(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第7277682号
(P7277682)
(45)発行日　令和5年5月19日(2023.5.19)　(24)登録日　令和5年5月11日(2023.5.11)

(51)Int.Cl.		F I	
H04L 49/201	*(2022.01)*	H04L	49/201
H04L 45/12	*(2022.01)*	H04L	45/12
H04L 45/24	*(2022.01)*	H04L	45/24
G06N 3/04	*(2023.01)*	G06N	3/04
G06N 3/063	*(2023.01)*	G06N	3/063

請求項の数 6　外国語出願　(全 18 頁)

(21)出願番号　特願2019-124541(P2019-124541)
(22)出願日　令和1年7月3日(2019.7.3)
(65)公開番号　特開2021-13048(P2021-13048A)
(43)公開日　令和3年2月4日(2021.2.4)
審査請求日　令和4年3月28日(2022.3.28)

(73)特許権者　506301140
公立大学法人会津大学
福島県会津若松市一箕町大字鶴賀字上居合
９０番地
(74)代理人　100094525
弁理士　土井　健二
(74)代理人　100094514
弁理士　林　恒徳
(72)発明者　ベン　アブダラ　アブデラゼク
福島県会津若松市一箕町大字鶴賀字上居合
９０番地　公立大学法人会津大学内
(72)発明者　ヴー　フィ　テー
福島県会津若松市一箕町大字鶴賀字上居合
９０番地　公立大学法人会津大学内

最終頁に続く

(54)【発明の名称】３次元ネットワークオンチップによるスパイキングニューラルネットワーク

(57)【特許請求の範囲】
【請求項１】
３次元ネットワークオンチップによるスパイキングニューラルネットワークであって、
複数の重心をランダムに決定し、
前記３次元ネットワークオンチップに実装された複数の送信先ルータのそれぞれから前記複数の重心のそれぞれまでの距離を算出し、
算出した前記距離に基づいて、前記複数の送信先ルータを前記複数の重心のそれぞれに対応する複数のサブグループのいずれかに割り当て、
前記複数のサブグループに対する前記複数の送信先ルータの割り当て結果に基づいて、前記複数の重心を再決定し、
前記３次元ネットワークオンチップに実装された送信元ルータから前記複数の送信先ルータに含まれる第１の送信先ルータに対してパケットが送信される場合、再決定した前記複数の重心のうちの前記第１の送信先ルータに対応する第１の重心を特定し、
特定した前記第１の重心に対応するサブグループに割り当てられた前記複数の送信先ルータのうち、前記送信元ルータからの距離が最短である第２の送信先ルータを特定し、
特定した前記第２の送信先ルータを通過するように、前記パケットの送信経路を特定し、
特定した前記送信経路を用いて前記パケットを送信する、
ことを特徴とする３次元ネットワークオンチップによるスパイキングニューラルネットワーク。

【請求項２】
請求項１において、
前記割り当てる処理では、
前記複数の送信先ルータごとに、前記複数の重心のうち、算出した前記距離が最も短い第１の重心を特定し、
前記複数の送信先ルータのそれぞれを、各送信先ルータに対応する前記第１の重心に対応するサブグループに割り当てる、
ことを特徴とする３次元ネットワークオンチップによるスパイキングニューラルネットワーク。
【請求項３】
請求項１において、
前記算出する処理では、前記複数の送信先ルータのそれぞれから前記複数の重心のそれぞれまでのマンハッタン距離を算出する、
ことを特徴とする３次元ネットワークオンチップによるスパイキングニューラルネットワーク。
【請求項４】
請求項１において、
前記算出する処理と、前記割り当てる処理と、前記再決定を行う処理とを、再選択した前記複数の重心が変更されなくなるまで繰り返す、
ことを特徴とする３次元ネットワークオンチップによるスパイキングニューラルネットワーク。
【請求項５】
請求項１において、
前記パケットの送信経路を特定する処理では、再決定した前記複数の重心のうちの前記第１の送信先ルータに対応する重心を通過するように、前記パケットの送信経路を特定する、
ことを特徴とする３次元ネットワークオンチップによるスパイキングニューラルネットワーク。
【請求項６】
請求項１において、
前記パケットの送信経路を特定する処理では、前記送信元ルータから前記第１の送信先ルータまでの間に位置する第３の送信先ルータと第４の送信先ルータとの間における複数の送信経路を特定し、
前記送信する処理では、特定した前記複数の送信経路のうちの第１の送信経路が使用可能である場合、前記第１の送信経路を用いて前記パケットを送信し、前記第１の送信経路が使用可能でない場合、特定した前記複数の送信経路のうちの第２の送信経路を用いて前記パケットを送信する、
ことを特徴とする３次元ネットワークオンチップによるスパイキングニューラルネットワーク。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、３次元ネットワークオンチップによるスパイキングニューラルネットワークに関する。
【背景技術】
【０００２】
近年、神経科学の研究は、個々のニューロンの構造及び動作について多くのことを明らかにし、医療ツールによって、脳のさまざまな領域の神経活動が感覚刺激に従う様子についての理解が可能になってきている。また、ソフトウェアベースの人工知能（ＡＩ：Ａｒｔｉｆｉｃｉａｌ　Ｉｎｔｅｌｌｉｇｅｎｃｅ）の進歩は、従来のフォンノイマンコンピ

ューティングスタイルのボトルネックを克服させる脳のような機能を有するデバイス及びシステムの構築技術の最先端に我々を到達させている。

【０００３】

ニューロインスパイアードシステムまたはニューロモルフィックシステムと従来の情報処理システムとの間の主な違いは、ニューロインスパイアードシステムやニューロモルフィックシステムがメモリ構造及び組織を使用していることにある。フォンノイマンスタイルに基づくシステムが、メインメモリ領域から物理的に分離された１つまたは複数の中央処理装置を有しているのに対し、生物学的（スパイキング）ニューラルネットワークシステム及び人工ニューラルネットワークシステムのそれぞれでは、共局在化されたメモリ及び計算の分散が行われている。スパイキングニューラルネットワーク（Ｓｐｉｋｉｎｇ Ｎｅｕｒａｌ Ｎｅｔｗｏｒｋｓ：以下、ＳＮＮと呼ぶ）に基づくニューロインスパイアードテクノロジーは、脳についてのより良い理解を獲得し、そして、生物学に触発された新しい計算を探求するために注目を集めている。ＳＮＮは、視覚認識タスクや分類タスク等のいくつかのアプリケーションに正常に適用されている（非特許文献１）。また、ニューロモルフィックハードウェアの実装は、大規模ネットワークをリアルタイムで実行することを可能にする。これは、ニューロロボティクス制御、ブレインマシンインタフェース及びロボットによる意思決定を含むいくつかのアプリケーションにおいて重要な要件である。

【０００４】

ＳＮＮは、スパイク事象を介して通信するニューロンの並列アレイに基づいて哺乳動物の脳における情報処理の模倣を試みる。ニューロンが各伝播サイクルにおいて発火する典型的な多層パーセプトロンネットワークとは異なり、ＳＮＮモデルのニューロンは、膜電位が特定の値に達したときにのみ発火する。ＳＮＮにおいて、情報は、一致符号化、レート符号化、時間符号化等のさまざまな符号化方式を用いることによって符号化される（特許文献１）。ＳＮＮでは、通常、他のニューロンからの外部刺激によって十分な刺激を受けた場合に、ニューロンが神経線維を伝達可能な電圧スパイク（スパイクあたり持続時間は約１ｍｓ）を生成する統合発火型ニューロンモデル（非特許文献２及び３）が採用される。これらのパルスは、振幅、形状及び持続時間が異なるが、一般的に、同一のイベントとして取り扱われる。また、Ｈｏｄｇｋｉｎ－Ｈｕｘｌｅｙのコンダクタンスに基づくニューロン（非特許文献４）は、生物学的ニューロンのイオンチャネルの非線形及び確率的な力を効率的にモデル化するためによく使用される。しかしながら、Ｈｏｄｇｋｉｎ－Ｈｕｘｌｅｙモデルは、大規模なシミュレーションやハードウェア実装に使用するには複雑すぎるという問題がある。

【０００５】

近年、多数のディープＳＮＮが提案されている（非特許文献５）。これらは、多くのスパイキングニューロンから構成されており、さまざまなパターン認識タスクにおいて成功している（非特許文献１及び６）。しかしながら、これらのモデルは、多層として知られているが、伝統的なディープニューラルネットワークと比較して多くの訓練可能な層を持っていないことに言及すべきである。これは、従来のＡＮＮ（Ａｒｔｉｆｉｃｉａｌ Ｎｅｕｒａｌ Ｎｅｔｗｏｒｋｓ）の逆伝播のように、スパイキングディープネットワークを直接的に訓練するための効率的な学習規則がないためである（非特許文献５）。一方、大規模なＳＮＮは、脳の複雑な活動をシミュレートするために求められる。例えば、Ｓｐａｎｕと呼ばれる２５０万ニューロンモデルが存在している（非特許文献７）。Ｓｐａｎｕは、神経解剖学、神経生理学及び心理的行動の多くの側面を捉え、数字認識タスクについても精度良く実行する。ディープＳＮＮにおいて、ニューロン間の通信は、実装時に不可欠な役割を果たす。多数のニューロンを平面構造にマッピングし、その結果として得られる平面のダイを貫通シリコンビア（ＴＳＶ：Ｔｈｒｏｕｇｈ－Ｓｉｌｉｃｏｎ ｖｉａ）を用いて積み重ねることによって、通信待ち時間を大幅に短縮することが可能になる。

【０００６】

ＳＮＮのソフトウェアシミュレーションは、ニューロシステムの挙動を調べるための適

切な方法である。しかしながら、ソフトウェアによる大規模な（深い）ＳＮＮシステムのシミュレーションは低速である。他の手法としては、独立したスパイクを正確に生成し、同時にスパイクをリアルタイムで出力する可能性を提供するハードウェア実装がある。ハードウェア実装は、ソフトウェアシミュレーションよりも計算速度が向上するという利点を有するため、固有の並列処理を行った場合における利点を最大限に活用することが可能である。そして、複数のニューロコアを持つ特殊なハードウェアキテクチャは、ニューラルネットワーク固有の並列処理を活用することで、低電力で高い処理速度を実現することが可能にある。そのため、ＳＮＮは、組み込みニューロモルフィックデバイスや制御アプリケーションに適している。

【０００７】

大量のシナプスを持つスパイキングニューラルネットワークアーキテクチャ（ニューロモルフィック）をハードウェアで構築する際に解決する必要がある課題には、低消費電力での小型の超並列アーキテクチャ、効率的なニューロコーディングスキーム、及び、軽量なオンチップ学習アルゴリズムの構築が含まれる。他の主要な課題は、ニューロコアとそのコアに転送されるオフチップデータとの間でデータを通信させるオンチップ通信及びルーティングネットワークである。さらに、接続されるニューロンの数は、現在のマルチコア/マルチプロセッサＳｏＣ（Ｓｙｓｔｅｍ ｏｎ ａ Ｃｈｉｐ）プラットフォームにおいて相互接続される必要があるＰＥ（Ｐｒｏｃｅｓｓｉｎｇ Ｅｌｅｍｅｎｔ）の数の少なくとも１０３倍である（非特許文献８）。上記の制約により、このような頭脳に似たＩＣ（Ｉｎｔｅｇｒａｔｅｄ Ｃｉｒｃｕｉｔ）の展開は、困難なオンチップ相互接続の問題となる（非特許文献９）。ＳＮＮにおいて、各ニューロンは、入力スパイク、シナプス荷重、現在の膜電位、及び、一定の漏れ係数を含むいくつかのパラメータの関数である内部膜電位を維持する（非特許文献１０）。ニューロンの活動は、ニューロン及びニューラルシステムの機能的特性を決定する上で重要な役割を果たすニューロンの連結性によって制約されている。脳の連結性は、一般的に、以下のようないくつかのスケールで記述される：（１）個々のニューロンをマイクロスケールでリンクする個々のシナプス接続、（２）メソスケールにおいてニューロンの細胞集団を結ぶネットワーク、そして、（３）マクロスケールで線維経路によって結び付けられた脳領域。

【０００８】

適切なニューロンとネットワークモデルとを備えた効率的なＳＮＮにおいて、ニューロンへのディラックデルタ関数や整形後シナプス電位（ＥＰＳＰ：Ｅｘｃｉｔａｔｏｒｙ ＰｏｓｔＳｙｎａｐｔｉｃ Ｐｏｔｅｎｔｉａｌ／ＩＰＳＰ：Ｉｎｈｉｂｉｔｏｒｙ ＰｏｓｔＳｙｎａｐｔｉｃ Ｐｏｔｅｎｔｉａｌ）等のシナプス入力の到着時間は、ニューロンの出力（スパイク）の時間に大きな影響を与える。その結果、図１に示すように、タイミング違反は、スパイキングニューロンの適切な機能（発火）やシステム全体のオンチップ学習機能に影響を与える。

【０００９】

通信媒体としての共有バスは、マルチキャストルーティングを備えた大規模で複雑なＳＮＮチップ／システムの実装に適していない。これは、ニューロンを追加すると、チップの通信容量が減少し、さらに、共有バスの長さが長くなるためにニューロンの発火率に影響を与える可能性があるからである。また、ニューラル接続における非線形の増加は、専用のポイントツーポイント通信方式を使用した直接的な実施において非常に重要である。

【００１０】

二次元パケット交換ネットワークオンチップ（２Ｄ－Ｎｏｃ：Ｔｗｏ－ｄｉｍｅｎｓｉｏｎａｌ ｐａｃｋｅｔ－ｓｗｉｔｃｈｅｄ Ｎｅｔｗｏｒｋ－ｏｎ－Ｃｈｉｐ）は、従来提案されてきたＳＮＮに基づく共有通信媒体に見られる相互接続問題に対処するための潜在的な解決策として考えられてきた（非特許文献９及び１２）。しかしながら、このような相互接続戦略は、特に大規模ＳＮＮチップにおいて、低い電力消費で高い拡張性を達成することを困難にする。パケット交換ＮｏＣとは別に、回線交換ＮｏＣを使用すると、さまざまなルーティング/スイッチングメカニズムのパフォーマンスを調べることが可能

になる。回線交換ＮｏＣは、パケット交換と比較して、ハードウェアの複雑さが小さくエネルギー効率が高いが、セットアップ時間が長くなる。

【００１１】

ここ数年で、３Ｄ－ＩＣとメッシュベースのＮｏＣの利点は、特にＡＩを搭載したチップにおいて、ＩＣ設計の新たな領域を開く有望なアーキテクチャに融合された。ＮｏＣの並列性は、短いワイヤ長と３Ｄ－ＩＣの相互接続の低消費電力のおかげで、３次元において強化することが可能である。その結果、３Ｄ-ＮｏＣパラダイムは、将来のＩＣ設計にとって最も先進的で好都合なアーキテクチャの１つであると考えられている。３Ｄ-ＮｏＣは、非常に高い帯域幅であって低消費電力の相互接続（非特許文献１３）を提供し、新たな人工知能（ＡＩ）アプリケーションの高い要件を満たすことが可能になる。３Ｄ－ＮｏＣとＳＮＮとを組み合わせる場合、スパイキングニューロンは、ＰＥ（ニューロコア）と見なすことが可能になる。ニューロン間の接続性は、拡張性のある相互接続ネットワークを介してスパイクパケットを送信する形で実装される。なお、この場合、ＰＥは、３Ｄ－ＮｏＣルータに接続されたＳＮＰＣ（Ｓｐｉｋｉｎｇ　Ｎｕｅｒｏｎ　Ｐｒｏｃｅｓｓｉｎｇ　Ｃｏｒｅ）を指しており、ＮｏＣチャネルは、ニューロンのシナプスに類似しており、さらに、ＮｏＣトポロジは、ニューロンがネットワーク内で相互接続される方法を指している。

【００１２】

ＳＮＮのハードウェア実装の主な問題の１つは、それらの信頼性に関する可能性である。ＳＮＮには、生物学的神経モデルによって触発された大規模で平行な構造のおかげで、いくつかの固有のフォールトトレランス特性があると言われているが、実際の場合に関しては必ずしもそうではない（非特許文献１４）。実際、半導体部品の継続的な縮小から引き継がれた課題により、ハードウェアでのＳＮＮの実装は、さまざまな障害にさらされる（非特許文献１４）。歩留まりが大きな問題となる場合、組み込みシステム向けの大規模なＳＮＮの統合に進むにつれて、障害リスクはさらに重要になる（非特許文献１５）。ニューロン間通信の信頼性を考慮する場合、特に重要なアプリケーション（航空宇宙、自動運転車、生物医学など）で発生する場合において、障害は、システムのパフォーマンスに影響を与える可能性がある。このような障害は、望ましくない不正確さ、または、不可逆的であって深刻な結果を招く可能性がある。ＳＮＮでは、ニューロン間接続に障害が発生すると、シナプス後ニューロンが無反応状態または無反応に近い状態（低発火活動状態）になる。図１（ｃ）に示すように、Ｎ１からＮ４への接続にリンク切れが存在する場合、Ｎ４の潜在的な膜では、図１（ｂ）の場合のように、出力スパイクを発火させる閾値に到達しない。これにより、シナプス後ニューロンの発火率が低下する。

【００１３】

従って、レートコーディング方法に基づいたＳＮＮモデルの全体的なパフォーマンスに影響を与える可能性がある（非特許文献１６）。発火率が低いニューロンは、発火率のノイズと分散を増加させるスパイクの一時的なジッタの影響を受けやすくなる（非特許文献１７）。その結果、効率的なフォールトトレラント技術が必要となる。このようなメカニズムでは、回復時間が重要な要件の１つになる。図１（ｄ）に示すように、フォールトトレラントルーティング方法の長い待ち時間が発火率に影響を与える可能性がある。特に、スパイク間の相対的なタイミングに基づく一時的なコーディング方法を使用するＳＮＮモデルに影響を与える可能性がある。

【００１４】

そのため、シリコンへの大規模なＳＮＮの統合により、効率的なフォールトトレラントソリューションを見つけるという課題がより重要になる。

【先行技術文献】

【特許文献】

【００１５】

【特許文献１】米国特許出願公開第２０１４／０３５１１９０号明細書
【特許文献２】特開２０１５－１１９３８７号公報

【非特許文献】
【００１６】
【非特許文献１】Y. Cao, Y. Chen, and D. Khosla, "Spiking deep convolutional neural networks for energy-efficient object recognition," Int. J. Comput. Vision, vol. 113, no. 1, pp. 54-66, May 2015.
【非特許文献２】N. Burkitt, "A review of the integrate-and-_re neuron model: I. homogeneous synaptic input," Biol. Cybern., vol. 95, no. 1, pp. 1-19, Jun. 2006. [Online]. Available: http://dx.doi.org/10.1007/s00422-006-0068-6
【非特許文献３】K. Suzuki, Y. Okuyama, and A. B. Abdallah, "Hardware design of a leaky integrate and fire neuron core towards the design of a low-power neuro-inspired spike-based multicore soc," in Information Processing Society Tohoku Branch Conference, February 2018.
【非特許文献４】J. H Goldwyn, N. S Imennov, M. Famulare, and E. Shea-Brown, "Stochastic differential equation models for ion channel noise in hodgkin-huxley neurons," in Phys. Rev. E, vol. 83, no. 1, 2011, pp. 4190-4208.
【非特許文献５】A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, "Deep learning in spiking neural networks," Neural Networks, 04 2018.
【非特許文献６】P. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," Frontiers in Computational Neuroscience, vol. 9, p. 99, 2015.
【非特許文献７】C. Eliasmith, T. C. Stewart, X. Choo, T. Bekolay, T. DeWolf, Y. Tang, and D. Rasmussen, "A large-scale model of the functioning brain." Science, vol. 338 6111, pp. 1202-1205, 2012.
【非特許文献８】S. Furber and S. Temple, "Neural systems engineering," Journal of the Royal Society Interface, vol. 4, no. 13, pp. 193-206, Sep 2006.
【非特許文献９】S. Carrillo, J. Harkin, L. J. McDaid, F. Morgan, S. Pande, S. Cawley, and B. McGinley, "Scalable hierarchical network-on-chip architecture for spiking neural network hardware implementations," IEEE Transactions on Parallel and Distributed Systems, vol. 24, no. 12, pp. 2451-2461, Dec 2013.
【非特許文献１０】W. Maas, "Networks of spiking neurons: The third generation of neural network models," Trans. Soc. Comput. Simul. Int.,vol. 14, no. 4, pp. 1659-1671, Dec. 1997. [Online]. Available: http://dl.acm.org/citation.cfm?id=281543.281637
【非特許文献１１】A. Ben Abdallah, Advanced Multicore Systems-On-Chip Architecture, On-Chip Network, Design. Springer, 2017.
【非特許文献１２】R. Hojabr, M. Modarressi, M. Daneshtalab, A. Yasoubi, and A. Khonsari, "Customizing clos network-on-chip for neural networks," IEEE Transactions on Computers, vol. 66, no. 11, pp. 1865-1877, Nov 2017.
【非特許文献１３】K. N. Dang, A. B. Ahmed, Y. Okuyama, and B. A. Abderazek, "Scalable design methodology and online algorithm for tsv-cluster defects recovery in highly reliable 3d-noc systems," IEEE Transactions on Emerging Topics in Computing, pp. 1-1, 2017.
【非特許文献１４】C. Torres-Huitzil and B. Girau, "Fault and error tolerance in neural networks: A review," IEEE Access, vol. 5, pp. 17322-17341, 2017.
【非特許文献１５】P. M. Furth and A. G. Andreou, "On fault probabilities and yield models for vlsi neural networks," IEEE Journal of Solid-State Circuits, vol. 32, no. 8, pp. 1284-1287, Aug 1997.
【非特許文献１６】P. U. Diehl, D. Neil, J. Binas, M. Cook, S. Liu, and M. Pfeiffer, "Fast-classifying, high-accuracy spiking deep networks through weight and t

hreshold balancing," in 2015 International Joint Conference on NeuralNetworks (IJCNN), July 2015, pp. 1-8.
【非特許文献１７】M. Pfeiffer and T. Pfeil, "Deep learning with spiking neurons: Opportunities and challenges," Frontiers in Neuroscience, vol. 12, p. 774, 2018. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2018.00774
【非特許文献１８】D. Vainbrand and R. Ginosar, "Scalable network-on-chip architecture for configurable neural networks," Microprocess. Microsyst., vol. 35, no. 2, pp. 152-166, Mar. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.micpro.2010.08.005
【非特許文献１９】B. A. Akram and B. A. Abderazek, "Adaptive fault-tolerant architecture and routing algorithm for reliable many-core 3d-noc systems," J. Parallel Distrib. Comput., vol. 93, no. C, pp. 30-43, Jul. 2016.
【非特許文献２０】W. Gerstner and W. Kistler, Spiking Neuron Models: Single Neurons, Populations, Plasticity. Cambridge University Press, 2002.
【非特許文献２１】K. N. Dang, M. Meyer, Y. Okuyama, and A. B. Abdallah, "Reliability assessment and quantitative evaluation of soft-error resilient 3d network-on-chip systems," in 2016 IEEE 25th Asian Test Symposium (ATS), Nov 2016, pp. 161-166.
【非特許文献２２】X. Lin and L. M. Ni, "Multicast communication in multicomputer networks," IEEE Transactions on Parallel and Distributed Systems, vol. 4, no. 10, pp. 1105-1117, Oct 1993.
【非特許文献２３】S. H. Strogatz, "Exploring complex networks," vol. 410, pp. 268-276, 03 2001.
【非特許文献２４】F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G. J. Nam, B. Taba, M. Beakes, B. Brezzo, J. B. Kuang, R. Manohar, W. P. Risk, B. Jackson, and D. S. Modha, "Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 34, no. 10, pp. 1537-1557, Oct 2015.

【発明の概要】
【発明が解決しようとする課題】
【００１７】
ルーティングアルゴリズムは、ニューロン通信のパフォーマンスに重要な役割を果たすために、ＳＮＮで最も効率的な回復メカニズムの１つと見なされている。ルーティングアルゴリズムは、ネットワーク全体の負荷分散と、障害のないシナリオでのシステムの全体的な遅延とに影響を与える可能性がある（非特許文献１１）。与えられたＳＮＮのトラフィックパターンは、シナプス前ニューロンがシナプス後ニューロンのサブセットにスパイクを送信する１対多の方法であるため、大規模なＳＮＮでの従来のユニキャストベースのルーティングの使用は、非効率的である（非特許文献１８）。さらに、フォールトトレランスの要件を考慮する場合、ニューロン間通信の遅延を最小限に抑えるために、ルーティングアルゴリズムを慎重に選択する必要がある。さもなければ、障害が回避されたという事実にもかかわらず、シナプス後ノードの精度が低下する可能性がある。図１（ｄ）は、このような場合の明確な例を示している。この図では、不適切なルーティングによる長い待ち時間が、シナプス後ニューロンによる出力スパイクのタイムリーな発火を妨げる可能性があることを示している。
【００１８】
そこで、本発明の目的は、脳の固有の３Ｄ構造を活用し、大規模なＳＮＮベースのコンピューティングシステムのシームレスな実装を可能にする新しいマルチキャストスパイクルーティングアルゴリズムを提案することにより、ニューロン間の通信遅延を削減するこ

とである。
【課題を解決するための手段】
【００１９】
本発明の一態様では、３次元ネットワークオンチップによるスパイキングニューラルネットワークであって、複数の重心をランダムに決定し、前記３次元ネットワークオンチップに実装された複数の送信先ルータのそれぞれから前記複数の重心のそれぞれまでの距離を算出し、算出した前記距離に基づいて、前記複数の送信先ルータを前記複数の重心のそれぞれに対応する複数のサブグループのいずれかに割り当て、前記複数のサブグループに対する前記複数の送信先ルータの割り当て結果に基づいて、前記複数の重心を再決定し、前記３次元ネットワークオンチップに実装された送信元ルータから前記複数の送信先ルータに含まれる第１の送信先ルータに対してパケットが送信される場合、再決定した前記複数の重心に基づいて、前記パケットの送信経路を特定し、特定した前記送信経路を用いて前記パケットを送信する。
【発明の効果】
【００２０】
脳の固有の３Ｄ構造を活用し、大規模なＳＮＮベースのコンピューティングシステムのシームレスな実装を可能にする新しいマルチキャストスパイクルーティングアルゴリズムを提案することにより、ニューロン間の通信遅延を削減する。
【図面の簡単な説明】
【００２１】
【図１】図１は、発火率に対する接続障害の影響の例を示す図である。
【図２】図２は、システムアーキテクチャの概要を示す図である。
【図３】図３は、ＳＮＰＣアーキテクチャを示す図である。
【図４】図４は、ＦＴＭＣ－３ＤＲアーキテクチャを示す図である。
【図５】図５は、ＫＭＣＲマルチキャストルーティング擬似コードのアルゴリズムを示す図である。
【図６】図６は、６×３×２メッシュの３ＤＮｏＣのＫＭＣＲアルゴリズムについての例を示す図である。
【図７】図７は、プライマリブランチとバックアップブランチとを示す図である。
【図８】図８は、プライマリブランチとバックアップブランチとのオフライン計算についてのＦＴＭＰ－ＫＭＣＲアルゴリズムを示す図である。
【図９】図９は、「ｓｏｎ」、バックアップ時、「ｆａｔｈｅｒ」、「ｇｒａｎｄｆａｔｈｅｒ」に対応する各ルータに適用される障害管理アルゴリズムを示す図である。
【発明を実施するための形態】
【００２２】
以下、図面を参照して本発明の実施の形態について説明する。各実施の形態は、本発明のより良い理解のために準備されている。ただし、かかる実施の形態は、本発明の技術的範囲を限定するものではない。また、本発明の範囲は、特許請求の範囲及びこれと同等のものを網羅している。
【００２３】
最初に、スパイクトラフィックルーティング用の低遅延マルチキャストルーティングスキームに基づく本発明による３ＤＦＴ－ＳＮＮアーキテクチャについて説明を行う。図２は、システムアーキテクチャの概要を示す図である。
【００２４】
図２に示されるように、システム１００（３ＤＦＴ－ＳＮＮシステム１００）は、スパイキングニューラルタイル１０がいくつかの積み重ねられた２Ｄ層からなり、従来の３Ｄ－ＮｏＣアーキテクチャに基づいている（非特許文献１３及び１９）。具体的に、図２では、４×４の２Ｄ層からなるスパイキングニューラルタイル１０が積み重ねられている例が示されている。
【００２５】

スパイキングニューラルタイル１０は、スパイキングニューラルプロセッシングコア（Ｓｐｉｋｉｎｇ Ｎｅｕｒａｌ Ｐｒｏｃｅｓｓｉｎｇ Ｃｏｒｅ：以下、ＳＮＰＣ1と呼ぶ）と、フォールトトレラントマルチキャストルータ（Ｆａｕｌｔ－Ｔｏｌｅｒａｎｔ Ｍｕｌｔｉｃａｓｔ Ｒｏｕｔｅｒ：以下、ＦＴＭＣ－３ＤＲ２とも呼ぶ）とから構成される。ＳＮＮに関連して、スパイクニューロンはＰＥを指しており、ニューロン間接続は、拡張性のある３Ｄ－ＮｏＣを介してスパイク（パケット）を送信する形で実装され、さらに、トポロジは、ネットワーク内でニューロンが相互接続される方法を指している。図３に示すように、３ＤＦＴ－ＳＮＮシステム１００内の各ＳＮＰＣ１は、スパイキングニューロンのアレイを使用して着信スパイクを処理する。

【００２６】

ＳＮＰＣ１は、システム１００の主要な処理ユニットである。図３に示す例では、入力スパイクが最初にデコードされて、それらのシナプス後ニューロンが決定される。重み値は、クロスバーベースのシナプスを介して、ＬＩＦ（Ｌｅａｋｙ Ｉｎｔｅｇｒａｔｅ－ａｎｄ－Ｆｉｒｅ）ニューロンの配列に蓄積される（非特許文献２０）。

【００２７】

ＳＮＰＣ１は、クロスバーアプローチに基づいている。ここでは、オンチップＳＲＡＭを使用してＮ×Ｎクロスバー（Ｎはニューロンの数）を実装する。各シナプスは、５ビットで表され、シナプスタイプ（すなわち、興奮性及び抑制性）のために１ビットが用いられ、重みのために４ビットが用いられる。以下、ＳＮＰＣ１の主要コンポーネントについて説明を行う。

【００２８】

デコーダ１１は、着信スパイク（パケット）ごとにシナプス後ニューロンを決定する。宛先ニューラルタイルに到着すると、着信スパイクは、ローカルルータによってローカルのＳＮＰＣ１に転送される。デコーダ１１は、ニューロンＩＤに基づいて、ルックアップテーブル（ＬＵＴ：Ｌｏｏｋｕｐ Ｔａｂｌｅ）を検索してシナプス後ニューロンを決定する。この情報は、ニューラル計算のために制御ユニット１２に送信される。

【００２９】

制御ユニット１２は、ニューラルコアの全体的な動作を制御するように設計されている。制御ユニット１２は、ニューラルコアの構成モードと動作モードの両方を制御する。制御ユニット１２は、単一のタイムステップの間にニューロンを更新することを保証する。

【００３０】

シナプスクロスバーには、シナプスのクロスポイント配列が含まれている。各シナプスは、行（軸）と列（樹状突起）間の接続（シナプス）の表示するビットであって、読み取り、設定またはリセットが可能なビットを格納する。このビットは、デコードの完了後に書き込まれている間に、ニューラル計算のために読み取られる。

【００３１】

シナプスメモリ１３（以下、ｓｙｍ＿ｍｅｍ１３とも呼ぶ）は、クロスバーとシナプス強度の設定に使用されるシナプス情報を格納する場所である。シナプス情報は、トレーニングフェーズにおいて更新され、推論操作において読み取りが行われる。

【００３２】

ニューラルメモリ１４（以下、ｎｅｕ＿ｍｅｍ１４とも呼ぶ）は、ニューラルパラメータに使用される。各パラメータは、ニューラル計算のために読み取られる。そして、ニューラル計算が行われた後、各パラメータは、ニューロンの現在の状態を保存するために更新される。

【００３３】

ＬＩＦアレイ１５は、ニューラル計算が実行されるニュートロンコアの主要な計算ユニットである。データは、シナプスクロスバーから読み取られ、ｓｙｍ＿ｍｅｍ１３及びｎｅｕ＿ｍｅｍ１４は、このユニットにおいて計算される。ここでは、複数のＬＩＦニューロンが実装されている。より正確には、複数のニューロンが順次実行される間に、物理的なＬＩＦ計算ユニットが実装される。これは、デジタルロジックの高速動作を利用するだ

けでなく、エリアコストと消費電力を削減する。

【００３４】

エンコーダ１６は、ＬＩＦアレイ１５から生成されたスパイクを詰めるように設計されている。ニューロンの膜電位が閾値を超えると、ニューロンは、スパイク（発火）を生成する。このスパイクは、エンコーダ１６に送信され、そこでローカルルータを介してネットワークに流入される前にパケットに詰められる。

【００３５】

構成情報１７は、ニューラルコアの構成に使用される。この情報には、シナプス及びニューロンモデルに関連する構成パラメータが含まれている。ニューラルコアの構成は、システムが動作する前であってアプリケーションのマッピングが行われる間に実行される。

【００３６】

次に、フォールトトレラントマルチキャスト３Ｄルータ（Ｆａｕｌｔ－Ｔｏｌｅｒａｎｔ　Ｍｕｌｔｉｃａｓｔ　３Ｄ　Ｒｏｕｔｅｒ：以下、ＦＴＭＣ－３ＤＲ２）アーキテクチャについて説明を行う。図４は、ＦＴＭＣ－３ＤＲアーキテクチャを示す図である。

【００３７】

各ニューロンは、数千の他のニューロンに接続できるため、ＦＴＭＣ－３ＤＲ２は、効率的なスパイク配信のためにマルチキャストルーティングをサポートする。ＦＴＭＣ－３ＤＲ２は、従来の３ＤＲアーキテクチャに基づいている（非特許文献１３、１９及び２１）。スパイク時間が情報のエンコードに使用されるため、ＦＴＭＣ－３ＤＲ２の遅延は、非常に短くなるはずである。システム１０の各ルータ２には、最大７つの入力ポートと７つの出力ポートがあり、そのうちの６つの入力/出力ポートが隣接ルータ専用であり、１つの入力/出力ポートがスイッチをＳＮＰＣ１に接続するために使用される。そして、ＦＴＭＣ－３ＤＲ２には、スイッチアロケーター２２に加えて、各方向のそれぞれに対応する７つの入力ポートモジュール２１が含まれる。また、ＦＴＭＣ－３ＤＲ２には、次のＳＮＰＣ１へのスパイクの転送を処理するクロスバーモジュール２３が含まれる。入力ポートモジュールは、入力バッファ２１ａとマルチキャストルーティングモジュール２１ｂとの２つの主要な要素で構成されている。

【００３８】

ルータ２は、バッファ書き込み（ＢＷ）、ルーティング計算（ＲＣ）、スイッチ調停（ＳＡ）、及びクロスバー横断（ＣＴ）の４つのパイプラインステージで設計されている。最初の段階において、着信スパイク（パケット）は、処理される前に入力バッファ２１ａに格納される。次に、パケットの送信元アドレス（Ｘｓ；Ｙｓ；Ｚｓ）が抽出及び計算され、出力ポートが決定される。ルーティング計算の後、選択された出力ポートを使用するために、リクエスト（ｓｗ＿ｒｅｑｕｅｓｔ信号）がスイッチアロケーター２２に送信される。スイッチアロケーター２２には、一般的なＳｔａｌｌ／Ｇｏフロー制御２２ａ（非特許文献１１）と、Ｍａｔｒｉｘ－ａｒｂｉｔｅｒスケジューラ２２ｂとの２つの主要コンポーネントが含まれる。ここでは、高速計算、安価な実装及び強力な公平性を提供するために、優先度が最も低いＭａｔｒｉｘ－ａｒｂｉｔｅｒが採用されている（非特許文献１１）。最後に、パケットは、（ｓｗ＿ｇｒａｎｔ信号を介して）許可された後、クロスバー２３を通過する目的の出力ポートに送信される。

【００３９】

ルータ２は、ルーティングパイプラインステージでのソフトエラーに加えて（非特許文献２１）、入力バッファ２１ａ、クロスバー２３及びリンクにおけるハードの欠点を処理するための冗長構造リソースを使用したシステム再構成に基づく高度な回復技術に依存している（非特許文献１３及び１９）。これらのメカニズムは、システムにおいて発生する障害を軽減することを目的としている。

【００４０】

次に、マルチキャストスパイクルーティングアルゴリズムに基づくＫ－ｍｅａｎｓクラスタリング（Ｋ－ｍｅａｎｓ　Ｃｌｕｓｔｅｒｉｎｇ：以下、ＫＭＣＲと呼ぶ）について説明を行う。

【００４１】
前述のように、３ＤメッシュのＮｏＣは、拡張性を有する状態で複数の２Ｄ－ＮＮレイヤーを積み重ねて大規模ネットワークを作成するのに適している。ＳＮＮでは、通常、１つのニューロンが他の多くのニューロンに接続される。したがって、ニューラルプロセッシングコア間には、大量の１対多の通信が存在する。
【００４２】
本発明におけるルーティングアルゴリズムは、Ｋ－ｍｅａｎｓクラスタリング法とツリーベースのルーティングとの組み合わせに基づいている。ツリーベースのメカニズムは、マルチキャスト通信で使用される一般的な方法である。このルーティングメカニズムでは、宛先グループがソースノードから分割されて、パケットの「ツリー」ルーティングパスが形成される。ツリーベースの方法の主な欠点の１つは、中間ノードでパケットがブロックされる可能性が高いために、トラフィックが競合することである（非特許文献２２）。この問題に対処するために、本発明では、宛先セットをサブセットに分割するＫ－ｍｅａｎｓを採用している。Ｋ－ｍｅａｎｓの採用は、シナプス後ニューロンがしばしば互いに隣接しているという観察結果から得られている。従来の研究では、ＳＮＮのニューロン間通信の局所性が高いことを示している（非特許文献２３）。これにより、同じ領域内にあるニューロングループは、着信スパイクを共有することが可能になる。したがって、３Ｄ－ＮｏＣシステムにマッピングされると、ＳＮＮレイヤーのニューロンは、１つのコアまたは近くのコアに分散される。これにより、Ｋ－ｍｅａｎｓを最大限に活用して効果的なパーティションを取得し、トラフィック負荷のバランスを取るとともにＮｏＣシステムの高い輻輳を緩和することが可能になる。
【００４３】
したがって、図５のアルゴリズムに示すように、提案されたルーティング方法は、最初に宛先をいくつかのサブグループに分割する。これを行うために、提案されたルーティング方法では、Ｋ－ｍｅａｎｓクラスタリングメカニズムを採用して、サブセットの重心と、そのラベル付きの目的地を見つける。ここでの重心は、そのサブグループ内の他のすべてとの平均距離が最小のノードである。
【００４４】
アルゴリズムは、重心を決定するために、まず、利用可能なターゲットからランダムに重心を選択する。次に、アルゴリズムは、次のステップを計算する。
【００４５】
（１）図５のアルゴリズムの１０行目に示すように、各目的地から重心までの距離は、マンハッタン距離を使用して計算される。
【００４６】
（２）これらの距離に基づいて、目的地は、最も近い重心を持つサブグループに割り当てられる。
【００４７】
（３）最後に、サブグループが一時的に形成された後、すべての要素の平均を取ることにより、重心の位置が更新される。そして、これらの更新は、重心が変更されなくなるまで反復して行われる。
【００４８】
重心を決定した後、ソースノードからターゲットへのルーティングパスは、２段階によって形成される。第１段階では、一般的な方法である次元順序ルーティング（Ｄｉｍｅｎｓｉｏｎ　Ｏｒｄｅｒ　Ｒｏｕｔｉｎｇ：以下、ＤＯＲと呼ぶ）を使用して、各ソースから重心までのルートを決定する（非特許文献１１）。この点から、与えられたソースから重心への同じルートがマージされる。これにより、ユニキャストベースの方法と比較して、ソースから送信する必要があるスパイクパケットの数を減らすことが可能になる。ＸＹＺやＺＹＸ等のＤＯＲの特定のバリエーションを使用することは、図６に示す例においてさらに説明されているように、最適化されてバランスが取れたトラフィックを得るためのアプリケーションマッピング方法に依存する。なお、ＺＹＸは、Ｚ次元がルーティング計

算で最初に実行され、次にＹ、Ｘが実行されることを意味している。そして、この段階の終わりに、ソースから重心までの「ツリー」の一部が形成される。続いて、第２段階では、第１段階と同様のルーティング計算を行い、重心から目的地までの「ツリー」の他の部分を確立する。２つの段階の後、与えられた送信元ノードからその宛先への「ツリー」ルートが構築され、さらに、計算されたルーティング情報を使用することによって、ルータに接続されたルーティングテーブルが更新される。

【００４９】

次に、６×３×２の３ＤＮｏＣ－ＳＮＮシステムにマッピングされた１８×１８の完全に接続されたＳＮＮアプリケーションの例について説明を行う。ここでは、各スパイキングタイルがＳＮＰＣにおいて１つのニューロンを持っているものと仮定する。

【００５０】

図６に示すように、Ｌ１におけるタイル/ノード（ソースノード）は、Ｌ２におけるすべてのノード（宛先ノード）に出力を送信する。特定の場合において、レイヤーＬ１におけるソースノードであるノード３（以下、このような場合、ノード３を「３」と表記する）は、レイヤーＬ２におけるすべてのノードにスパイクパケットを送信する必要がある。クラスター数ｋが２の場合、宛先セットは、「２６」及び「２９」を重心とする２つのサブセットに分割される（図６（ａ））。次に、図６（ｂ）に示すように、ソースから両方の重心への「ツリー」ルートが決定される。このマッピング方法では、ＤＯＲにおけるＺＹＸバージョンが選択されている。これにより、スパイクが複数の層間リンクを通過するため、第１層の中間ノード（すなわち、「８」及び「１１」）のトラフィック競合を緩和できる。一方、ＸＹＺまたはＹＸＺのいずれかを使用する場合、Ｌ１におけるすべてのソースノードは、「１１」及び「８」を介してスパイクを重心（すなわち、「２６」及び「２９」）に送信する必要がある。そのため、「１１」と「２６」とのレイヤー間のリンク、及び、「８」と「２９」とのレイヤー間のリンクにおいて高いトラフィック輻輳が発生する。図６（ｃ）に示すように、重心から目的地へのルートが計算された後、ツリーの他の部分が形成される。最後に、図６（ｄ）に示すように、「３」からＬ２におけるそれぞれへのルーティング「ツリー」が形成される。

【００５１】

最適なクラスター数の選択：前述のように、クラスターの数（ｋ）は、提案されたルーティングアルゴリズム（ＫＭＣＲ）を適用する前に決定する必要がある。直感的に、ｋが小さい場合、宛先セットは、大きなサブセットに分割される。これにより、中間ノード（すなわち、重心）での輻輳が大きくなり、ネットワークの輻輳が大きくなる可能性がある。一方、ｋが大きい場合、各ソースノードは、与えられたパケットの複数のコピーを重心に送信できる。これにより、待ち時間が長くなる場合がある。ｋが宛先の数と等しい場合、本発明におけるルーティングアルゴリズムは、ユニキャストベースのマルチキャストのように動作する。ｋの選択は、主に宛先ノードの分布に依存することに言及することが重要である。幸いなことに、ｋの最適値を選択するために採用可能ないくつかの優れた観測が存在する。まず、前述のように、ＳＮＮには、高いニューロン間通信の局所性がある。これにより、同じグループ（レイヤー）内のニューロンが近くのニューラルプロセッシングコアにマッピングされる状況が発生する。これは、Ｋ－ｍｅａｎｓクラスタリングアルゴリズムを効率的に機能させることを可能にする。第二に、一般的なＳＮＮアプリケーションにおける宛先ノードの数は多くない。実際、レイヤー内のニューロンの数は、多層モデルに基づいた深層学習の場合、数百から数千であり、それぞれ数百のニューロン（ＳＮＰＣの場合は２５６のニューロン）を含めることが可能な数十のコアに収容することが可能である（非特許文献２４）。したがって、ＳＮＮアプリケーションをマッピングした後、宛先の分布を視覚化することによってクラスターの数を決定することが可能になる。ただし、特定の場合における最適なｋの値を選択するには、ｋ以外の異なる値によってパフォーマンスシステムを評価する必要がある。

【００５２】

上記の観察結果に基づいて、最適なｋは、次の２つのステップによって決定することが

可能である。

【００５３】

（１）ＳＮＮアプリケーションをマッピングした後、宛先セットを視覚化することによってクラスターの数を見つける。

【００５４】

（２）：ｋの値（（１）で見つけられたクラスターの数及び他のいくつかの値を含む）を変化させることによってシステムを評価し、最適なケースを選択する。

【００５５】

次に、マルチキャストルーティングアルゴリズムに基づく最短経路のＫ－ｍｅａｎｓクラスタリング（Ｓｈｏｒｔｅｓｔ　Ｐａｔｈ　Ｋ－ｍｅａｎｓ　Ｃｌｕｓｔｅｒｉｎｇ：以下、ＳＰ－ＫＭＣＲと呼ぶ）について説明を行う。

【００５６】

前述のように、ＫＭＣＲでは、送信元ノードがスパイクパケットを重心に送信し、次に重心がスパイクを宛先に送信する。重心の使用において、重心から目的地までの全体の距離が最小であることが保証される。ただし、これにより、異なるソースからのトラフィックが重心に集中するため、重心へのリンクにおいてトラフィックの輻輳が発生する可能性がある。

【００５７】

この問題に対処するために、本発明では、新しいルーティング方法の提案を行う。Ｋ－ｍｅａｎｓを採用することによって宛先サブセットを決定した後、本発明では、初めに、与えられたソースからサブセット内のすべてのノードまでのホップ数を計算する。次に、本発明では、各サブセットについて、ソースへの最短パスを持つノードを選択する。ＫＭＣＲの場合とは異なり、ソースは、重心ノードではなく各サブセットの最短パスノードにスパイクパケットを送信する。以下、この方法をＳＰ－ＫＭＣＲと呼ぶ。これにより、トラフィックの輻輳の潜在的な問題が解消され、平均遅延も削減される。なお、新しい方法では、ＫＭＣＲと比較した場合、最短パスを見つけるためにより多くの計算が必要になる。ただし、新しい方法とＫＭＣＲの両方の計算は、オフラインで実行される。したがって、実行時のオーバーヘッドは、両方のアルゴリズムで同じになる。

【００５８】

次に、マルチキャストルーティングアルゴリズムに基づくフォールトトレラントにおける最短経路のＫ－ｍｅａｎｓクラスタリング（Ｆａｕｌｔ－Ｔｏｌｅｒａｎｔ　Ｓｈｏｒｔｅｓｔ　Ｐａｔｈ　Ｋ－ｍｅａｎｓ　Ｃｌｕｓｔｅｒｉｎｇ：以下、ＦＴＳＰ－ＫＭＣＲと呼ぶ）について説明を行う。ＦＴＳＰ－ＫＭＣＲは、ＳＰ－ＫＭＣＲに基づいている。

【００５９】

ＦＴＳＰ－ＫＭＣＲの基本的な考え方は、次の通りである。

【００６０】

（１）与えられたソースノードからその宛先へのプライマリルーティングツリー及びバックアップルーティングブランチのオフライン計算が実行される。

【００６１】

（２）オフライン計算の後、ルーティングテーブルが構成される。

【００６２】

図７は、プライマリ及びバックアップルーティングブランチを示す図である。障害のあるプライマリブランチが検出された場合、事前に計画されたバックアップブランチが使用され、障害のあるリンクが回避される。ＳＰ－ＫＭＣＲメカニズムは、プライマリブランチ（実線）を計算するために使用される。一方、バックアップブランチは、プライマリブランチの代替ルートである。そして、検討中のルータ（すなわち、「ｓｏｎ」）のために、プライマリ接続において障害が発生した場合に使用されるバックアップブランチ（破線）が計算される。例えば、「ｆａｔｈｅｒ」と「ｓｏｎ」との間のプライマリ接続において障害がある場合（すなわち、pl_1）、bl_1及びbl_2は、「ｆａｔｈｅｒ」と「ｓ

ｏｎ」との間のトラフィックを維持するために使用されるバックアップブランチである。これは、ｐｌ$_2$とｐｌ$_1$との両方に障害がある場合においても同じである。

【００６３】

アルゴリズムでは、プライマリルートとバックアップルートの計算は重要な計算タスクである。これらの計算は、オフラインで実行される。これにより、提案されたルーティングアルゴリズムの実行時におけるオーバーヘッドを削減することが可能になり、ＳＮＮで発生する可能性があるタイミング違反を回避する。図８のアルゴリズムに示されているように、ソース及び宛先アドレス（Ｓ；Ｔ）及びサブセットの数（ｋ）は、入力として事前に定義され、出力部分は、各ソースから宛先へのプライマリツリー（Ｐ$_{pr}$）及びバックアップブランチ（Ｐ$_{bk}$）である。

【００６４】

その後、次の手順に従ってルーティングの計算が行われる。

【００６５】

ステップ１：６行目～１９行目に示すように、宛先アドレスから、Ｋ－ｍｅａｎｓを採用して宛先サブセットを決定する。

【００６６】

ステップ２：２０行目～２５行目に示すように、各ソースから各サブセットのノードまでの最短経路を見つける。

【００６７】

ステップ３：プライマリツリーの最初の部分は、ソースノードからＳＰノードまで形成されます。これは、ソースから各ＳＰノードへの次元順序ルーティング（ＤＯＲ）アルゴリズムを採用し、同じルートとマージすることにより行われる。次に、ＤＯＲの代替バリエーションを採用してバックアップブランチを計算し、バックアップブランチがプライマリルートから分離されることを保証する。例えば、プライマリツリーの形成においてＺＹＸのＤＯＲが使用されている場合、バックアップブランチには、ＹＺＸやＸＺＹ等の他のバリエーションのＤＯＲを使用する。

【００６８】

ステップ４：ステップ２と同じ計算に従って、ＳＰノードから同じグループに含まれるその宛先へのプライマリツリーの２番目の部分とバックアップブランチとを計算する。

【００６９】

なお、プライマリ及びバックアップルーティングパスのみがオフライン計算であることに注意が必要である。これらの計算結果は、ルータにおけるルーティングテーブルの構成に使用される。設定プロセスは、実行前のアプリケーションマッピング中に行われるため、シナプス強化（重み）が更新されるオンライン学習プロセスのカテゴリに影響しない。さらに、これにより、バックアップブランチの計算オーバーヘッドが、提案されたルーティングアルゴリズムの回復時間に影響を与えないことが保証され、システムに必要なハードウェアコストも削減される。

【００７０】

次に、障害管理アルゴリズムについて説明する。ルーティング情報が構成された後、図９に示すように、着信パケットを処理するために障害管理アルゴリズムが実装される。

【００７１】

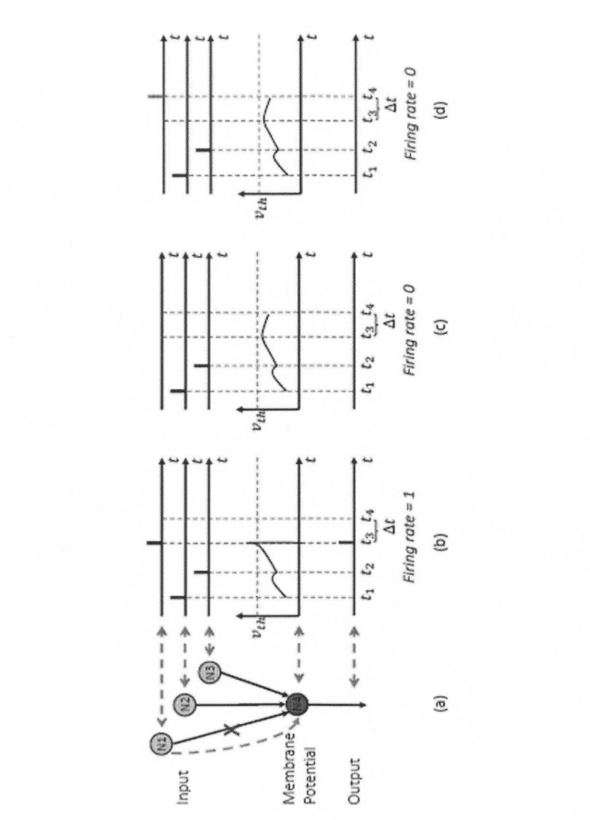
Ｓ１：与えられた着信パケットについて、そのパケットがプライマリブランチにあるかバックアップブランチにあるかを示すために、ｆａｕｌｔ＿ｆｌａｇ＿ｖａｌが抽出される。同時に、送信元アドレスは、予想されるプライマリ出力ポートの計算にも使用される。

【００７２】

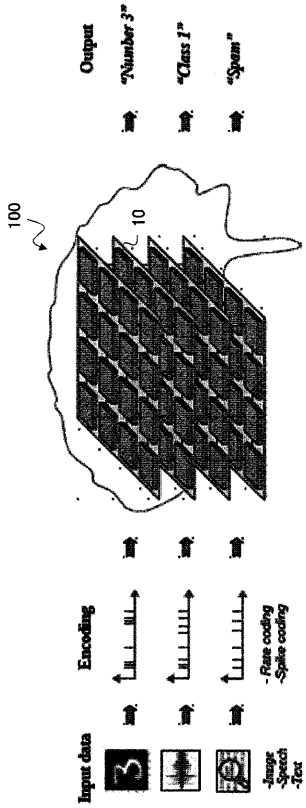
Ｓ２及びＳ３：ｆａｕｌｔ＿ｆｌａｇ＿ｖａｌ＝０である場合（すなわち、ルータが「ｆａｔｈｅｒ」または「ｇｒａｎｄｆａｔｈｅｒ」の役割を果たしている場合）、計算された出力ポートでは、各ルータに接続された障害検出器を使用することによって障害があるかどうかが判定される。

【００７３】
Ｓ４：予想される出力ポートが判定されると、転送する前においてｆａｕｌｔ＿ｆｌａｇ＿ｖａｌがパケットに付加される。
【００７４】
Ｓ５：これ以外の場合、出力ポートは、バックアップブランチを使用するように切り替えられ、このパケットがバックアップブランチ上にあることを次のバックアップルータに通知するために、ｆａｕｌｔ＿ｆｌａｇ＿ｖａｌに初期値（バックアップパスにおけるホップ数と等しい値）を設定する。
【００７５】
Ｓ６：ｆａｕｌｔ＿ｆｌａｇ＿ｖａｌ≠０である場合（すなわち、ルータの役割がバックアップまたは「ｓｏｎ」ルータである場合）、出力ポートは、バックアップルートを介してルーティングされ、さらに、ｆａｕｌｔ＿ｆｌａｇ＿ｖａｌは、０になってバックアップパスが終了するまで１ずつ減少される。
【符号の説明】
【００７６】
１：ＳＮＰＣ
２：ＦＴＭＣ－３ＤＲ
１０：スパイキングニューラルタイル
１００：３ＤＦＴ－ＳＮＮシステム

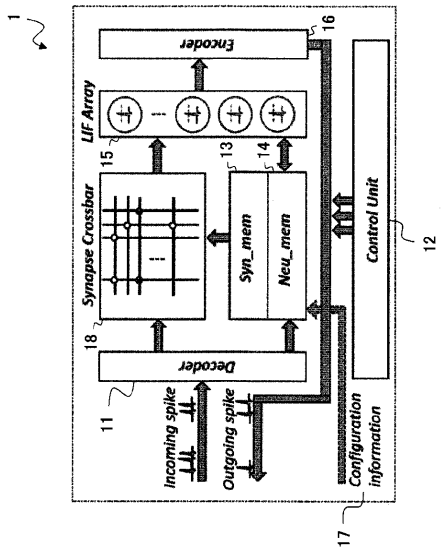
【図１】



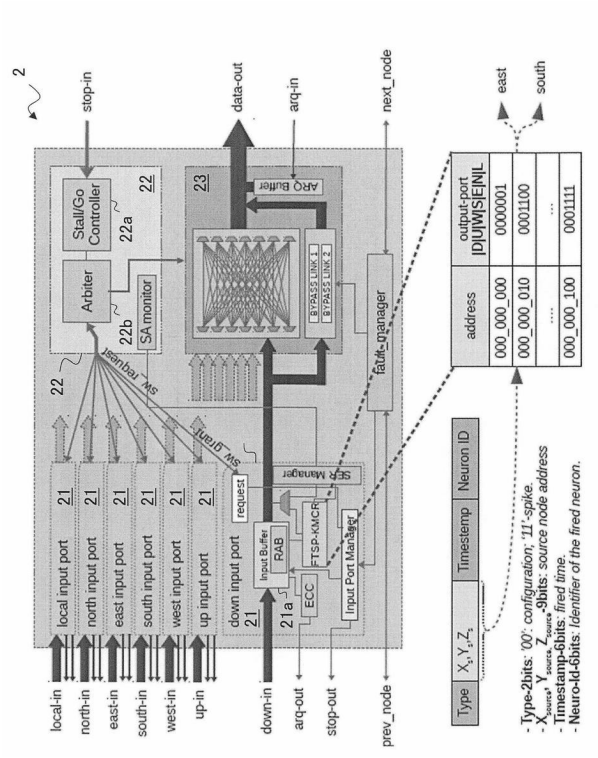
【図２】



【図３】



【図４】

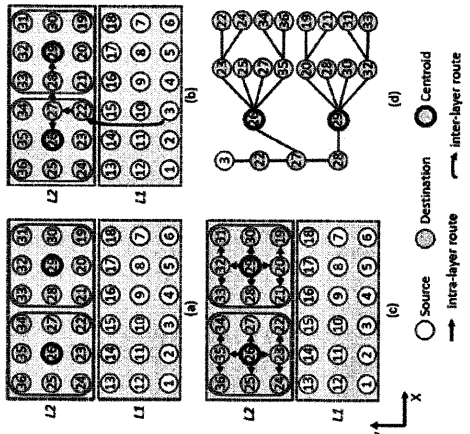


【図５】

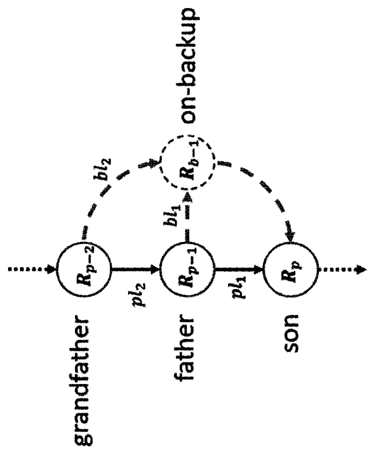
Algorithm 1: KMCR multicast routing pseudo-code

```
/* Input and output
Input // Source node address (S), destination node addresses (T), and the
      number of centroid nodes (k)
S = {s1(x1, y1, z1), s2(x2, y2, z2),...sn(xn, yn, zn)}
T = {t1(x1, y1, z1), t2(x2, y2, z2),...tm(xm, ym, zm)}
k
Output // Routing paths from S to T
P = {p(s1 → T),ps(s2 → T),...ps(sn → T)}
/* Centroid node assignment
// initial centroid nodes by randomly select from T */
foreach ci ∈ C do
    ci ← tj ∈ T
end
// Evaluate centroid nodes
do
    // Calculate the distance between ti ∈ T to cj ∈ C
    foreach ti ∈ T do
        d(ti, cj) = |xi − xj| + |yi − yj| + |zi − zj|
    end
    // Assign each destination to its centroid by
       minimum distance
    foreach ti ∈ T do
        I(ti) ← argmind(ci, tj)
    end
    // Update centroid
    foreach ci ∈ C do
        ci ← update(mean(ttj))
    end
while C != conv;
/* Creating routing path from each source to centroids
foreach si ∈ S do
    p(si, cj) ← DOR_based_tree(si, cj)
end
/* Creating routing tree from each centroid to its
   destinations                                        */
foreach ci ∈ C do
    p(ci, tj) ← DOR_based_tree(ci, tj)
end
```

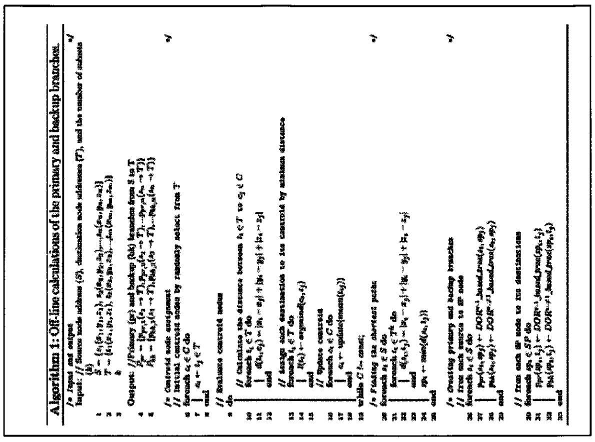
【図６】



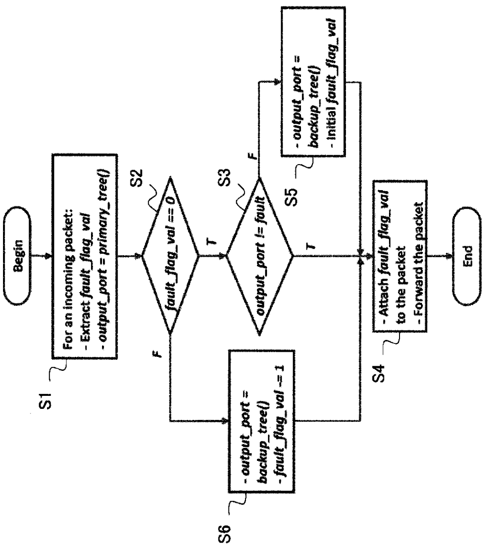
【図７】


grandfather
father
son
on-backup
R_{p-2}
R_{p-1}
R_p
R_{b-1}
pl_2
pl_1
bl_1
bl_2

【図８】


Algorithm 1: Off-line calculations of the primary and backup branches.

【図９】


Begin
For an incoming packet:
- Extract fault_flag_val
- output_port = primary_tree()
fault_flag_val == 0
output_port != fault
- output_port = backup_tree()
- Initial fault_flag_val
- output_port = backup_tree()
- fault_flag_val -= 1
- Attach fault_flag_val to the packet
- Forward the packet
End
T
F
S1
S2
S3
S4
S5
S6

フロントページの続き

特許法第３０条第２項適用　（１）平成３１年１月１日にｈｔｔｐ：／／ｗｅｂ－ｅｘｔ．ｕ－ａｉｚｕ．ａｃ．ｊｐ／▲～▼ｂｅｎａｂ／ａｌｌｐｒｏｊｅｃｔｓ．ｈｔｍｌにて発表。　（２）平成３１年３月１日にＴｈｅ　６ｔｈ　ＩＥＥＥ　Ｉｎｔｅｒｎａｔｉｏｎａｌ　Ｃｏｎｆｅｒｅｎｃｅ　ｏｎ　Ｂｉｇ　Ｄａｔａ　ａｎｄ　Ｓｍａｒｔ　Ｃｏｍｐｕｔｉｎｇ（ＢｉｇＣｏｍｐ　２０１９）にて発表。　（３）平成３１年３月１２日にｈｔｔｐｓ：／／ｗｗｗ．ｕ－ａｉｚｕ．ａｃ．ｊｐ／ｅｎ／ｉｎｆｏｒｍａｔｉｏｎ／ｉｅｅｅ．ｈｔｍｌにて発表。

(72)発明者　久田　雅之
福島県会津若松市東栄町１番７７号　株式会社会津コンピュータサイエンス研究所内

審査官　佐々木　洋

(56)参考文献　特表２０１９－５０５０６５（ＪＰ，Ａ）
特開２０１９－０９２０２０（ＪＰ，Ａ）
特開２０１５－１１９３８７（ＪＰ，Ａ）
国際公開第２０１９／０２６５２３（ＷＯ，Ａ１）
特開２０１０－１９９９７２（ＪＰ，Ａ）
特開２００７－２２１４２８（ＪＰ，Ａ）
特開平０７－２３９８３４（ＪＰ，Ａ）
米国特許出願公開第２００７／００９７９５１（ＵＳ，Ａ１）
中国特許出願公開第１０６１６１２７０（ＣＮ，Ａ）
大崎　功一　Koichi　OHSAKI，多対多マルチキャストにおけるメンバークラスタリング手法　A Member-Clustering Scheme for Many to Many Multicast，情報処理学会研究報告　Ｖｏｌ．２００１　Ｎｏ．５９　IPSJ SIG Notes，日本，社団法人情報処理学会　Information Processing Society of Japan，2001年06月08日，第2001巻，pp. 101-106

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｈ０４Ｌ　　１２／００－１２／６６
Ｈ０４Ｌ　　４１／００－１０１／６９５
Ｇ０６Ｎ　　３／０６３
Ｇ０６Ｎ　　３／０４