

(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7595014号
(P7595014)

(45)発行日 令和6年12月5日(2024.12.5)

(24)登録日 令和6年11月27日(2024.11.27)

(51)国際特許分類

F I

H O 4 L 41/0668(2022.01)

H O 4 L 41/0668

請求項の数 32 (全42頁)

(21)出願番号	特願2021-545309(P2021-545309)	(73)特許権者	521343220
(86)(22)出願日	令和2年2月3日(2020.2.3)		ヌオディービー インコーポレイテッド
(65)公表番号	特表2022-524931(P2022-524931 A)		アメリカ合衆国 マサチューセッツ州 0
(43)公表日	令和4年5月11日(2022.5.11)		2 1 4 0 ケンブリッジ ケンブリッジパ
(86)国際出願番号	PCT/US2020/016449	(74)代理人	100094569
(87)国際公開番号	WO2020/160557		弁理士 田中 伸一郎
(87)国際公開日	令和2年8月6日(2020.8.6)	(74)代理人	100103610
審査請求日	令和4年12月9日(2022.12.9)		弁理士 ▲吉▼田 和彦
(31)優先権主張番号	62/800,009	(74)代理人	100109070
(32)優先日	平成31年2月1日(2019.2.1)		弁理士 須田 洋之
(33)優先権主張国・地域又は機関	米国(US)	(74)代理人	100067013
			弁理士 大塚 文昭
		(74)代理人	100086771
			弁理士 西島 孝喜

最終頁に続く

(54)【発明の名称】 分散データベースにおけるノード障害の検出及び解決

(57)【特許請求の範囲】

【請求項1】

分散データベースにおける障害を解決する方法であって、前記分散データベースは複数のノードを含み、前記複数のノード内の各ノードは前記複数のノード内の他の各ノードに直接接続され、前記方法は、

前記複数のノード内の第1のノードにおいて、

前記複数のノード内の被疑ノードを識別することであって、前記被疑ノードは、前記障害の結果として前記第1のノードに接続されなくなった前記複数のノード内のノードである、前記識別することと、

第1の被疑ノードのリストを前記複数のノード内の隣接ノードにブロードキャストすることであって、前記第1の被疑ノードのリストは前記被疑ノードを含み、前記隣接ノードは、前記障害後に前記第1のノードに直接接続されたままである前記複数のノード内のノードである、前記ブロードキャストすることと、

前記隣接ノードのうちの少なくとも1つから第2の被疑ノードのリストを受信することと、

前記第1の被疑ノードのリスト及び前記第2の被疑ノードのリストに少なくとも部分的に基づいて、前記複数のノードの接続情報を決定することと、

前記接続情報に基づいて、前記第1のノードが前記分散データベースの勝利全接続コンポーネント内にあるかどうかを判定することであって、前記勝利全接続コンポーネントは、前記複数のノード内の前記ノードの半数超を含み、前記勝利全接続コンポーネント内の

各ノードは、前記勝利全接続コンポーネント内の他の各ノードに直接接続されている、前記判定することと、

前記第1のノードが前記複数のノードの前記勝利全接続コンポーネント内にあると判定したことに応答して、前記第1のノードの動作を継続させることと、

前記第1のノードが前記複数のノードの前記勝利全接続コンポーネント内にないと判定したことに応答して、前記障害を解決するために前記第1のノードを機能停止させることと、

を含む、前記方法。

【請求項2】

前記第1の被疑ノードのリストをブロードキャストすることは、前記第1のノードによって呼び出された前記方法の反復を示すプロトコル反復番号をブロードキャストすることを含む、請求項1に記載の方法。

【請求項3】

前記プロトコル反復番号を、前記第2の被疑ノードのリストと共に受信されたプロトコル反復番号と比較すること

をさらに含む、請求項2に記載の方法。

【請求項4】

マスターカタログの一部として前記プロトコル反復番号をシリアルライズすることであって、前記マスターカタログは、前記複数のノード内の前記ノードのリストを含む、前記シリアルライズすること

をさらに含む、請求項2に記載の方法。

【請求項5】

前記接続情報を決定することは、前記第1のノードにおいて、

前記接続情報に少なくとも部分的に基づいて接続グラフを決定することと、

前記接続グラフから前記勝利全接続コンポーネントを特定することと、

をさらに含む、請求項1に記載の方法。

【請求項6】

前記勝利全接続コンポーネントを特定することは、

前記勝利全接続コンポーネントのサイズ及び前記複数のノードのサイズに基づいて、前記勝利全接続コンポーネントを決定すること

を含む、請求項5に記載の方法。

【請求項7】

前記第1のノードが前記勝利全接続コンポーネント内にあるかどうかを判定することは、前記第1のノードにおいて、前記接続情報に基づいて前記勝利全接続コンポーネントを特定することを含む、請求項1に記載の方法。

【請求項8】

前記勝利全接続コンポーネントを特定することは、

前記接続情報に基づいて前記分散データベースの第1の全接続コンポーネントを特定することであって、前記第1の全接続コンポーネント内の各ノードは前記第1の全接続コンポーネント内の他の各ノードに直接接続されている、前記特定することと、

前記接続情報に基づいて前記分散データベースの第2の全接続コンポーネントを特定することであって、前記第2の全接続コンポーネントは前記第1の全接続コンポーネントとは異なり、前記第2の全接続コンポーネント内の各ノードは前記第2の全接続コンポーネント内の他の各ノードに直接接続されている、前記特定することと、

前記第1の全接続コンポーネントが、(i) 前記第2の全接続コンポーネントよりも多くのノードと、(ii) 前記複数のノード内の前記ノードの半数超と、を含むと判定することと、

前記第1の全接続コンポーネントを前記勝利全接続コンポーネントとして選択することと、

を含む、請求項7に記載の方法。

10

20

30

40

50

【請求項 9】

前記勝利全接続コンポーネントを特定することは、

前記接続情報に基づいて前記分散データベースの第 1 の全接続コンポーネントを特定することであって、前記第 1 の全接続コンポーネント内の各ノードは前記第 1 の全接続コンポーネント内の他の各ノードに直接接続されている、前記特定することと、

前記接続情報に基づいて前記分散データベースの第 2 の全接続コンポーネントを特定することであって、前記第 2 の全接続コンポーネントは前記第 1 の全接続コンポーネントとは異なり、前記第 2 の全接続コンポーネント内の各ノードは前記第 2 の全接続コンポーネント内の他の各ノードに直接接続されている、前記特定することと、

前記第 1 の全接続コンポーネントが、(i) 前記第 2 の全接続コンポーネントと同数のノードと、(ii) 前記複数のノード内の前記ノードの半数超と、を含むと判定することと、

10

前記第 1 の全接続コンポーネント及び前記第 2 の全接続コンポーネント内の前記ノードの一意の識別子に基づいて、前記第 1 の全接続コンポーネントを前記勝利全接続コンポーネントとして選択することと、

を含む、請求項 7 に記載の方法。

【請求項 10】

前記第 2 の被疑ノードのリストを前記第 1 のノードから少なくとも 1 つの隣接ノードに送信すること

をさらに含む、請求項 1 に記載の方法。

20

【請求項 11】

前記第 2 の被疑ノードのリストに少なくとも部分的に基づいて、前記第 1 の被疑ノードのリストを更新することと、

前記更新された第 1 の被疑ノードのリストを前記第 1 のノードから前記隣接ノードにブロードキャストすることと、

をさらに含む、請求項 1 に記載の方法。

【請求項 12】

前記第 1 のノードは、前記複数のノードのうちの半数未満を被疑ノードとして識別する、請求項 1 に記載の方法。

【請求項 13】

前記複数のノード内の第 3 のノードにおいて、前記複数のノードの半数超を被疑ノードとして識別することと、

30

前記障害を解決するために前記第 3 のノードを機能停止させることと、

をさらに含む、請求項 1 に記載の方法。

【請求項 14】

前記複数のノードに加入しようとしている第 3 のノードを機能停止させること

をさらに含む、請求項 1 に記載の方法。

【請求項 15】

前記障害は第 1 の障害であり、前記被疑ノードは第 1 の被疑ノードであり、前記複数のノード内の第 3 のノードにおいて、

40

前記分散データベースにおける第 2 の障害を検出することと、

前記複数のノード内の第 2 の被疑ノードを識別することであって、前記第 2 の被疑ノードは、前記第 2 の障害の結果として前記第 3 のノードに直接接続されなくなった前記複数のノード内のノードである、前記識別することと、

第 3 の被疑ノードのリストを前記第 1 のノードにブロードキャストすることと、

前記第 1 のノードによって前記方法を再開始することと、

をさらに含む、請求項 1 に記載の方法。

【請求項 16】

分散データベースにおける障害を解決するための方法であって、前記分散データベースは複数のノードを含み、前記複数のノード内の各ノードは前記複数のノード内の他の各ノ

50

ードに直接接続され、前記方法は、

前記複数のノード内の第1のノードにおいて、
前記分散データベースにおける障害によって引き起こされる、前記複数のノード内の第2のノードとの通信の中断を検出することと、

前記中断を検出したことに応答して、前記複数のノード内の隣接ノード間でそれぞれの被疑ノードのリストの協調的なブロードキャストを開始することであって、前記隣接ノードは、前記第1のノードに直接接続されたままである前記複数のノード内のノードであり、前記それぞれの被疑ノードのリストは、前記第2のノードを含む前記第1のノードの前記被疑ノードのリストであり、前記それぞれの被疑ノードのリストの被疑ノードのリストの各々は、少なくとも1つの他のノードに接続されなくなった少なくとも1つのノードを含む、前記開始することと、

10

前記それぞれの被疑ノードのリストに基づいて勝利全接続コンポーネントを識別することであって、前記勝利全接続コンポーネントは、前記複数のノード内の前記ノードの半数超を含み、前記勝利全接続コンポーネント内の各ノードは、前記勝利全接続コンポーネント内の他の各ノードに直接接続されている、前記識別することと、

前記第1のノードまたは前記第2のノードのうちの少なくとも1つが、前記勝利全接続コンポーネント内にないと判定することと、

前記第1のノードまたは前記第2のノードのうちの前記少なくとも1つを機能停止させることによって、前記障害を解決することと、

を含む、前記方法。

20

【請求項17】

各隣接ノードにおいて、

前記隣接ノードの被疑ノードのリストと、少なくとも1つの他の隣接ノードからの前記被疑ノードのリストとの比較を実行することと、

をさらに含む、請求項16に記載の方法。

【請求項18】

前記隣接ノードのそれぞれにおいて、前記それぞれの被疑ノードのリストに少なくとも部分的に基づいて接続グラフを用意すること

をさらに含む、請求項16に記載の方法。

【請求項19】

前記第1のノードまたは前記第2のノードのうちの少なくとも1つは、前記接続グラフに少なくとも部分的に基づいて自身を機能停止させる、請求項18に記載の方法。

30

【請求項20】

前記協調的なブロードキャストは、前記障害を検出するために前記第1のノードが呼び出した障害解決プロトコルを示すものを前記第1のノードによってブロードキャストすることを含む、請求項18に記載の方法。

【請求項21】

分散データベースにおける障害を解決するための方法であって、前記分散データベースは複数のノードを含み、前記複数のノード内の各ノードは前記複数のノード内の他の各ノードに接続され、前記方法は、

40

前記複数のノード内の第1のノードにおいて、

前記第1のノードが前記複数のノード内の前記ノードの少なくとも半数に直接接続されているかどうかを判定することと、

前記第1のノードが前記複数のノード内の前記ノードの半数未満に直接接続されていると判定したことに応答して、前記障害を少なくとも部分的に解決するために前記第1のノードを機能停止させることと、

前記第1のノードが前記複数のノード内の前記ノードの少なくとも半数に直接接続されていると判定したことに応答して、第1の被疑ノードのリストを前記複数のノード内の隣接ノードにブロードキャストすることであって、前記第1の被疑ノードのリストは前記第1のノードに直接接続されていないノードを含み、前記隣接ノードは、前記障害後に前記

50

第1のノードに直接接続されたままのノードである、前記ブロードキャストすることと、
前記隣接ノードのうちの少なくとも1つから第2の被疑ノードのリストを受信することと、

前記第1の被疑ノードのリストが前記第2の被疑ノードのリストと一致するかどうかを判定することと、

前記第1の被疑ノードのリストが前記第2の被疑ノードのリストと一致すると判定したことに応答して、前記障害の少なくとも部分的な解決において前記第1のノードを動作可能に維持することと、

前記第1の被疑ノードのリストが前記第2の被疑ノードのリストと一致しないと判定したことに応答して、前記第1の被疑ノードのリスト及び前記第2の被疑ノードのリストに基づいて第1の更新された被疑ノードのリストを前記隣接ノードにブロードキャストすることと、

10

前記隣接ノードのうちの少なくとも1つから少なくとも1つの第2の更新された被疑ノードのリストを受信することと、

前記第1の更新された被疑ノードのリスト及び前記第2の更新された被疑ノードのリストに少なくとも部分的に基づいて、前記複数のノードの接続情報を決定することと、

前記接続情報に基づいて前記分散データベースの勝利全接続コンポーネントを決定することであって、前記勝利全接続コンポーネントは前記複数のノード内の前記ノードの半数超を含み、前記勝利全接続コンポーネント内の各ノードは前記勝利全接続コンポーネント内の他の各ノードに直接接続されている、前記決定することと、

20

前記第1のノードが前記勝利全接続コンポーネント内にあるかどうかを判定することと、

前記第1のノードが前記複数のノードの前記勝利全接続コンポーネント内にあると判定したことに応答して、前記障害を少なくとも部分的に解決するために前記第1のノードの動作を継続させることと、

前記第1のノードが前記複数のノードの前記勝利全接続コンポーネント内にないと判定したことに応答して、前記障害を少なくとも部分的に解決するために前記第1のノードを機能停止させることと、

を含む、前記方法。

【請求項22】

分散データベースシステムであって、

30

複数のノードを備え、前記複数のノード内の各ノードは、対応するプロセッサ及び対応するメモリを含み、前記複数のノード内の他の全てのノードに直接接続され、前記複数のノード内の第1のノードの前記プロセッサは、前記分散データベースシステム内の障害を解決することを、

前記複数のノード内の被疑ノードを識別することであって、前記被疑ノードは、前記分散データベースシステム内の障害の結果として前記第1のノードに接続されなくなった前記複数のノード内のノードである、前記識別することと、

第1の被疑ノードのリストを前記複数のノード内の隣接ノードにブロードキャストすることであって、前記第1の被疑ノードのリストは前記被疑ノードを含み、前記隣接ノードは、前記障害後に前記第1のノードに直接接続されたままである前記複数のノード内のノードである、前記ブロードキャストすることと、

40

前記隣接ノードのうちの少なくとも1つから第2の被疑ノードのリストを受信することと、

前記第1の被疑ノードのリスト及び前記第2の被疑ノードのリストに少なくとも部分的に基づいて、前記複数のノードの接続情報を決定することと、

前記接続情報に基づいて、前記第1のノードが前記分散データベースシステムの勝利全接続コンポーネント内にあるかどうかを判定することであって、前記勝利全接続コンポーネントは、前記複数のノード内の前記ノードの半数超を含み、前記勝利全接続コンポーネント内の各ノードは、前記勝利全接続コンポーネント内の他の各ノードに直接接続されている、前記判定することと、

50

前記第 1 のノードが前記複数のノードの前記勝利全接続コンポーネント内にあると判定したことに応答して、前記第 1 のノードの動作を継続させることと、

前記第 1 のノードが前記複数のノードの前記勝利全接続コンポーネント内にないと判定したことに応答して、前記障害を解決するために前記第 1 のノードを機能停止させることと、

によって行うように構成される、前記分散データベースシステム。

【請求項 2 3】

前記第 1 のノードの前記プロセッサは、障害解決反復番号を前記第 1 の被疑ノードのリストと共にブロードキャストするように構成される、請求項 2 2 に記載の分散データベースシステム。

10

【請求項 2 4】

前記第 1 のノードの前記プロセッサは、前記障害解決反復番号を、前記第 2 の被疑ノードのリストと共に受信された障害解決反復番号と比較するように構成される、請求項 2 3 に記載の分散データベースシステム。

【請求項 2 5】

前記第 1 のノードの前記プロセッサは、マスターカタログの一部として前記障害解決反復番号をシリアルライズするようにさらに構成され、前記マスターカタログは、前記複数のノード内の前記ノードのリストを含む、請求項 2 3 に記載の分散データベースシステム。

【請求項 2 6】

前記第 1 のノードの前記プロセッサは、前記接続情報を決定することと、
前記接続情報に少なくとも部分的に基づいて接続グラフを決定することと、
前記接続グラフから前記勝利全接続コンポーネントを特定することと、
によって行うように構成される、請求項 2 2 に記載の分散データベースシステム。

20

【請求項 2 7】

前記勝利全接続コンポーネントを特定することは、
前記勝利全接続コンポーネントのサイズ及び前記複数のノードのサイズに基づいて、前記勝利全接続コンポーネントを決定すること
を含む、請求項 2 6 に記載の分散データベースシステム。

【請求項 2 8】

前記第 1 のノードが前記勝利全接続コンポーネント内にあるかどうかを判定することは、
前記接続情報に基づいて前記勝利全接続コンポーネントを特定することを含む、請求項 2 2 に記載の分散データベースシステム。

30

【請求項 2 9】

前記勝利全接続コンポーネントを特定することは、
前記接続情報に基づいて前記分散データベースシステムの第 1 の全接続コンポーネントを特定することであって、前記第 1 の全接続コンポーネント内の各ノードは前記第 1 の全接続コンポーネント内の他の各ノードに直接接続されている、前記特定することと、

前記接続情報に基づいて前記分散データベースシステムの第 2 の全接続コンポーネントを特定することであって、前記第 2 の全接続コンポーネントは前記第 1 の全接続コンポーネントとは異なり、前記第 2 の全接続コンポーネント内の各ノードは前記第 2 の全接続コンポーネント内の他の各ノードに直接接続されている、前記特定することと、

40

前記第 1 の全接続コンポーネントが、(i) 前記第 2 の全接続コンポーネントよりも多くのノードと、(i i) 前記複数のノード内の前記ノードの半数超と、を含むと判定することと、

前記第 1 の全接続コンポーネントを前記勝利全接続コンポーネントとして選択することと、

を含む、請求項 2 8 に記載の分散データベースシステム。

【請求項 3 0】

前記勝利全接続コンポーネントを特定することは、
前記接続情報に基づいて前記分散データベースシステムの第 1 の全接続コンポーネント

50

を特定することであって、前記第1の全接続コンポーネント内の各ノードは前記第1の全接続コンポーネント内の他の各ノードに直接接続されている、前記特定することと、

前記接続情報に基づいて前記分散データベースシステムの第2の全接続コンポーネントを特定することであって、前記第2の全接続コンポーネントは前記第1の全接続コンポーネントとは異なり、前記第2の全接続コンポーネント内の各ノードは前記第2の全接続コンポーネント内の他の各ノードに直接接続されている、前記特定することと、

前記第1の全接続コンポーネントが、(i)前記第2の全接続コンポーネントと同数のノードと、(ii)前記複数のノード内の前記ノードの半数超と、を含むと判定することと、

前記第1の全接続コンポーネント及び前記第2の全接続コンポーネント内の前記ノードの一意の識別子に基づいて、前記第1の全接続コンポーネントを前記勝利全接続コンポーネントとして選択することと、

を含む、請求項28に記載の分散データベースシステム。

【請求項31】

前記第1のノードの前記プロセッサは、前記分散データベースシステム内の前記障害を解決することを、

前記第2の被疑ノードのリストを前記第1のノードから少なくとも1つの隣接ノードに送信することによって行うようにさらに構成される、請求項22に記載の分散データベースシステム。

【請求項32】

前記第1のノードの前記プロセッサは、前記分散データベースシステム内の前記障害を解決することを、

前記第2の被疑ノードのリストに少なくとも部分的に基づいて、前記第1の被疑ノードのリストを更新することと、

前記更新された第1の被疑ノードのリストを前記第1のノードから前記隣接ノードにブロードキャストすることと、

によって行うようにさらに構成される、請求項22に記載の分散データベースシステム。

【発明の詳細な説明】

【技術分野】

【0001】

関連出願の相互参照

本出願は、2019年2月1日に出願された「Node Failure Detection and Resolution」と題された米国出願第62/800,009号の、米国特許法第119条(e)に基づく優先権の利益を主張し、その開示は全体が引用により本明細書に組み込まれている。

【背景技術】

【0002】

分散データベースにおけるデータ及びメタデータは、相互に通信する複数のノードにわたって記憶される。しかしながら、ノード間で通信の中断が発生することがある。たとえば、分散データベースシステム内のノードは、自身が不整合状態にあることに気付いて、クラッシュまたは機能停止し得る。他の例では、分散データベースシステム内のノード上で動作している仮想マシンまたはプロセスが、クラッシュまたは機能停止し得る。さらに他の例では、分散データベースシステム内の第1のノード及び第2のノードの間の通信リンクに障害が発生し得る。たとえば、分散データベースシステム内の2つ以上のノードを接続するネットワーク（たとえば、ローカルエリアネットワーク、ワイドエリアネットワーク、イーサネットなど）に障害が発生して、ノード間の通信が中断され得る。

【発明の概要】

【0003】

本明細書では、分散データベースシステムについて説明する。分散データベースシステムは、複数のノードを含むことができる。複数のノード内の各ノードは、対応するプロセ

10

20

30

40

50

ッサ及び対応するメモリを含むことができる。複数のノード内の各ノードは、複数のノード内の他の全てのノードに接続することができる。複数のノード内の第1のノードのプロセッサは、分散データベースシステム内の障害を解決することを、複数のノード内の被疑ノードを識別することと、第1の被疑ノードのリストを複数のノード内の隣接ノードにブロードキャストすることと、少なくとも1つの他の隣接ノードから第2の被疑ノードのリストを受信することと、接続情報に基づいて、第1のノードが分散データベースの勝利全接続コンポーネント (winning fully connected component) 内にあるかどうかを判定することと、第1のノードが複数のノードの勝利全接続コンポーネント内にあると判定したことに応答して、第1のノードの動作を継続させることと、第1のノードが複数のノードの勝利全接続コンポーネント内にはないと判定したことに応答して、障害を解決するために第1のノードを機能停止させることと、によって行うように構成することができる。被疑ノードは、分散データベースシステム内の障害の結果として第1のノードに接続されなくなった複数のノード内のノードとすることができる。第1の被疑ノードのリストは被疑ノードを含むことができる。隣接ノードは、ネットワーク障害後に第1のノードに直接接続されたままである複数のノード内のノードとすることができる。勝利全接続コンポーネントは、複数のノード内のノードの半数超を含むことができ、勝利全接続コンポーネントノード内の各ノードは、勝利全接続コンポーネントノード内の他の各ノードに直接接続されている。

【0004】

本明細書では、分散データベースにおける障害を解決するための方法について説明する。分散データベースは複数のノードを含むことができ、複数のノード内の各ノードは複数のノード内の他の各ノードに直接接続することができる。この方法は、複数のノード内の第1のノードにおいて、複数のノード内の第2のノードとの通信の中断を検出することと、中断を検出したことに応答して、複数のノード内の隣接ノード間でそれぞれの被疑ノードのリストの協調的なブロードキャストを開始することと、それぞれの被疑ノードのリストに基づいて接続情報を決定することと、接続情報に少なくとも部分的に基づいて障害を解決することと、を含むことができる。隣接ノードは、第1のノードに直接接続されたままである複数のノード内のノードとすることができる。第1のノードの被疑ノードのリストは第2のノードを含む。

【0005】

本明細書では、分散データベースにおける障害を解決するための方法について説明する。分散データベースは複数のノードを含むことができ、複数のノード内の各ノードは複数のノード内の他の各ノードに接続することができる。この方法は、障害を検出したことに応答して、複数のノード内の第1のノードにおいて、第1のノードが複数のノード内のノードの少なくとも半数に接続されているかどうかを判定することと、第1のノードが複数のノード内のノードの半数未満に直接接続されていると判定したことに応答して、障害を少なくとも部分的に解決するために第1のノードを機能停止させることと、第1のノードが複数のノード内のノードの少なくとも半数に直接接続されていると判定したことに応答して、第1の被疑ノードのリストを複数のノード内の隣接ノードにブロードキャストすることと、隣接ノードのうちの少なくとも1つから第2の被疑ノードのリストを受信することと、第1の被疑ノードのリストが第2の被疑ノードのリストと一致するかどうかを判定することと、第1の被疑ノードのリストが第2の被疑ノードのリストと一致すると判定したことに応答して、障害の少なくとも部分的な解決において第1のノードを動作可能に維持することと、第1の被疑ノードのリストが第2の被疑ノードのリストと一致しないと判定したことに応答して、第1の被疑ノードのリスト及び第2の被疑ノードのリストに基づいて第1の更新された被疑ノードのリストを隣接ノードにブロードキャストすることと、隣接ノードのうちの少なくとも1つから少なくとも1つの第2の更新された被疑ノードのリストを受信することと、第1の更新された被疑ノードのリスト及び第2の更新された被疑ノードのリストに少なくとも部分的に基づいて、複数のノードの接続情報を決定することと、接続情報に基づいて分散データベースの勝利全接続コンポーネントを決定すること

10

20

30

40

50

と、第1のノードが勝利全接続コンポーネント内にあるかどうかを判定することと、第1のノードが複数のノードの勝利全接続コンポーネント内にあると判定したことに応答して、障害を少なくとも部分的に解決するために第1のノードの動作を継続させることと、第1のノードが複数のノードの勝利全接続コンポーネント内にないと判定したことに応答して、障害を少なくとも部分的に解決するために第1のノードを機能停止させることと、を含むことができる。第1の被疑ノードのリストは第1のノードに直接接続されていないノードを含むことができる。隣接ノードは、障害後に第1のノードに直接接続されたままのノードとすることができる。勝利全接続コンポーネントは複数のノード内のノードの半数超を含み、勝利全接続コンポーネント内の各ノードは勝利全接続コンポーネント内の他の各ノードに直接接続されている。

10

【0006】

前述の概念及びさらなる概念の全ての組み合わせについて、以下でより詳細に論じ（ただし、そのような概念が相互に矛盾していない場合に限る）、それらは本明細書で開示した本発明の主題の一部である。詳細には、本開示の最後にある特許請求した主題の全ての組み合わせは、本明細書で開示した本発明の主題の一部である。引用により組み込まれているあらゆる開示にも登場し得る、本明細書で使用する用語には、本明細書で開示した特定の概念に最も一致する意味が与えられるものとする。

【0007】

当業者であれば理解するように、図面は主に説明を目的としており、本明細書で説明する本発明の主題の範囲を限定することは意図していない。図面は必ずしも一定の比率ではなく、場合によっては、異なる特徴の理解を容易にするために、本明細書で開示した本発明の主題の種々の態様を図面内で誇張または拡大して示す場合がある。図面では、同様の参照文字は通常、同様の特徴（たとえば、機能的に同様及び／または構造的に同様の要素）を指す。

20

【図面の簡単な説明】

【0008】

【図1】ネットワーク障害を解決して分散データベース内のノード間の完全な接続を修復する処理を示す図である。

【図2】ネットワーク分断イベントによって分散データベースシステムが全接続されたノードの2つの非連結グループに分割される、分散データベースシステムにおけるネットワーク障害の典型的なケースの一例を示す図である。

30

【図3】図1に示す処理によって解決することができる部分接続を有する3つのノードの例示的な分散データベースシステムを示す図である。

【図4】図1に示す処理によって解決することができる2つのリンク障害を有する5つのノードの例示的な分散データベースシステムを示す図である。

【図5】図1に示す処理によって解決することができる4つのリンク障害を有する5つのノードの例示的な分散データベースシステムを示す図である。

【図6】図5に示す部分接続のケースの一例を示す図である。

【図7】図1に示す処理によって解決することができる3つのリンク障害を有する5つのノードの例示的な分散データベースシステムを示す図である。

40

【図8】図1に示す処理によって解決することができる5つのリンク障害を有する5つのノードの例示的な分散データベースシステムを示す図である。

【図9】図1に示す処理によって解決することができる単方向リンク障害を有する特殊な部分接続の例示的なケースを示す図である。

【図10】図1に示す処理によって解決することができるメンバーシップ変更中のネットワーク障害の例示的なケースを示す図である。

【図11】図1に示す処理の一例を示す図である。

【図12】ネットワーク障害を解決する拡張された処理を示すフローチャートである。

【図13】分散データベースシステムにおいてネットワーク分断によって新しいノード及びエントリノードが残りのノードから分離されるとき、分散データベースシステムにお

50

けるメンバーシップ変更を示す図である。

【図 1 4】分散データベースシステムにおいてネットワーク分断によって新しいノード及びエントリノードが残りのノードから分離されるとき、図 1 3 のシナリオの変形例の図である。

【図 1 5】分散データベースシステムにおいてネットワーク分断によって新しいノード、エントリノード、及び一部のピアが残りのピアから分離されるとき、分散データベースシステムにおけるメンバーシップ変更を示す図である。

【図 1 6】分散データベースシステムにおいてネットワーク分断によってエントリノードが残りのノードから分離されるとき、分散データベースシステムにおけるメンバーシップ変更を示す図である。

10

【図 1 7】分散データベースシステムにおいてネットワーク分断によって新しいノード、エントリノード、及び一部のピアが残りのノードから分離されるとき、分散データベースシステムにおけるメンバーシップ変更を示す図である。

【図 1 8】分散データベースシステムにおいてネットワーク分断によって新しいノード、エントリノード、及び一部のピアが残りのノードから分離されるとき、分散データベースシステムにおけるメンバーシップ変更の他の図である。

【図 1 9】ノードが障害検出メッセージを交換してネットワーク障害イベントを解決することを示す図である。

【図 2 0】障害検出メッセージを交換しながらノード障害に対処することを示す図である。

【図 2 1】フェイルオーバーを実行しながらノード障害に対処することを示す図である。

20

【発明を実施するための形態】

【0 0 0 9】

分散データベースシステムは、分散データベースのデータ及び／またはメタデータのフラグメントを記憶する複数のノードを含む。分散データベースシステム内の全てのノードは、相互に通信できるように相互に直接接続される。しかしながら、分散データベースシステム内の1つまたは複数のノードが、ネットワーク障害が原因で通信の中断を被る場合があり得る。これらの通信の中断は、2つ以上のノード間の通信リンクの障害、あるいは1つまたは複数のノードの障害が原因であり得る。これらの障害は、以下でさらに詳細に説明するように、まだ相互に直接接続されているノードを識別し、直接接続されたノードの最大のグループを特定し、そのグループの一部ではないノードを機能停止させることによって、解決することができる。

30

【0 0 1 0】

分散データベースシステム

分散データベースシステムは、2タイプのノード、すなわち、分散データベースへのユーザアクセスを提供するトランザクションエンジン（TE）ノードと、分散データベース全体のそれぞれのディスクアーカイブを維持するストレージマネージャ（SM）ノードとを含むことができる。各ストレージマネージャノードは通常、分散データベース全体のコピーを記憶するが、単一のトランザクションエンジンノードは、そのトランザクションエンジンノードでそのときに実行されているトランザクションをサポートするために必要な分散データベースの一部のみを含み得る。

40

【0 0 1 1】

分散データベースシステム内の各ノードは、独自のプロセッサ、メモリ、及び通信インターフェース（複数可）を有し、データベースシステムネットワークを介して分散データベースシステム内の他の全てのノードと直接通信することができる。任意の2つのノード間の通信は、シリアルライズされたメッセージの送信を含むことができる。シリアルライズされたメッセージは、伝送制御プロトコル（TCP）または他の任意の適切なメッセージングプロトコルに従うことができる。

【0 0 1 2】

分散データベースシステム内の各ノードは、一意の識別子（たとえば、辞書式ID）を有し、他の全てのノードのリストを一意の識別子の順に分散データベースシステムに記憶

50

する。各ノードはこのリストを使用して、分散データベースシステム内の全てのトランザクションエンジンノード及びストレージマネージャノードのステータスを追跡する。さらに、各ノードは、全てのデータベーストランザクションと、全てのデータベースレコードの場所（すなわち、どのノードがどのデータフラグメントを記憶しているか）とを追跡し得る。ノードは、このノード及びトランザクション情報をマスターカタログのそれぞれのコピーに記憶し得、マスターカタログは、分散データベースシステムに関するメタデータを含み、データベース内の全てのノードに複製される。新しいノードは、分散データベースシステムに加入する場合に、エントリノードと呼ばれる他のノードからマスターカタログのコピーを受信する。

【0013】

データベーストランザクションとデータベースフラグメントの場所とを追跡することは、分散データベースシステムが、分散データベース内のデータの正確性、完全性、及び整合性を確保するために、一般にACIDプロパティとして知られている、不可分性、一貫性、独立性、及び永続性を維持するのに役立つ。

【0014】

ネットワーク障害及び障害検出

分散データベースシステム内の各ノードは、「ハートビート」メッセージを分散データベースシステム内の他の全てのノードに頻繁な間隔で送信する。たとえば、各ノードは毎秒または数秒ごとに他の全てのノードにハートビートメッセージを送信する。（任意選択により、ハートビートメッセージを受信したノードは、ハートビートメッセージを送信したノードに確認応答メッセージを送信することができる。）通信の中断がない場合、分散データベースシステム内の全てのノードは、分散データベースシステム内の他の全てのノードにハートビートメッセージを直接送信し、これらからハートビートを直接受信し続ける。しかしながら、ネットワーク障害によって、そのような通信が中断され得る。通信の中断を検出したノード（たとえば、他のノードから所定の時間内にハートビートメッセージを受信していないもの）は、ネットワーク障害を解決するために障害解決プロトコルを開始する。

【0015】

ネットワーク障害の解決

ここで示す障害解決処理では、分散データベース内のノードは、ネットワーク障害に応じて自身らを再グループ化し、自身らが、辞書式IDの順位が最も先の、多数派のサイズの、最大の全接続されたノードグループの一部でない場合に、自身らを機能停止させる。最大の全接続されたグループが分散データベースシステム内の半数未満のノードを含む場合、全てのノードが自身らを機能停止させ得る。切断または部分接続されたノードを機能停止させることにより、データベースの一部または全部が無効になり得る可能性が低くなる。これらの障害解決処理は、リーダーなしの方法で、進行中のデータベーストランザクションをブロックまたは中止することなく、実行することができる。

【0016】

図1は、ネットワーク障害を解決する処理100を示している。分散データベースシステム内の任意のノードが、ネットワーク障害（たとえば、所定の期間内の他のノードからのハートビートメッセージの受信の失敗）を検出したことに応答して、この処理100を開始することができる。102において、第1のノードはネットワーク障害を検出し、「被疑ノード」、すなわち、第1のノードが障害の発生を疑っているノードのリストを作成することによって、障害解決処理100を開始する。たとえば、第1のノードの被疑リストは、（a）第1のノードが所定のタイムアウト間隔（たとえば、ping Time out秒）内にそれらのノードからハートビートメッセージを受信していないという条件、及び（b）オペレーティングシステムが第1のノード及びその他のノード（複数可）の間の接続（複数可）を閉じたという条件、の一方または両方を満たすノードのリストである。この時点で、第1のノードの被疑リストが分散データベースシステム内の他の全てのノードを含む場合、第1のノードはネットワーク障害を少なくとも部分的に解決するために、

10

20

30

40

50

自身を機能停止させ得る。

【0017】

104において、第1のノード（すなわち、処理100を開始したノード）は、自身の被疑ノードリストを自身の隣接ノードにブロードキャストし、隣接ノードは、ネットワーク障害後に第1のノードがまだ直接通信することができるノードである。（ネットワーク障害がない場合、分散データベースにおいて、全てのノードは他の全てのノードの隣接機（neighbor）である。）隣接ノードはこの被疑リストを受信し、自身らの被疑リストを自身らの隣接機にブロードキャストする。隣接ノードの被疑リストは、ネットワーク障害の性質に応じて、第1のノードの被疑リストと同一であり得、または異なり得る。

【0018】

106において、第1のノードは、自身の隣接ノードから被疑リストを受信し、それら及び自身の被疑リストを使用して、接続グラフを構築する。接続グラフは、分散データベースシステム内のどのノードに第1のノードが実際に直接接続されているか（すなわち、どのノードが実際に第1のノードの隣接ノードであるか）を示す。その他のノードも接続グラフを構築する。ネットワーク障害の性質に応じて、これらの接続グラフは、第1のノードの接続グラフと同じであり得、または異なり得る。同様に、各接続グラフは、対応するノードの被疑リストを補うものであり得る。

【0019】

各ノードは、自身の接続グラフを使用して、ネットワーク障害後に相互に直接接続されたままのノードのグループを識別する。直接接続されたノードの各グループを、「全接続コンポーネント」と呼ぶ。全接続コンポーネント内では、各ノードは、ネットワーク障害後も全接続コンポーネント内の他の全てのノードとの通信を継続する。各ノードは、分散データベースシステム内で全接続コンポーネントを特定すると、自身が「勝利全接続コンポーネント」の一部であるかどうかを判定する（110）。自身が全接続コンポーネントの一部でない場合、各ノードはネットワーク障害を解決するために自身を機能停止させる（112）。自身が勝利全接続コンポーネントの一部である場合、動作を継続する（114）。

【0020】

勝利全接続コンポーネントは、データベース内の全てのデータを含むことができるが、必須ではない（たとえば、ストレージマネージャノードを含む必要はない）。この手順は、勝利全接続コンポーネントを形成するノードのタイプを考慮しない。（ただし、場合によっては、この処理は、勝利全接続コンポーネントを決定する場合に、全接続コンポーネント内のノードのタイプに注意を払うように修正することができる。）勝利全接続コンポーネントが分散データベース内の全てのデータを含まない場合、ユーザは適切な動作を確保するために介入し得る。

【0021】

各ノードは、自身が勝利全接続コンポーネントの一部であるかどうかを次のように判定することができる。まず、各ノードは、自身の接続グラフに基づいて、自身が全接続コンポーネントの一部であるかどうかを判定し得る。一部でない場合、自身を機能停止させる。しかしながら、ノードが全接続コンポーネント（または場合によっては2つ以上の全接続コンポーネント）の一部である場合、ノードは自身の接続グラフに基づいて自身の全接続コンポーネント（複数可）のサイズを特定する。（自身の接続グラフと、各ノードが分散データベースシステム内のその他のノードに関して記憶している情報とに基づいて）自身が最大の全接続コンポーネントの一部ではないとノードが判定した場合、ノードは自身を機能停止させる（112）。ノードが最大の全接続コンポーネントの一部であって、その全接続コンポーネントが、ネットワーク障害前の分散データベースシステム内のノードの総数の半数超を含む場合、ノードは動作可能なままとなる（114）。この全接続コンポーネントを、「勝利全接続コンポーネント」と呼び、その理由は、障害解決処理100の終了時に、分散データベースシステム内の全ての動作可能なノードを含むためである。

【0022】

同じサイズで、それぞれが分散データベース内のノードの半数超を有し、他の全ての全接続コンポーネントよりも大きい2つ以上の全接続コンポーネントが存在するとノードが判定した場合、勝利全接続コンポーネントを特定するためのタイブレーク処理を実施する。タイブレーク処理は、各全接続コンポーネント内のノードを、ノードの一意の識別子の順にソートすることを含み得る。一意の識別子がソートされると、ノードは一意の識別子の辞書式順序に基づいて、勝利全接続コンポーネントを選ぶ。たとえば、ノードは、共通プレフィックスに続くノードIDが最小の全接続コンポーネントを、勝利全接続コンポーネントとして選び得る。

【0023】

他の障害解決処理に対する技術的利点

図1に示す障害解決処理は、分散データベースにおける障害を解決するための他の処理に比べていくつかの相違点及び利点を有する。初めに、ブロッキング処理とは異なり、図1に示す障害解決処理は、ネットワーク障害後に分散データベース内の1つまたは複数のノードを排除して、完全な全接続を修復する。ブロッキングは、分散データベース内のデータに対して行われた更新をロールバックし得るので、望ましくない。他の方法論とは異なり、本明細書に記載の処理は、いかなる種類のブロッキングメカニズムも含まない。

【0024】

さらに、図1に示す障害解決処理は、リーダーノードを必要とせず、使用もしない。逆に、分散データベースにおける障害を解決するための他の方法論は、強力なリーダーシップモデルを実装する。基本的に、この方法論はリーダーノードを使用して障害解決の判断を行う。このリーダーベースの方法論とは異なり、本明細書に記載の処理は、障害解決の判断を行うリーダーノードを有さない。代わりに、図1に関して上述したように、任意のノードが障害解決処理を開始することができ、各ノードは、処理の一部として、リーダーノードからの命令なしに、自身を機能停止させるべきか、動作可能なままにすべきかを判定する。

【0025】

ブロッキング及びリーダーベースの障害解決処理とは異なり、本明細書に記載の非ブロッキングのリーダーなしの障害解決処理は、部分接続のネットワーク障害に一貫した方法で対処することができる。部分接続のネットワーク障害では、分散データベースシステム内のネットワーク分断によって、ノードまたはノードのセットが、分散データベースシステム内のノードのサブセットとしか通信しないようになり得る。部分接続のケースに対処するために、他の処理は、ローテーション式リーダーモデルを適用して、リーダー及び情報提供機(informer)に明示的なメッセージ確認応答を使用させる。場合によっては、通信の中断を被っているノード間でリーダーシップが絶えずシフトして、ネットワーク障害の解決が(場合によっては無期限に)遅れる可能性がある。

【0026】

ネットワーク障害の様々なケース

処理100は、ネットワーク障害イベントの後に、2つ以上の非連結ノードグループ(すなわち、異なる全接続コンポーネント)の稼働を継続させない。瑣末な解決策(たとえば、全てのノードを機能停止させる)を回避するために、処理100は、可能な場合には、単一のノードグループが稼働を継続できるようにする。

【0027】

さらに、ユーザが分散データベースシステム内の生き残ったノードの半数以上をシャットダウンすることを選択した場合、処理100は必ずしも残りのノードを機能停止させなくてもよい。処理100はまた、リンク障害に加えて、低速リンク(すなわち、接続が低速な2つ以上のノード間の通信経路)に対処することができる。換言すれば、処理100は、低速リンク及びリンク障害を同じ方法で処理する。

【0028】

図2～図10は、図1の処理100を使用して解決することができる様々なタイプのネットワーク障害を示している。

10

20

30

40

50

【0029】

ケースA：図2は、ネットワーク分断イベントによって分散データベースシステム200が全接続されたノードの2つ以上の非連結グループに分割される典型的な障害のケースを示している。図2の左側に見られるように、分散データベースシステム200は、3つのトランザクションエンジンノードTE1、TE2、及びTE3と、2つのストレージマネージャノードSM1及びSM2とを含み、これらは全て相互に接続されている。これらのノードは、それぞれの通信リンク212a～212jを介して相互に通信し、TE1はリンク212aを介してTE2と通信し、TE2はリンク212dを介してTE3と通信し、TE3はリンク212eを介してSM2と通信し、SM2はリンク212fを介してSM1と通信し、TE2はリンク212bを介してSM1と通信し、TE2はリンク212cを介してSM1と通信し、TE1はリンク212hを介してSM2と通信し、TE3はリンク212jを介してSM1と通信し、TE2はリンク212iを介してSM2と通信する。

10

【0030】

図2の中央では、ネットワーク分断によって、コーラスが2つの非連結ノードグループ（2つの全接続コンポーネント202'及び202''）に分割されている。（コーラスまたはコーラスグループとは、分散データベースシステム内の全てのノードのセットである。）この例では、第1の全接続コンポーネント202'は、{TE1, TE2, SM1}を含み、第2の全接続コンポーネント202''は、{TE3, SM2}を含む。次に、処理100は、第1の全接続コンポーネント202'が、第2の全接続コンポーネント202''よりも大きく、分散データベース200内のノードの半数超を含むので、勝利全接続コンポーネント204'であると判断する。勝利全接続コンポーネント204'内のノード{TE1, TE2, SM1}は稼働を継続し、ノードTE3及びSM2は、自身らが勝利全接続コンポーネント204'内にないと気づいたことに応答して自身らを機能停止させる。

20

【0031】

ケースB：図3～図8は、部分接続の様々な例を示している。これらの各例では、ネットワーク分断または（双方向）リンク障害（複数可）によって、ノードまたはノードのセットが分散データベースシステム内の他のノードのサブセットのみと通信ようになる。部分接続のケースでは、ノード間の接続は推移性（transitive property）を満たさず、すなわち、たとえば、ノードTE1はノードTE2と直接通信することが可能であり得、ノードTE2はノードSM1と直接通信することができるが、ノードSM1はノードTE1と直接通信することができない。

30

【0032】

例B1：図3は、3つのノードTE1、TE2、及びSM1を有する分散データベースシステム300を示している。図3に見られるように、3つのノードTE1、TE2、及びSM1は、TE1がリンク212aを介してTE2と通信し、TE2がリンク212bを介してSM1と通信し、SM1がリンク212cを介してTE1と通信するコーラスグループを形成する。TE1及びTE2の間のリンク212aの障害（またはTE1及びTE2が異なるデータセンターにあると仮定すると、TE1及びTE2のデータセンター間のネットワーク分断）によって、2つの全接続コンポーネント202' {SM1, TE1}及び202'' {SM1, TE2}が、ノードTE1及びTE2の部分接続と共に作成される。全接続コンポーネント202'及び202''は同じサイズであり、リンク障害前に稼働していたノード数の半数超を有するので、これらのノードは、上記で論じた辞書式順序付けなどのタイブレーク処理を実施して、勝利全接続コンポーネント204'を決定する。図3において、{SM1, TE1}は、（辞書式順序などのタイブレーク処理によって決定される）勝利全接続コンポーネント204'であるので、SM1及びTE1は稼働を継続し、TE2は自身を機能停止させる。

40

【0033】

例B2：図4は、5つのノードTE1、TE2、TE3、SM1、及びSM2のコーラスグループを有する分散データベースシステム400を示している。この例では、2つの

50

リンク障害が発生し、1つはSM1及びSM2の間（リンク212f）であり、もう1つはSM2及びTE3の間（リンク212e）である。これらの障害により、全接続コンポーネント402' {TE1, TE2, TE3, SM1} 及び402'' {TE1, TE2, SM2} が生成され、ノードSM2はその他のノードに部分接続されており、その他のノードは互いに直接接続されたままである。第1の全接続コンポーネント402' {TE1, TE2, TE3, SM1} は、半数超のノードを含み、他の勝利全接続コンポーネント402'' よりも大きいので、勝利全接続コンポーネント404' である。ノードSM2は機能停止し、その他のノードは稼働を継続する。

【0034】

例B3：図5は、4つのリンク障害を被っている5つのノードTE1、TE2、TE3、SM1、及びSM2のコーラスグループを有する5ノードの分散データベースシステム500を示している。この例では、TE1及びSM1の間（リンク212c）、TE1及びSM2の間（リンク212h）、TE2及びTE3の間（リンク212g）、ならびにTE3及びSM1の間（リンク212j）で4つのリンク障害が発生している。これらの障害により、いくつかの全接続コンポーネントが生成されるが、1つのみが少なくとも3つのノード {TE2, SM1, SM2} を有し、これを右に示す。ノードTE1及びTE3は、分散データベースに部分接続されたままであるが、分散データベース内の他の全てのノードと直接通信することはできない。その結果、ノードTE1及びTE3は機能停止して、{TE2, SM1, SM2} が勝利全接続コンポーネント404' として残る。

【0035】

図6は、図5の部分接続のケースには、ローテーション式リーダーモデルの方法論を使用して対処できない理由を示している。図6に示すように、5つのノードTE1、TE2、TE3、SM1、及びSM2は、ステップ1（左）でグループを形成する。ステップ1では、これらのノード全てが中断されことなく相互に通信することができる。しかしながら、ステップ2（右）に示すように、TE1及びSM1の間（リンク212c）、TE1及びSM2の間（リンク212h）、TE2及びTE3の間（リンク212g）、ならびにTE3及びSM1の間（リンク212j）で通信リンクに障害が発生する。これらの障害によって、TE1及びSM1の間、TE1及びSM2の間、TE2及びTE3の間、ならびにTE3及びSM1の間の直接通信が中断される。

【0036】

SM1は、ネットワーク障害の直前の現在のリーダーである。リンク障害後にローテーション式リーダーの方法論に基づいて、SM1はTE2及びSM2からハートビートメッセージを受信するので、自身がリーダーであるという想定を継続する。TE1及びSM1の間（リンク212c）のリンク障害が原因で、TE1がSM1からハートビートメッセージを受信しないので、TE1はリーダーシップをTE2にローテーションする。同様に、TE3及びSM1の間（リンク212j）のリンク障害が原因で、TE3はリーダーシップをTE1にローテーションする。したがって、SM1、TE2、TE1が立て続けに（必ずしもこの順序でない）リーダーシップをとるが、TE1はSM1にもSM2にも接続されていないので、SM1がSM2に接続されているか否かさえない。このローテーション式リーダーシップでは、障害（複数可）を解決することが困難になる。

【0037】

概念的には、上記のように、リーダーノードが他の全てのノードに接続されていない場合がある（ひいてはリーダーが他の全てのノードの接続情報を知らない場合がある）ので、中央集権的なリーダーベースのソリューションに部分接続のケースに適切に対処させることは困難である。しかしながら、本明細書に記載のリーダーなしの障害解決処理は、これらの部分接続のケースの全てに信頼できる方法で対処するので、リーダーベースの障害解決方法よりも改善される。

【0038】

例B4：図7は、3つのリンク障害を有する5つのノードのコーラスグループを有する分散データベースシステム700を示しており、1つのリンク障害はTE1及びSM1の

10

20

30

40

50

間（リンク 2 1 2 c）であり、もう 1 つは T E 2 及び T E 3 の間（リンク 2 1 2 g）であり、もう 1 つは S M 1 及び T E 3 の間（リンク 2 1 2 j）である。これらの障害により、全接続コンポーネント { T E 1, T E 2, S M 2 }、{ T E 2, S M 1, S M 2 }、及び { T E 1, S M 2, T E 3 } が生成され、ノード T E 1 及び T E 2 は部分接続される。これらの 3 つの全接続コンポーネントのそれぞれは、リンク障害前のコーラスグループのノード数の半数超を含む。さらに、これら 3 つの全接続された多数派グループは同じサイズである。したがって、ノードは、辞書式順序付けなどのタイブレーク処理を実施して、勝利全接続コンポーネント 7 0 4 ' を特定する。この例では、{ T E 2, S M 1, S M 2 } が、（タイブレーク処理によって決定される）勝利全接続コンポーネント 7 0 4 ' である。したがって、ノード T E 1 及び T E 3 は、ネットワーク障害を解決するために自身らを機能停止させる。

10

【 0 0 3 9 】

例 B 5：図 8 は、5 つのノード T E 1、T E 2、T E 3、S M 1、及び S M 2 のコーラスグループを有する分散データベースシステム 8 0 0 を示している。この例では、T E 1 及び S M 1 の間（リンク 2 1 2 c）、T E 1 及び S M 2 の間（リンク 2 1 2 h）、T E 2 及び T E 3 の間（リンク 2 1 2 g）、T E 2 及び S M 2 の間（リンク 2 1 2 i）、ならびに T E 3 及び S M 1 の間（リンク 2 1 2 j）に 5 つのリンク障害が発生している。図 8 からわかるように、これらの障害の後に 5 つの全接続されたノードグループが存在し、それぞれが 2 つのノードのサイズである。これは、リンク障害前のコーラスグループ内のノード数の半数超より少ない。したがって、リンク障害後に全接続された多数派グループが存在しないので、全てのノードが自身らを機能停止させる。

20

【 0 0 4 0 】

ケース C：図 9 は、（単方向）リンク障害（複数可）により、ノードまたはノードのセットが、一方向では他のノードのサブセットと通信できるが、他の方向では通信できない、部分接続の特殊なケースを示している。図 9 に見られるように、分散データベースシステム 9 0 0 内の 3 つのノード T E 1、T E 2、及び S M 1 が、コーラスグループを形成する。しかしながら、T E 1 及び T E 2 の間（リンク 2 1 2 a ' '）で単方向リンク障害が発生して、T E 2 は T E 1 にメッセージを送信できるが（リンク 2 1 2 a '）、T E 1 は T E 2 にメッセージを送信できなくなる（リンク 2 1 2 a ' '）。この単方向リンク障害により（T E 1 及び T E 2 の間の双方向リンク障害と同様に）、全接続コンポーネント 9 0 2 ' { T E 1, S M 1 } 及び 9 0 2 ' ' { T E 2, S M 1 } が作成される。全接続コンポーネントの 2 つのセットは同じサイズであり、リンク障害前に稼働していたノード数の半数超（すなわち、計 3 つのノードのうち 2 つ）を含むので、ノードはタイブレーク処理を実施して、勝利全接続コンポーネント 9 0 4 ' を決定する。この例では、{ T E 1, S M 1 } が、（上記のタイブレーク処理によって決定される）勝利全接続コンポーネント 9 0 4 ' である。したがって、ノード T E 1 及び S M 1 は稼働を継続し、ノード T E 2 は自身を機能停止させる。

30

【 0 0 4 1 】

ケース D：処理 1 0 0 はまた、メンバーシップ変更中にネットワーク障害が発生したことが原因で、分散データベースシステムが複数の多数派グループに分割しないようにする。メンバーシップ変更とは、新しいノードが分散データベースシステムに加入すること、または分散データベースシステムの既存のノードが分散データベースシステムから離脱することを指す。図 1 0 は、ケース D の一例を示している。この例では、コーラス 1 0 0 0 は、3 つのノード T E 1、S M 1 及び S M 2 から始まる。2 つのノード T E 2 及び T E 3 がコーラス 1 0 0 0 に加入しようとする。それらが加入している最中に、ネットワーク分断が発生して、分散データベースが全接続コンポーネント 1 0 0 2 ' { T E 2, T E 3, T E 1 } 及び 1 0 0 2 ' ' { S M 1, S M 2 } に分離される。両方のグループが稼働を継続し得、その理由は、グループ { T E 2, T E 3, T E 1 } のメンバーが、自身らがコーラス { T E 2, T E 3, T E 1, S M 1, S M 2 } の一部であるために、多数派を形成すると考え、グループ { S M 1, S M 2 } のメンバーが、自身らがコーラス { T E 1, S M 1,

40

50

SM2}の一部であるために、同様に多数派を形成すると考えるためである。処理100は、1つのグループのみが稼働を継続するようにする。換言すれば、処理100は、{TE2, TE3, TE1}及び{SM1, SM2}の両方が同時に稼働を継続しないようにする。

【0042】

被疑ノードに関する情報の収集及び共有

本明細書で開示する障害解決処理（たとえば、処理100）は、リーダーなしの処理である。ネットワーク障害イベントに応答して、各ノードは自身の被疑リストを特定し、接続情報（自身のもの、及び任意選択により他のノードのもの）を他のノードと交換したのち、障害解決の判断を行う。この処理は、パーティション内の全てのノードが同じ障害解決の判断（複数可）に至るようにするために、この処理の通信フェーズの終了時に各ノードが自身のパーティション内の他のノードに関する十分な接続情報を有するような方法で、ノードに通信させて接続情報を交換させる。プロトコルの進行中に発生するいずれかの新しいネットワーク障害イベントにより、全てのノードがプロトコルを再開始する。

【0043】

一般に、本発明の障害解決処理は、各ノードが他のノードの被疑リスト／接続に関する情報を収集するフェーズ1と、各ノードがフェーズ1中に収集した情報に基づいて障害解決の判断を行う（たとえば、自身を機能停止させる）フェーズ2との2つのフェーズを含むことができる。

【0044】

フェーズ1中に、各ノードは最大で2ラウンドの協調的なブロードキャストに参加する。被疑リストのこれらの協調的なブロードキャストは、パーティション内のノード間の接続情報／被疑リストの交換を含む。上記のケースAでは、各ノードは1回の協調的なブロードキャストを実行する。上記のケースB及びCでは、各ノードは2回の協調的なブロードキャストを実行する。ケースA、B、及びCにおいて全てのノードがグループメンバーシップ変更について合意するには、2ラウンドの協調的なブロードキャストで十分である。

【0045】

この処理を直感的に理解できるようにするために、まず、最適化されていない接続情報交換処理を以下に示し、これは $(n-1)$ ラウンドのブロードキャストを含み、ここで n はフェーズ1中のコーラス内のノード数である。その後、最適化されたバージョンの接続情報交換処理を以下に示し、この処理では、コーラス内のノード数に関係なく、各ノードが最大で2ラウンドのブロードキャストに参加する。

【0046】

明確さ及び単純さのために、接続情報交換処理の進行中に、新しいネットワーク障害イベントも、加入する新しいノードも、コーラスメンバーノードの障害もないと仮定する。しかしながら、本明細書に記載の接続情報交換処理は、これら全てのイベントにも拡張することができる。これらの仮定及び／または制限は、コア処理の説明後、後のセクションで解除する。

【0047】

最適化されていない被疑ノードリストの配布

まず、コーラスは n 個の全接続されたノードを含む。ネットワーク障害イベントが発生したとする。各ノードは、ネットワーク障害イベントを解決するために以下のプロトコルを実行する。

【0048】

各ノードは自身の被疑リストを用意する（この被疑リストは空のリストであり得、これが起こり得るのは、ネットワーク障害イベント後にノードが他の全てのノードに全接続されている（または少なくともそう考えている）場合である）。

【0049】

フェーズ1：各ノードは、他のノードの被疑リスト／接続に関する情報を収集するために、 $(n-1)$ ラウンドの協調的なブロードキャストを実行する。ラウンド1では、各ノ

10

20

30

40

50

ードは自身の被疑リストを自身の隣接ノードに送信し、自身の隣接ノードの被疑リストを受信するまで待機する。ラウンド2からラウンド $(n-1)$ までにおいて、各ノードは、前回のラウンドで受信した他のノードの被疑リストを自身の隣接機に送信し、自身の隣接機からそのような情報を受信するまで待機する。

【0050】

フェーズ2：各ノードはこのとき、自身のパーティション内の他の全てのノードの接続情報を受信している（コーラスは n 個のノードを含むので、ノードが上記の方法で $(n-1)$ ラウンドのブロードキャストを実行することにより、各ノードが自身のパーティション内の他の全てのノードの接続情報を得るようになる）。各ノードは、自身のパーティションの接続グラフを用意し、接続グラフの最大サイズ（または最大クリーク）の全接続コンポーネントを見つける。そのような全接続コンポーネントが2つ以上ある場合、ノードは、（たとえば、完全コンポーネント内のノードの一意の識別子の辞書式順序に基づいて）タイブレーク処理によって決定された1つの全接続コンポーネントを勝利全接続コンポーネントとして選択する。勝利全接続コンポーネントのサイズが少なくとも $(n/2+1)$ であり、ノードが勝利全接続コンポーネントのメンバーである場合、ノードは稼働を継続すると決断し（そしてプロトコルを終了し）、そうでなければ、ノードは自身を機能停止させる。

【0051】

以下は、最大で2ラウンドのブロードキャストの後にノードがメンバーシップ変更について合意するようにする最適化である。

【0052】

最適化1：これは、（上記のセクションの）ケースAでカバーされるシナリオの場合に適用可能な最適化である。これは、ネットワーク障害イベントによってデータベースが全接続されたノードの非連結グループに分割された場合に、グループ／パーティション内の全てのノードの被疑リストが同じになるという観察結果に基づいている。たとえば、図2について考察する。図2では、ノードTE1、TE2、及びSM1はTE3及びSM2を疑い、ノードTE3及びSM2はTE1、TE2、及びSM1を疑う。フェーズ1中の第1ラウンドの協調的なブロードキャストの後、ノードの被疑リストが自身の全ての隣接機の被疑リストと一致する場合、ノードは、（a）自身が全接続コンポーネントの一部であり、（b）全接続コンポーネントのサイズ（これはコーラスのサイズから自身の被疑リストのサイズを引いたものに等しい）を特定できると推測することができる。したがって、全てのノードは、フェーズ1中の第1ラウンドのブロードキャスト後に、メンバーシップ変更について合意することができる。

【0053】

最適化2：これは、主に上記のケースB及びCに適用可能であり、部分的にケースAに適用可能である最適化である。最適化されていない処理では、全てのノードが $(n-1)$ ラウンドの協調的なブロードキャストに参加する。これにより、各ノードは、自身のパーティション内の他の全てのノードの接続情報を認識する。しかしながら、最適な障害解決の判断に至るために、各ノードは自身のパーティション内の他の全てのノードの接続情報を知る必要が本当にあるだろうか。ネットワーク障害イベント後にノードをそれらの被疑リストに基づいて2つのカテゴリに分割することについて考察する。カテゴリ（M）は、 $n/2$ 個未満の他のノードを疑っているノードを含み、カテゴリ（N）は、 $n/2$ 個より多くのノードを疑っているノードを含む。 $n/2$ 個より多くを疑っているノードは、被疑リストをブロードキャストするのではなく、即刻自身らを機能停止させ得、その理由は、それらが勝利全接続コンポーネントの一部になり得ないためである。

【0054】

たとえば、図11について考察する。ネットワーク障害イベントの後、ノードTE2、SM1、及びSM2はカテゴリ（M）に入り、ノードTE1及びTE3はカテゴリ（N）に入る。カテゴリ（M）について考察する。カテゴリ（M）のノードは、最適な障害解決の判断を行うために、カテゴリ（M）の他のノードの接続情報について知る必要があるだ

ろうか。必要はある。この理由は、カテゴリ (M) のノードが、カテゴリ (M) の他のノードと共に少なくとも $(n/2 + 1)$ のサイズの全接続コンポーネントを形成できるためであり、カテゴリ (M) の他のノードの接続情報について知ることは、(a) 自身が少なくとも $(n/2 + 1)$ のサイズの全接続コンポーネントの一部であるかどうかを判別し、(b) 少なくとも $(n/2 + 1)$ のサイズの全ての全接続コンポーネントを識別し、(c) 自身が勝利全接続コンポーネントの一部であるかどうかを判別するのに役立つ。カテゴリ (M) のノードは、最適な障害解決の判断を行うために、カテゴリ (N) のノードの接続情報について知る必要があるだろうか。必要はない。これは、カテゴリ (M) のノードが、カテゴリ (N) のノードと共に少なくとも $(n/2 + 1)$ のサイズの全接続コンポーネントを形成することはないためであり、その理由は、カテゴリ (N) のノードが $(n/2)$ 個より多くの他のノードを疑っているためである。

10

【0055】

ここでカテゴリ (N) について考察する。カテゴリ (N) のノードは、最適な障害解決の判断を行うために、カテゴリ (M) 及びカテゴリ (N) のノードの接続情報について知る必要があるだろうか。必要はない。この理由は、カテゴリ (N) のノードが $(n/2)$ 個より多くの他のノードを疑っているために、他の任意のノード (複数可) と共に少なくとも $(n/2 + 1)$ のサイズの全接続コンポーネントを形成することはないためである。他の全てのノードの接続情報を用意することは、カテゴリ (N) のノードが、稼働を継続する他のノードを知るのに役立つが、そのノードが他のノードと共に少なくとも $(n/2 + 1)$ のサイズの全接続コンポーネントを形成できないという事実は変わらない。

20

【0056】

したがって、分散データベースシステム内の全てのノードが最適な障害解決の結果について合意するには、カテゴリ (M) 内の各ノードにカテゴリ (M) 内の他の各ノードの接続情報を認識させるのに十分なラウンドの協調的なブロードキャストで事足りるであろう。したがって、最適化されていない処理への修正として、最適化された処理は、フェーズ 1 の開始前にカテゴリ (N) のノードを機能停止させるが、同時にそれらをコーラスのメンバーとして保持することから始まる。換言すれば、カテゴリ (M) のノードは、フェーズ 1 が開始する前にカテゴリ (N) のノードが自身らを機能停止させるが、フェーズ 2 までカテゴリ (N) のノードを自身らのノードリスト上に保持する。機能停止されたノード (フェーズ 1 の開始前に機能停止させることができるカテゴリ (N) のノード) をフェーズ 2 までコーラスのメンバーとして保持することによって、正確性が確保され、すなわち、障害解決の結果は、少なくとも $(n/2 + 1)$ 個のノードを有する全接続されたセットであり、ここで n はフェーズ 1 の前に最適化として機能停止されたノードを含む。(カテゴリ (N) のノード (または任意のタイプのノード) を含めないと、 n の値 (グループサイズ) 及び多数派のサイズが変化し得、結果の正確性を証明することがより困難になり得る。)

30

【0057】

カテゴリ (N) のノードを機能停止させても、カテゴリ (M) のノード間の接続には影響せず (すなわち、カテゴリ (N) のノードの機能停止が原因で、カテゴリ (M) のノードは切断されない)、その理由は、カテゴリ (M) の任意の 2 つのノードが、相互に直接接続されているか、またはカテゴリ (M) の他のノードを介して接続されているためである。したがって、カテゴリ (N) ノードを機能停止させても、障害解決の結果の最適性に影響を与えるはずがない。

40

【0058】

概念的には、最適化によって、基本的にはカテゴリ (M) のノードが障害解決の結果について合意に達し、カテゴリ (N) のノードがその結果に従うようになる。この最適化により、フェーズ 1 を開始した各ノードは、少なくとも $(n/2)$ 個の他のノードに接続されているので、接続グラフの直径 (すなわち、接続グラフ内の任意の 2 つのノード間の最大距離) は最大で 2 である。したがって、フェーズ 1 を開始した各ノードがフェーズ 1 を開始した他の各ノードの接続について知るには、2 ラウンドのブロードキャストしか必要

50

ない。フェーズ1の各ノードは少なくとも $n/2$ 個の他のノードに接続されているので、接続グラフの直径は最大で2であり、したがって、任意の2つのノードは最大で1つのノードだけ離れている。

【0059】

最適化された被疑ノードリストの配布

n 個の全接続されたノードを含むコーラスについて考察する。ネットワーク障害が発生したとする。各ノードは、ネットワーク障害を解決するために以下のプロトコルを実行する。各ノードは自身の被疑リストを用意する（注：この被疑リストは空のリストであり得、これが起こり得るのは、ネットワーク障害イベント後にノードが他の全てのノードに全接続されている（またはそう考えている）場合である）。

【0060】

フェーズ0：各ノードは、自身が $(n-1/2)$ 個より多くの他のノードを疑っているかどうかをチェックする。疑っている場合、ノードは自身を機能停止させる。（他のノードは、フェーズ1中に、この機能停止について聞き得る。聞いた場合、それらのノードはプロトコルを再開始し、フェーズ0からやり直す。）

【0061】

フェーズ1のラウンド1：各ノードは、自身の被疑リストを自身の隣接ノードに送信し、自身の隣接ノードの被疑リストを受信するまで待機する。上記のように、ノードの隣接機のうちの1つまたは複数がフェーズ0で機能停止した場合、そのノードは自身の隣接機の被疑リストを待っている間にそれらの機能停止について聞き得る。そのような機能停止（複数可）について聞くと、ノードはプロトコルを再開始し、フェーズ0からやり直す。これにより、他のノードもプロトコルを再開始する。同様に、隣接ノードがプロトコルを再開始する場合、ノードはフェーズ0からやり直す。また、上記のように、このノードは、このステージでどの機能停止されたノードのフェイルオーバーも開始しない（すなわち、勝利全接続コンポーネントを決定する目的で、全てのノードを自身のコーラスに保持する）。これは、フェーズ0の複数のラウンドにも当てはまる。

【0062】

各ノードは、自身の被疑リストが自身の全ての隣接ノードの被疑リストと同じであるかどうかをチェックする。ノードの被疑リストが自身の全ての隣接ノードの被疑リストと一致する場合、これはノードが自身の隣接ノードと全接続されていることを示している。このシナリオは、上記のケースA（たとえば、図2）でカバーされる。フェーズ1を開始した各ノードは、少なくとも $(n/2)$ 個の他のノードに接続されているので、ノードの隣接機のリストのサイズは少なくとも $(n/2)$ であり得る（ノードはその隣接ノードと共に、少なくとも $(n/2+1)$ 個のノードを含むグループを形成する）。ノードは稼働を継続すると決断し、プロトコルを終了する。

【0063】

ノードの被疑リストが、自身の隣接機のうちの少なくとも1つの被疑リストと一致しない場合、これは、ノードが自身のパーティション内の他の全てのノードと全接続されていないことを示している。このシナリオは、上記のケースB及びC（たとえば、図3～図9）でカバーされている。そのようなノードは、ラウンド1で受信した情報に基づいて、稼働を継続すべきか否かを判断することができない。したがって、フェーズ1のラウンド2を実施する。

【0064】

フェーズ1のラウンド2：各ノードは、ラウンド1で受信した他のノードの被疑リストを自身の隣接機に送信し、自身の隣接機から自身の隣接機の隣接機のそのような被疑リストを受信するまで待機する。

【0065】

フェーズ2：各ノードはこのとき、自身のパーティション内の他の全てのノードの接続情報を受信している。各ノードは、自身のパーティションの接続グラフを用意し、接続グラフの少なくとも $(n/2+1)$ 個のノードを有する最大の全接続コンポーネント（また

10

20

30

40

50

は少なくとも $(n/2 + 1)$ のサイズの最大クリーク)を見つける。2つ以上の全接続コンポーネントが存在する場合(たとえば、図7)、ノードは、障害解決を決定論的にするために、タイブレーク処理(たとえば、辞書式順序)によって決定される1つの全接続コンポーネントを勝利全接続コンポーネントとして選択する。ノードが勝利全接続コンポーネントのメンバーである場合、ノードは稼働を継続すると決断し(そしてプロトコルを終了し)、そうでなければ、ノードは自身を機能停止させる。

【0066】

分散データベースシステムがネットワークイベントを解決している最中に、新しいネットワーク障害イベントが発生した場合、プロトコルはノードに逆戻りさせ、新しいネットワークイベントの影響を考慮して、ノードの接続を再検査させたのち、障害解決の判断を行わせる。

【0067】

新しいネットワーク障害イベントに加えて、分散データベースシステム内のノードがネットワーク障害を解決している間に、ノードの機能停止(たとえば、ノードの手動シャットダウンが原因で発生するもの)も発生し得る。ノードの機能停止に応答して、プロトコルはノードにフェーズ0から再開始させつつ、機能停止されたノードをコーラスのメンバーとしてフェーズ2まで保持する(機能停止されたノードに対してフェイルオーバーを実行しないことにより、残りのノードが機能停止されたノードをそれらのノードリストから削除するのを阻止する)。上記で説明したように、機能停止されたノードをコーラスのメンバーとしてフェーズ2まで保持することによって、正確性が確保され、すなわち、障害解決の結果は、少なくとも $(n/2 + 1)$ 個のノードを有する全接続されたセットであり、ここで n は機能停止したノードを含み、したがって、フェーズ2の後で稼働を継続するようなセットは1つのみとすることができる。

【0068】

図12は、ネットワーク障害を解決するための最適化された処理1200を示すフローチャートである。各ノードは同じ処理に従うので、フローチャートは単一ノードの視点から処理1200を示している。処理1200は、ステージの観点から詳細に説明する。

【0069】

ステージ0：初期ステージ。1202において、ノードはコーラス内の他の全てのノードに全接続される。ローカルまたはリモートで被疑ノードを検出すると、ノードはステージ1に移動する。

【0070】

ステージ1：1210において、ノードは1ping(ハートビート)サイクルの間、さらなるping(ハートビート)タイムアウトが発生するのを待機し、自身の被疑リストを用意し、受信した被疑リストメッセージを取り込んだのち、ステージ2に入る。

【0071】

ステージ2：1220において、ノードは自身が $(n-1/2)$ 個より多くの他のノードを疑っているかどうかをチェックする(ここで n はコーラス内のノード数である)。疑っている場合、1299において、ノードは自身を機能停止させる。そうでない場合、ノードは、ステージ1で自身の被疑リストを用意してから、新しい被疑機(suspect)がいるかどうかをチェックする。また、ノードは、自身の隣接機のいずれかが新しい被疑機を検出したために、プロトコルを再開始したかどうかをチェックする。各ノードは、実行する処理1200の反復ごとに、protocolIterationNumberと呼ぶ番号を割り当て得る。各ノードは、送信する被疑リストメッセージにこの番号を設定し、自身のローカルのprotocolIterationNumberを他のノードから受信した被疑リスト内のprotocolIterationNumberと比較する。ノードは、自身のprotocolIterationNumberが隣接機のprotocolIterationNumberよりも小さいと判定した場合、自身の隣接機が処理を再開始したと判定し、ステージ1に戻る。そうでなければ、ノードはステージ3に入る。(ノードのprotocolIterationNumberが隣接機のpr

protocolIterationNumberよりも大きい場合、そのノードはプロトコルを再開始しており（おそらく新しい被疑機を見つけたため）、これにより隣接機もプロトコルを再開始するはずである。）

【0072】

ステージ3：1230において、ノードは自身のラウンド1被疑リストを自身の隣接ノードにブロードキャストする。ノードは、1232において、ラウンド1被疑リストメッセージを待機している間に、新しい被疑機を検出し得、あるいは自身の隣接機のうちの1つまたは複数が新しい被疑機を検出したことを聞き得る。その場合、ノードはそれ以上の応答の待機を中止し、ステージ1に戻る。1234において、ノードは自身の全ての隣接ノードからラウンド1被疑リストメッセージを受信する。ノードが自身の隣接機のいずれからも適時に（たとえば、所定の期間内に）応答を受信しなかった場合、1236において、ノードはそのような隣接機を被疑機としてマークし、ステージ1に戻る。ノードが自身のprotocolIterationNumberよりも大きいprotocolIterationNumberを有するラウンド1被疑リストを受信した場合、1238において、ノードはステージ1の最初に戻る。自身の全ての隣接機からラウンド1応答を受信すると、ノードはステージ4に入る。

【0073】

ステージ4：1240において、ノードの被疑リストが自身の全ての隣接機の被疑リストと一致する場合、ノードは、自身が自身の隣接ノードと全接続されていると判定する（たとえば、図2の通り）。ステージ3を開始した各ノードは、少なくとも $(n/2)$ 個の他のノードに接続されているので、ノードの隣接機のリストのサイズは少なくとも $(n/2)$ であり得る（すなわち、ノード及びその隣接機は、少なくとも $(n/2+1)$ 個のノードを含む全接続コンポーネントまたはグループを形成する）。1201において、ノードは、稼働を継続すると決断し、被疑ノードを排除し、処理1200を終了する。

【0074】

ノードの被疑リストが自身の隣接機のうちの少なくとも1つの被疑リストと一致しない場合、ノードは自身のパーティション内の他の全てのノードと全接続されていない（たとえば、図3～図9の通り）。ノードは、ラウンド1で受信した情報に基づいて、稼働を継続すべきか、機能停止すべきかを判断できないので、ノードはステージ5に入り、ステージ5は1250においてラウンド2被疑リストメッセージをブロードキャストすることを含む。

【0075】

ステージ5：1250において、ノードは、自身の元の被疑機と自身の隣接ノードの被疑機とを加えたものを含む自身のラウンド2被疑リストを自身の隣接ノードにブロードキャストし、自身の全ての隣接ノードからラウンド2被疑リストメッセージを受信するまで待機する。ノードは、1230において自身のラウンド1被疑リストメッセージをブロードキャストした後、いつでもその他のノードからラウンド2被疑リストメッセージを受信し得る。ノードは、これらのラウンド2被疑リストメッセージを蓄積する。1252において、新しいネットワーク障害が発生した場合、もしくはノードが他のノードからラウンド1メッセージを受信した場合、またはノードが他のノードの機能停止について聞いた場合、ノードはステージ1に戻る。ステージ1に戻ると、ノードは蓄積したラウンド2被疑リストメッセージを全て破棄する。しかしながら、他のノードが戻り、他のメッセージを送信した場合、そのメッセージは保持される。ノードは、ラウンド1及びラウンド2の被疑リストメッセージ内のprotocolIterationNumberに基づいて、これら2つのタイプのメッセージを区別する。換言すれば、protocolIterationNumberに基づくメッセージは、protocolIterationNumber及びラウンド番号を含む。

【0076】

1254において、ノードは、自身の全ての隣接ノードからラウンド2被疑リストメッセージを受信すると、ステージ6に入る。ノードが自身のラウンド2被疑リストメッセー

ジをブロードキャストした後に、新しいネットワークイベントが発生した場合、またはノードが他のノードの機能停止について聞いた場合、障害解決の判断は最適なものではなくなり得る。可能性のあるケースは少なくとも2つあり、ケース（a）では、ノードは新しい被疑ノードまたは機能停止したノードからラウンド2メッセージを既に受信しており、ケース（b）では、ノードは新しい被疑機または機能停止したノードからラウンド2メッセージを受信していない。

【0077】

ケース（a）では、ノードはステージ6に進み得、現在のネットワークイベントの障害解決を実行したのち、プロトコルを再開始することにより新しいネットワークイベントに対処するか、または（現在のネットワークイベントを解決せずに）ステージ1に戻って、処理1200を再開始する（これにより、その後、現在のネットワーク障害及び新しいネットワーク障害の両方が解決される）。ケース（b）では、ノードは新しい被疑機からも機能停止したノードからもラウンド2メッセージを受信しないので、ノードはステージ1に戻る。しかしながら、その他のノードもステージ6を完了する前にステージ1に戻る保証はない（その理由は、それらが新しい被疑機または機能停止したノードからラウンド2メッセージを受信している場合があるためである）。このケースでは、障害解決の結果は準最適であり得る（すなわち、生存セットは、なれたはずのものよりも小さくなるが、生存セットは依然として1つのみである）。しかしながら、このノードがステージ1に移動しても、このノードは既に自身のラウンド2メッセージを送信しているので、他のノードの進行は阻止されない。

【0078】

ステージ6：1260において、ノードは自身のパーティションの接続グラフを用意し、接続グラフの少なくとも $(n/2 + 1)$ のサイズの最大の全接続コンポーネント（または少なくとも $(n/2 + 1)$ のサイズの最大クリーク）を見つける。そのようなコンポーネントが2つ以上ある場合、ノードは、タイブレーク処理によって決定されたそれらの中の1つを勝利全接続コンポーネントとして選択する。ノードが勝利全接続コンポーネントのメンバーである場合、1201において、ノードは稼働を継続すると決断し、勝利全接続コンポーネントの一部ではないノードを排除する。そうでない場合、1299において、ノードは自身を機能停止させる。

【0079】

プロトコル反復番号

上記で論じたように、分散データベースシステム内の任意のノードは、1つまたは複数の被疑ノードを検出したことに応答して、障害解決プロトコル（たとえば、図12の処理1200）を開始することができる。また、障害解決プロトコルの実行中に発生したいずれかの新しいネットワーク障害イベントによって、プロトコルの再開始がトリガーされる。ノードが受信した被疑リストメッセージ（ラウンド1またはラウンド2）が、プロトコルの現在の呼び出しに属しているか、またはプロトコルの再開始による次の呼び出しに属しているか（さらには、プロトコルを再開始したノードの場合、プロトコルの前の呼び出しに属しているか）をノードが検出できるようにするために、ノードは、`protocolIterationNumber`と呼ばれる番号を、障害解決プロトコルの各呼び出しに関連付ける。

【0080】

各ノードは自身のローカルの`protocolIterationNumber`を維持し、送信する被疑リストメッセージにこの番号を設定し、各ノードは自身のローカルの`protocolIterationNumber`を、受信した被疑リストメッセージ内の`protocolIterationNumber`と比較する。番号が一致する場合、ノードは、受信した被疑リストメッセージがプロトコルの現在の呼び出しに対応していると推測する。受信した被疑リストメッセージ内の`protocolIterationNumber`が自身の`protocolIterationNumber`より大きい場合、ノードは送信元がプロトコルの再開始を開始した（したがって、プロトコルを再開始する）

と推測する。また、受信した被疑リストメッセージ内の `protocolIterationNumber` が自身の `protocolIterationNumber` よりも小さい場合、ノードは、送信元がまだプロトコルの前回の反復を実行しているために、メッセージを無視すると推測する。

【0081】

各ノードは、以下の方法で自身のローカルの `protocolIterationNumber` を維持することができる。

(a) `ProtocolIterationNumber` は第1のノードにおいて、データベースの初期化中及びデータベースの再開始中にゼロに設定される。

(b) `ProtocolIterationNumber` はマスターカタログの一部としてシリアル化され、分散データベースシステムに加入する新しいノードは、マスターカタログをフェッチするときに、マスターカタログチェアマンから現在の `protocolIterationNumber` を受信する。新しいノードは、分散データベースシステムに加入する場合に、マスターカタログを受信する。現在の `protocolIterationNumber` をマスターカタログに記憶することにより、現在の `protocolIterationNumber` が新しいノードで利用可能になる。

(c) ノードがどのノードも疑っておらず、被疑ノードを検出しており、他のノードからいずれの被疑リストメッセージも受信しておらず、障害解決プロトコルを呼び出している場合、ノードは自身の `protocolIterationNumber` をインクリメントする。

(d) ノードがどのノードも疑っておらず、被疑ノードを検出しており、他のノードから1つまたは複数の被疑リストメッセージを受信しており、障害解決プロトコルを呼び出している場合、ノードは自身の `protocolIterationNumber` を受信した被疑リストメッセージ内の最大の `protocolIterationNumber` に設定する。

(e) ノードが被疑ノードを検出しておらず、他のノードから1つまたは複数の被疑リストメッセージを受信しており、障害解決プロトコルを呼び出している場合、ノードは自身の `protocolIterationNumber` を、他のノードから受信した被疑リストメッセージ内の最大の `protocolIterationNumber` に設定する。

(f) ノードが障害解決プロトコルを実行していて、自身のローカルの番号よりも大きい `protocolIterationNumber` を有する被疑リストメッセージを受信しておらず、新しいネットワーク障害イベントを検出した場合、ノードは自身の `protocolIterationNumber` をインクリメントする（そしてプロトコルを再開始する）。

(g) ノードが障害解決プロトコルを実行していて、自身のローカルの番号よりも大きい `protocolIterationNumber` を有する被疑リストメッセージを受信しており、新しいネットワーク障害イベントを検出した場合、ノードは自身のローカルの `protocolIterationNumber` を受信した被疑リストメッセージ内の番号に設定する（そしてプロトコルを再開始する）。

(h) ノードが障害解決プロトコルを実行していて、自身のローカルの番号よりも大きい `protocolIterationNumber` を有する被疑リストメッセージを受信しており、新しいネットワーク障害イベントを検出なかった場合、ノードは自身のローカルの `protocolIterationNumber` を受信した被疑リストメッセージ内の番号に設定する（そしてプロトコルを再開始する）。

(i) ノードが障害解決プロトコルを実行していて、自身のローカルの番号よりも小さい `protocolIterationNumber` を有する被疑リストメッセージを受信した場合、ノードはそのメッセージを無視する。

【0082】

これらのポイントは以下のようにまとめることができる。

(A) `ProtocolIterationNumber` は第1のノードにおいて、デー

データベースの初期化中及びデータベースの再開始中にゼロに設定される。

(B) `ProtocolIterationNumber` はマスターカタログの一部としてシリアル化され、データベースに加入する新しいノードは、マスターカタログをフェッチするときに、マスターカタログチェアマンから現在の `protocolIterationNumber` を受信する。

(C) ノードが障害解決プロトコルを呼び出している場合（被疑ノードを検出したため、及び／または他のノードから被疑リストメッセージを受信したため）、ノードは、自身のローカルの `protocolIterationNumber` よりも大きい `protocolIterationNumber` を有する被疑リストメッセージを受信したかどうかをチェックする。受信した場合、ノードは自身のローカルの `protocolIterationNumber` を、受信した被疑リストメッセージ（複数可）内の最大の `protocolIterationNumber` に設定し、そうでない場合、自身の `protocolIterationNumber` をインクリメントする。

【0083】

単方向リンク障害への対処

上記のケース D（図 10）のような単方向リンク障害は、それらに双方向リンク障害として対処することによって（すなわち、障害のあるリンクの両側のノードに互いを疑わせることによって）解決することができる。たとえば、分散データベースシステム内のノード A 及びノード B の 2 つのノードについて考察する。ノード A はノード B にメッセージを送信できるが、ノード B はノード A にメッセージを送信できないと仮定する。A はノード B に ping メッセージを送信できるが、ノード B から確認応答メッセージを受信しないので、ノード A はノード B を疑うようになる。この時点で、ノード B はまだノード A を疑っていない。しかしながら、ノード A はノード B を疑うようになるので、ノード B への ping メッセージの送信をやめる。これによりノード B はノード A を疑うようになるので、単方向リンク障害が双方向リンク障害に変換される。

【0084】

本明細書に記載の処理では、ノードは `MsgPing` メッセージ（たとえば、ping メッセージ）を送信し、特定のノードに対して、そのノードが前の `MsgPing` メッセージに確認応答した場合にのみ、`Node::lastPingTime` を設定する。これにより、単方向リンク障害によって、リンクの両側のノードが互いを疑うことを確実にする。したがって、上記のプロトコルは、単方向リンク障害、または単方向及び双方向リンク障害の混合を解決することができる。

【0085】

コーラスメンバーシップ変更

新しいノード（または新しいノードのセット）がコーラスに加入している最中にネットワーク障害イベントが発生した場合、この処理は、コーラスが複数の多数派グループに分割されないようにする必要がある。図 10 では、たとえば、ネットワーク分断によって、コーラスが多数派グループ {SM1, SM2} と、少数派グループ {TE1} とに分割される。しかしながら、少数派グループ {TE1} は、新しいノード {TE2, TE3} と共に、多数派グループ {TE1, TE2, TE3} を形成し、その結果、2 つの「多数派」グループ {TE1, TE2, TE3} 及び {SM1, SM2} となる。

【0086】

新しいノードをコーラスに加入させることに関連する問題を解決する 1 つの方法は、新しいノード（複数可）がコーラスに加入している最中にネットワーク障害イベントが発生した場合、新しいノード（複数可）を機能停止させることである。これにより、現在のコーラス内の少数派のノードセットが、新しいノード（複数可）と共に多数派グループを形成することが防がれる。図 10 では、新しいノード TE2 及び TE3（これらはまだコーラスに加入している最中である）を機能停止させることができ、これにより TE1 も機能停止して、データベースに単一の多数派グループ {SM1, SM2} が残る。この処理では、コーラスに同時に加入できるノード数に制限はない。しかしながら、新しいノード（

複数可)が機能停止され、現在のコーラス内の一部のノードが新しいノード(複数可)を認識している場合があるので、この処理は(現在のコーラスの奇数または偶数のノード数、コーラスに加入しようとしているノード数、ネットワーク障害時に新しいノード(複数可)を認識しているコーラス内のノード数などに応じて)システムの可用性に影響を与え得る。

【0087】

この処理は、現在のコーラスメンバーに、新しいノードがコーラスに加入することについて合意させるために、分散データベースにおいてデータのフラグメントを要求するための処理(オリジネータが利用可能なフラグメントを送信し、ピアがオリジネータに確認応答を送信し、オリジネータが完全なデータを要求元へ送信する)に便乗することもできる。この処理は、ノード加入処理中にノードがコーラスサイズについて一致するために図12の障害解決処理1200に対する以下の変更を含む。

【0088】

ノードは、ラウンド1及びラウンド2のブロードキャスト中に、自身らの完全な接続情報(すなわち、自身らの隣接ノードリストならびに自身らの被疑ノードリスト)を交換する。ノードは、ラウンド1／ラウンド2のメッセージを受信したことに応答して、自身らの被疑ノードリスト及び隣接ノードリストを、自身らの隣接機の被疑ノードリスト及び隣接ノードリストと比較する。ノードは、自身が認識していない n_j 個のノードについて自身の隣接機が認識していることに気づいた場合、自身のコーラスサイズを n_j だけインクリメントし、処理を再開する。

【0089】

この処理により、正確性を確保することができ、ネットワーク分断が原因で、新しいノード(複数可)がコーラス内の全てのノードのノードリストに入ることができない場合、その新しいノード(複数可)は障害解決中に自身を機能停止させる。 n をコーラス内のノード数とし、 n_j を、同時にコーラスに加入しようとしているが、ネットワーク分断が原因で n 個全てのノードのノードリストに入ることができないノード数とした場合、 n_j 個のノード(新しいノード)は、自身らのパーティションに関係なく、処理の実行中に自身らを機能停止させる。したがって、最大で n 個のノードが、稼働を継続すべきか否かを判断するために、ラウンド1の後で、自身らが多数派パーティション内にいるかどうかをチェックする。各パーティション内のノードがコーラスサイズ s ($n \leq s \leq n + n_j$)で動作し、ラウンド1の後、コーラス内に最大で n 個のノードが存在するので、最大で1つのパーティションが多数派グループを形成することができ、これにより正確性が確保される。

【0090】

しかしながら、パーティション内の全てのノードが、障害解決プロトコルを開始した後、自身らのノードリストに新しいノード(複数可)を追加したらどうなるだろうか。(なお、ノードは、プロトコルの開始時に、ステージ1中に、自身らの被疑ノードリスト及び隣接ノードリストを用意し、その情報をキャッシュすることに留意されたい)。いずれのノードも、自身らのノードリストに新しいノード(複数可)が追加されたことを検出できない。結果として、新しいノード(複数可)のマスターカタログは完全な状態に遷移し得、新しいノード(複数可)が障害解決処理に参加することになり、その結果、複数の多数派グループがもたらされ得る。

【0091】

たとえば、このシナリオについて考察する。コーラスはノードA、B、及びCを含み、Aは分散データベースのフラグメント(たとえば、フラグメント「マスターカタログ」)のチェアマン／リーダーである。新しいノードD及びEが、同時にコーラスに加入しようとする。ノードAは、D及びEの利用可能メッセージをB及びCに送信する。B及びCは、Aからpingメッセージを受信せず、Aを疑い、プロトコルを開始する。B及びCは(まだ)Aからの利用可能メッセージを適用していないので、コーラスメンバー{A, B, C}でプロトコルを開始する。その後、B及びCは利用可能メッセージを適用し、確認応答メッセージをAに送信し、その後、ネットワーク分割が発生する。D及びEのマスタ

10

20

30

40

50

ーカタログは完全なものとなるので、A、D、Eはコーラスメンバー {A, B, C, D, E} でプロトコルを開始する。両方のグループ {A, D, E} 及び {B, C} は、自身らが多数派グループを形成できると考える。

【0092】

次の拡張により、そのような状況を防ぐことができる。利用可能メッセージを適用した後（または、チェアマンノードの場合は、マスターカタログを新しいノードに送信した後）、ノードは障害解決プロトコルを再開始し（1つが進行中の場合）、これにより、そのノードは自身がキャッシュした被疑リスト及び隣接リストを無効化し、より大きなコーラスサイズでそれらを再計算することになる。

【0093】

図13～図18は、いくつかの例示的な障害シナリオと、本発明の障害解決処理によるそれらへの対処方法とを示している。

【0094】

シナリオ（A）：ネットワーク分断が発生して、新しいノード及びエントリノード（マスターカタログのオリジネータ）が残りのノードから分離される。

【0095】

図13では、SM3はTE1（マスターカタログのチェアマン）にマスターカタログを要求してTE1から受信し、TE1がMsgObjectAvailable（たとえば、送信元ノードが分散データベースシステムに加入することを受信側ノードに通知するメッセージ）をSM1及びSM2に送信する前にネットワーク分断が発生する。SM3を含む全てのノードが、解決プロトコルを開始する。SM3及びTE1はノードSM1及びSM2を疑い、SM1及びSM2はTE1を疑う（SM1及びSM2はSM3については知らない）。SM3は、まだコーラスに加入している最中であるので（TE1から完了を受信していないので）機能停止し、TE1は、コーラス {SM1, SM2, SM3, TE1} 内の2つのノードを疑っているので（フェーズ0で）機能停止し、SM1及びSM2は、コーラス {SM1, SM2, TE1} の多数派を形成するので稼働を継続する。

【0096】

シナリオ（B）：シナリオ（A）の変形例。ネットワーク分断が発生して、新しいノード及びエントリノード（マスターカタログのオリジネータ）が残りのノードから分離される。

【0097】

図14では、SM3はTE1（マスターカタログのチェアマン）にマスターカタログを要求してTE1から受信し、SM1はTE1からMsgObjectAvailableを受信し、SM2がTE1からMsgObjectAvailableを受信する前にネットワーク分断が発生する。SM3及びTE1はSM1及びSM2を疑い、SM1はSM3及びTE1を疑い、SM2はTE1を疑う（SM2はSM3について知らない）。SM3はまだコーラスに加入している最中である（TE1から加入の最終確認を受信していない）ので機能停止し、TE1及びSM1はコーラス {SM1, SM2, SM3, TE1} 内の2つのノードを疑っているので（フェーズ0で）機能停止する。SM2は最初はTE1のみを疑うので（ $n=3$ の場合、これはノード数の $n/2$ 未満である）、フェーズ0で機能停止せず、フェーズ1のラウンド1のメッセージをSM1に送信するが、SM1の機能停止について聞いた後、解決プロトコルを再開始したのち、自身を機能停止させる。

【0098】

シナリオ（D）：ネットワーク分断が発生して、新しいノード、エントリノード、及び一部のピアが残りのピアから分離される。

【0099】

図15では、SM3はTE1（マスターカタログのチェアマン）にマスターカタログを要求してTE1から受信し、SM1はTE1からMsgObjectAvailableを受信し、SM2がTE1からMsgObjectAvailableを受信する前に、ネットワーク分断によってSM2が残りのノードから分離される。SM3は、まだコーラ

10

20

30

40

50

スに加入している最中である（TE1から完了を受信していない）ので機能停止する。SM2は、コーラス {SM1, SM2, TE1} の少数派パーティション内にいるので機能停止する。TE1及びSM1はプロトコルを開始し、SM3から（ラウンド1）応答を受信せず、最終的にSM3を疑ったのち、自身らを機能停止させる。

【0100】

シナリオ（E）：ネットワーク分断によって、エントリノード（マスターカタログのチェアマン）が残りのノードから分離される。

【0101】

図16では、SM3はTE1（マスターカタログのチェアマン）にマスターカタログを要求してTE1から受信し、SM1及びSM2はTE1からMsgObjectAvailableを受信し、ネットワーク分断によって、エントリノードTE1が残りのノードから分断される。SM3は、まだコーラスに加入している最中であるので、機能停止する。TE1は、コーラス {SM1, SM2, SM3, TE1} の少数派パーティション内にいるので、機能停止する。SM1及びSM2は障害解決処理を開始し、SM3からの応答を受信せず、最終的にSM3を疑ったのち、自身らを機能停止させる。

10

【0102】

シナリオ（H）：ネットワーク分断によって、新しいノード、エントリノード、及び一部のピアが残りのノードから分離される。

【0103】

図17では、SM4はTE1（マスターカタログのチェアマン）にマスターカタログを要求してTE1から受信し、SM1及びSM3はTE1からMsgObjectAvailableを受信し、ネットワーク分断が発生してSM2が残りのノードから分離される。SM4はまだコーラスに加入している最中であるので、機能停止する。SM2は、コーラス {SM1, SM2, SM3, TE1} の少数派パーティション内にいるので、機能停止する。TE1、SM1、及びSM3は、コーラス {SM1, SM2, SM3, SM4, TE1} において多数派グループを形成するので、稼働を継続する。このケースでは、グループ {TE1, SM1, SM3} は、元のコーラス {TE1, SM1, SM2, SM3} において多数派であったが、新しいコーラス {TE1, SM1, SM2, SM3, SM4} においてもまだ多数派である。これにより、TE1、SM1、及びSM3は、自身らのノードリストにSM4が追加された後でも、稼働を継続することが可能になる。一般に、この動作は、（ネットワーク障害時に1つのノードがコーラスに加入しようとして）コーラスサイズが偶数のノード数から奇数のノード数に変化するときはいつでも発生する。

20

30

【0104】

シナリオ（I）：ネットワーク分断によって、新しいノード、エントリノード、及び一部のピアが残りのノードから分割される。

【0105】

図18では、SM4及びSM5はTE1（マスターカタログのチェアマン）にマスターカタログを要求してTE1から受信し、SM1及びSM3はTE1からSM4及びSM5の両方のMsgObjectAvailableを受信し、ネットワーク分断によってSM2が残りのノードから分離される。SM4及びSM5は、まだコーラスに加入している最中であるので機能停止し、SM2はコーラス {SM1, SM2, SM3, TE1} の少数派グループ内にいるので機能停止する。TE1、SM1、及びSM3も、コーラス {SM1, SM2, SM3, SM4, SM5, TE1} において少数派グループを形成するので機能停止する。シナリオ（H）で稼働を継続したノードTE1、SM1、及びSM3はここで機能停止し、その理由は、コーラスに加入しようとしているノードが2つであるために、これらのノードが新しいコーラスで少数派グループになるためである。

40

【0106】

概念的には、 n 個のノードを有するコーラスは、最大で $(n - (n / 2 + 1))$ 個のノードがコーラス内の残りのノードから分離されるネットワーク分断（またはコーラス内の最大で $(n - (n / 2 + 1))$ 個のノードの同時障害）を許容し、依然として稼働を継続

50

することができる。単一のノードがコーラスに加入しようとしている場合、 n が奇数の場合に、コーラスは $(n - (n / 2 + 1) - 1)$ 個のノードの分離を許容し、依然として稼働を継続することができる。単一の新しいノードの場合、 n が偶数の場合に、コーラスは $(n - (n / 2 + 1))$ 個の分離を許容し、依然として稼働を継続することができる。

【0107】

コーラスの障害許容数を、コーラス内の全てのノードが機能停止せずにコーラスが許容できる最大のノード障害数とする。 n 個のノードを有するコーラスでは、コーラスに加入する新しいノードがない場合、コーラスの障害許容数は $(n - (n / 2 + 1))$ である（表1の列1）。コーラスに加入しようとしているノードが1つである場合、コーラスの故障（*fault*）許容数は、 n が奇数の場合は $(n - (n / 2 + 1))$ に低下し、 n が偶数の場合は $(n - (n / 2 + 1))$ のままである（表1の列2）。コーラスに（同時に）加入しようとしている新しいノードの数が1より大きい場合、コーラスの障害許容数はさらに低下し得る。表1は、コーラス内の様々なノード数（ n ）と、同時にコーラスに加入しようとしている様々なノード数（ n_j ）についてのコーラスの障害許容数をまとめたものである。以下の表1では、同時にコーラスに加入しようとしている n_j 個のノードが存在し、多数派パーティション内の少なくとも1つのノードが n_j 個全てのノードの*MsgObj* *jectAvailable*を受信している。

	$n_j=0$	$n_j=1$	$n_j=2$	$n_j=3$	$n_j=4$	$n_j=5$	$n_j=6$	$n_j=7$	$n_j=8$
$n=3$	1	0	0	0	0	0	0	0	0
$n=4$	1	1	0	0	0	0	0	0	0
$n=5$	2	1	1	0	0	0	0	0	0
$n=6$	2	2	1	1	0	0	0	0	0
$n=7$	3	2	2	1	1	0	0	0	0
$n=8$	3	3	2	2	1	1	0	0	0
$n=9$	4	3	3	2	2	1	1	0	0
$n=10$	4	4	3	3	2	2	1	1	0

表1

【0108】

シナリオB、D、及びF（上記）の障害解決は、 $n = 3$ 及び $n_j = 1$ のテーブルエントリに取り込まれている。この構成のコーラス障害許容数はゼロであるので、（ノードの少なくとも1つが*MsgObj* *jectAvailable*を受信して）新しいノードが加入している最中に、ネットワーク分断（または任意のノード障害）が発生したことにより、コーラス全体が機能停止する。シナリオAは表1には取り込まれておらず、その理由は、シナリオAの多数派グループのいずれのノードも*MsgObj* *jectAvailable*を受信しなかったためである。シナリオHは、 $n = 4$ 及び $n_j = 1$ のエントリに取り込まれている。シナリオHのコーラス障害許容数は1である。コーラスは少数派パーティションに単一のノードを有するので、コーラスは稼働を継続する。シナリオIは、 $n = 4$ 及び $n_j = 2$ のエントリに取り込まれている。この構成のコーラス障害許容数はゼロであるので、ノードが加入している間のネットワーク分断によって、コーラス全体が機能停止する。

【0109】

ノード障害への対処

このセクションでは、分散データベースシステムがネットワーク障害を解決している間の1つまたは複数のノード障害（またはシャットダウン）への対処について論じる。上記で論じたように、ネットワーク障害イベントを解決する処理は、ノードが障害検出メッセージを交換することと、ノードが交換されたメッセージに基づいて稼働を継続すべきか否かを判断することと、稼働を継続すると決断したノードが被疑ノードのフェイルオーバー

を実行することと、を含む。この処理は図 19 に示している。

【0110】

図 19 では、コーラスはメンバー {A, B, C, D} を含む。ネットワーク分断によって {A, B, C} が D から分離される。ノード A、B、及び C はノード D を疑い、障害検出メッセージを交換し、稼働を継続すると決断し、D のフェイルオーバーを実行する。ノード D はノード A、B、及び C を疑い、障害解決プロトコルを開始し、自身を機能停止させる。

【0111】

ping (ハートビート) タイムアウトが有効になっている場合、ノード障害によって、障害が発生したノードの隣接機が障害解決プロトコルを開始 (または再開) し、障害が発生したノードを排除することについて合意し、障害が発生したノードをコーラスから排除する。分散データベースシステムがネットワーク障害イベントを解決している最中にノード障害が発生した場合、障害が発生したノードは、障害が発生したノードの隣接機にとって、新しい被疑機となって現れ得る。これにより、隣接機にプロトコルを再開させることができる。したがって、分断の解決中にノード障害に対処するための特別なメカニズムは存在しない。代わりに、本明細書に記載の処理により、ノード障害に応答して障害解決プロトコルを開始／再開したノードは、コーラスメンバーシップについて一致することを確実にする。

【0112】

障害検出メッセージの交換中のノード障害への対処

障害検出メッセージを交換している間にノードに障害が発生した場合、そのノードは自身の隣接機の被疑ノードリスト上に存在しなくなる。その結果、隣接ノードは、上記で論じた処理と同様に、コーラスメンバーシップ／コーラスサイズについて意見が一致することになる。ノード障害によって生じた新しい被疑機を検出したことに応答して、隣接機は更新された被疑リストを使用して障害解決処理を再開する。この更新された被疑リストは、ネットワーク障害によって生じた被疑ノードと、障害が発生したノードとの和集合である。隣接機は、更新された被疑リストに基づいて多数派グループを形成する場合、稼働を継続する。

【0113】

図 20 では、ネットワーク分断によって {A, B, C} が D から分離され、ノードがメッセージを交換している間に、ノード C に障害が発生する。ノード A 及び B は、C を疑うと、プロトコルを再開する。A 及び B は、コーラス {A, B, C, D} において多数派グループを形成しないので、機能停止する。

【0114】

フェイルオーバー実行中のノード障害への対処

ノードがフェイルオーバーを実行している間に (機能停止したノードをコーラスメンバーシップリストから削除している間に) ノードに障害が発生した場合、その隣接機は他の被疑ノードのフェイルオーバーを開始または完了している場合がある。その結果、隣接機は、自身らのノードリストから 1 つまたは複数の被疑ノードを削除している場合があるので、隣接機はプロトコルの開始／再開時にコーラスメンバーシップ／コーラスサイズが一致しない場合がある。

【0115】

図 21 では、コーラスはメンバー {A, B, C, D} を含む。ネットワーク分断によって、{A, B, C} が D から分離される。ノード A、B、及び C は、プロトコルを開始し、障害検出メッセージを交換し、稼働を継続すると決断する。ノード A、B、及び C はノード D のフェイルオーバーを開始する。A が D のフェイルオーバーを完了した後 (及び D を自身のノードリストから削除した後)、B がまだ D のフェイルオーバーを実行している間に、ノード C に障害が発生する。これにより、A は C を疑い、コーラス {A, B, C} 及び被疑リスト {C} でノード障害処理を開始する。また、それによって、B はコーラス {A, B, C, D} 及び被疑リスト {C, D} でノード障害処理を開始する。その結果、

10

20

30

40

50

A及びBはコーラスメンバーシップについて一致しない。

【0116】

この場合、次のようにして、ノードはコーラスサイズについて一致するようになる。

【0117】

機能停止していないノードは、ブロードキャストのラウンド1及び2の間に、自身らの完全な接続情報（すなわち、自身らの隣接ノードリストならびに自身らの被疑ノードリスト）を交換する。ノードは、ラウンド1／ラウンド2メッセージを受信した後、自身らの被疑ノードリスト及び隣接ノードリストを自身らの隣接機の被疑ノードリスト及び隣接ノードリストと比較する。ノードは、自身が認識していない n_j 個のノードについて自身の隣接機が認識していることに気づいた場合、自身のコーラスサイズを n_j だけインクリメントし、障害解決処理を再開始する。

10

【0118】

したがって、 n を多数派パーティション内のノード数とし、 f を機能停止したノード数とし、 e をフェイルオーバーが実行されている排除されるノード数とした場合、 $(n - f) \geq (s / 2 + 1)$ であれば、パーティション内のノードは稼働を継続することになり、ここで $(n \leq s \leq n + e)$ である。

【0119】

しかしながら、ノードが障害解決処理を実行している間に、ノードでフェイルオーバーが完了したらどうなるだろうか。コーラスメンバーの稼働を維持する可能性を高めるために、障害解決処理に次の変更を加えることができる。ノードが排除されるノードのフェイルオーバーを完了した後、障害解決処理を再開始し（1つが進行中の場合）、これにより、この処理はより小さいコーラスサイズで実行される。

20

【0120】

ノードがより小さいコーラスサイズで再開始した場合に全てのノードがコーラスサイズについて一致するようにするために、この処理は次のようにさらに拡張することができる。ノードはブロードキャストのラウンド1及び2の間に、自身らの完全な接続情報（すなわち、自身らの隣接ノードリストならびに自身らの被疑ノードリスト）を交換する。次に、ノードは、自身らの被疑ノードリスト及び隣接ノードリストを、自身らの隣接機の被疑ノードリスト及び隣接ノードリストと比較する。ノードは、自身が認識していない n_j 個のノードについて自身の隣接機が認識していることに気づいた場合、自身のコーラスサイズを n_j だけインクリメントし、処理を再開始する。その後、ノードの隣接機が自身のコーラスリストから r_j 個のノードを削除して処理を再開始した場合、ノードは自身のコーラスサイズを r_j だけデクリメントし、処理を再開始する。

30

【0121】

これにより、ノードはコーラスサイズについて一致するが、疑わしいメンバーシップを有するノードと新しいノードとが機能停止している限り、ノードはコーラスメンバーシップ（またはコーラスサイズ）について一致する必要はない。言い換えると、各ノードは、そのノードのマスターカタログノードリストによって決定されたコーラスメンバーシップに基づいて、障害解決処理を実行することができる。この処理では、メンバーシップが合意されていないノードが、処理の開始前に機能停止されるか、処理中に機能停止される限り、全てのノードが正しい結果に到達することが保証される。

40

【0122】

これが成り立つ理由を理解するために、 $n + n_j$ をコーラス内のノード数とする。 n はマスターカタログが完全であるノード数であり、 n_j は機能停止したノード数と、機能停止することになるノード数（ノード障害のケースのように、これらのノードのマスターカタログは機能停止時に完全である場合があり、完全でない場合もある）、または障害解決プロトコルを開始すると機能停止することになる新しいノードの数（ノード加入のケースのように、これらのノードのマスターカタログは機能停止時に完全にならない）との合計である。

【0123】

50

s を障害解決プロトコルに参加するノードのマスターカタログノードリストのサイズとすると、 $n \leq s \leq n + n_j$ である。なお、 s が障害解決プロトコルに参加する全てのノードと同じではない場合があることに留意されたい。

【0124】

障害解決プロトコルは、各ノードが自身のマスターカタログノードリストサイズに設定されたコーラスサイズを使用してプロトコルを実行する場合、最大で1つのパーティション内のノードが稼働を継続することを保証できるだろうか。保証できるが、その理由は、 $n \leq s \leq n + n_j$ であるので、各ノードによって計算される多数派グループサイズは少なくとも $(n / 2 + 1)$ であるためである。パーティション内の各ノードが多数派グループ内にいる $(n / 2 + 1 \leq \text{多数派グループサイズ} < (n + n_j) / 2 + 1)$ と結論付けることができる場合、そのパーティションは少なくとも $(n / 2 + 1)$ 個のノードを有する。プロトコルに参加しているノードは n 個のみであるので、そのようなパーティションを最大で1つにすることができる。したがって、最大で1つのパーティション内のノードは、障害解決プロトコルを正常に完了し、稼働を継続することができる。

【0125】

パーティション内の全てのノードが、そのパーティションが勝利全接続コンポーネントになるために、自身が多数派全接続コンポーネント内にいると結論付ける必要はない。パーティション内のノードのサブセットは、自身らのマスターカタログノードリストサイズに応じて、自身らが多数派グループ内にいないと結論付け得る。これらのノードは、処理（図12）のステージ2の間に機能停止して、そのパーティション内の残りのノードが処理を再開する。しかしながら、残りのノードが処理の再開時に自身らが多数派グループ内にいると結論付けることができる場合は、その全接続コンポーネントを勝利全接続コンポーネントにするために、それで十分であり得る。

【0126】

追加の性能目標の達成

ユーザがコーラスメンバーノードの半数以上をシャットダウンすることを選択した場合、障害検出をトリガーすることができない。これは、手動でシャットダウンされたノードを被疑ノードとして扱わないように障害解決処理を修正することで実現される。

【0127】

シャットダウン中のノードの識別

管理レイヤからシャットダウン要求を受信すると、ノードは、（たとえば、ノード状態 `NODE__STATE__SHUTTING__DOWN` によって）自身がシャットダウン中であることを示すメッセージノード状態 (`MsgNodeState`) メッセージをブロードキャストする。分散データベースシステム内の管理レイヤは、ユーザが分散データベースとやりとりできるノードのレイヤである。管理レイヤは、分散データベースシステム内のノードを追跡することができ、ユーザと分散データベースシステム内のノードとの間のやりとりを容易にすることができる。たとえば、ユーザがノードをシャットダウンしたい場合、ユーザは管理レイヤにシャットダウンコマンドを出すことができ、次いで管理レイヤは、ユーザが指定したノードにシャットダウンメッセージを送信する。この処理は、少なくとも1つのコーラスメンバーが、シャットダウン中のノードからこのノード状態メッセージを受信することに依存している。

【0128】

障害解決プロトコルの変更

障害解決プロトコルに以下の変更を加えることができる。

(a) 処理（図12）のステージ1の間に、シャットダウン中であることがわかっているノードを被疑ノードとは見なさない。

(b) コーラスメンバーに、シャットダウン中のノードのゴシップをさせる（たとえば、処理のステージ3及び4の間に障害検出メッセージを交換する）。

【0129】

どのようにしてこれらの変更が手動でシャットダウンされているノードを識別したいと

いう要望を満たすことができるかについての一例を以下に示す。ノードA、B、C、及びDを有するコーラスについて考察する。ユーザがノードC及びDをほぼ同時にシャットダウンすると仮定する。ノードAのみがCからノード状態メッセージを受信し、ノードBのみがDからノード状態メッセージを受信すると仮定する。ノードAは、コーラス{A、B、C、D}、被疑リスト{D}、及びシャットダウン中ノードリスト{C}で障害解決処理を開始し、ラウンド1障害検出メッセージをBに送信する。ノードBは、コーラス{A、B、C、D}、被疑リスト{C}、及びシャットダウン中ノードリスト{D}でプロトコルを開始し、ラウンド1障害検出メッセージをAに送信する。障害検出メッセージを受信したことに応答して、ノードAは自身のシャットダウン中ノードリストを{C、D}に更新し、被疑リストを{}に更新し、プロトコルを再開する。ノードBも同様にする。ラウンド1の後、ノードA及びBは、コーラスサイズ=4及び被疑ノードリストサイズ=0に基づいて、自身らが多数派のパーティション内にいると結論付け、稼働を継続する。

【0130】

しかしながら、ノードがシャットダウンしている間にネットワーク分断またはリンク障害が発生した場合、修正されたプロトコルはどのようにして正しい処理に到達するだろうか。このシナリオを考察する。コーラスはノードA、B、C、D、及びEを含む。ユーザはノードEをシャットダウンし、ほぼ同時にネットワーク分断によって{A、B}が{C、D}から分離される。全てのノードがEからノード状態メッセージを受信すると仮定する。ノードAはコーラス{A、B、C、D、E}、被疑リスト{C、D}、及びシャットダウン中ノードリスト{E}でプロトコルを開始し、ラウンド1障害検出メッセージをBに送信する。ノードBも、コーラス{A、B、C、D、E}、被疑リスト{C、D}、及びシャットダウン中ノードリスト{E}でプロトコルを開始し、ラウンド1障害検出メッセージをAに送信する。障害検出メッセージを受信すると、ノードA及びBは、(コーラスサイズ=5及び被疑ノードリストサイズ=2に基づいて)自身らが多数派のパーティションにいると結論付け、稼働を継続する。ノードC及びDも同じロジックで稼働を継続する。次のアプローチにより、このシナリオでプロトコルが正しい処理に到達することを保証することができる。ノード(複数可)がシャットダウンしている間にネットワーク分断(またはリンク障害)が発生した場合、シャットダウン中のノードを被疑ノードとして扱う。

【0131】

要約すると、ユーザがコーラスメンバーノードの半数以上をシャットダウンする場合(NDをこのノードセットとする)、この処理により、以下の条件が成り立つ場合、残りのノード(NSDをこのノードセットとする)が稼働を継続することになる。

(A) NSDの少なくとも1つのノードは、SDの各ノードからノード状態変更メッセージを受信する。

(B) NSDの各ノードは、SDの少なくとも1つのノードからノード状態メッセージを受信する。(この理由は、NSDのノードがSDのいずれのノードからもノード状態メッセージを受信しない場合、そのノードはSDの全てのノードを疑うことになり、他のノードからのシャットダウン中のノードについて知る前に、ステージ2で自身を機能停止させることになるためである。)

(C) NSDのノードが障害解決プロトコルを完了し、SDのノードをコーラスから排除するまで、ネットワーク分断もリンク障害も発生しない。

【0132】

結び

様々な本発明の実施形態を本明細書で説明及び図示してきたが、本明細書に記載の機能を実行するための、ならびに／あるいは結果及び／または利点のうちの1つもしくは複数を得るための様々な他の手段及び／または構造を当業者は容易に想像すると思われ、また、そのような変形例及び／または変更例のそれぞれは、本明細書に記載の本発明の実施形態の範囲内にあると考えられる。より一般的には、本明細書に記載の全てのパラメータ、寸法、材料、及び構成は典型的なものであることが意図されており、実際のパラメータ、

10

20

30

40

50

寸法、材料、及び／または構成は、本発明の教示が使用される1つまたは複数の特定の用途に依存することを当業者は容易に理解するであろう。当業者は、ほんの日常的な実験によって、本明細書に記載した本発明の特定の実施形態に対する多くの均等物を認識または確認することができるであろう。したがって、前述の実施形態は単なる例として示しており、添付の特許請求の範囲及びその均等物の範囲において、本発明の実施形態が、具体的に説明し、特許請求した以外の方法で実施され得ることを理解されたい。本開示の発明の実施形態は、本明細書に記載した各個別の特徴、システム、物品、材料、キット、及び／または方法に向けられている。加えて、2つ以上のそのような特徴、システム、物品、材料、キット、及び／または方法の任意の組み合わせは、そのような特徴、システム、物品、材料、キット、及び／または方法が相互に矛盾していない場合には、本開示の発明の範囲に含まれる。

10

【0133】

また、種々の発明概念は、1つまたは複数の方法として具現化され得、その一例を提供している。方法の一部として実行される行為は、任意の適切な方法で並べられ得る。したがって、図示とは異なる順序で行為が実施される実施態様が構築され得、これには、いくつかの行為を、例示的な実施形態では順次的な行為として示していたとしても、同時に行うことが含まれ得る。

【0134】

本明細書で定義及び使用する全ての定義は、辞書の定義、引用により組み込まれた文献における定義、及び／または定義した用語の通常の意味に優先するものと理解されたい。

20

【0135】

不定冠詞「a」及び「an」は、本明細書及び特許請求の範囲で使用する場合、明瞭にそうでないという指示がない限り、「少なくとも1つ」を意味するものと理解されたい。

【0136】

「及び／または」という語句は、本明細書及び特許請求の範囲で使用する場合、そのように等位接続された要素の「一方または両方」、すなわち、あるケースでは連言的に存在し、他のケースでは選言的に存在する要素を意味するものと理解されたい。「及び／または」を用いて列記された複数の要素は、同様に、すなわち、そのように等位接続された要素のうちの「1つまたは複数」として解釈されたい。「及び／または」の節によって具体的に特定した要素以外の他の要素が、具体的に特定したそれらの要素に関係しているか関係していないかに関わらず、任意選択で存在してもよい。したがって、非限定的な一例として、「A及び／またはB」への言及は、「備える」などのオープンエンドの文言と共に使用する場合、一実施形態ではAのみ（任意選択でB以外の要素を含む）を指し、他の実施形態ではBのみ（任意選択でA以外の要素を含む）を指し、さらに他の実施形態ではA及びBの両方（任意選択で他の要素を含む）などを指すことができる。

30

【0137】

本明細書及び特許請求の範囲で使用する場合、「または」は、上記で定義した「及び／または」と同じ意味を有するものと理解されたい。たとえば、リスト内の項目を分ける場合、「または」あるいは「及び／または」は包括的であると解釈され、すなわち、いくつかの要素または要素のリスト、及び任意選択で列記されていないさらなる項目のうちの少なくとも1つを含むだけでなく、2つ以上も含むと解釈されるものとする。明瞭にそうでないという指示がある用語、たとえば、「～のうちの1つのみ」もしくは「～のうちのちょうど1つ」、または特許請求の範囲で使用する場合の「～から成る」のみが、いくつかの要素または要素のリストのうちのちょうど1つの要素を含むことを指す。全般的に、「または」という用語は、本明細書で使用する場合、排他性の用語、たとえば、「～のいずれか」、「～のうちの1つ」、「～のうちの1つのみ」、または「～のうちのちょうど1つ」などを伴う場合には、排他的な選択肢（すなわち、「一方または他方であるが両方ではない」）のみを示すと解釈されるものとする。「本質的に～から成る」は、特許請求の範囲で使用する場合には、特許法の分野で用いられる場合のその通常の意味を有するものとする。

40

50

【0138】

本明細書及び特許請求の範囲で使用する場合、1つまたは複数の要素のリストに関連する「少なくとも1つ」という語句は、その要素のリスト内の要素のうちの任意の1つまたは複数から選択された少なくとも1つの要素を意味するものと理解されるべきであり、必ずしもその要素のリスト内に具体的に列記されたありとあらゆる要素のうちの少なくとも1つを含む必要はなく、その要素のリスト内の要素のあらゆる組み合わせを除外するものでもない。また、この定義は、「少なくとも1つの」という語句が指す要素のリスト内に具体的に特定した要素以外の要素が、具体的に特定したそれらの要素に関係しているか関係していないかに関わらず、任意選択で存在してもよいということも認める。したがって、非限定的な一例として、「A及びBのうちの少なくとも1つ」（もしくは同義的に「AまたはBのうちの少なくとも1つ」、または同義的に「A及び／またはBのうちの少なくとも1つ」）は、一実施形態では、少なくとも1つのAがあり、任意選択で2つ以上のAを含み、Bは存在しないこと（及び任意選択でB以外の要素を含む）、他の実施形態では、少なくとも1つのBがあり、任意選択で2つ以上のBを含み、Aは存在しないこと（及び任意選択でA以外の要素を含む）、さらに他の実施形態では、少なくとも1つのAがあり、任意選択で2つ以上のAを含み、少なくとも1つのBがあり、任意選択で2つ以上のBを含むこと（及び任意選択で他の要素を含む）、などを指す場合がある。

【0139】

特許請求の範囲ならびに上記の明細書では、全ての移行句、たとえば、「備える」、「含む」、「運ぶ」、「有する」、「包含する」、「伴う」、「保持する」、「～から成る」などは、オープンエンドであることを、すなわち、「～を含むがこれらに限定されない」を意味することを理解されたい。「～から成る」及び「本質的に～から成る」という移行句のみが、それぞれクローズド移行句またはセミクローズド移行句であるものとし、これらについては米国特許庁の特許審査便覧、セクション2111.03に記載されている。

10

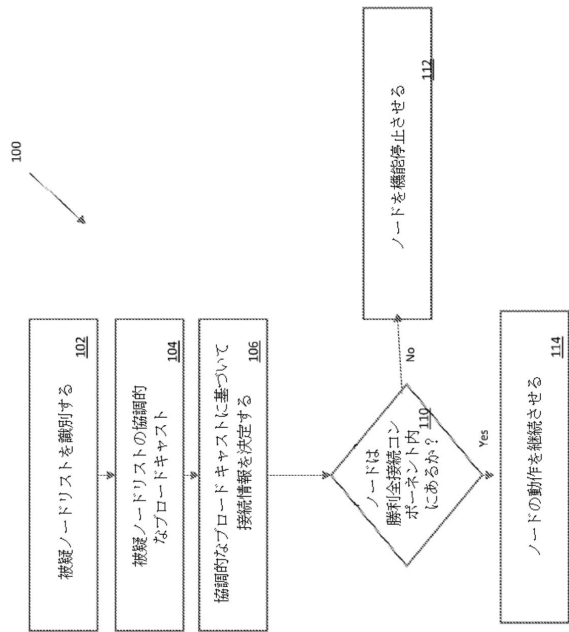
20

30

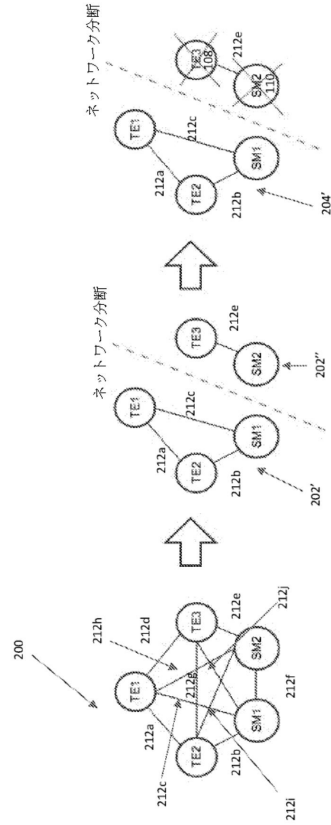
40

50

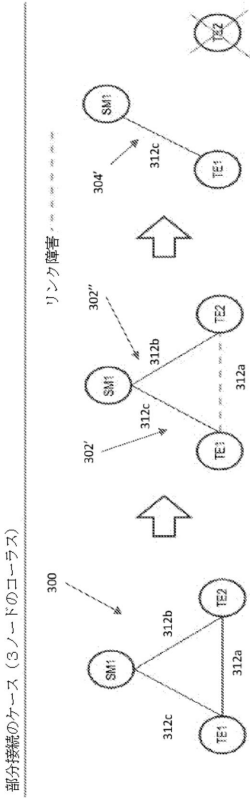
【図面】
【図 1】



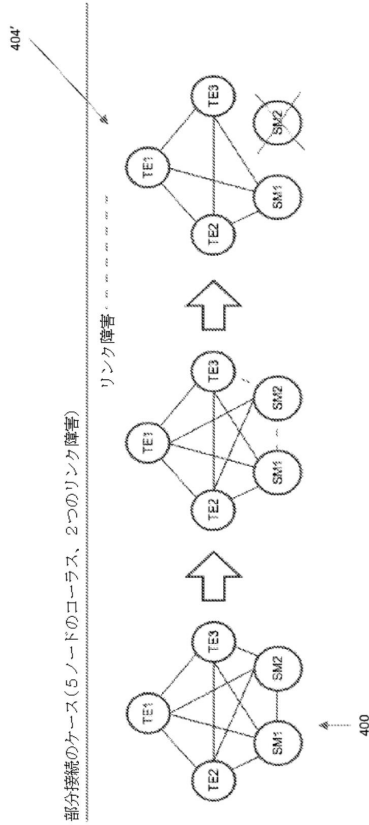
【図 2】



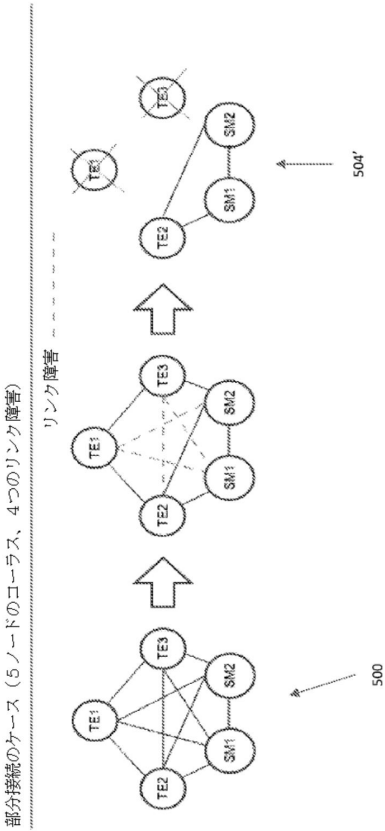
【図 3】



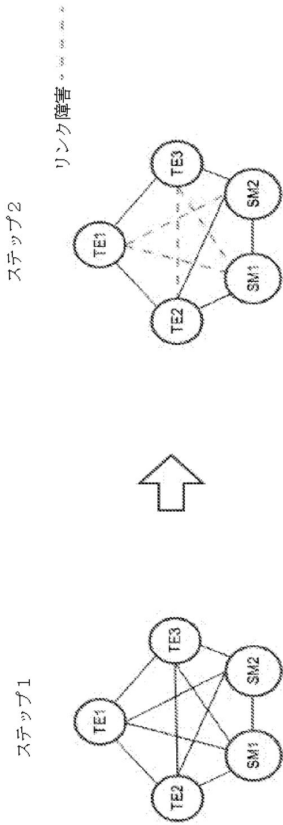
【図 4】



【図 5】



【図 6】



10

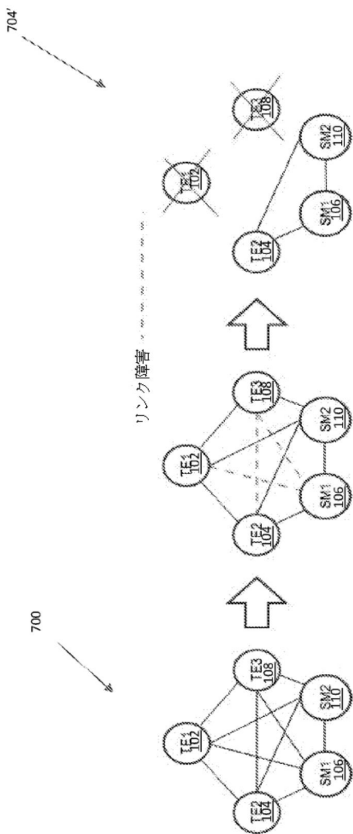
20

30

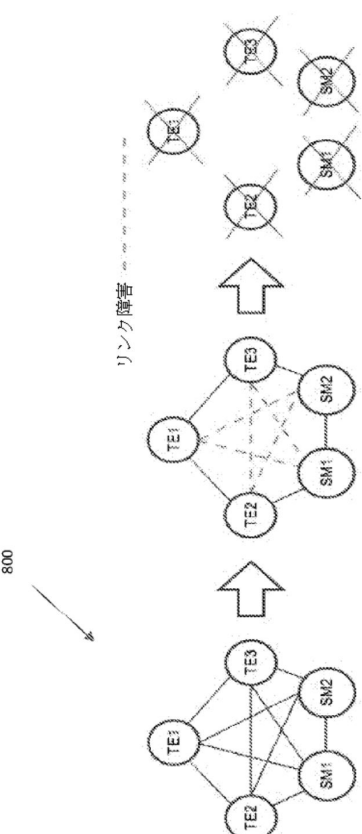
40

50

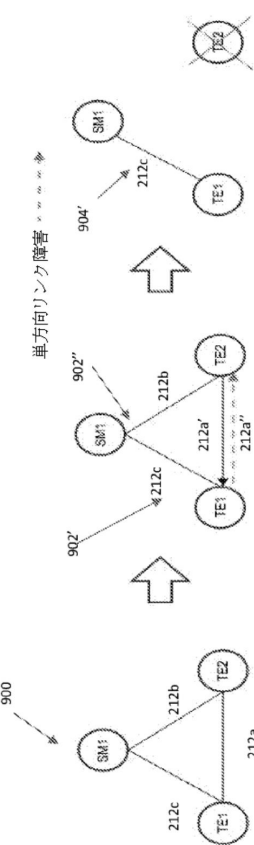
【図 7】



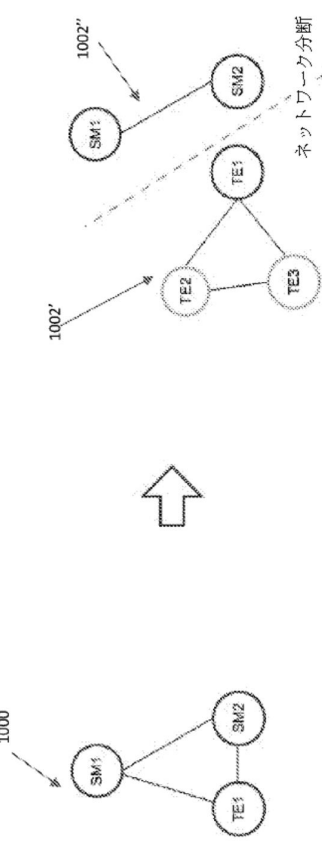
【図 8】



【図 9】



【図 10】



10

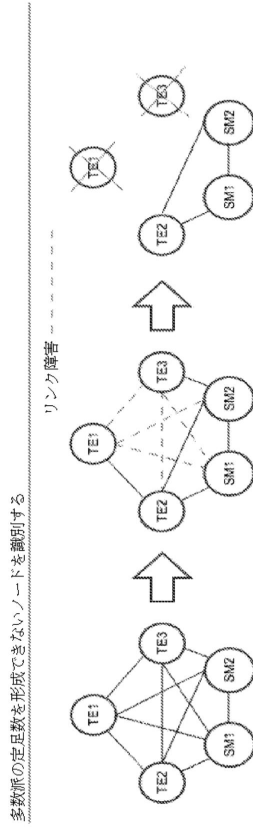
20

30

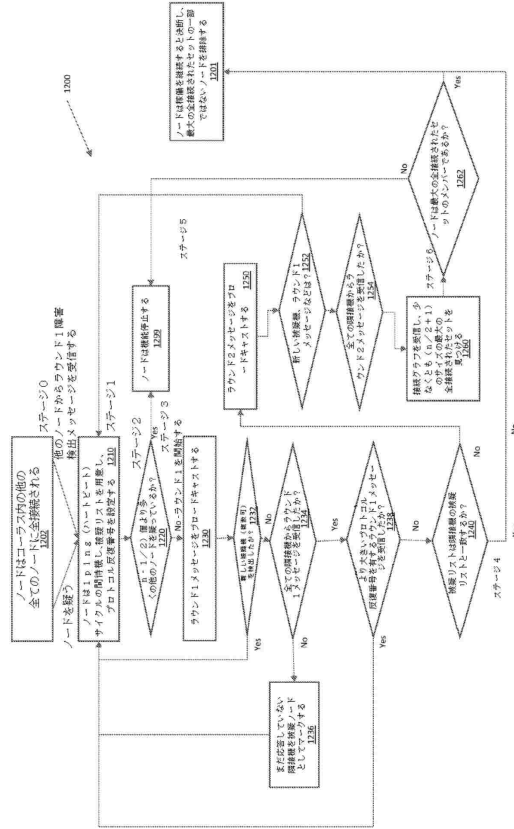
40

50

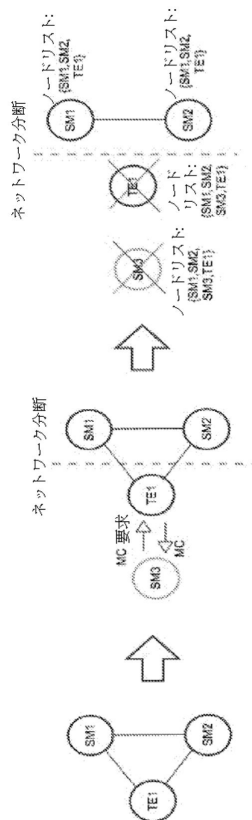
【図 1 1】



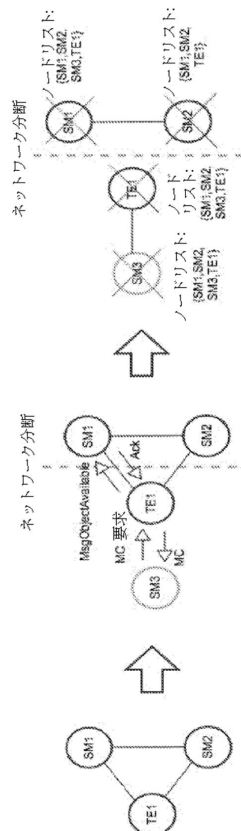
【図 12】



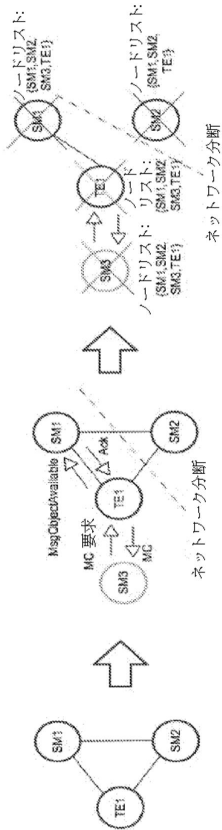
【図 13】



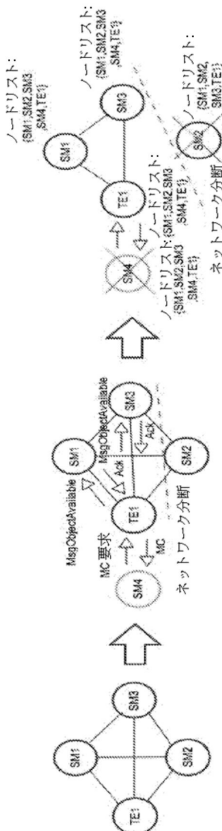
【図 14】



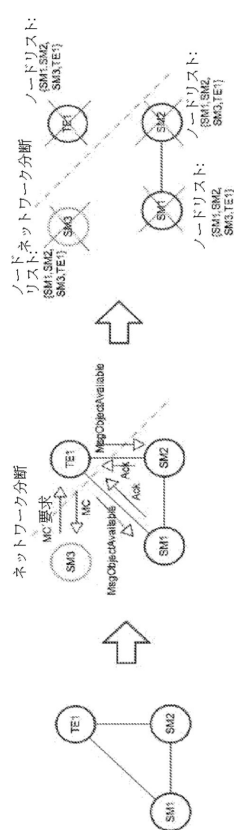
【図 15】



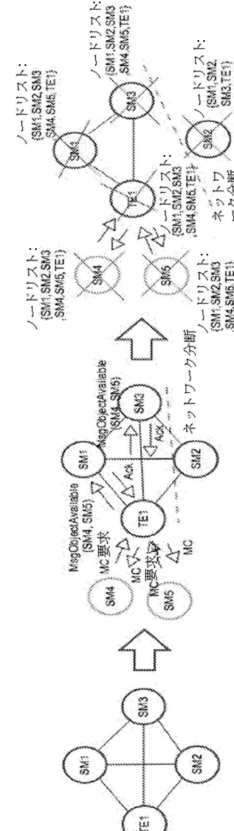
【図 17】



【図 16】



【図 18】



10

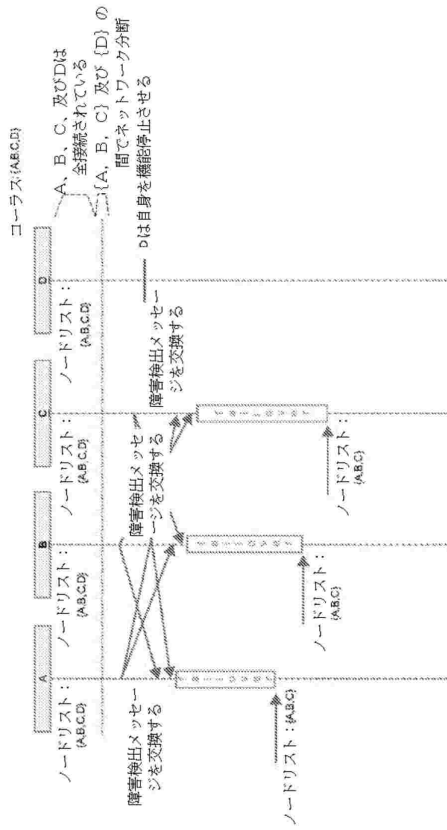
20

30

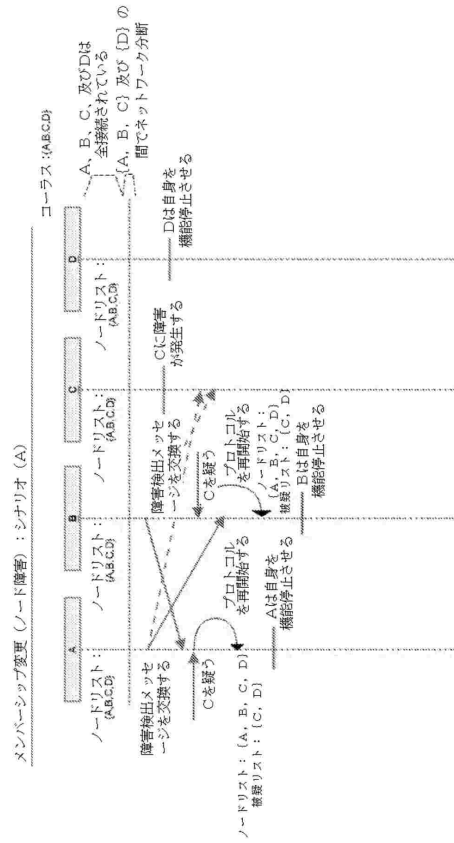
40

50

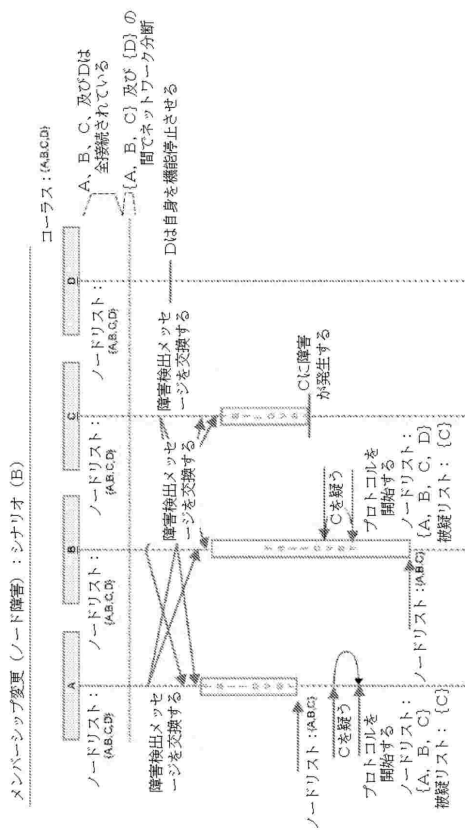
【図 19】



【図 20】



【図 2 1】



フロントページの続き

- (74)代理人 上杉 浩
- (74)代理人 100120525
弁理士 近藤 直樹
- (74)代理人 100139712
弁理士 那須 威夫
- (74)代理人 100151987
弁理士 谷口 信行
- (72)発明者 ボダガラ スリーナト
アメリカ合衆国 ニューハンプシャー州 03060 ナシュア ニュー キャッスル ドライブ 7
アパートメント 4
- (72)発明者 シャウル ロス
アメリカ合衆国 マサチューセッツ州 01776 サドベリー ロビンズ ロード 62
- (72)発明者 スミス ポール ディー
アメリカ合衆国 マサチューセッツ州 02421 レキシントン シェード ストリート 153
- 審査官 羽岡 さやか
- (56)参考文献 特表2012-504807 (JP, A)
特表2016-520904 (JP, A)
米国特許出願公開第2018/0276251 (US, A1)
米国特許出願公開第2010/0262717 (US, A1)
特表2016-524846 (JP, A)
- (58)調査した分野 (Int.Cl., DB名)
H04L 12/00-69/00
G06F 16/00