2004－AL－93　(7)
2004／1／30

確率的弱コーラムシステムを用いた P2P 環境オブジェクト検索アルゴリズム

三浦　健[†]　角川裕次[†]

分散システムとは複数のプロセスとそれらを繋ぐ通信路により構成されるシステムである．近年，P2P (Peer-to-Peer) と呼ばれる大規模かつ動的なシステムが注目されているが，P2P 環境ではその特徴によりシステムに参加しているプロセスの情報を完全に把握するのは難しい．本稿では，P2P 環境におけるオブジェクト検索問題を取り扱う．ここで，オブジェクトには同一内容の複製が存在すると仮定し，その複製の少なくとも一つを知ることをオブジェクト検索問題とする．

本稿では P2P 環境におけるオブジェクト検索に適した情報構造としてコータリを拡張した確率的弱コーラムシステム（PWQS）を提案し，さらに，PWQS を用いたオブジェクト検索アルゴリズムの提案と評価を行う．

An Object Search Algorithm in P2P Environment with Probabilistic Weak Quorum Systems

KEN MIURA [†] and HIROTSUGU KAKUGAWA[†]

A distributed sytem consists of a set of processes and set of communication links. Recently, large scale and dynamic distributed system, called P2P (Peer-to-Peer), has attracted attention. In P2P environment, caused by its nature, it is not easy to obatin complete information of the system. In this paper, we investigate an object search problem in P2P environment. We suppose that there are some replicas, and the object search problem can be stated as finding an object itself or its replica in the system.

To solve the problem, we propose a probabilistic weak quorum system (PWQS) as an information structure, which is an extention of coterie. Then we propose an algorithm based on PWQS which solves the object search problem.

1. はじめに

分散システムとは複数のプロセスとそれらを繋ぐ通信路により構成される．P2P やアドホックネットワークなどの動的なネットワークでは，分散システムの情報を完全に把握することは困難であるといえる．P2P 環境における興味有る問題としてファイル等に代表されるオブジェクトの検索問題が挙げられる．本稿では各オブジェクトに同一内容の複製が存在している場合のオブジェクト検索問題を解くアルゴリズムを提案する．

近年のインターネットの普及によるクライアント数の増加により従来の単純なクライアント/サーバ型システムでは，サーバとネットワーク帯域の負荷が限界を越えることもある．特にファイルの配布はデータ転送量も多く，その負荷は深刻である．具体的な例としては，ソフトウェアの不具合に対するパッチが公開されると同時にアクセスが集中してしまい，サーバが停止するということはしばしば起きている．このような背景から P2P によるファイル共有は注目されている．実際に，音楽ファイル等の共有を目的に Gnutella[1] に代表されるファイル共有ソフトが多数開発されている．このようなファイル共有ソフトでは，自分が手に入れたファイルを新たに共有することで，その複製がネットワークに分散される．本稿でのオブジェクト検索問題は，このようなファイル共有において目的のファイルもしくはその複製を持っているプロセスの少なくとも一つを見付ける問題である．

これまで，分散システム上のある種の問題を解決するための情報構造としてコータリ[3] と呼ばれる構造が広く研究されてきた．コータリは，コーラムと呼ばれるプロセスのグループの集合で，各コーラムは他のコーラムと少なくとも一つのプロセスを共有しなければならないという，*Intersection Property* を持っている．コーラムを基にしたアルゴリズムの基本的な考え方は以下のとおりである．

- 情報を伝えたいプロセスは，あるコーラムに属するプロセスにその情報を送信する．
- 情報を得たいプロセスは，あるコーラムに属するプロセスに問い合わせる．
- *Intersection Property* により任意のコーラムへの情報の送信/問い合わせに対しても情報の一貫性が保持さ

† 広島大学大学院工学研究科情報工学専攻
Department of Information Engineering, Graduate School of Engineering, Hiroshima University

れる．

通常コーラムを基にしたアルゴリズムではプロセスの全体と通信する必要は無く，あるコーラムに属するプロセスとのみ通信すれば良い．したがって，必要なメッセージ数はコーラムサイズに比例し，効率的である．また，システムのいくつかのプロセスが故障したとしても，少なくとも一つのコーラムに属するプロセスが残っていればアルゴリズムが動作可能であり，耐故障性を持つ．しかし，現実的に *Intersection Property* を持った構造をどのように構成するか，また耐故障性と効率性のトレードオフが存在するといった問題があった．確率的コーラムシステム (PQS)[6] は *Intersection Property* を確率的な保証に緩めることでコータリにおけるそれらの問題を解決した．しかし，コータリ，PQS 両者ともその構造を決定するためにはシステムに存在するプロセスの完全な情報を必要とする．

本稿では P2P 環境でのオブジェクト検索に適した構造として，PQS を拡張した確率的弱コーラムシステム (PWQS) の提案を行う．PWQS ではあるプロセスが選択したコーラムが，他の幾つかのプロセスにより選択されたコーラムのうちの少なくとも一つと共通部分を持つ確率が保証された構造となっている．この拡張は，オブジェクト検索問題における複製の少なくとも一つを知ることが出来れば良いという目的と合致する．また，PWQS では必ずしもシステムに存在するプロセスの完全な情報を必要とせず一部のプロセスの情報を得ることでコーラムが構成可能であり，P2P 環境に適しているといえる．

分散システムにおける問題をコーラムを用いて解決する研究は相互排除問題[5] など，これまでに広く行われている．PQS の応用としては負荷分散アルゴリズム[9] が挙げられる．また近年，より柔軟で現実的なコーラムの構成法として，Dynamic Paths Quorum Systems (DP-QS)[7]，Non-uniform PQS (NPQS)[2] といった動的なコーラムの構造が提案されている．

以下では，まず本研究で仮定するシステムモデルの設定とオブジェクト検索問題の定義を第 2 章で行う．第 3 章では本研究で提案する確率的弱コーラムシステム (PWQS) の定義，第 4 章では PWQS を用いたオブジェクト検索アルゴリズムの説明を行う．第 5 章では提案したアルゴリズムが適切な PWQS を形成しオブジェクト検索問題を解くことを示し，その複雑度について議論する．そして第 6 章でまとめを行う．

2. モ デ ル

2.1 システムモデル

本稿で仮定するシステムは，プロセス集合 $U = \{p_1, p_2, \ldots, p_n\}$ とそれらを繋ぐ通信リンクにより構成される．各プロセスは論理的に完全結合されておりメッセージ交換により互いに通信可能である．ただし，各プロセス p_i は初期状態では U の部分集合 $N_0(p_i)$ のみを把握している．そして把握していないプロセスにはメッセージを送ることはできない．システムは同期しており，あるプロセスから送信されたメッセージが送信先のプロセスに届くまでに必要な時間を 1 単位時間とし，さらに単位時間により区切られるアルゴリズムの実行をラウンドと呼ぶことにする．各プロセスは十分な長さのメッセージキューを持っており，あるプロセスにより送信されたメッセージは送信先のプロセスに到着するとそのプロセスのメッセージキューに順番に格納される．各プロセスはメッセージキューの先頭から順番にメッセージを取り出すことでメッセージの受信を実現する．システムを構成するプロセスは参加/離脱が可能であり U は時刻により変化をする．

システムには種々のデータや資源が存在するが，それらを本研究ではオブジェクトという単位で統一的に取り扱うことにする．オブジェクトの種類や容量，内容等を表したデータとそれを保持しているプロセスの組を本研究ではオブジェクト情報とし，各プロセスはオブジェクト情報を用いてオブジェクトの管理を行う．

2.2 オブジェクトの検索問題

本稿で仮定するオブジェクトは，同一内容の複製が存在している．プロセスの総数が n 個のネットワークにおいて，あるオブジェクトが n' 個の複製を持っていた場合，その割合をパラメータ $r = \frac{n'}{n}$ で表すことにする．オブジェクトの各複製はシステムのプロセスに分散され保持されている．また，システムにはオブジェクト検索に対する信頼度パラメータ ρ が存在し，要求に応じてその値を設定できるものとする．

あるプロセスにおいて，オブジェクト o に対する要求が発生したとし，その複製の割合が r であるとする．そのとき，o の複製を持っているプロセスの内の一つを知ることが出来れば，そのプロセスに対しオブジェクトの要求を行うことが出来る．本稿でのオブジェクトの検索問題の定義は，そのようなオブジェクトの複製を持つプロセスを知る確率を保証することとする．

定義 1 (オブジェクトの検索問題) あるオブジェクト o 又はその任意の複製を保持するプロセスの集合を $\mathcal{P}_o$ とし，r を o の複製の個数と全体プロセス数の割合とする．任意のあるプロセス $p_i \in U$ においてオブジェクト o に対する要求が発生したときに，$\mathcal{P}_o$ に属するプロセスを少なくとも一つ，確率 ρ 以上で p_i が知ることを保証する． □

3. 確率的弱コーラムシステム (PWQS)

定義 2 (セットシステム[8]) 集合 U の部分集合の集合を U のセットシステムという． □

定義 3 (コータリ[3]) セットシステム $\mathcal{Q}$ が次の二つの条件を満たすときコータリである．

(Intersection property) $Q_i, Q_j \in \mathcal{Q} \Rightarrow Q_i \cap Q_j \neq \emptyset$

(Minimality) $Q_i, Q_j \in \mathcal{Q} \Rightarrow Q_i \not\subseteq Q_j$

□

定義 4 (アクセス戦術[8]) セットシステム $\mathcal{Q}$ への確率分布 $w : \mathcal{Q} \mapsto [0,1], \sum_{Q\in\mathcal{Q}} w(Q) = 1$ を $\mathcal{Q}$ のアクセス戦術と呼ぶ. □

定義 5 (確率的コーラムシステム (PQS)[6]) セットシステム $\mathcal{Q}$，アクセス戦術 w，正定数 ϵ の三つ組 $\langle \mathcal{Q}, w, \epsilon\rangle$ が

$$\sum_{Q_i,Q_j\in\mathcal{Q}, Q_i\cap Q_j\neq\emptyset} w(Q_i)w(Q_j) \geq 1-\epsilon$$

を満たすなら，PQS である. □

例 1 (PQS の例[6]) $U = \{p_1, p_2, \ldots, p_n\}$,$\mathcal{Q} = \{Q \subseteq U : |Q| = \beta\sqrt{n}\}$, $w(Q) = \frac{1}{|\mathcal{Q}|}$ とすると，$\langle \mathcal{Q}, w, e^{-\beta^2}\rangle$ は PQS である. □

本研究ではコータリの拡張として弱コーラムシステムおよび確率的弱コーラムシステムを提案する．弱コーラムシステムは *Intersection Property* を，複数のコーラムの少なくとも一つと共通部分を持てばよいと拡張した *Weak Intersection Property* を持ち，確率的弱コーラムはそれを更に確率的に拡張した構造となっている.

定義 6 (弱コーラムシステム (WQS)) $\mathcal{Q}$ を U のセットシステムとし，$\mathcal{Q}$ の任意のコーラム Q と大きさ k の任意のコーラム集合 $\mathcal{Q}_k \subseteq \mathcal{Q}$ について，

(Weak Intersection Property) $Q \cap (\bigcup_{Q'\in\mathcal{Q}_k} Q') \neq \emptyset$

を満たすなら，$\mathcal{Q}$ は位数 k の弱コーラムシステム (k-WQS) である. □

定義 7 (アクセス戦術ベクトル) プロセス集合 $U = \{p_1, p_2, \ldots, p_n\}$ に対し $\mathcal{Q}$ をそのセットシステムとする．各プロセス p_i について $\mathcal{Q}$ へのアクセス戦術を w_i とするとき，ベクトル $\overline{w} = (w_1, \ldots, w_n)$ をアクセス戦術ベクトルと呼ぶ. □

定義 8 (確率的弱コーラムシステム (PWQS)) プロセス集合 $U = \{p_1, p_2, \ldots, p_n\}$ に対し，$\mathcal{Q}$ をそのセットシステムとし，$\overline{w} = (w_1, w_2, \ldots, w_n)$ をアクセス戦術ベクトルとする．任意のプロセス p_x と任意のプロセス集合 $P_k = \{p_{y_1}, \ldots, p_{y_k}\}$ について，p_x が選択するコーラムが P_k の各プロセスが選択するコーラムの少なくとも一つと共通部分を持つ確率が少なくとも $1-\epsilon$，つまり，

$$\sum_{\substack{Q_k = Q_{y_1}\cup\cdots\cup Q_{y_k},\\ Q_x\cap Q_k\neq\emptyset}} w_x(Q_x)\prod_{j=1}^{k} w_{y_j}(Q_{y_j}) \geq 1-\epsilon$$

を満たすならば，$\langle \mathcal{Q}, \overline{w}, \epsilon, k\rangle$ は位数 k の確率的弱コーラムシステム (k-PWQS) である. □

例 2 (PWQS の例) $U = \{p_1, p_2, \ldots, p_n\}$, $\mathcal{Q} = \{Q \subseteq U : |Q| = \frac{\beta}{\sqrt{k}}\sqrt{n}\}$, $w_i(Q) = \frac{1}{|\mathcal{Q}|}$ とすると，$\langle \mathcal{Q}, \overline{w}, e^{-\beta^2}, k\rangle$ は k-PWQS である. □

4. アルゴリズム

本稿で提案するアルゴリズムは，コーラムを基にしてオブジェクトの検索を行う．つまり，オブジェクトを持っているプロセスはそのオブジェクト情報を自身のコーラムに伝え，オブジェクトを検索するプロセスは自身のコーラムに問い合わせをすることで，目的のオブジェクトを保持するプロセスを知る.

本稿で扱うモデルでは各プロセスは初期状態では全体プロセスの部分集合しか知らず，コーラムの形成には不十分である．したがって，まずはコーラムを形成するのに十分な個数のプロセスを知り，プロセスリストとして保持する．そして，そのプロセスリストからコーラムを形成しオブジェクトの検索を行う．また，プロセスのシステムからの参加/離脱を許しており，それらのイベントによるプロセスリストの変化を適切に処理する必要がある．さらに，プロセスの離脱に対応するために各プロセスが持つプロセスリストにはその生存時間が設定されており，一定時間内に更新されなければその情報は捨てる．プロセスリストとその生存時間をあわせてプロセス情報とする.

アルゴリズムは以下のように構成される.

プロセス情報の管理 (図 1) コーラム形成に必要なプロセス情報の管理を行う．主に以下の三つの項目について考えなければならない.

- **プロセス情報の収集** コーラム形成に必要な個数のプロセス情報を集める.
- **プロセス情報の更新** 収集したプロセス情報に含まれるプロセスリスト内の生存時間を更新する.
- **プロセスの参加/離脱** プロセスの参加/離脱に伴うプロセス情報の変更を行う.

オブジェクト情報の管理 各プロセスが保持しているオブジェクトに関する情報はコーラムの各プロセスに通知される．コーラムの決定はプロセス情報の管理によって得られたプロセスリストに従い行われる.

- **コーラム形成** その時点で知っているプロセスリストからコーラム集合を決定する.
- **オブジェクト検索** コーラム形成により決定されたコーラム集合からコーラムを選択しオブジェクト検索を行う.

4.1 入　　力

各プロセスは入力として以下のパラメータが与えられるものとする.

- $N_0(p_i)$: プロセス p_i が把握しているプロセスの初期集合．$p_i \in N_0(p_i)$ と仮定する.
- 複製パラメータ r : プロセス数 n に対してオブジェクトの複製が存在している割合．複製の個数が n' とすると，$r = \frac{n'}{n}$.
- 信頼度パラメータ ρ : オブジェクトの検索を行ったと

```
procedure ManageProcInfo at p_i:
begin
    loop: /* beginning of a round */
        if wish to leave the system then
            Leave;
            leave the system;
        /* receive all the messages
          arrived at this round */
        MSGS = ∅;
        while msg_queue_i is not empty do
            MSGS = MSGS ∪ top of msg_queue_i;
            remove top of msg_queue;
        if ∃p_j :LEAVE_j ∈ MSGS then
            RcptLeave;
        UpdateProcInfo;
        CollectProcInfo;
    goto loop; /* end of a round */
end
```

図 1 プロセス情報管理の流れ

きに，そのオブジェクトを持ったプロセスの存在を少なくとも 1 つく知る確率の下限値.

- TTL : プロセス情報の生存時間.

4.2 変　　数

各プロセス p_i は以下の変数を持つ.

- $state_i$: プロセス p_i の現在の状態を表す．取り得る値は以下の 3 種類で，初期値は UNREADY である．

  UNREADY$\cdots$　コーラム形成に十分な個数のプロセス情報が得られていない状態．コーラム形成に十分なプロセス個数のプロセス情報を得れば READY に遷移する．もし，現在知っているプロセス全てと通信を行った場合でも十分な個数のプロセス情報が得られてない場合，TEMPORARY に遷移する.

  READY$\cdots$　コーラムを形成するのに十分な個数のプロセス情報を得た状態．プロセスの離脱によりコーラムの再構成を行わなければならない場合 UNREADY に遷移する.

  TEMPORARY$\cdots$　READY，UNREADY どちらとも判断が出来ない状態である．考えられる状況としては,

  – システム全体のプロセス数が，$\frac{2}{r}$ 個未満である.

  – 単に，未だ情報を得ていないプロセスが存在する.

  前者の場合，既にシステム全体のプロセスを知ってしまっている状態であり，実質的には READY 状態と等価である．システムに新たにプロセスが参加することでシステム状況が変化すれば再び UNREADY に遷移する．後者の場合も時間の経過により他のプロセスから未知のプロセスの情報を得ることで再び UNREADY に遷移する.

- *msg_queue*$_i$: プロセス p_i に届いたメッセージが格納されるメッセージキュー.
- *ttu*$_i$: プロセス p_i による情報更新までの時間.
- *init_ready* : READY 状態で必要な初期化を行うためのフラグ.
- $N_t(p_i)$: プロセス p_i が現在情報を得ているプロセスの集合とそのプロセスの情報の生存時間の組.

  $N_t(p_i) = \{(p_j, \mathit{ttl}_j) \mid p_i \text{ knows } p_j\}$.

  初期値は $\{(p_j, TTL) \mid p_j \in N_0(p_i)\}$.
- $N(p_i)$: プロセス p_i が現在把握しているプロセスの集合．$N_t(p_i)$ から直ちに得られる.

  $N(p_i) = \{p_j \mid (p_j, \mathit{ttl}_j) \in N_t(p_i)\}$.
- $D(p_i)$: プロセス p_i が過去に少なくとも一度メッセージを送ったことのあるプロセスの集合． 初期値は空集合.

  $D(p_i) = \{p_l \mid p_i \text{ has ever sent messages to } p_l\}$.
- $W(p_i)$: コーラム集合を定義するのに用いるプロセス集合．各時点でシステムに存在していると思われるプロセスの集合． 初期値は空集合.

  $W(p_i) = \{p_m \mid p_i \text{ guesses that } p_m \text{ is in the system}\}$.

4.3 プロセス情報の収集

各プロセス p_i は初期状態では $N_0(p_i)$ のプロセスの情報しか知らない．しかし，信頼度パラメータ ρ を保証するコーラムを形成するためには $\frac{2}{r}$ 個以上のプロセスの情報を得る必要がある ($\frac{2}{r}$ という値の根拠については後述)．そこで，本研究では文献 4) の Name-Dropper アルゴリズムを基にしたアルゴリズムを提案する．Name-Dropper アルゴリズムの基本的な考えは，以下の通りである.

(1)　$N(p_i)$ をプロセス p_i が現在知っているプロセス集合とする．p_i は $N(p_i)$ からランダムに一つプロセス p_j を選択し，p_j に $N(p_i)$ を送信する.

(2)　$N(p_i)$ を受け取ったプロセス p_j は自身のプロセスリストを更新する；$N(p_j) = N(p_j) \cup N(p_i)$.

(3)　全てのプロセスが，完全なプロセスの情報を知るまで，(1)-(2) を繰り返す.

一方，本稿の設定ではシステム全体のプロセスを知る必要は無い．そこで，全てのプロセスがコーラム形成に十分な個数のプロセスを知った時点でプロセス情報の収集を停止し，時間及び通信量の減少を図った (図 2).

[$state_i$ =UNREADY での p_i の動作]

(1)　$N(p_i) \setminus \{p_i\}$ の中からプロセス p_j をランダムに一つ選び $N_t(p_i)$ を送る．もしプロセス p_k から $N_t(p_k)$ を受け取ったなら，自身の $N_t(p_i)$ を $N_t(p_i) = N_t(p_i) \cup N_t(p_k)$ と更新する.

(2)　もし $|N_t(p_i)| \geq \frac{2}{r}$，つまりコーラム形成に十分な個数のプロセス情報を得たならば状態を $READY$ に変更および，初期化処理を行う．そして，各 $p_j \in N(p_j)$ に SELECTED$_i$ メッセージを送り p_i が READY 状態に遷移したことを伝えると同時に，$N_t(p_i)$ に含まれる各プロセスの存在確認を行う.

(3)　もし，プロセス p_s から SELECTED$_s$ メッセージを

```
procedure CollectProcInfo at p_i:
begin
  if state_i = UNREADY then
    /* Name-Dropper アルゴリズムの基本ステップ */
    p_j = randomly choosen from N(p_i) \ {p_i};
    send N_t(p_i) to p_j ;
    D(p_i) = D(p_i) ∪ {p_j};
    for each p_k in which N_t(p_k) ∈ MSGS do
      UpdateN_t(N_t(p_k)) ;

    /* p_s は既に 2/r 個のプロセスを知っている．*/
    for each p_s in which SELECTED_s ∈ MSGS do
      UpdateN_t({(p_s, TTL)});
      /* 自身のプロセス情報を送信し p_s からの応答を期待 */
      send N_t(P_i) to p_s;
      D(p_i) = D(p_i) ∪ {p_s};

    if |N_t(p_i)| ≥ 2/r then
      state_i = READY;
      /* READY 状態で必要な初期化 */
      W(p_i) = ∅;
      init_ready = false;

      /* SELECTED メッセージにより READY 状態に遷移したことを
         通知すると同時にプロセスの存在を調査 */
      for each p_j ∈ N(p_j) do
        send SELECTED_i to p_j;
        D(p_i) = D(p_i) ∪ {p_j};

    else if N(p_i) ⊆ D(p_i) then
      W(p_i) = N(p_i);
      state_i = TEMPORARY;

  else if state_i = TEMPORARY then
    /* UNREADY の場合と同様 */
    for each p_j in which N_t(p_j) ∈ MSGS do
      UpdateN_t(N_t(p_j));

    /* UNREADY の場合と同様 */
    for each p_s in which SELECTED_s ∈ MSGS do
      UpdateN_t({(p_s, TTL)});
      send N_t(p_i) to p_s;
      D(p_i) = D(p_i) ∪ {p_s};

    /* 新たなプロセスを知ったなら状態遷移 */
    if N(p_i) ⊈ D(p_i) or |N_t(p_i)| ≥ 2/r then
      state_i = UNREADY;

  else if state_i = READY then
    /* p_j は存在していると判断 */
    for each p_j in which N_t(p_j) ∈ MSGS do
      W(p_i) = W(p_i) ∪ {p_j};
      UpdateN_t({(p_j, TTL)});
      /* p_j が UNREADY 状態の可能性があるなら
         READY 状態へと促す．*/
      if |N_t(p_j)| < 2/r then
        send N_t(p_i) to p_j;
        D(p_i) = D(p_i) ∪ {p_j};

    /* p_s は存在していると判断 */
    for each p_s in which SELECTED_s ∈ MSGS do
      W_t(p_i) = W_t(p_i) ∪ {p_s};
      UpdateN_t({p_s, TTL});
      /* p_s に，自身の存在を伝える */
      send ACK_i to p_s;
      D(p_i) = D(p_i) ∪ {p_s};

    /* p_a は存在していると判断 */
    for each p_a in which ACK_a ∈ MSGS do
      W(p_i) = W(p_i) ∪ {p_a};

    /* READY 状態に遷移した最初のラウンドでは
       ACK が帰ってくるのを待つ */
    if init_ready = false then
      init_ready = true;
    else
      /* p_r は既に存在していないので削除 */
      for each p_r ∈ N(p_i) \ W(p_i) do
        N_t(p_i) = N_t(p_i) \ {(p_r, ⋆)};
        D(p_i) = D(p_i) \ {p_r};
      if |N(p_i)| < 2/r then
        state_i = UNREADY;
end

macro UpdateN_t(N)
begin
  for each (p_t, ttl_t) ∈ N do
    if ttl_t − 1 > 0 then
      N_t(p_i) = N_t(p_i) \ {(p_t, ⋆)};
      N_t(p_i) = N_t(p_i) ∪ {(p_t, ttl_t − 1)};
end
```

図 2 プロセス情報の管理

受け取ったなら p_s を $N_t(p_i)$ に加える．SELECTED$_s$ メッセージはプロセス p_s において $N_t(p_s) \geq \frac{2}{r}$ となり READY 状態になったことを各 $p_i \in N_t(p_s)$ に伝えると同時に，それらのプロセスの存在確認を行うためのメッセージである．p_i は $N_t(p_i)$ を p_s に送信することで p_i の存在を通知し，さらに p_i が未だ十分な個数のプロセス情報を得ていない，つまり $N_t(p_i) < \frac{2}{r}$ であることが同時に p_s へと伝わる．

(4) もし $N(p_i) \subseteq D(p_i)$ なら，つまり現在知っている全てのプロセスに少なくとも一度メッセージを送信したことがあるならば状態を TEMPORARY とし，$W(p_i) = N(p_i)$ とする．

[$state_i = READY$ での p_i の動作]

(1) p_i から SELECTED$_i$ を受けとったプロセス p_j はその応答として $N_t(p_j)$ もしくは，ACK$_j$ を返す．それらのメッセージを送ってきたプロセスは，システムに存在していると判断し $W(p_i)$ に加える．また，SELECTED$_i$ メッセージへの応答としてではなく，UNREADY 状態の最初のステップとして $N_t(p_j)$ を送ってきた場合も考えられる．その場合は $p_j \notin$

$N(p_i)$ であるので $N_t(p_i)$ に p_j を加えておく．ここで，$N_t(p_j)$ を $N_t(p_i)$ に加えないのは，$N_t(p_i)$ の大きさを出来るだけ増やさないためである．
また，$|N_t(p_j)| < \frac{2}{r}$ であるプロセス p_j は未だ UNREADY 状態である可能性があるので自身の $N_t(p_i)$ を p_j に送る．$N_t(p_i) \geq \frac{2}{r}$ であるので，p_j は次のラウンドで READY 状態へ遷移する．

(2) もし，プロセス p_s から SELECTED$_s$ メッセージを受け取ったなら p_s に ACK$_i$ メッセージを返し自身の存在を知らせる．さらに，p_s は現在システムに存在していると考え $W(p_s)$ に加える．

(3) 上記のステップにより，$N_t(p_i)$ に含まれる各プロセスで現在システムに存在しているものを把握する．その結果，システムに既に存在していないと思われるプロセスを $N_t(p_i)$ から取り除き，その結果 $|N_t(p_i)| < \frac{2}{r}$ となったら，状態を UNREADY に戻す．しかし，READY 状態に遷移した最初のラウンドでは SELECTED$_i$ メッセージに対する応答が返って来ないので，*init_ready* により 1 ラウンド待つ．

[$state_i = TEMPORARY$ での p_i の動作]

(1) 余計なメッセージを抑制するために，自ら $N_t(p_i)$ を送信しない事以外は，UNREADY 状態と同様である．

(2) もし，$N(p_i) \not\subseteq D(p_i)$ か $|N_t(p_i)| \geq \frac{2}{r}$ であるならば再び状態を UNREADY とする．

4.4 プロセス情報の更新

各プロセスにより保持されるプロセス情報には生存時間が設定されており，各ラウンド開始時に更新を行う (図 3).

```
procedure UpdateProcInfo at p_i:
begin
    /* p_j の情報を更新 */
    for each p_j in which UPDATE_j ∈ MSGS do
        N_t(p_i) = N_t(p_i) \ {(p_j, ⋆)}
        N_t(p_i) = N_t(p_i) ∪ {(p_j, TTL)}

    /* 各プロセス情報の ttl を減らす */
    for each (p_j, ttl_j) ∈ N_t(p_i) do
        N_t(p_i) = N_t(p_i) \ {(p_j, ttl_j)};
        if ttl_j − 1 > 0 then
            N_t(p_i) = N_t(p_i) ∪ {(p_j, ttl_j − 1)};
        else
            W(p_i) = W(p_i) \ {p_j}
            D(p_i) = D(p_i) \ {p_j}

    if |N_t(p_i)| < 2/r then
        state_i = UNREADY

    /* 自身の ttu_i の更新 */
    ttu_i = ttu_i − 1;
    if ttu_i = 1 then
        ttu_i = TTL;
        for each p_j ∈ N(p_i) do
            send UPDATE_i to p_j;
end
```

図 3 プロセス情報の更新

4.5 オブジェクトの検索

4.5.1 コーラムの形成

各プロセス p_i は，$W(p_i)$ から大きさ $\beta\sqrt{\frac{2}{r}}$ の大きさの部分集合を選び，それらをコーラム集合とする; $\mathcal{Q}_i = \{Q \subseteq W(p_i) : |Q| = \beta\sqrt{\frac{2}{r}}\}$ ここで，$\beta = \sqrt{\log\left(\frac{1}{1-\rho}\right)}$．各プロセス p_i は，コーラム集合 $\mathcal{Q}_i$ からコーラムをランダムかつ一様に選択する．

4.5.2 オブジェクト検索

プロセスの状態が READY もしくは TEMPORARY であれば，オブジェクトの検索を行うことが出来る．ただし，TEMPORARY の状態では必ずしも確率 ρ で正しい結果が得られることは保証されない．p_o をオブジェクト o の複製を持つプロセス，$\mathcal{Q}_o$ を p_o のコーラムの集合とする．

[オブジェクトを持っているプロセス]

- p_o は，$\mathcal{Q}_o$ からランダムかつ一様に選んだ Q_o の各メンバに，o を持っていることを伝える．

[オブジェクトを検索したいプロセス]

- オブジェクト o を得たいプロセス p_i は自身のコーラム Q_i のメンバに，o を持っているプロセスはどれかを問い合わせる．
- 問い合わせを受けたプロセスは，知っている情報を p_i に返す．このとき，$q \in Q_i \cap Q_o$ であるプロセスは p_i に p_o を教えることとなる．
- p_i は p_o にオブジェクトの要求を行う．

4.6 プロセスの参加/離脱

4.6.1 プロセスの参加

(1) システムに参加したいプロセス p_i はシステムに存在しているプロセスを少なくとも一つ知っているとし，それを p_j とする．そして，プロセスの初期集合を $N_0(p_i) = \{p_i, p_j\}$ として，アルゴリズムの実行を開始する．

4.6.2 プロセスの離脱 (図 4)

(1) システムから離脱したいプロセス p_i は 各 $p_j \in N(p_i)$ に **LEAVE** メッセージを送り，システムから離脱する．

(2) **LEAVE** メッセージを受け取ったプロセス p_j は $N_t(p_j), D(p_j), W(p_j)$ を更新する．

(3) もし，p_j において $|N_t(p_j)| < \frac{2}{r}$ となった場合 p_j は再びコーラム形成に必要な個数のプロセス情報を得なければならない．p_j は自身の状態を UNREADY とし，コーラムの再構成を行う．

5. 解　　析

5.1 PWQS の構造について

補題 1 大きさ n の集合 U に対し，Q_1, Q_2 を U から $\ell\sqrt{n}$ 個の要素をランダムかつ独立に (要素の重複を許して) それぞれ選んだ集合とする．このとき，次の二つの式が成り立つ．

```
procedure Leave at p_i:
begin
    for each p_j ∈ N(p_i) do
        send LEAVE_i to p_j;
end
```

```
procedure RcptLeave at p_j:
begin
    for each p_i in which LEAVE_i ∈ MSGS do
        N_t(p_j) = N_t(p_j) \ {p_i, ⋆};
        D(p_j) = D(p_j) \ {p_i};
        W(p_j) = W(p_j) \ {p_i};
        if (|N_t(p_j)| < 2/r) then
            state_j = UNREADY;
end
```

図 4 プロセスの離脱処理

$$E[|Q_1 \cap Q_2|] \geq \ell^2,$$
$$\Pr[Q_1 \cap Q_2 = \emptyset] \leq e^{-\frac{\ell^2}{2}}.$$

証明

文献 2) Lemma 1 の証明より. □

補題 2 大きさ n のプロセス集合 U に対し，大きさ $\alpha\sqrt{n}$ である U の部分集合の集合を $\mathcal{W} = \{W \subseteq U : |W| = \alpha\sqrt{n}\}$ とし，W_1, W_2 を $\mathcal{W}$ からランダムかつ一様に選択する．各 $i \in \{1,2\}$ に対し W_i から $\beta\sqrt{|W_i|} = \beta\sqrt{\alpha}\sqrt[4]{n}$ 個の要素をランダムかつ独立に (要素の重複を許して) 選んだ集合を Q_i とすると，Q_1 と Q_2 が共通部分を持たない確率 X は,

$$X = \Pr[Q_1 \cap Q_2 = \emptyset] \leq e^{-\frac{\beta^2}{2}\frac{\alpha}{\sqrt{n}}}$$

である.

証明

W_1 と W_2 の持つ共通部分を $I = W_1 \cap W_2$ とすると，その大きさの期待値 $E[|I|]$ は，補題 1 より

$$E[|I|] \geq \alpha^2$$

となる．そのとき Q_i のメンバの内で I に含まれるものを $S_i = \{q \in Q_i \mid q \in I\}$ とすると，その大きさの期待値は,

$$\begin{aligned}
\mathrm{E}[|S_i|] &= |Q_i|\frac{E[|I|]}{|W_i|} \\
&= \beta\sqrt{\alpha}\sqrt[4]{n} \cdot \frac{\sqrt{E[|I|]}\sqrt{E[|I|]}}{\alpha\sqrt{n}} \\
&\geq \beta\sqrt{\alpha}\sqrt[4]{n} \cdot \frac{\sqrt{E[|I|]}\alpha}{\alpha\sqrt{n}} \\
&= \frac{\beta\sqrt{\alpha}}{\sqrt[4]{n}} \cdot \sqrt{E[|I|]}
\end{aligned}$$

となる．このとき，S_1 と S_2 は大きさ $m = E[|I|]$ の集合から $k\sqrt{m}$ 個の要素をランダムかつ独立に (要素の重複を許して) 選んだものと考えることができる (ここで，$k = \frac{\beta\sqrt{\alpha}}{\sqrt[4]{n}}$). よって，$S_1$ と S_2 が共通部分を持たない確率 X は，補題 1 より,

$$X \leq e^{-k^2/2} = e^{-\frac{1}{2}\left(\frac{\beta\sqrt{\alpha}}{\sqrt[4]{n}}\right)^2} = e^{-\frac{\beta^2}{2}\frac{\alpha}{\sqrt{n}}}$$

となる. □

補題 3 大きさ n のプロセス集合 U に対して n' 個 $(2 \leq n' \leq n)$ のプロセス集合を $P_{n'} \subseteq U$ とし，説明のために $P_{n'} = \{p_1, \ldots, p_{n'}\}$ とおく．各プロセス $p_j \in P_{n'}$ はそれぞれ U の大きさ m の部分集合をランダムかつ一様に選び，さらにその中から $\beta\sqrt{m}$ 個の要素をコーラムとして選択するとする．Q_i を U のあるプロセス $p_i \in U$ が同様に選んだコーラムとし，Q_i が $P_{n'}$ の選んだコーラム $\mathcal{Q}_{n'} = \{Q_1, \ldots, Q_{n'}\}$ の全てと共通部分を持たない確率を Y とする．つまり,

$$Y = \Pr[\bigwedge_{Q_j \in \mathcal{Q}_{n'}} Q_i \cap Q_j = \emptyset].$$

とする．このとき,

$$Y \leq e^{-\beta^2}$$

となるための条件は,

$$m \geq \frac{2}{r}$$

である．ここで，$r = \frac{n'}{n}$.

証明

$m = \alpha\sqrt{n}$ とすると補題 2 より,

$$Y = X^{n'} \leq e^{-\frac{\beta^2}{2}\frac{\alpha}{\sqrt{n}} \cdot n'}$$

である．$Y \leq e^{-\beta^2}$ の条件を得るため,

$$Y \leq e^{-\frac{\beta^2}{2}\frac{\alpha}{\sqrt{n}} \cdot n'} \leq e^{-\beta^2}$$

を，α について解くと,

$$\alpha \geq 2\frac{\sqrt{n}}{n'} = \frac{2}{r\sqrt{n}}$$

を得る．したがって，m の条件として次式を得る,

$$m = \alpha\sqrt{n} \geq \frac{2}{r}.$$

□

定理 1 $U = \{p_1, p_2, \ldots, p_n\}$ をプロセス集合, $\mathcal{W} = \{W \subseteq U : |W| = \frac{2n}{k}\}$ を大きさ $\frac{2n}{k}$ の U の部分集合の集合とし，各 p_i について $W_i \in \mathcal{W}$ をランダムかつ一様に選ぶ．さらに，各 $p_i \in U$ に対しセットシステム $\mathcal{Q}_i$，アクセス戦術 w_i を以下のように定める．

$$\mathcal{Q}_i = \{Q \subseteq W_i : |Q| = \beta\sqrt{|W_i|}\}$$

表 1 既存のコーラム構造との比較

	PWQS (本論文)	DP-QS[7]	NPQS[2]
コーラムの構造	確率的	決定的	確率的
コーラム構成に必要な時間	$O\left(\log^2\left(\frac{2}{r}\right)\right)$	$O(\log n)$	$O(1)$
コーラム構成に必要なメッセージ	$O(n)$ (最悪時)	$O(\log n)$	$O(\log n)$
コーラムへのアクセスに必要な時間	$O(1)$	$O(\sqrt{n})$	$O(\log n)$
コーラムへのアクセスに必要なメッセージ	$O\left(\sqrt{\log\frac{1}{1-\rho}}\sqrt{\frac{2}{r}}\right)$	$O(\sqrt{n}\log n)$	$O\left(\sqrt{\log\frac{1}{1-\rho}}\sqrt{n}\log n\right)$

$$w_i(Q) = \begin{cases} \frac{1}{|\mathcal{Q}_i|}, & Q \in \mathcal{Q}_i \\ 0, & Q \notin \mathcal{Q}_i \end{cases}$$

このとき，$\langle \mathcal{Q}, \overline{w}, e^{-\beta^2}, k\rangle$ は k-PWQS である．ここで，$\mathcal{Q} = \bigcup_{p_i \in U} \mathcal{Q}_i$，$\overline{w} = (w_1, w_2, \ldots, w_n)$ である．

証明

補題 3 に従う． □

定理 2 提案アルゴリズムはオブジェクト検索問題を解く．

証明

定理 1 において $k = n'$, $\beta = \sqrt{\log(\frac{1}{1-\rho})}$ とおくと，$|W| = \frac{2}{r}$，$Y \leq 1-\rho$ であり，アルゴリズムによって形成されるコーラム集合は n'-PWQS である．したがって，PWQS の定義より複製が n' 個であるオブジェクトを持つてりうプロセスの少なくとも 1 つをアルゴリズムによって少なくとも確率 ρ で知ることが出来，オブジェクト検索問題を解くことが出来る． □

5.2 複 雑 度

アルゴリズムの複雑度について解析をする．時間複雑度とは実行に必要なラウンド数であり，通信複雑度とは必要なメッセージ数である．Name-Dropper アルゴリズム[4] の結果に従うと，コーラム形成に関して次の補題を得る．

補題 4 (コーラム形成) 全体のコーラムの形成に必要な時間複雑度は $O\left(\log^2(\frac{2}{r})\right)$，各プロセスの通信複雑度は $O\left(\log^2\left(\frac{2}{r}\right) + N_{\max}\right)$ である．ここで，$N_{\max} = \max_{p_i \in U}\{|N(p_i)|\}$ である． □

$N_{\max}$ のとり得る値は $\frac{2}{r} \leq N_{\max} \leq n$ であり，したがってコーラム形成における通信複雑度は $O(n)$ となる．しかし，提案手法では $N(p_i)$ の大きさが $\frac{2}{r}$ を越えた時点でその大きさを出来るだけ増やさないようにしており，$N_{\max}$ の大きさは平均的には $O(\frac{2}{r})$ 程度になるものと思われる．また，オブジェクト検索に関しても直ちに次の補題を得る．

補題 5 (オブジェクト検索) あるプロセスがオブジェクトの検索に必要な時間複雑度は $O(1)$，通信複雑度は $O\left(\sqrt{\log(\frac{1}{1-\rho})}\sqrt{\frac{2}{r}}\right)$ である． □

6. お わ り に

本稿では，複製を持ったオブジェクト検索問題を考え，そのような問題に適した情報構造である確率的弱コーラムシステム (PWQS) の提案，及び PWQS を用いたアルゴリズムの提案と解析を行った．本稿で提案したアルゴリズムではシステム全体の情報を把握する必要が無く，またプロセスの参加/離脱によるシステムの変化にも対応しており，P2P 環境に適しているといえる．

提案手法と同様に動的にコーラムを形成する DP-QS[7], NPQS[2] との比較を表 1 に示す．この表からわかるように既存手法ではコーラムの構成については優れているが，コーラムへのアクセスに関しては提案手法が優れている．

今後の課題としては，

- 提案手法の現実的な性能の評価．
- PWQS のオブジェクト検索以外の応用の可能性についての考察．

などが挙げられる．

謝辞 広島大学工学部第二類コンピュータ・アーキテクチャ学研究室の田川太郎君には提案アルゴリズムのシミュレーションを行うことで，バグの発見，及び，有益なデータの提供を受けた．心より感謝を申し上げたい．

参 考 文 献

1) Gnutella Web site : http://www.gnutella.com.
2) I. Abraham and D. Malkhi. Probabilistic quorums for dynamic systems. *DISC*, pp. 60–74, 2003.
3) H. Garcia-Molina and D. Barbara. How to assign votes in a distributed system. *J. ACM*, 32(4):841–860, 1985.
4) M. Harchol-Balter, T. Leighton, and D. Lewin. Resource discovery in distributed networks. In *PODC*, 1999.
5) M. Maekawa. A $\sqrt{n}$ algorithm for mutual exclusion in decentralized systems. *ACM Trans. Comp. Sys.*, 3(2):145–159, 1985.
6) D. Malkhi, M. K. Reiter, A. Wool, and R. N. Wright. Probabilistic quorum systems. *The Information and Computation Journal*, 170(2):184–206, 2001.
7) M. Naor and U. Wieder. Scalable and dynamic quorum systems. In *PODC*, 2003.
8) M. Naor and A. Wool. The load, capacity and availability of quorum systems. *SIAM Journal on Computing*, 27(2):423–447, 1998.
9) 吉村, 角川, 阿江. 確率的コーラムシステムに基づく負荷分散アルゴリズムとその実験的評価. 情報処理学会論文誌, 43(3):776–783, Mar 2002.