のある実施の形態に従い、ファットツリートポロジーにおいてスイッチ間のデッドロックフリールーティングを提供するための代表的なフローチャートを示す。図４に示されるように、ステップ４０１において、ミドルウェアマシン環境は、ルーティングアルゴリズムを用いて、ミドルウェアマシン環境におけるスイッチ間トラフィックのためのルーティングを実行することができる。次に、ステップ４０２において、ミドルウェアマシン環境は、ルーティングアルゴリズムを用いて、ミドルウェアマシン環境内のあるスイッチを、宛先に到達できないスイッチ間トラフィックのためのハブスイッチとして選択することができる。さらに、ステップ４０３で、ミドルウェアマシン環境は、ソーススイッチと宛先スイッチとの間にハブスイッチを介した経路が存在するとき、ハブスイッチと関連付けられたルーティングテーブルを更新することができる。

【００５６】

図５は本発明に従うスイッチの代表的な実施の形態を示す。このスイッチは、ミドルウ

ェアマシン環境におけるスイッチ間トラフィックルーティングをサポートし複数のスイッチを含む上記システムのうちのいずれかに含まれるものであってもよい。図５を参照して、代表的なスイッチ１０は、上記のように本発明の原理に従って構成された、ルーティング部１１と、選択部１３と、更新部１４とを含み得る。部１１、１３、および１４を含むがこれらに限定されないスイッチ１０の機能部は、本発明の原理を実施するハードウェア、ソフトウェア、またはハードウェアとソフトウェアの組合せによって実現し得る。図５に示される各機能ブロックを組合わせてまたはサブユニットに分割して上記本発明の原理を実現し得ることが、当業者に理解される。したがって、本明細書における説明は、本明細書に記載の機能ブロックの可能な組合せまたは分割またはさらに他の定義をサポートし得る。

【００５７】

ルーティング部１１は、先に開示されているように、第１のルーティングアルゴリズムを用いて、ミドルウェアマシン環境におけるスイッチ間トラフィックのルーティングを実行してもよい。選択部１３は、第１のルーティングアルゴリズムを用いて、ミドルウェアマシン環境内のスイッチを、宛先に到達できないスイッチ間トラフィックのためのハブスイッチとして選択してもよい。ソーススイッチと宛先スイッチとの間にハブスイッチを介した経路が存在するとき、更新部１４はこのハブスイッチと関連付けられたルーティングテーブルを更新してもよい。

【００５８】

ある代表的な実施の形態において、システムの複数のスイッチはファットツリートポロジーを形成する。選択部１３がファットツリートポロジー内のリーフスイッチをハブスイッチとして選択してもよく、このリーフスイッチはファットツリートポロジー内の複数のスイッチ宛先に到達できる。第１のルーティングアルゴリズムは、アップ／ダウンターンモデルを用いることによってファットツリートポロジー内のスイッチ間トラフィックをルーティングしてもよい。スイッチ間トラフィックは、アップ／ダウンターンモデルが機能しないときは、ダウン／アップターンを行ない、すべてのダウン／アップターンは、ファットツリートポロジーにおけるデッドロックを防止するために、ハブスイッチをサブツリールートノードとする単一のデッドロックフリーサブツリーに、ローカライズされる。

【００５９】

ある代表的な実施の形態において、スイッチ１０は任意で、ファットツリートポロジーにおける複数のスイッチ宛先に到達できる１つ以上のリーフスイッチを検知するために、ファットツリートポロジー内の複数のリーフスイッチを通して繰返すための繰返し部１２を含んでもよい。したがって、選択部１３は、この１つ以上のリーフスイッチからハブスイッチを選択してもよい。

【００６０】

ある代表的な実施の形態において、スイッチ１０は確認部１５をさらに含んでもよい。確認部１５は、ハブスイッチが兄弟スイッチを有しているか否かを確認するとともに、宛先スイッチへの経路が存在しないとき、兄弟スイッチと関連付けられたルーティングテーブルがこの宛先スイッチへの経路を含むか否かを確認してもよい。

【００６１】

ある代表的な実施の形態において、スイッチ１０は経路設定部１６をさらに含んでもよい。兄弟スイッチが存在しこの兄弟スイッチと関連付けられたルーティングテーブルが宛先スイッチへの経路を含むとき、経路設定部１６は、兄弟スイッチを通して、宛先スイッチへの経路を、ソーススイッチおよびサブツリールートスイッチに設定してもよい。

【００６２】

ある代表的な実施の形態において、スイッチ１０は、宛先スイッチに対する出力ポートがハブスイッチに対する出力ポートと同一になるように設定するためのポート設定部１７をさらに含んでもよい。

【００６３】

本発明は、１つ以上のプロセッサ、メモリ、および／または本開示の教示に従いプログ

ラムされたコンピュータ読取可能な記憶媒体を含む、従来の汎用または専用デジタルコンピュータ、コンピューティングデバイス、マシン、またはマイクロプロセッサを１つ以上用いて、適宜実現し得る。適切なソフトウェアコーディングは、熟練したプログラマが本開示の教示に基づいて容易に準備できる。これはソフトウェア技術の当業者には明らかであろう。

【００６４】

実施の形態によっては、本発明は、本発明のプロセスのうちいずれかを実行するためにコンピュータをプログラムするのに使用できる命令が格納された記憶媒体またはコンピュータ読取可能な媒体（複数の媒体）であるコンピュータプログラムプロダクトを含む。この記憶媒体は、フロッピーディスク（登録商標）、光ディスク、ＤＶＤ、ＣＤ－ＲＯＭ、マイクロドライブ、および光磁気ディスクを含む、任意の種類のディスク、ＲＯＭ、ＲＡＭ、ＥＰＲＯＭ、ＥＥＰＲＯＭ、ＤＲＡＭ、ＶＲＡＭ、フラッシュメモリデバイス、磁気もしくは光カード、ナノシステム（分子メモリＩＣを含む）、または、命令および／またはデータを格納するのに適した任意の種類の媒体もしくはデバイスを含み得る。

【００６５】

本発明に関するこれまでの記載は例示および説明を目的として提供されている。すべてを網羅するまたは本発明を開示された形態そのものに限定することは意図されていない。当業者には数多くの変更および変形が明らかであろう。変形は、開示された特徴および／または要素の任意の組合せを含み得る。実施の形態は、本発明の原理およびその実際の応用を最もうまく説明することによって当業者が本発明のさまざまな実施の形態および意図している特定の用途に適したさまざまな変形を理解できるようにするために、選択され説明されている。本発明の範囲は、以下の特許請求の範囲およびその均等物によって定められることが意図されている。

【００６６】

付録Ａ

以下は、ＳＦＴＲＥＥアルゴリズムがデッドロックフリーであることを示す証拠である。この証拠には関連用語の代表的な定義も含まれる。

【００６７】

定義１．スイッチS_nは、ランクｎのスイッチを意味する。ルートスイッチのランクはｎ＝０である。

【００６８】

定義２．下方向のチャネルは、トラフィックがs_{n-1}からs_nに流れるs_{n-1}とs_nの間のリンクを意味する。上方向のチャネルは、トラフィックがs_nからs_{n-1}に流れるs_nとs_{n-1}の間のリンクを意味する。

【００６９】

定義３．経路は、チャネルによって接続された一連のスイッチであると定義される。経路はソーススイッチで始まり宛先スイッチで終わる。

【００７０】

定義４．Ｕターンは、下方向のチャネルの次に上方向のチャネルが続くターンである。

定義５．チャネルc_iとチャネルc_jとの間のチャネル従属性は、パケットを保持しているチャネルc_iがチャネルc_jの使用を要求するときに生じる。

【００７１】

定義６．チャネル従属性グラフＧ＝（Ｖ，Ｅ）は、頂点ＶがネットワークＮのチャネルでありエッジＥがチャネル対（c_i，c_j）でありc_iからc_jへの（チャネル）従属性が存在する有向グラフである。

【００７２】

定義７．サブツリーは、サブツリーの単一のルートである１つのリーフスイッチｓに収束するファットツリー内の上下逆の論理ツリー構造を意味する。サブツリーは、そのルートから拡張しそのリーフはすべてファットツリー内のトップレベルスイッチである。上方向から下方向へのターンによってサブツリー構造の外に出る。

【００７３】
以下の定理はルーティング関数のデッドロックフリーにとって十分な条件を示す。
定理１．ネットワークＮに対するルーティング確定関数Ｒは、チャネル従属性グラフＧに循環がない場合に限り、デッドロックフリーである。
【００７４】
次に補助定理を示すことができる。この補助定理の後にＳＦＴＲＥＥアルゴリズムのデッドロックフリー定理が続く。
【００７５】
補助定理８　ファットツリー内に少なくとも１つのサブツリーが存在する。これにより、このサブツリー内のＵターンのみを用いてスイッチ間の完全接続を構築できる。
【００７６】
証拠．ファットツリーにおいて、アップ／ダウン経路または水平兄弟経路を用いて、ファブリック内の任意のリーフから任意の他のノード（すべてのスイッチを含む）に到達することができる。その理由は、すべての終端ノードがその他すべての終端ノードと通信できること、および終端ノードが直接リーフスイッチに接続されていることにある。すなわち、トポロジー内においてリーフスイッチはその他任意のノードに到達することができる。結果として、リンク障害が含まれていなければ、いずれのリーフスイッチもサブツリーのルートとして機能できる。
【００７７】
補助定理９．デッドロックを生じさせずに１つのサブツリー内で無制限の数のＵターンが可能である。
【００７８】
証拠．サブツリーは論理ツリー構造でありその中で２種類のターンが可能である。これらのターンは、Ｕターンすなわち下方向から上方向へのターンと、通常の上方向から下方向へのターンである。Ｕターンは、サブツリー（およびファットツリー）内の最上部のスイッチでは不可能である。なぜなら、これらのスイッチには上向きの出力ポートがないからである。定義７に従い、上方向から下方向への任意のターンによってサブツリーの外に出る。
【００７９】
サブツリーは循環がない連結グラフであるため、それ自体がデットロックフリーである。いかなるデッドロックにもサブツリー外部のＵターンが関わっているはずである。その理由は、Ｕターンの次には上方向から下方向へのターンが続いてサブツリーから出ることにある。サブツリーから始まりアップ／ダウンターンを通して進むすべてのトラフィックは、ターンが行なわれるとサブツリーから外に出て、宛先への下方向のチャネルのみを通る。このような従属性は、再びサブツリーに入ることはできず、その他のＵターンに決して到達できず、したがって循環を形成できない。
【００８０】
定理２．　サブツリー法を用いたＳＦＴＲＥＥアルゴリズムはデッドロックフリーである。
【００８１】
証拠．　補助定理８および補助定理９に従うと、ファットツリー内のデッドロックは、サブツリーの外部でＵターンがあった場合にしか生じない。このような状況が生じ得ない理由は、すべてのＵターンがサブツリー内で生じるＳＦＴＲＥＥアルゴリズムの設計にある。サブツリーから出たトラフィックが循環してサブツリーに戻ることはない。なぜなら、トラフィックは、一旦サブツリーから出ると、下向きのチャネルのみを通して宛先に送られ、よって、トポロジー内では循環クレジット従属性が発生しないと考えられるからである。

【図５】

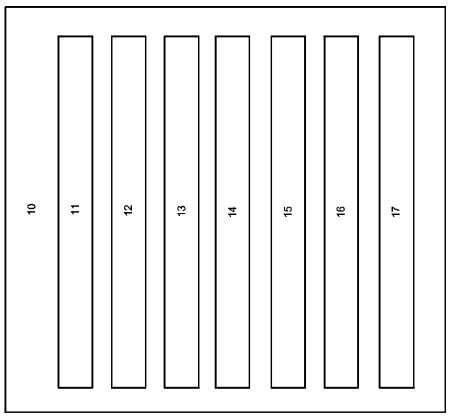

FIGURE 5

【図１】

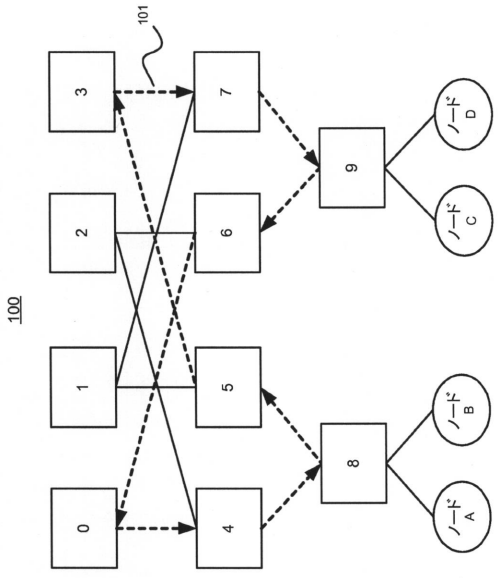

FIGURE 1

【図２】

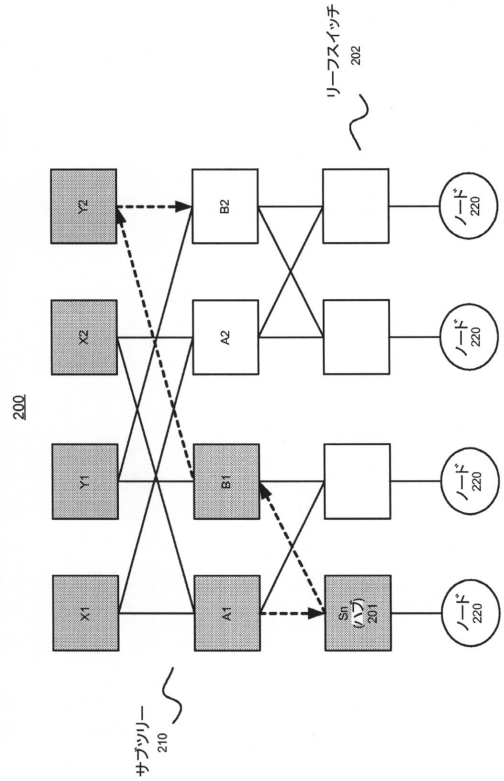

FIGURE 2

【図３】

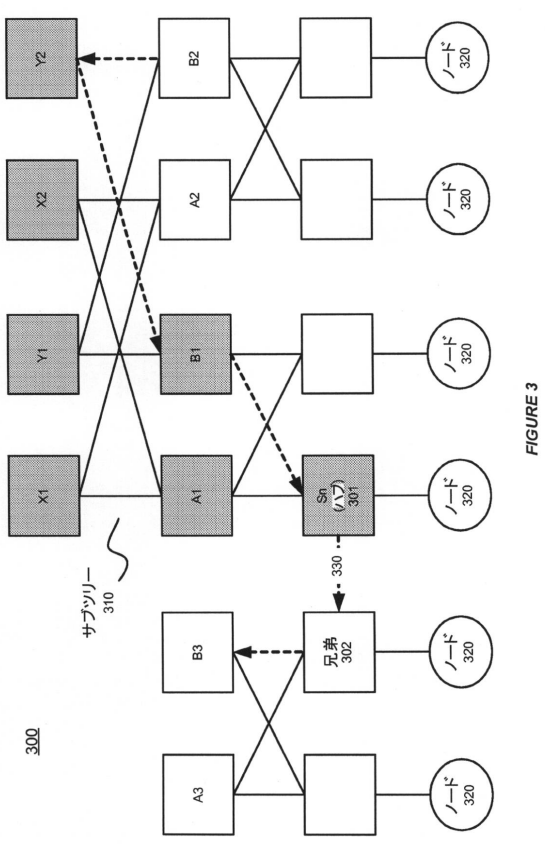

FIGURE 3

【図４】

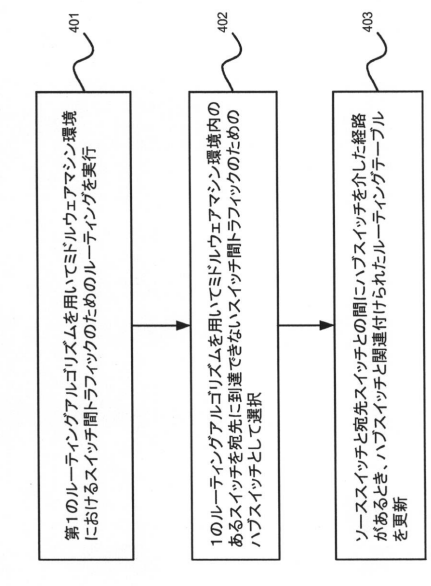
401
第1のルーティングアルゴリズムを用いてミドルウェアマシン環境におけるスイッチ間トラフィックのためのルーティングを実行
402
1のルーティングアルゴリズムを用いてミドルウェアマシン環境内のあるスイッチを宛先に到達できないスイッチ間トラフィックのためのハブスイッチとして選択
403
ソーススイッチと宛先スイッチとの間にハブスイッチを介した経路があるとき、ハブスイッチと関連付けられたルーティングテーブルを更新

FIGURE 4

フロントページの続き

(56)参考文献 特開２０１１－２２３４１９（ＪＰ，Ａ）
Frank Olaf Sem-Jacobsen 他, Dynamic Fault Tolerance in Fat Trees, IEEE TRANSACTIONS ON COMPUTERS, IEEE Computer Society, ２０１１年 4月, Vol.60, No.4, p.508-525

(58)調査した分野(Int.Cl., ＤＢ名)
Ｈ０４Ｌ １２／７５３