Vol.2009-OS-112 No.12
2009/8/6

単一IPアドレスクラスタにおける耐故障機構の設計と実装

加藤 純[†1] 藤田 肇[†1] 石川 裕[†1]

本研究では，TCP アプリケーション向けに単一 IP アドレスクラスタを実現する FTCS 機構を対象としている．FTCS 機構は TCP 接続要求処理中に動作を行い，クライアントからの接続ごとに対応する一意のノードを選び接続の確立を行う．しかし，故障が発生した場合にクライアントからの接続に対して複数のノードが接続の確立を行う問題を抱えている．そこで，本研究では耐故障性とともにこの問題の解決を行う 3 つのアルゴリズム，Discarding,Gathering,Scattering を提案する．Discarding アルゴリズムでは故障時に TCP 接続要求処理中の接続の破棄を行う．Gathering アルゴリズムでは故障時に TCP 接続要求処理中の接続に関する情報の収集を行う．Scattering アルゴリズムでは常に TCP 接続要求処理中の接続に関する情報の同期を行う．

The Design and Implementation of Fault Tolerant Single IP Address Cluster

Jun Kato ,[†1] Hajime Fujita [†1]
and Yutaka Ishikawa [†1]

This paper focuses on the FTCS mechanism, which constructs a single IP address cluster for TCP applications. It handles a TCP connection request and establishes the TCP connection between the client and the unique node of the cluster. When a failure occurs, TCP connection can be duplicated in nodes. In order to prevent this issue, three algorithms, Discarding, Gathering, Scattering are proposed for solving this problem with fault tolerance. In the Discarding algorithm, processing TCP connection request is discarded in the case of failure. The Gathering algorithm collects information about processing TCP connection requests. The Scattering algorithm always synchronizes information of processing TCP connection requests.

1. はじめに

近年，ハードウェア技術の進歩による高速化と価格低下により，安価に高性能な計算機が導入できるようになった．Web サーバやストリーミングサーバなど，クライアント数に応じて要求される性能が増減するようなサーバをクラスター構成にすることにより，拡張性と対価格性能に優れたサーバが実現できる．

このようなクラスタでは，単一のドメイン名，IP アドレスでアクセスできるようにラウンドロビン DNS[11)] や単一 IP アドレスクラスタ[1)–3),8)–10),16)–18)] などの研究開発が行われてきた．通常のクラスタでは，クライアントは各ノードに割り振られた独自の IP アドレスを用いる．ラウンドロビン DNS ではこれらノードごとに割り当てられた IP アドレスと単一のドメイン名を関連付けることでクラスタへの単一ドメイン名によるアクセスを実現している．これに対して，単一 IP アドレスクラスタはクラスタノード全体に単一の IP アドレスを割り当てることでアクセスの利便性を向上させるクラスタである．ラウンドロビン DNS においてクラスタ外部で行われていた負荷分散を単一 IP アドレスクラスタではクラスタ内部で行うことができる．そのため，高負荷の環境下でもサーバの処理を均等に分散することができる．

クライアント数の増大に伴いクラスタの性能を上げるためには，ノード数を増大しなければならない．ノード数の増大により故障率が増大し，可用性が低下する．ノード数増大に伴う可用性の低下を防ぐために，本研究では，単一 IP アドレスクラスタに耐故障性を付与する機構を提案する．

単一 IP アドレスクラスタとしては，TCP アプリケーション向けに単一 IP アドレスクラスタを実現する FTCS 機構[3)] を対象とする．FTCS 機構は TCP 接続要求処理時に接続要求に応答する一意のノードを決めることで，単一 IP アドレスクラスタを実現している．しかし，ノードが故障した場合に一意に応答するノードを決められない問題を抱えている．本研究では耐故障性とともにこの問題の解決を行う 3 つのアルゴリズム，Discarding,Gathering,Scattering を提案する．Discarding アルゴリズムでは，TCP 接続要求処理中の接続を破棄することでこの問題の解決を行う．Gathering アルゴリズムでは，TCP 接続要求処理中の接続に関する情報を集めることで応答するノードの重複を防ぐ．Scattering

†1 東京大学
The University of Tokyo

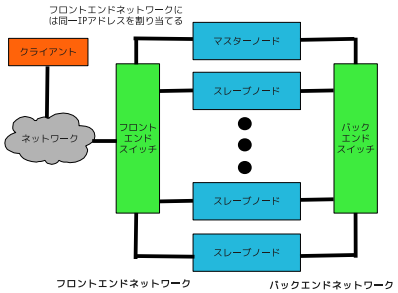

図 1 FTCS 機構の動作環境

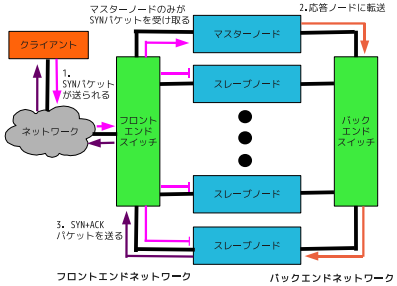

図 2 FTCS 機構の処理の流れ その 1

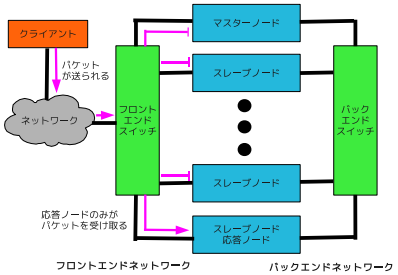

図 3 FTCS 機構の処理の流れ その 2

アルゴリズムでは，正常動作時にノード間で TCP 接続要求処理中の接続に関する情報の同期を行うことで問題解決を図る．

2. 単一 IP アドレスクラスタ FTCS 機構

2.1 環境と動作概略

FTCS 機構の動作環境を図 1 に示す．各ノードはマスターノードとスレーブノードの 2 種類に分類される．マスターノードはクラスタ内で多くても 1 つと約束する．各ノードはそれぞれフロントエンドネットワーク側にフロントエンドスイッチ，バックエンドネットワーク側にバックエンドスイッチの 2 つのスイッチに接続されている．フロントエンドスイッチはクライアント-クラスタ間のスイッチ，バックエンドスイッチはクラスタ内部での通信用スイッチとして使われる．フロントエンドスイッチはクライアント側からパケットが届けられた場合に全ノードに転送するように設定しておく．FTCS 機構は単一 IP アドレスクラスタを実現するために，各ノードのフロントエンドネットワーク側の IP アドレスには同一の IP アドレスを割り当てている．各ノードのバックエンドネットワーク側にはノード間通信用として異なる IP アドレスを割り当てる．

SYN+ACK パケットをクライアントに届けるまでの FTCS 機構における TCP 接続要求処理の流れを図 2 に示す．その処理は次の通りである．

(1) まずクライアントは TCP 接続処理を行う際 TCP プロトコル[12] に従い SYN パケットをクラスタに送る．FTCS 機構の設定によりフロントエンドスイッチにて全ノードに SYN パケットが送られる．この SYN パケットはマスターノードのみが受け取り，スレーブノードは破棄する．SYN パケットを受け取ったマスターノードはこの接続要求に応答するノード（以後，応答ノードとする）の決定を行う．

(2) 応答ノードが決定すると，バックエンドネットワークを通して応答ノードに SYN パケットを転送する．

(3) SYN パケットを受け取った応答ノードは SYN-RECEIVED 状態に遷移を行い，続きの TCP 接続要求処理を行うため SYN+ACK パケットをクライアントに送る．

クライアントから届けられる ACK パケットの処理と接続確立後のパケット処理の流れを図 3 に示す．SYN+ACK パケットを受け取ったクライアントは TCP プロトコルに従い ACK パケットをクラスタに送る．SYN パケット同様 ACK パケットもフロントエンドスイッチにより全ノードに送られる．この ACK パケットは SYN-RECEIVED 状態に遷移しているノード，すなわち応答ノードのみが受け取りこれにより TCP 接続要求処理が完了する．他ノードは ACK パケットを破棄する．TCP 接続要求完了後に届けられるパケットは ACK パケットと同様に処理する．すなわち，フロントエンドスイッチにより全ノードにパケットが送られるがクライアントとの接続が確立した応答パケットのみが応答する．

2.2 故障問題

FTCS 機構は SYN パケット転送履歴をマスターノードが保持している．これにより，ある接続に対して 2 つ以上の応答ノードがあること，つまり SYN-RECEIVED 状態に遷移するノードが複数存在することを以下の通り防いでいる．SYN パケット転送履歴がない場合に応答ノードが重複してしまう場合の処理の流れを図 4 に示す．処理の流れは次の通りである．

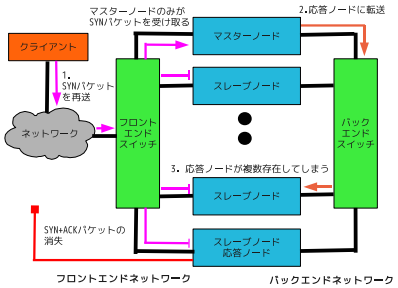

図 4 応答ノードの重複

(1) 応答ノードが送る SYN+ACK パケットが消失すると，クライアントは一定時間経過後に SYN パケットをクラスタに再送する．再送された SYN パケットもフロントエンドスイッチにより全ノードに届けられるが，マスターノードのみがこれを受け取る．マスターノードは応答ノードの決定を行うが，SYN パケット転送履歴を保持していないため前回選ばれた応答ノードが分からずに別のノードを応答ノードとして選択する．

(2) マスターノードは新しく選ばれた応答ノードに SYN パケットを転送する．

(3) 新しく選ばれた応答ノードが SYN パケットを受け取り SYN-RECEIVED 状態に遷移を行うと，応答ノードが重複することになる．

このようにして，応答ノードが重複してしまう．応答ノードが複数存在すると，クライアントから ACK パケットが届けられた場合に接続を確立するノードが複数存在することになる．これは TCP プロトコルが 1 対 1 通信であるということに矛盾する．このような応答ノードの重複を防ぐために，FTCS 機構ではマスターノードが SYN パケット転送履歴を保持している．SYN パケット転送履歴を利用することで，マスターノードは再送された SYN パケットを同一の応答ノードに転送することができる．このため，応答ノードの重複を防ぐことができる．

FTCS 機構に耐故障性を付与する際の問題点として，マスターノード故障に伴う SYN パケット転送履歴の消失が挙げられる．FTCS 機構では，マスターノードのみが SYN パケット転送履歴を保持していた．そのため，マスターノードが故障してしまうと SYN パケット転送履歴も同時に消失することになる．SYN パケット転送履歴が消失してしまうと，再送された SYN パケットを同一の応答ノードに転送する保証ができなくなる．つまり，再び応答ノードが重複する可能性が浮上する．

例えば，応答ノードからの SYN+ACK パケットが消失かつマスターノードが故障した場合を考える．マスターノードが故障していなければ SYN パケット転送履歴によって再送された SYN パケットを同一の応答ノードに転送することができる．ところがマスターノードの故障に伴い SYN パケット転送履歴が消失してしまうので，新しいマスターノードでは同一の応答ノードに再送された SYN パケットを転送することができない．従って，応答ノードが重複してしまうことがある．

3. 提 案 手 法

本研究では FTCS 機構に故障検出機構，故障からの回復機構，マスターノード故障後に応答ノードを重複させない機構 (以後，耐故障化同一応答ノード選択機構とする) の 3 つの機構を付与することにより耐故障性を実現する．故障検出機構と回復機構の連携によりノード故障からの耐故障性の実現を行い，耐故障化同一応答ノード選択機構により FTCS 機構特有の故障問題の解決を行う．

本研究ではこれら 3 つの機構を付与するアルゴリズムとして Discarding,Gathering,Scattering の 3 つのアルゴリズムを提案する．これら 3 つのアルゴリズムは同一の回復機構を用いる．故障検出機構では Discarding,Gathering アルゴリズムは同一の機構を用いるが，Scattering アルゴリズムでは別の機構を用いる．耐故障化同一応答ノード選択機構はそれぞれのアルゴリズムで異なる機構を用いる．なお，本研究ではビザンチン耐故障[5] は考慮せず，動的ノード追加，故障ノードの復帰も考慮しない．また，故障としてはノード故障のみを考慮する．

3.1 故障検出機構

故障検出機構とはバックエンドネットワークでの通信を用いることでノード故障を検出する機構である．故障検出機構はマスターノード故障検出とスレーブノード故障検出を分けて検出しなければならない．それは後述する回復機構と耐故障化同一応答ノード選択機構がマスターノード故障時とスレーブノード故障時で異なる動作をするためである．これらの機構は故障検出機構により故障が検出された際に動作を行うため，故障検出の段階で分類しなければ正しく動作することができない．

まず，マスターノードの故障検出に期待される動作を考えてみる．後述する回復機構において，マスターノード故障が検出された場合に全ノード間で合意のとれた新しいマスターノードの選出が期待されている．この要求を満たすために，故障検出機構では全ノードが

マスターノードの故障を検出することが期待される．マスターノード故障を検出できないノードが存在すると，そのノードが新しいマスターノードを選出することはない．つまり，全ノード間で合意がとることができないことになるからである．そのため，全ノードがマスターノードの故障を検出する必要がある．

全ノードがマスターノード故障を検出する方法としては 2 種類ある．一つは各スレーブノードがそれぞれマスターノード故障を検出する方法である．もう一つは，特定のスレーブノードがマスターノード故障の検出を行い，残りのスレーブノードに通知する方法である．ところが，耐故障性を考慮すると後者の方法には問題がある．マスターノードの故障を検出したノードが故障すると，残りのスレーブノードはマスターノードの故障を検出する手段がない．つまり，マスターノードの故障は永久に通知されず，故障検出に続く処理を行えない．そのため，マスターノードの故障検出では各スレーブノードが独自にマスターノードの故障検出を行うことを期待する．

次に，スレーブノードの故障検出に期待される動作を考えてみる．スレーブノードの故障検出を必要とするのはマスターノードのみである．スレーブノードはスレーブノード間での通信を行わないため，スレーブノード故障による影響を受けない．影響を受けるのは，SYN パケットの転送を行うマスターノードのみである．従って，スレーブノードがスレーブノード故障の検出を行う必要はない．

マスターノードの故障検出では，各スレーブノードが独自にマスターノードの故障検出を行うことが期待された．これはマスターノードからみると，各スレーブノードごとに何らかのタイミングでマスターノードに対して故障検出が行われるということである．つまり，スレーブノードが故障しているかどうかは，スレーブノード自身が行うマスターノードの故障検出の有無によってマスターノードは判断することができる．このように，スレーブノード故障はマスターノードの故障検出機構で期待される動作をもとにすることで検出を行うことができる．

3.2 回 復 機 構

回復機構では，故障の検出後に FTCS 機構が正しく動作するように回復を行う機構である．回復機構はマスターノード故障時とスレーブノード故障時では異なる動作を行う．マスターノード故障時，回復機構では全ノード間で合意のとれた新しいマスターノードの選択が期待される．新しいマスターノードに関して全ノード間で合意がとれていないと，マスターノードが複数存在する可能性がある．例えば，どのスレーブノードも自身をマスターノードとして選択する場合が挙げられる．マスターノードが複数存在する状況下でクライアントから SYN パケットが届けられた時，各マスターノードはそれぞれ独自の応答ノードを決定する．この時マスターノード間で応答ノードの合意は取られないため，応答ノードが重複してしまう可能性がある．このような事態を防ぐために，新しいマスターノードは全ノード間で合意のとれたものでなければならない．

スレーブノード故障時，故障検出機構によりマスターノードのみがスレーブノード故障を検出する．故障を検出したマスターノードでは，故障したスレーブノードをマスターノードの応答ノード選択候補から取り除く動作が期待される．これにより故障したスレーブノードに SYN パケットを転送することがなくなる．

故障検出機構ではスレーブノードは他スレーブノードの故障検出を期待していない．したがって，スレーブノード故障時に他スレーブノードは何も動作を行わない．スレーブノードはマスターノード故障に伴い新しくマスターノードとして選択された場合に限り，スレーブノード故障の検出と回復を行う．

3.2.1 マスターノード故障からの回復機構

マスターノードが故障した際，回復機構は動作を再開するためにスレーブノード間で一意の新しいマスターノードを選ばなければならない．この問題はリーダー選挙問題[4),14),15)] に帰着される．文献 4)，14)，15) で挙げられるリーダー選挙問題の解決法は動的ノードの追加を想定しているが，本研究では動的ノードの追加は想定していない．このことを利用して以下のようなアルゴリズムで全ノード間で合意のとれた新しいマスターノードを選出する．

FTCS 機構では，マスターノードは SYN パケット転送のためにクラスタに参加している全ノードの情報を持っている．スレーブノードはマスターノードの情報のみを持っている．しかし，これでは故障を考慮した際に問題がある．マスターノードが故障した場合，スレーブノードが新しいマスターノードとして動作する必要があるためである．新しいマスターノードでも故障前と同様に動作するためには，どのノードもクラスタに参加している全ノードの情報を持っている必要がある．そこで，どのノードでもクラスタに参加している全ノードの情報をマスターノードに選出される順に従って整列して持たせることにする．すなわち，あらかじめマスターノードに選出される順番を定めておき，マスターノードが故障した際はその順番に従い新しいマスターノードを選出する．あらかじめマスターノードの選択順番を定めて合意がとられているので，故障時に通信を行って合意をとる必要がない．

3.3 耐故障化同一応答ノード選択機構

故障検出機構，故障検出からの回復機構の 2 つがあれば耐故障性は実現できる．故障が発生した場合は，故障検出機構が故障の検出を行い，マスターノード故障，スレーブノード故

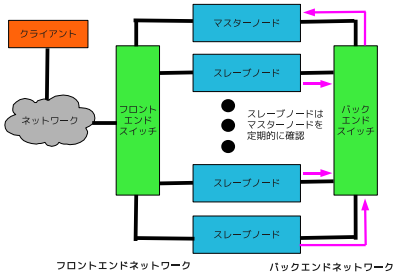

図 5 Discarding アルゴリズムにおける故障検出機構

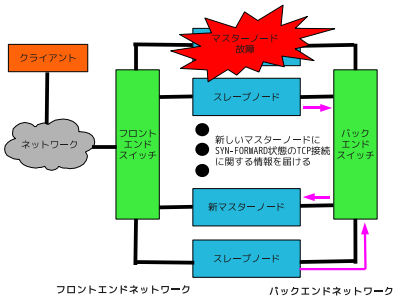

図 6 Gathering アルゴリズムにおける耐故障化同一応答ノード選択機構

障に対応した回復機構を動作させることで故障から回復できる．しかし，FTCS 機構では上述のようにマスターノード故障時に SYN パケット転送履歴が消失してしまうために応答ノードが重複してしまうという特有の問題がある．そのために，マスターノード故障時に応答ノードが重複しないことを保証しなければならない．したがって，耐故障化同一応答ノード選択機構に期待される動作は，新しいマスターノードの SYN パケット転送履歴がクラスタ内に存在する SYN-RECEIVED 状態の TCP 接続に関する情報を全て保持するようにする，ということである．これにより，新しいマスターノードは重複なく応答ノードに SYN パケットを転送することができる．

3.4 Discarding アルゴリズム

Discarding アルゴリズムは故障検出機構として定期的に故障の確認を行い，マスターノード故障が検出された際に各ノードは自身のもつ SYN-RECEIVED 状態の TCP 接続をすべて破棄するアルゴリズムである．Discarding アルゴリズムにおける故障検出機構を図 5 に示す．Discarding アルゴリズムでは各スレーブノードが定期的にマスターノードと通信を行うことで，マスターノードの故障の検出を行う．Discarding アルゴリズムにおける耐故障化同一応答ノード選択機構として，マスターノード故障の際に各ノードは自身の持つ SYN-RECEIVED 状態の TCP 接続をすべて破棄する．これにより，クラスタ全体が持っている SYN-RECEIVED 状態の TCP 接続は存在しなくなる．したがって，新しいマスターノードが持つべき SYN パケット転送履歴は空でよいために新しいマスターノードのもと動作を再開した際に応答ノードが重複することはなくなる．

3.5 Gathering アルゴリズム

Gathering アルゴリズムは Discarding アルゴリズムと同じ故障検出機構を用い，マスターノード故障時には各スレーブノードが持つ SYN-RECEIVED 状態の TCP 接続に関する情報を新しいマスターノードに集め SYN パケット転送履歴を作成するアルゴリズムである．Gathering アルゴリズムにおける耐故障化同一応答ノード選択機構を図 6 に示す．マスターノード故障時に各スレーブノードは自身の持つ SYN-RECEIVED 状態の TCP 接続に関する情報を新しいマスターノードに送る．新しいマスターノードはスレーブノードから送られてきた情報をもとに新しく SYN パケット転送履歴を作成する．これにより，SYN-RECEIVED 状態の TCP 接続に関する情報は新しいマスターノードに全て集められるので，動作を再開した際に応答ノードが重複することはない．

3.6 Scattering アルゴリズム

Scattering アルゴリズムでは，SYN パケット転送履歴をマスターノードのみならずスレーブノードでも持つようにする．スレーブノードが持つ SYN パケット転送履歴は図 7 に示すように，SYN パケット転送前にマスターノードと同期をとる．これにより，応答ノードに関する情報を全ノードが SYN パケット転送履歴にもつ．そのため，いつマスターノードが故障しても同一の応答ノードに SYN パケットを転送することができる．Scattering アルゴリズムではこのようにして耐故障化同一応答ノード選択機構を実現している．また，SYN パケット転送履歴の同期手法としては，マスターノードが転送した応答ノードに関する情報を送り，それを受け取ったスレーブノードがマスターノードに受領確認を送ることで同期を行う．

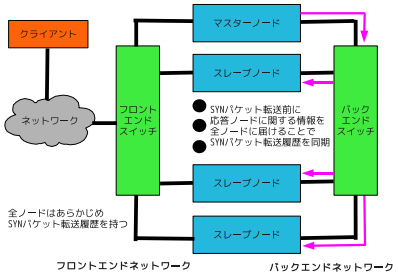

図 7 Scattering アルゴリズムにおける SYN パケット転送履歴の同期

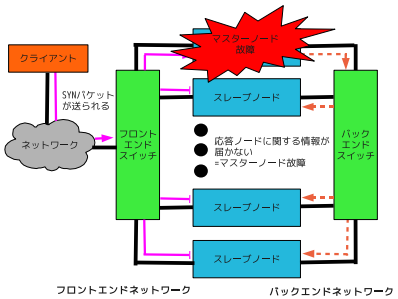

図 8 Scattering アルゴリズムにおける故障検出機構

表 1 正常時のオーバーヘッド

アルゴリズム	オーバーヘッド
Discarding	0
Gathering	0
Scattering	SYN パケット転送履歴の同期にかかる時間

表 2 故障検出にかかる最悪時間

アルゴリズム	最悪時間
Discarding	定期的にマスターノードを確認する間隔
Gathering	定期的にマスターノードを確認する間隔
Scattering	SYN パケットが届いてから SYN パケット転送履歴の同期が完了するまでの時間

Scattering アルゴリズムの故障検出機構も SYN パケット転送履歴の同期を利用している．マスターノードが故障した場合，マスターノードからの SYN パケット転送履歴の同期が行われない．このことを利用して，図 8 のようにして故障検出機構を実現する．スレーブノードではフロントサイドスイッチにより全ノードに届けられ，破棄される．その後，一定時間以内にマスターノードからの同期が行われなければマスターノードの故障として故障を検出する．逆にマスターノードはスレーブノードとの同期の失敗によりスレーブノード故障を検出する．

3.7 アルゴリズム評価

各アルゴリズムにおいて，FTCS 機構に耐故障性を付与することで生じる正常時のオーバーヘッド比較を表 1 に示す．Discarding,Gathering アルゴリズムではオーバーヘッドは存在しない．これらのアルゴリズムは通常の動作範囲ではスレーブノードがマスターノードを定期的に確認する故障検出機構を付与しただけだからである．Scattering アルゴリズムでは SYN パケット転送前の SYN パケット転送履歴同期時間がオーバーヘッドとして生じることになる．

各アルゴリズムにおいて故障検出にかかる最悪時間を表 2 に示す．Discarding,Gathering アルゴリズムでは定期的にマスターノードを確認する事で故障の検出を行う．そのため，故障検出にかかる最悪時間はマスターノードの確認を行う間隔となる．Scattering アルゴリズムでは，SYN パケット転送履歴の同期と兼ねて故障検出が行われる．そのため，最悪で SYN パケットが届いてから SYN パケット転送履歴の同期が終了すると予想される時間が故障検出に必要となる．

各アルゴリズムについて，耐故障化同一応答ノード選択機構の比較を表 3 に示す．Dis-

表 3 耐故障化同一応答ノード選択機構の比較

アルゴリズム	SYN-RECEIVED 状態の接続	最悪時間
Discarding	破棄	SYN-RECEIVED 状態の接続を検索，破棄する時間
Gathering	保存	全ノードから SYN-RECEIVED 状態の接続情報を集め SYN パケット転送履歴を作成する時間
Scattering	保存	0

carding アルゴリズムでは処理として SYN-RECEIVED 状態の接続を全て破棄する．そのため，最悪時間としては SYN-RECEIVED 状態の接続を検索して破棄する時間が必要となる．Gathering アルゴリズムでは新しいマスターノードに SYN-RECEIVED 状態の TCP 接続に関する情報を集め，SYN パケット転送履歴の複製を行う．そのため，全ノードから SYN-RECEIVED 状態の接続情報を集め，SYN パケット転送履歴を作成する時間が必要となる．Scattering アルゴリズムでは常に SYN パケット転送履歴を同期しているため時間はかからない．

以上の比較から，各アルゴリズムの長所，短所を議論する．Discarding アルゴリズムは，通常時のオーバーヘッドは少なく，マスターノード故障を確認する間隔を短くすればその分故障検出時間が短くなる．しかし，マスターノード故障の際に耐故障化同一応答ノード選択機構により，全ての SYN-RECEIVED 状態の接続を破棄してしまう．そのため，SYN-RECEIVED 状態にあった接続は再びはじめから TCP 接続要求処理を行うことになる．それゆえ，TCP 接続が確立されるまでにかかる時間が増大することになる．これは大量の TCP 接続要求処理を行わなければならない環境で問題になる．大量の TCP 接続要求処理を行う環境ではマスター故障時に SYN-RECEIVED 状態の接続数も多いため，その分のオーバーヘッドが増加によってスループットが低下することになる．

Gathering アルゴリズムでは，SYN-RECEIVED 状態の接続をそのまま保持しているので，Discarding アルゴリズムのように再びはじめから TCP 接続要求処理を行うということはない．しかし，Gathering アルゴリズムでは Discarding アルゴリズムに比べて耐故障化同一応答ノード選択機構の処理に時間がかかる傾向にある．Discarding アルゴリズムでは，スレーブノードは自身の持つ SYN-RECEIVED 状態の接続を破棄するだけである．どのノードも自身のノード内で完結した処理によって回復を行うことができる．Gathering アルゴリズムでは，マスターノードは SYN パケット転送履歴を作成しなければならない．各ノードが持つ応答ノードに関する情報を集めるために，新しいマスターノードとスレーブノード間で通信が必要となる．Discarding アルゴリズムでは不必要であった他ノードとの通信が存在することにより，耐故障化同一応答ノード選択機構の処理にがかかる傾向にある．

Scattering アルゴリズムは他のアルゴリズムに比べて通常時のオーバーヘッドが大きい．それは SYN パケット転送履歴の同期をしなければならないためである．逆に，常に SYN パケット転送履歴の同期をとるのでマスターノード故障時は最も耐故障化同一応答ノード選択機構の処理にかかる時間が少ない．そのため，故障検出にかかる時間が他のアルゴリズムより短い場合に最も早くマスターノード故障から回復することができる．

4. 関 連 研 究

耐故障性単一 IP アドレスクラスタとして Linux Virtual Server(LVS)[10),18)] が挙げられる．LVS ではクライアントから届けられたパケットはロードバランサーを介してサーバに転送される．FTCS 機構との環境の違いとして，LVS ではロードバランサーのみにクライアントからのパケットが届けられる点である．FTCS 機構ではフロントエンドスイッチを用いることにより，全ノードに届けられた．LVS の耐故障性を見てみると，FTCS 機構のマスターノードと同様にロードバランサーが単一障害点である．このため LVS では Heartbeat[6),13)] と組み合わせてロードバランサーを冗長化する構成が用いられることが多い．しかしロードバランサーが切り替わった際にはロードバランサーに記憶されていた TCP 接続に関する情報は全て失われてしまうため，クライアントとの接続が途切れてしまうことになる．2 台のロードバランサー間で TCP 接続表を同期する機構[7)] も提案されているが，障害発生からロードバランサーを切り替えるまでの間は通信が不能となる．FTCS 機構で提案する手法で耐故障性を実現する場合，どのアルゴリズムを用いても一旦 TCP 接続が確立すると単一障害点となるマスターノードは関与しない．そのため，マスターノードが故障した場合でも確立された TCP 接続が途切れることはない．

5. ま と め

単一 IP アドレスクラスタを実現する FTCS 機構に耐故障性を付与するためには，故障検出機構，故障検出からの回復機構，耐故障化同一応答ノード選択機構の 3 つが必要である．これら 3 つを付与するアルゴリズムとして，本研究では Discarding,Gathering,Scattering の 3 つのアルゴリズムを提案した．故障検出からの回復機構としては，マスターノード故障時にリーダー選挙問題を解くことで新しいマスターノードを選出することにより回復を行う．スレーブノード故障時は，故障したスレーブノードを応答ノード選択候補から除外することで回復を行う．Discarding アルゴリズムではスレーブノードがマスターノードを定

期的に確認することで故障検出を行う．また，耐故障化同一応答ノード選択機構を実現するために，マスターノード故障時には自身の持つ SYN-RECEIVED 状態の TCP 接続をすべて破棄する．Gathering アルゴリズムでは Discarding アルゴリズムと同様の方法で故障検出を行う．違いは，マスターノード故障時に SYN-RECEIVED 状態の破棄を行わずに，新しいマスターノードにその情報を届けることである．新しいマスターノードは各スレーブノードから届けられる情報から SYN パケット転送履歴を作成することで，耐故障化同一応答ノード選択機構を実現している．Scattering アルゴリズムでは各ノードが SYN パケット転送履歴を持ち，マスターノードが SYN パケットを転送する前に同期をとることで常に応答ノードの重複を防いでいる．故障検出機構も SYN パケット転送履歴の同期を用いて実現される．

Discarding アルゴリズムの利点としては通常時のオーバーヘッドが少なく，マスターノードの確認の間隔を短くすることで故障検出時間を短くすることができる．しかし，マスターノード故障時に SYN-RECEIVED 状態の接続をすべて破棄してしまう点が欠点として挙げられる．これにより TCP 接続要求処理にかかる時間が増大するため，大量の TCP 接続要求を処理する環境ではスループットが低下することになる．Gathering アルゴリズムでは，SYN-RECEIVED 状態の接続は保持したままなのでそのようなことにはならない．しかし，そのために新しいマスターノードで SYN パケット転送履歴を作成することになるので，その分 Discarding アルゴリズムより時間がかかる傾向にある．Scattering アルゴリズムでは，常に SYN パケット転送履歴の同期を行うので通常時のオーバーヘッドが最も大きくなる．しかし，常に SYN パケット転送履歴を同期しているので，耐故障化同一ノード選択機構での処理が最も早い．

謝辞 本研究の一部は，科学技術振興機構 戦略的創造研究推進事業 (CREST) (領域名：実用化を目指した組込みシステム用ディペンダブル・オペレーティングシステム) 技術課題：「高信頼組込みシングルシステムイメージ OS」による．

参 考 文 献

1) Damani, O.P., Chung, P.E., Huang, Y., Kintala, C. and Wang, Y.-M.: ONE-IP: techniques for hosting a service on a cluster of machines, *Selected papers from the sixth international conference on World Wide Web*, Essex, UK, Elsevier Science Publishers, Ltd., pp.1019–1027 (1997).
2) Dias, D. M., Kish, W., Mukherjee, R. and Tewari, R.: A scalable and highly available web server, *COMPCON '96: Proceedings of the 41st IEEE International Computer Conference*, Washington, DC, USA, IEEE Computer Society, pp.85–92 (1996).
3) Fujita, H., Matsuba, H. and Ishikawa, Y.: TCP Connection Scheduler in single IP Address Cluster, *Proceedings of the 2008 Eighth IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*, pp.366–375 (2008).
4) Garcia-Molina, H.: Elections in a Distributed Computing System, *Computers, IEEE Transactions on*, Vol.C-31, No.1, pp.48–59 (1982).
5) Lamport, L., Shostak, R. and Pease, M.: The Byzantine Generals Problem, *ACM Trans. Program. Lang. Syst.*, Vol.4, No.3, pp.382–401 (1982).
6) LinuxHA: The High Availability Linux Project. http://www.linux-ha.org/.
7) LVS: IPVS Connection Synchronization. http://www.linuxvirtualserver.org/docs/sync.html.
8) Matsuba, H. and Ishikawa, Y.: Single IP address cluster for internet servers, *Proceedings of 21st IEEE International Parallel and Distributed Processing Symposium (IPDPS2007)* (2007).
9) Microsoft: Network Load Balancing Technical Overview, `http://www.microsoft.com/technet/prodtechnol/windows2000serv/deploy/confeat/nlbovw.mspx`.
10) O'Rourke, P. and Keefe, M.: Performance Evaluation of Linux Virtual Server, *LISA 2001 15th Systems Administration Conference* (2001).
11) RFC: DNS Support for Load Balancing, `http://www.ietf.org/rfc/rfc1794.txt`.
12) RFC: Transmission Control Protocol, `http://www.ietf.org/rfc/rfc0793.txt`.
13) Robertson, A.: Linux-HA heartbeat system design, *ALS'00: Proceedings of the 4th conference on 4th Annual Linux Showcase & Conference, Atlanta*, Berkeley, CA, USA, USENIX Association, pp.20–20 (2000).
14) Stoller, S.D.: Leader Election in Distributed Systems with Crash Failures, Technical report (1997).
15) Stoller, S.D.: Leader Election in Asynchronous Distributed Systems, *IEEE Transactions on Computers*, Vol.49, No.3, pp.283–284 (2000).
16) Takigahira, T.: Hive server: high reliable cluster Web server based on request multicasting, *Proceedings of The Third International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'02)*, pp. 289–294 (2002).
17) Vaidya, S. and Christensen, K.J.: A Single System Image Server Cluster using Duplicated MAC and IP Addresses, *Proceedings of the 26th Annual IEEE Conference on Local Computer Networks*, pp.206–214 (2001).
18) Zhang, W.: Linux Virtual Servers for Scalable Network Services, *Linux Symposium* (2000).