RL-006

ノードの局所性と管理の公平性を考慮した OneHop-P2P 拡張方式

Improvement of OneHop-P2P for Locality of Node and Fairness of Management

金子 豊† 竹内 真也† 南 浩樹† 和泉 吉則†
Yutaka Kaneko Shinya Takeuchi Hiroki Minami Yoshinori Izumi

1．はじめに

放送局では，VTR テープを使った制作・送出システムからファイルとネットワークを使ったシステムへの移行が進められている．番組素材がファイル化され，全ての放送局のサーバーが広帯域ネットワークで結ばれ，各放送局に保管される膨大な素材を全局でいつでも共用できる環境の構築を検討している．

P2P による分散コンテンツ管理方式[1]は，中央集権的な管理機構を持たないため管理コスト低減ができ，また障害耐性，拡張性に優れているなど，大規模な分散共有に向いた方式である．放送局の素材共有環境では，サーバーは各放送局に配置され，それぞれのサーバーは局内ネットワークと局間ネットワークとで結ばれる．また，各放送局が保有するサーバーの台数，サーバーの性能，公開するファイル数などは，放送局の規模により異なる．そのため，規模の異なる放送局間で素材の分散共有を実現するには，このような放送局の環境に適した P2P 方式が必要となる．

OneHop-P2P[2]はスケーラビリティでは劣るが，ノード管理の容易性では有利な方式である．本稿では，放送局の閉じた環境での利用を前提とし，次の２点の要求を満たす OneHop 拡張方式を提案する．(1)放送局のネットワーク構成を考慮し，局間ネットワークに流れるパケット量ができるだけ少ない．(2) 各サーバーが公開するファイル数に応じて管理するファイル数が公平に割り当てられる．

2．OneHop 方式[2]

放送局のシステムでは，システム障害が生じたときに，障害箇所の特定が迅速に行えることが重要である．オーバーレイネットワークに参加しているノードが分散してルーティング情報を管理する P2P 方式では，参加しているノードを一箇所で把握できず，障害探索が困難である．

DHT(Distributed Hash Table)を用いた構造型 P2P の一方式として OneHop が提案されている．OneHop では，オーバーレイネットワークに参加する全てのノードが，参加している全てのノードのテーブルを保持する．そのため，参加しているノードを，常時どこからでも把握でき，障害時の対応が比較的容易となる．また，ファイル ID（キー）の検索は，各ノードが保持するノードテーブルを使って完了するため，参加ノード数に関係なく検索ホップ数がオーダー O(1)の高速な検索が可能である．

OneHop は，Chord[3]と同様，各ノードに固有のハッシュ値を付与し，それを円周上に並べたオーバーレイネットワークを構成する．ファイルの管理は，そのファイル ID に近い値のノード ID のノードが行う．OneHope では，図 1 に示すように，ノードの参加，離脱を，参加する全ノードに高速に通知するため，階層的な通知処理機構を持つ．ハッシュ空間内を k 個のスライスに分割し，各スライス内は u 個のユニットに分割する．各スライスおよびユニットにはリーダーが決められる．ノードの参加や離脱を検知したノードは，スライスリーダーに通知する．通知を受けたスライスリーダーは他のスライスリーダーに通知し，その通知を受けた各スライスリーダーはユニットリーダーに通知する．また，ユニットリーダーからユニット内のメンバへの通知は，隣接ノード間で行う keep-alive メッセージに多重することによって行われる．

3. 拡張方式

本稿では，地域ごとにサイト（放送局に相当）があり，サイト内に複数のノード（サーバーに相当）が存在する環境を想定する．各サーバーが公開するファイル数は異なる数とする．

提案する OneHop 拡張方式は，サイト間に流れるパケット量を少なくするために localID を用い，さらに各ノードが公開するファイル数に応じたファイル管理の負荷を実現するために仮想ノード ID を用いる．これらは，OneHop に対して，それぞれ単独で適用することも，併用することも可能である．

3.1 ローカルノード ID による拡張方式

ノード ID を並べて構成されるオーバーレイネットワークは，物理ネットワーク上でのノードの配置とは無関係である．そのため，オーバーレイネットワーク上では隣接ノード間の通信であっても，実際のネットワーク上では離れた位置のノード間で通信が行われる．その結果，サイト内，サイト外のノードの位置関係とは無関係に通信が行われる．

ノードの物理的な位置をオーバーレイネットワークに反映させるため，localID を導入する．各サイトに固有の localID を割り当て，元のノード ID の上位ビットを localID に置き換える[4]．以下，この ID をローカルノード ID と呼ぶ．各ノードは，通知処理，Stabilization 処理など，ネットワーク維持の処理にはこのローカルノード ID による管理テーブルを使う．これにより，物理ネットワーク上での近隣ノードが，オーバーレイネットワーク上でも近隣ノード

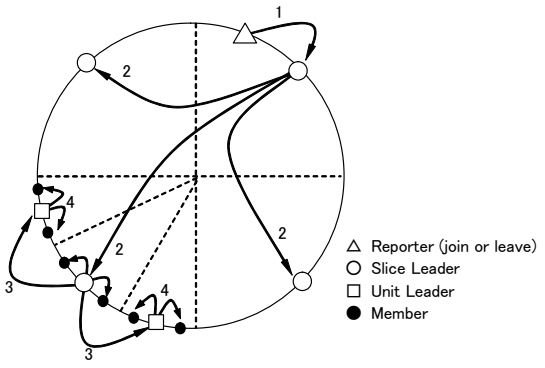


図 1 通知処理の流れ

† 日本放送協会 放送技術研究所, NHK

となり，サイト間を流れるパケット数が減少する．

なお，ローカルノード ID によるテーブルを，ファイル ID の検索用テーブルに用いると，各サイトのローカルノード ID の最小値を持つノードにファイルの管理が集中してしまう．そのため，検索用テーブルには，元のノード ID を使ったテーブルを用いる必要がある．

3.2 仮想ノード ID による拡張方式

ハッシュ関数を用いて生成される ID は一様に分布するため，ノード ID，ファイル ID ともにオーバーレイネットワーク上では均等に配置される．そのため，オーバーレイネットワーク上で管理するファイルは，参加しているノードで均等に分散管理される．参加ノードの公開するファイル数が不均一な場合，公開ファイル数が少ないノードは，相対的に多くのファイルの管理を割り当てられることになり，管理の負荷で不公平性が生じる．

この不公平性を緩和するため，仮想ノード ID を導入する．提案方式では，ネットワーク維持のための管理テーブルとは別に，仮想ノード ID による検索用テーブルを持たせ，ファイル ID の検索に用いる．ここでノードが公開するファイル数を N_p とした場合，各ノードは元のノード ID に加え，(1)式で示す N_{vhost} 個の仮想ノード ID を，検索用テーブルに追加することとする．R は比例係数である．

$$N_{vhost} = [N_p \times R] \quad (1)$$

i 番目の仮想ノード ID は元のノード ID の上位 m ビットを次式により置き換える．

$$\text{仮想ノード}ID\text{の上位}m\text{ビット} = (a \bullet \tilde{f} + \tilde{a} \bullet f) \quad (2)$$

$$a = \text{ノード}ID\text{の上位}m\text{ビット}$$

$$f = din < i >$$

ここで，din<>はビットリバース，~はビット反転，・は論理積，＋は論理和を表す．(2)式により，元のノード ID を中心にハッシュ空間上で均等に仮想ノード ID が割り当てられる．

3.3 提案方式の特長

提案する仮想ノード ID は，文献[5]における virtual node と同等であるが，文献[5]の方式の場合，virtual node 毎に参加や離脱，Stabilization 処理が必要となるため，ネットワークの帯域を消費する．提案方式では，元のノードの参加・離脱の通知メッセージに仮想ノード数を含めることで，各ノードが仮想ノード ID を生成でき，仮想ノードのための参加，離脱，Stabilization 処理は不要で，ネットワークの帯域を消費しない．また，文献[4], [5]など，ルーティングテーブルを分散管理する方式では，検索用テーブルと管理用テーブルとを別にした場合，検索用テーブルにもテーブルを維持するための Stabilization 処理が必要となる．検索用テーブルは局所性が考慮されないため，3.1 で示した局所性を考慮した方法との併用が困難である．提案方式では，検索用テーブルはノード内で管理用テーブルから生成できるので，検索用テーブルを維持するための処理は必要なく，localID と仮想ノード ID の併用が可能である．

4. 実装方法

実験用として Linux 上に OneHop および提案する OneHop 拡張方式を実装した．

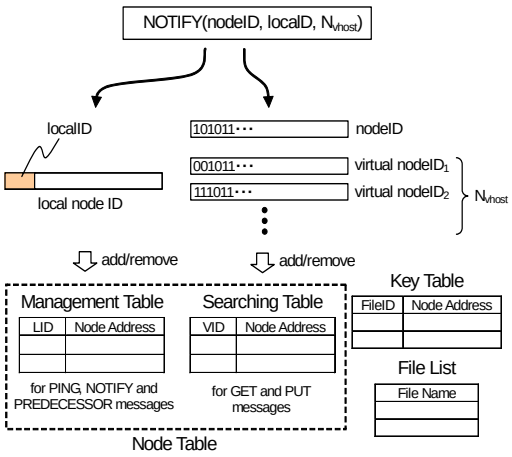


図 2 ノードが持つテーブル

図 2 にノードが持つテーブルおよび，ノード ID，localID，仮想ノード ID との関係を示す．ノードは，ノードテーブル，キーテーブル，ファイルリストを持つ．ノードテーブルは，オーバーレイネットワークに参加しているノード ID が保存される．提案方式ではノードテーブルとして，ローカルノード ID による管理用テーブルと，仮想ノード ID による検索用テーブルの２つを持つ．キーテーブルは，外部から登録（PUT）された(key, value)(ここでは，ファイル ID と，そのファイルを持っているノードアドレス)を保持するテーブルである．本実装では，キーテーブルをソフトステートで管理[6]するため，一定時間(T_l)以内に再登録（PUT）が無いと，そのキーはテーブルから削除する．ファイルリストは，ノードが公開するファイルのリストであり，このリスト内のファイルのファイル ID を，時間間隔 T_p でオーバーレイネットワークに登録(PUT)する．

ノード間で通信されるメッセージは PING, NOTIFY, PREDECESSOR, PUT, GET の５種類である．メッセージの通信には UDP を用いた．これらのメッセージにより，各ノードは，Stabilization 処理，登録/検索処理，通知処理を行う．Stabilization 処理は，隣接ノード(Successor)のチェックを行う隣接ノードチェックと，管理用テーブルに登録されているノードを順番にチェックするテーブルメンテナンス処理の２つを定期的に行う．それぞれのチェック間隔を T_n, T_m とする．近隣ノードチェックおよびテーブルメンテナンス処理では，PING メッセージによるノードの生存確認後，PREDESEDDOR メッセージにより隣接ノード ID を取得し，自分が持つテーブル内のノード ID を比較する．違いがある場合は管理用および検索用テーブルを修正する．

新規に参加するノードは，既知のノードから隣接ノードを教えてもらい，隣接ノードから管理用テーブルをコピーする．ノードリストの取得は，PREDECESSOR メッセージを繰り返し呼び出すことで行う．

文献[2]では，スライスリーダー間，スライスリーダーからユニットリーダーへの通知処理は，一定時間間隔のバッチ処理で行っているが，本実装では，通知速度を速くするため，通知メッセージ毎に処理する．同じ理由から，ユニットリーダーから各メンバへの通知は，keep-alive(PING)メッセージに多重する方法ではなく，ユニットリーダーから各メンバへ直接通知する方法とした．

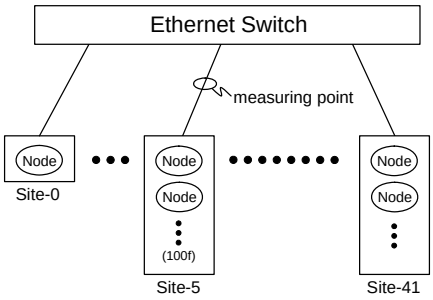


図 3 実験システムの概要

表 1 サイト内のノード数

Site no	Number of node per site	Total number of node
0	1	1
1 - 4	60F	240F
5 - 8	100F	400F
9 - 31	1F	23F
32 - 41	40F	400F
	Total	1063F+1

F: scale factor (1-10)

表 2 実験に用いたパラメータ

Number of slice (k)	256 (8bit)
Number of unit (u)	16 (4bit)
Length of node ID	160bit
Length of local ID	8bit
Interval time of neighbor's check (T_n)	1.05 sec
Interval time of table maintenance (T_m)	1.00 sec
Time to live for key table (T_l)	30 min
Interval time of publication (T_p)	1000 sec

5. 実験結果と考察

実験システムの概要を図 3 に示す．サイト数を 42 とし，各サイトで複数のノードを起動した．実験には 1 サイトを 1 台の PC を使って行った．Site-5 を観測対象とし，サイトに流れるパケットを測定した．また，起動している全ノードの 1 分毎の状態データを収集した．各サイト内で起動したノード数を表 1 に示す．ここで，F はスケールファクターであり，F を 1 から 10(ノード数 1064 から 10641)とした．実験に用いたパラメータを表 2 に示す．

3.1 ローカルノード ID の実験結果

各ノードが公開するファイル数を 100 とし，ノードの参加，離脱および検索がない定常状態での Site-5 を流れるパケット量の測定値および計算値を図 4 に示す．localID を用いることで，サイト間を流れるパケット量を削減する効果があり，この実験での削減量は約 47%であった．これは，localID を用いることで，Stabilization 処理のうち，隣接ノードのチェック処理のパケットがサイトの外に出なくなったためである．

次に，ノードの参加処理，検索処理を含めたときの実験結果を図 5 に示す．実験はノードの参加状態，検索をしない状態，検索をする状態をスケールファクター単位で繰り返した．検索は全ノードが約毎秒 2 回（ポアソン分布）の頻度とした．実験結果から，ノードの参加時にサイト間を流れる NOTIFY および PREDECESSOR メッセージの量が削減されている．NOTIFY メッセージの減少は，localID を使うことで，OneHop の通知処理の階層構造と，実ネットワーク上のノードの配置が一致するため，サイト間を流れるメッセージがスライスリーダー間の通信パケットのみとなるためである．PREDECESSOR メッセージの削減は，ノードが参加するときの初期ノードリストの取得を，localID を用いることで，同サイト内の隣接ノードから取得するようになったためである．

3.2 仮想ノード ID の実験結果

仮想ノード ID の効果を調べるため，各サイト内のノードが公開するファイル数を表 3 で示すようにサイト毎に異なる数にして実験を行った．(1)式の R を 0, 1/50, 1/20, 1/10

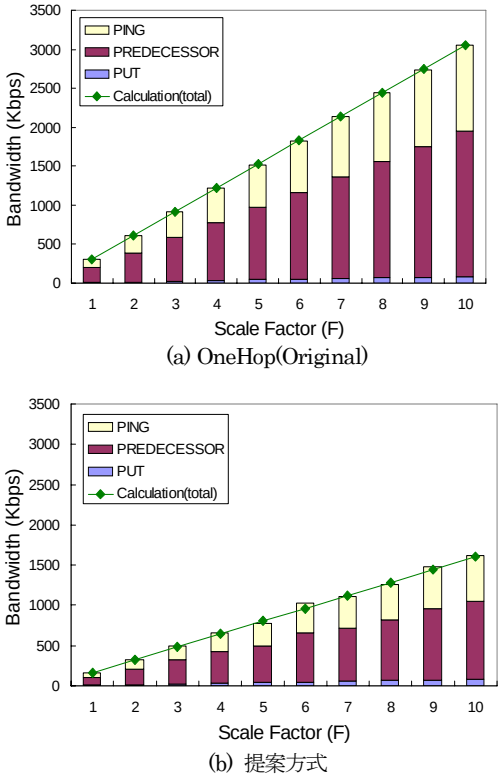


図 4 Site-5 のネットワーク帯域（定常状態）

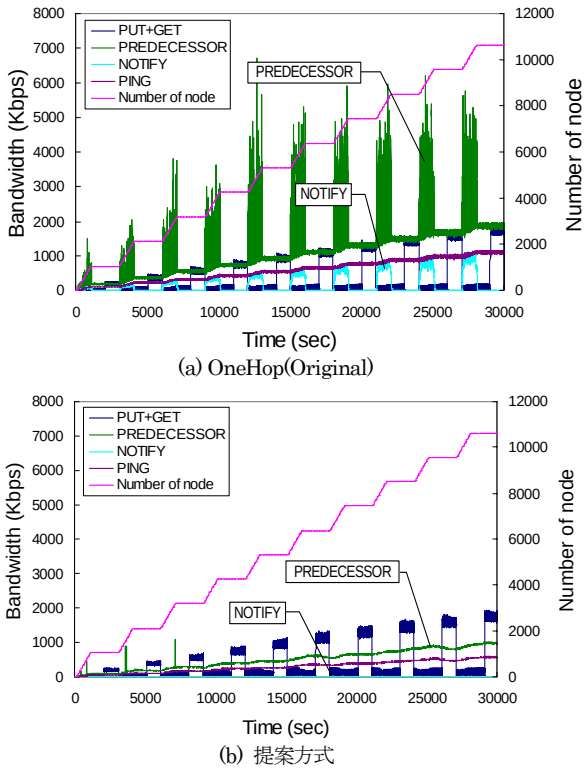


図 5 Site-5 のネットワーク帯域（ノードの参加，検索）

とし，仮想ノード数 N_{vhost} を決定した．

図 6 に，各サイト内のノードが管理するファイル数（キーテーブルのサイズ）の平均値を示す．仮想ノード ID を使わない(R=0)場合，管理ファイル数は公開するファイル数に無関係であり，ばらつきも大きい．一方，仮想ノード ID を使うことで，各サイトとも，公開するファイル数に応じた管理ファイル数となっている．

ノードが公開するファイル数を N_p，ノードが管理するファイル数を N_k としたとき，ファイル管理比率を RF= N_k/ N_p と定義する．このとき，全ノードの RF のヒストグラムを図 7 に示す．また，積算数が全ノードの 90%となる $RF_{90\%}$および，RF の最大値を図 8 に示す．図 7 から，仮想ノード ID を使わない場合には，ファイルの管理をしないノードや，多くのファイルを管理するノードが多数存在する．一方，仮想ノード ID を使った場合には，RF=1(公開するファイル数と管理するファイル数が同数)近辺のノードが増加している．図 8 から，$RF_{90\%}$値，RF 最大値ともに仮想ノード ID を使うことで減少しており，ファイルの管理数を公開ファイル数に応じて制御できることがわかる．R の値は大きいほど精度の高い制御が可能となるが，ノード内部の検索用テーブルサイズが増加する．ノードの公開するファイル数が 1000 ファイル以下を想定した今回の実験では，R=1/50 程度でも効果があることが分かった．

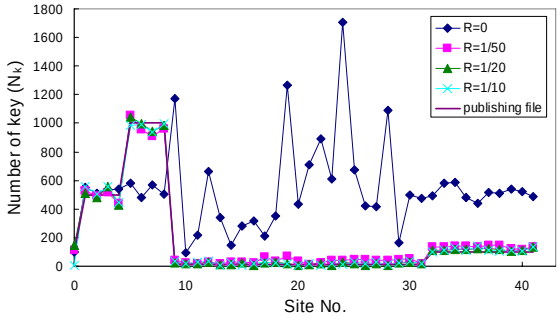
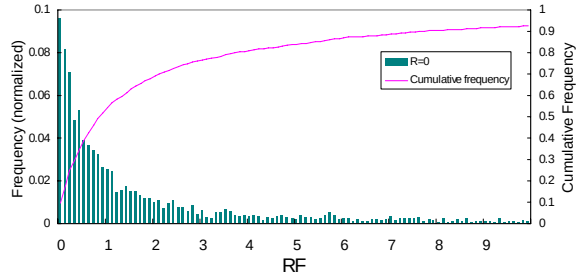
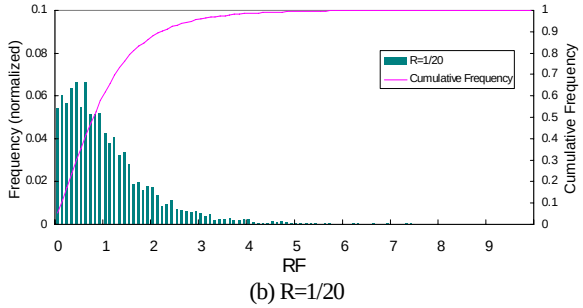


図 6 平均管理ファイル数(F=3(3190 ノード))



(a) R=0



(b) R=1/20

図 7 RF のヒストグラム(F=3(3190 ノード))

表 3 ノードが公開するファイル数

	Site No.				
	0	1-4	5-8	9-31	32-41
Number of file per node	100	500	1000	10	100

6. まとめ

放送局で番組素材ファイルを広域に分散共有することを想定し，コンテンツ管理方法として OneHop 拡張方式を提案し，実験によりその有効性と実現可能性を示した．実験結果から，localID を用いることで，サイト間に流れるパケット量を削減できることを示した．また，仮想ノード ID を用いることで，各ノードの公開ファイル数に応じて，管理ファイル数を制御できることを確認した．

参考文献

[1] H. Balakrishman, M. F. Kaashoek, D. Karger, R. Morris and I. Stoic, “Looking Up Data in P2P Systems,” Communications of the ACM, vol.46, No.2, pp.43-48, Feb. 2003.

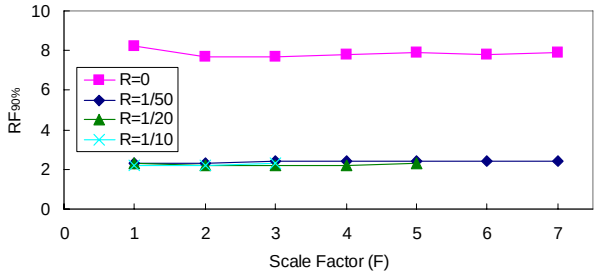
[2] A. Gupta, B. Liskov and R. Rodrigues, “One Hop Lookups for Peer-to-Peer Overlays,” 9th Workshop on Hot Topics in Operating Systems (HotOS-IX), May 2003.

[3] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek and H. Balakrishnan, “Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications,” ACM SIGCOMM’01, Aug. 2001.

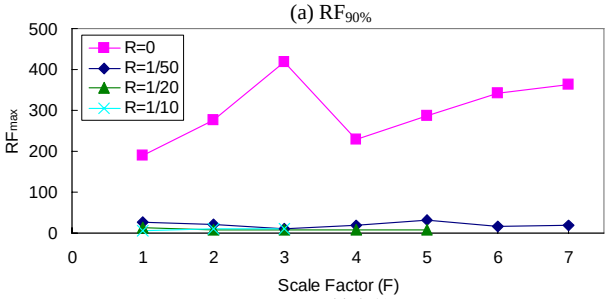
[4] 縣, 金子, 堀内, “DHT を利用したデータ検索システムの高速化手法,” 信学技報, IN2006-70, pp.121-126, Sep. 2006.

[5] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, I. Stoica, “Wide-area cooperative storage with CFS,” SOSP’01, Oct. 2001.

[6] 多田, 今瀬, 村田, “ディレクトリサービスにおけるソフトステートとハードステートの性能比較,” 情報処理学会論文誌, Vol.48, No.5, pp.1936-1951, May 2007.



(a) $RF_{90\%}$



(b) RF 最大値

図 8 RF の比較