

複数の異なる時間間隔に基づく 構造化オーバーレイネットワーク構築手法の評価

Evaluation of a Construction Technique for Structured Overlay Networks Based on Multiple Different Time Intervals

川上 朋也†
Tomoya Kawakami

1. はじめに

IoT (Internet of Things, モノのインターネット) [1]においては多種多様なデバイスがネットワークにつながり、それら IoT デバイスによって発信されるデータも膨大な量となる[2]。このとき、発信されるデータの一例として、温度や消費電力量などのセンサモジュールを用いて測定されたセンサデータが考えられる。センサデータは測定された地点や範囲、日時の情報に深く結びついた空間的かつ時間的なデータで、利用者やシステムはそれらの情報に基づいてセンサデータの検索や利活用を行う。そのため、膨大な量のデバイス (ノード) やデータ、利用者を扱うために高いスケーラビリティを維持しつつ、要求されたデータを効率的に検索可能な仕組みが求められる。

高いスケーラビリティの実現手法として現在、分散ハッシュテーブル (distributed hash table, DHT) や地理的な情報に基づくオーバーレイネットワークなどの構築手法が提案されている[3-14]。これらオーバーレイネットワーク構築手法は各ノードの一次元の ID や多次元の情報に基づいてノード間に論理ネットワークを構築する。ノードの探索はキー (key) と呼ぶ値や範囲を指定することで行い、論理ネットワークを表す各ノードのルーティングテーブルを参照することで、該当するノードへ到達するようにクエリを転送する。既存のオーバーレイネットワーク構築手法では一次元の時間情報や二次元の位置情報に基づいてノード間に論理ネットワークを構築し、さらに各ノードに特定の範囲を割り当てることで、センサデータの分散管理や検索を実現できる。一方、「過去の 8 月の気温を知りたい」「過去の日曜日の消費電力量のみ表示する」など、利用者やシステムは特定の時間的な間隔を指定してセンサデータを要求することが考えられる。特定の時間間隔を持つデータ要求の例を図 1 に示し、本研究ではそれらを「間隔クエリ (interval query)」と呼ぶ。間隔クエリでは指定される間隔も年や月、週、日、時間などの複数が考えられるが、既存のオーバーレイネットワーク構築手法ではこれらの複数の時間間隔に基づく検索を効率的に行えない。

筆者は現在、年や月、週、日、時間など、間隔クエリを効率的に処理可能なオーバーレイネットワーク構築手法を提案している[15, 16]。提案手法では既存の Chord[2]と同様の環状のオーバーレイネットワークを想定し、一次元の時間情報に基づいてキー空間上にノードを配置する。また、年、月、週、日、時間の間隔に基づくクエリを効率的に処理するため、各ノードはキー空間上のそれらの間隔で他ノードへの仮想リンクを構築する。それら複数の間隔で構築した仮想リンクを用いることで、検討手法は要求データの担当

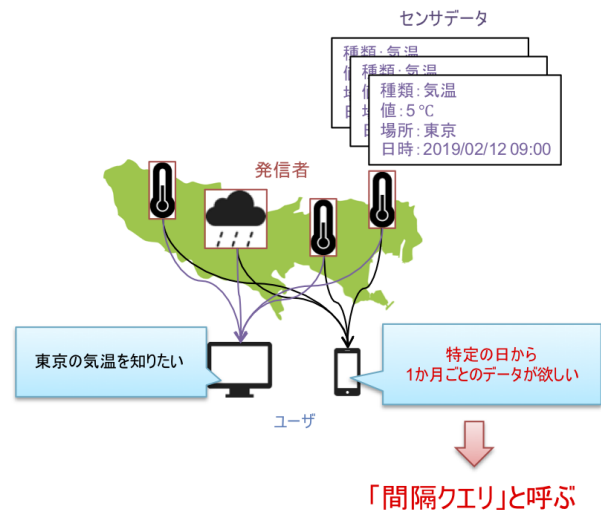


図 1: 特定の時間間隔を持つデータ要求

ノードまでクエリを転送する際のホップ数を削減できる。本研究では、提案手法の評価について述べる。

以下、本研究に関連する既存手法の Chord について 2 章で述べる。本研究の提案手法は 3 章で説明し、その評価について 4 章で述べる。5 章では関連研究について述べ、まとめと今後の課題を 6 章で述べる。

2. Chord

Chord[2]は代表的な環状のオーバーレイネットワークで、各ノードの ID などに基づいて一次元のキー空間上にノードを配置する。キーの長さ (ビット数) はノード数を想定してあらかじめ与え、両端を結んで環状とする。Chord では特定のノードからキー空間上で時計回りに次のノードを successor, 反時計回りに次のノードを predecessor と呼び、各ノードは自身の successor と predecessor へのリンクをそれぞれ保持する。また、各ノードは predecessor から自身までの部分的なキー空間を担当し、その範囲に関するデータや情報を保持する。さらに、Chord ではノード探索におけるホップ数を削減するため、各ノードは finger table と呼ぶショートカットリンクも保持する。キー空間の長さが m ビットかつ各ノードの配置されたキーが重複しない場合、各ノードは $2^1, 2^2, 2^3, \dots, 2^{m-1}$ ビット先のキーを担当するノードへのリンクを finger table としてそれぞれ保持する (2^0 ビット先のキーの担当は successor)。finger table の例を図 2 に示す。図 2 ではキー空間の長さは 2^6 ビット (0~63) で、数字の書かれた円はノードとキーを、矢印や数字、右上の

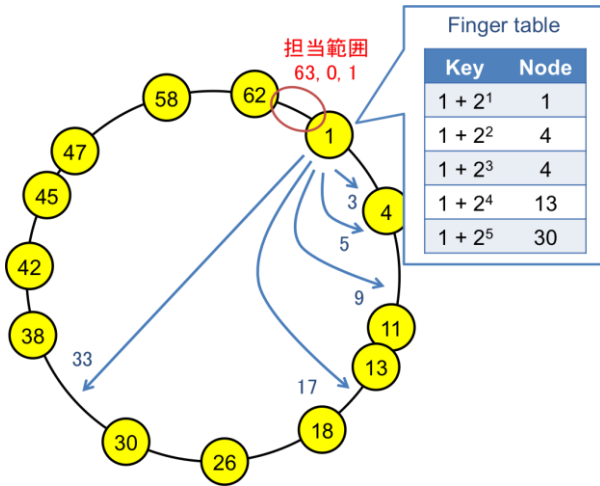


図 2 : finger table の例

表はキー1のノードにおける finger table の仮想リンクや対象のキー、宛て先ノードを表している。また、図 2 ではキー1のノードの predecessor はキー62のノードのため、キー1の担当範囲はキー63, 0, 1となる。Chordでは finger tableにより、各ノードのルーティングテーブルの大きさを最大で $m + 1$ に抑えつつ、 N をノード数とした場合の任意のキーに基づくノード探索で $O(\log N)$ のホップ数を実現している。

Chord は各ノードの担当範囲とノード探索機能によって、データの分散管理にも用いることができる。また、文献[4]では空間充填曲線の Z-ordering を用いて二次元の位置情報を一次元のキーに変換し、二次元の範囲探索 (range query) にも対応している。一方、本研究で想定するセンサデータでは、「過去の8月の気温を知りたい」「過去の日曜日の消費電力量のみ表示する」など、利用者やシステムは特定の時間的な間隔を指定してセンサデータを要求することも考えられる。また、指定される間隔も年や月、週、日、時間などの複数が考えられるが、既存手法では単一のキー間隔のクエリやそれぞれ独立した多次元のクエリを効率的に処理するオーバーレイネットワークを構築できても、一次元かつ複数のキー間隔のクエリを効率的に処理できない。

3. 提案手法

筆者は現在、年や月、週、月、時間など、間隔クエリを効率的に処理可能なオーバーレイネットワーク構築手法を提案している[15, 16]。本章では、提案手法のアイデアとショートカットリンクの構築について述べる。

3.1 アイデア

提案手法では既存の Chord[2]と同様の環状のオーバーレイネットワークを想定し、一次元の時間情報に基づいてキー空間上にノードを配置する。ここでの一次元の時間情報とは、協定世界時 (UTC) や UNIX 時刻のような基準時刻と年や月、週、日、時間、分、秒などの単位を示すキーのビット値、さらにキー空間の長さ (ビット長) で表されるものとする。このとき、既存の finger table ではビット長 m として 2^i ビット先 ($i=1, 2, \dots, m-1$) の担当ノードへのショートカットリンクを構築する。一方、提案手法ではキーが示

表 1 : 構築ショートカットの種類と数

種類	対象キー (間隔)	数
年	2^0 年, 2^1 年, 2^2 年, ..., 2^{m-1} 年	m
月	6か月, 3か月, 1か月	3
日	15日, 7日 (曜日), 3日, 1日	4
時刻	12時間, 6時間, 3時間, 1時間	4
隣接	successor, predecessor	2

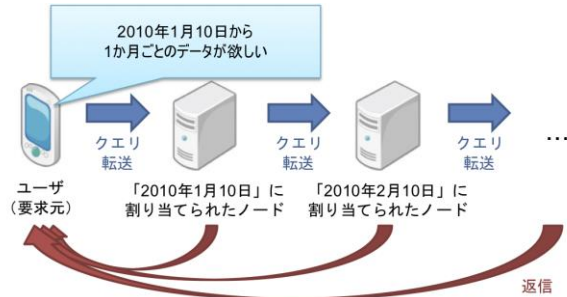


図 3 : 想定する間隔クエリの転送

す最小の時間単位をもとに、1週間先や1か月先、1年先など、実際に利用者から要求されそうな間隔で担当ノードへのショートカットリンクを構築する。さらに、同じ年の場合は1か月先、3か月先、6か月先、12か月先 (1年先) など、同じ時間基準内でもホップ数を削減できる宛て先ノードへも仮想リンクを構築する。本研究では年、月、週、日、時刻に関して考慮し、表 2 に示すショートカットリンクを構築するものとする。また、表 2 ではキーの値を時刻 (1時間) としており、キー空間の長さを m ビットの年で表した場合の各リンク数も示している。なお、分や秒などの基準でも同様に設定できる。

想定する間隔クエリの転送の流れを図 3 に示す。図 3 は環状のネットワークポロジを省略しているが、クエリは関連するノードへリンクを経由して再帰的に転送される。クエリを受信したノードは該当するデータを要求元へ返信する。

3.2 ショートカットリンクの構築

本節では 3.1 節で述べたアイデアに基づいて、各ノードにおけるショートカットリンクの構築について述べる。

提案手法におけるショートカットリンクの構築例を図 4 に示す。図 4 では 2001 年 1 月 1 日からの追加日数をキーとし、キー空間の長さは 64 日間 (2001 年 1 月 1 日~3 月 5 日) である。数字の書かれた円は図 2 と同様にノードとキー (日数) を、矢印や数字、右上の表はキー1のノードにおけるショートカットリンクの仮想リンクや対象のキー、宛て先ノードを表している。図 4 ではキーが日数で、キー空間の長さが 64 日間のため、表 1 に基づき、キー1のノードは以下の間隔を進んだキーの担当ノードへショートカットリンクを構築する。

- ・月 : 1か月 (31日)
- ・日 : 15日, 7日 (曜日), 3日
- ・隣接 : successor (1日), predecessor

図 4 に示したショートカットリンクは図 2 の finger table と比べ、次の月の同じ日を担当するノードや次に同じ曜日を担当するノードへ仮想リンクを構築している。そのため、

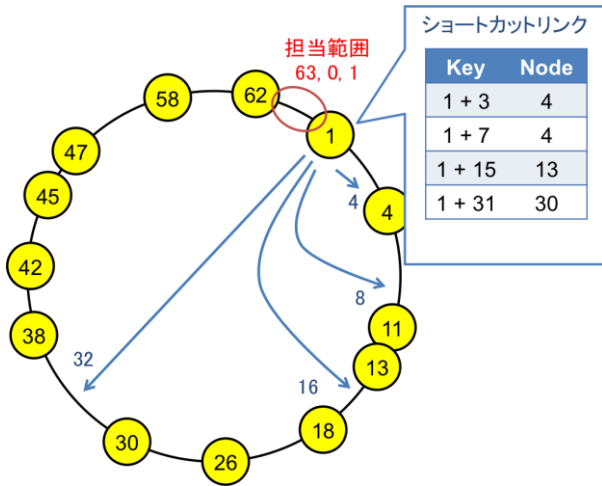


図 4：ショートカットリンクの構築例

「特定の日から 1 か月間隔のデータを知りたい」「同じ曜日のデータを表示したい」など、実際に発生しそうな利用者からの要求に対して、該当する複数のノードへ少ないホップ数でクエリを転送できる。また、ルーティングテーブルのエントリ数はキーとする値やキー空間の長さに依存するが、図 4 ではショートカットリンクの数は 4 で、図 2 の finger table のエントリ数の 5 から減少している。

4. 評価

本研究では、提案手法をシミュレーションによって評価する。シミュレータと各手法は Java によって実装した。シミュレーションにおけるパラメータを表 2 に示し、それらの詳細を以降で述べる。

4.1 シミュレーション環境

本シミュレーションではキー空間の最小単位を 1 時間とし、キー空間の長さを 16 ビット ($2^{16} = 65536$ 時間) とした。ノード数は 1000 から 4000 まで変化させ、各ノードにランダムなキーを割り当て、ノード間にショートカットリンクを構築する。比較手法として、Chord と同様に 2 を基底とした指数先のノードへのショートカットリンクを用いた。要求元から送信されるクエリは 1 つのみであるが、クエリは複数のキー情報を含み、関係するノードへ転送される。本シミュレーションではクエリのパターンに 6 種類を用いた。1 つ目のパターンはキーの数が 12 で、キー空間上でランダムな地点から 720 時間の間隔である（長さの合計は 8640 時間 = 360 日）。2 つ目のパターンもキーの数が 12 であるが、各キーはランダムに決定する。3 つ目のパターンはキーの数が 52 で、キー空間上でランダムな地点から 168 時間の間隔である（長さの合計は 8736 時間 = 52 週）。4 つ目のパターンはキーの数が 52 で、各キーはランダムに決定する。5 つ目のパターンはキーの数が 120 で、キー空間上でランダムな地点から 72 時間の間隔である（長さの合計は 8640 時間 = 360 日）。最後のパターンはキーの数が 120 で、各キーはランダムに決定する。シミュレーションは各環境で 20 回ずつ行い、評価指標としてノード間のメッセージ数とホップ数、各ノードが持つルーティングテーブルの長さ（エントリ数）の平均値を算出した。

表 2：シミュレーションにおけるパラメータ

パラメータ	値
キー空間の最小単位	1 時間
キー空間の長さ	16 ビット ($2^{16} = 65536$ 時間)
ノード数	1000, 2000, 3000, 4000
各ノードのキー	ランダムに割り当て
比較手法	Chord (2 を底とする指数)
クエリ数	1
1 クエリのキー数	15, 52, 120
クエリのパターン	12 キー (間隔: 720 時間), 12 キー (間隔: ランダム), 54 キー (間隔: 168 時間), 54 キー (間隔: ランダム), 120 キー (間隔: 72 時間), 120 キー (間隔: ランダム)
試行回数	20
評価指標	平均メッセージ数, 最大ホップ数, ルーティングテーブルの長さ

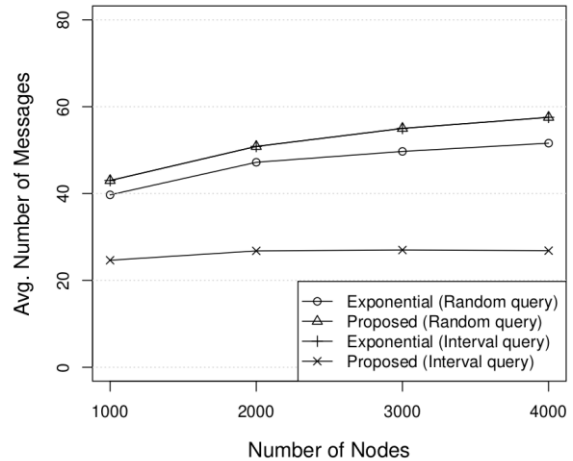


図 5：12 キーにおける平均メッセージ数

4.2 クエリ転送に関する結果

1 つ目 (12 キー, 720 時間間隔) と 2 つ目 (12 キー, ランダム間隔) のクエリパターンにおけるノード間の平均メッセージ数を図 5 に示す。図の横軸はノード数を表している。図 5 より、キーの数が 12 の場合、提案手法で特定のキー間隔 (720 時間) のクエリが最もメッセージ数は小さい。一方、提案手法はランダムなキー間隔のクエリにおいて、比較手法よりもメッセージ数が大きい。また、1 つ目と 2 つ目のクエリパターンにおけるノード間の最大ホップ数を図 6 に示す。平均メッセージ数と同様、提案手法はランダムなキー間隔のクエリではホップ数が最も大きい、特定のキー間隔のクエリではホップ数が最も小さい。

キー数が 52 または 120 のクエリパターンにおける平均メッセージ数や最大ホップ数を図 7 から図 10 に示す。図 7 から図 10 より、提案手法はキー数が 52 や 120 のクエリにおいても、同様に特定のキー間隔のクエリで最も結果が良い。これらの結果から、実際のユーザが要求しうる特定の

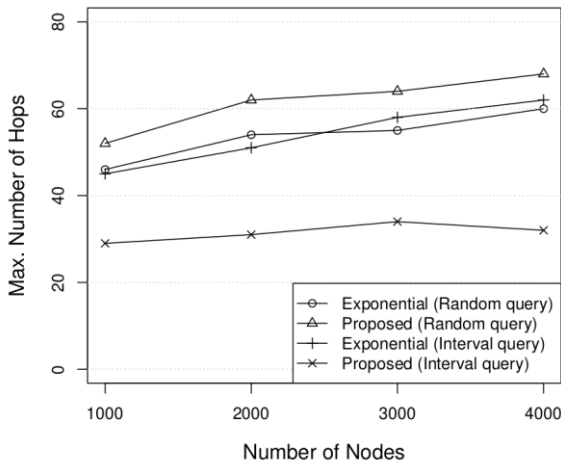


図 6 : 12 キーにおける最大ホップ数

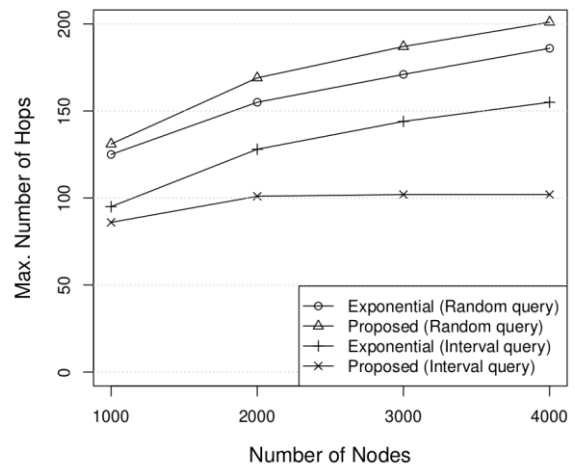


図 8 : 54 キーにおける最大ホップ数

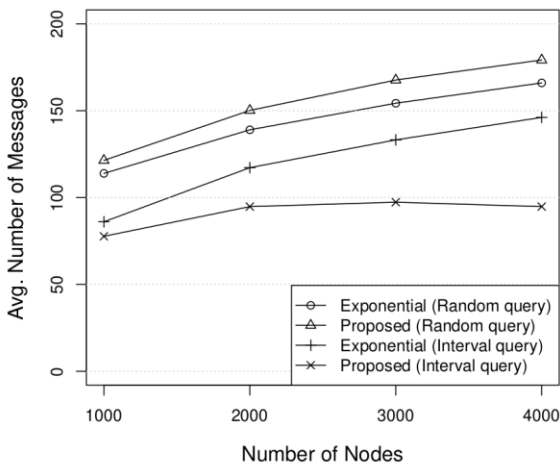


図 7 : 54 キーにおける平均メッセージ数

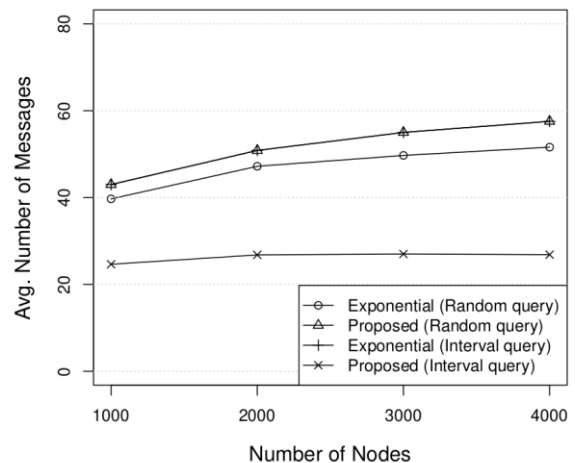


図 9 : 120 キーにおける平均メッセージ数

キー間隔のクエリにおいて、提案手法はメッセージ数やホップ数を削減できる。

4.3 ルーティングテーブルに関する結果

各ノードのルーティングテーブルのエントリ数を図 11 に示す。図の横軸はノード数である。図 11 では、2本の直線はルーティングテーブル内で登録されたキーの数を表している。一方、残りの2本の直線はルーティングテーブル内の一意の宛て先の数を表している。

登録されたキーの数はノード数ではなく、キー空間の長さに影響される。本シミュレーションでは図 11 より、登録されたキーの数は比較手法が約 19 で、提案手法が約 16 である。また、一意の宛て先の数は提案手法が比較手法よりも小さい。この結果から、提案手法はノードの追加や削除が起きる場合でも、ルーティングテーブル維持のためのコストを削減できる。

5. 関連研究

Chord などの DHT 以外にも、多くのオーバーレイネットワーク構築手法が研究されている[3-14]。スキップグラフとその関連手法はスキップリストを P2P モデルへ適用し、ノ

ード (ピア) 間に階層的なリンクを構築する[5-9]。スキップグラフはキーの始点と終点を指定する範囲クエリ (range query) に対応し、クエリはその範囲内か終点を超えないキーのノードへ再帰的に転送される。キーに基づくノード探索のホップ数は $O(\log N)$ で表され、 N はノード数である。また、各ピアのルーティングテーブルの長さは平均で $\log N$ である。一次元の範囲クエリのみでなく、多次元のクエリを扱うオーバーレイネットワーク構築手法も研究されている[10-14]。それらの手法は 2D/3D の地図上の位置や地域に関係するデータの分散管理にも用いられ、「地理的オーバーレイネットワーク (geographical overlay network)」とも呼ばれる。地理的オーバーレイネットワークの多くは R 木や 4 分木、ドロネー図 (ボロノイ図) などの幾何学の要素に基づいている。しかし、これらの既存手法は「間隔クエリ」を想定しておらず、本研究の提案手法と比べて効率的にクエリを転送できない。

オーバーレイネットワークにおける到達性や耐障害性、信頼性を高めるため、関連技術も研究されている[17-24]。まず、複製に関する仕組みとして、非構造化オーバーレイネットワークにおいて path replication や関連手法が提案されている[17-20]。path replication では、コンテンツの要求元か

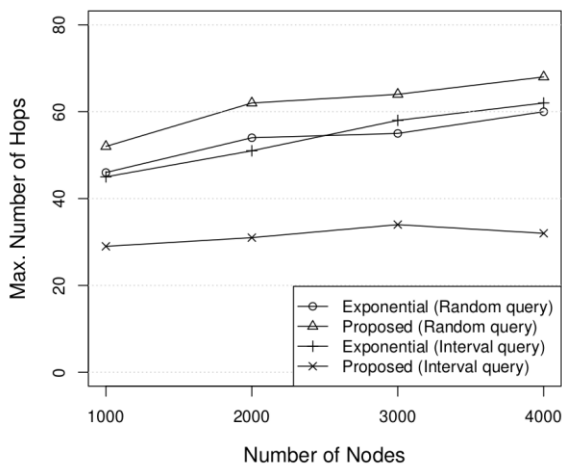


図 10 : 120 キーにおける最大ホップ数

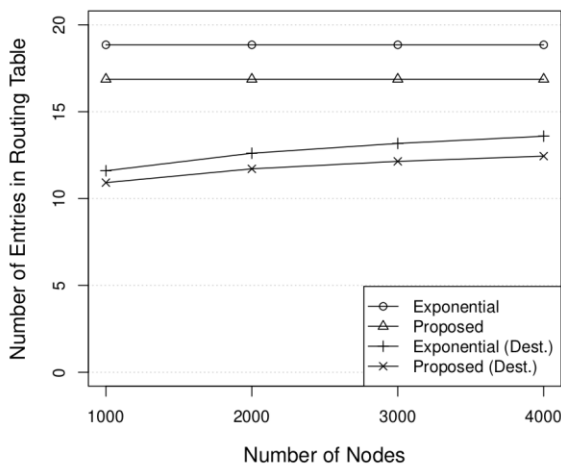


図 11 : ルーティングテーブルの平均エントリ数

ら発信者へ他ノード経由でクエリが転送された際、経路上のノードの一部がコンテンツの複製を保持する。複製に関する仕組みは構造化オーバーレイネットワークにおいても研究されており、複製の管理や発見の効率化が求められる[21-24]。例えば、分散 key/value ストアの Erlang 実装である Scalaris ではデータの可用性のために複製が配置され、データの一貫性のために多数決を用いている[21]。RelaxDHT ではノードはデータのブロックに分けられ、各ブロックでルートノードが複製のメタデータを持つ[23]。これらの手法も指定間隔クエリを想定していないが、本研究の提案手法は到達性や耐障害性、信頼性を高めるため、同様にしくは類似した複製の仕組みを取り入れられる。

6. まとめ

本研究では、年や月、週、月、時間など、特定の複数の時間間隔に基づくクエリ（間隔クエリ）を効率的に処理可能なオーバーレイネットワーク構築手法とその評価について述べた。本研究の提案手法では既存の Chord と同様の環状のオーバーレイネットワークを想定し、一次元の時間情報に基づいてキー空間上にノードを配置する。また、間隔クエ

リを効率的に処理するため、各ノードはキー空間上のそれらの間隔で他ノードへの仮想リンクを構築する。それら複数の異なる間隔で構築した仮想リンクを用いることで、提案手法は要求データの担当ノードまでクエリを転送する際のホップ数を削減できる。提案手法はシミュレーションによって評価し、間隔クエリにおいてメッセージ数やホップ数を削減しつつ、ルーティングテーブルのエントリ数も削減できることを確認した。

今後の課題として、ノード間の負荷の偏りを軽減するため、仮想ノード[3]の仕組みの導入を検討している。また、オーバーレイネットワークの到達性や耐障害性、信頼性の向上のため、割り当てノードやデータを複製する仕組みを検討している。P2P エージェントプラットフォームである PIAX[25]への提案手法の実装やテストベッド上での評価も検討している。

謝辞

本研究の一部は JSPS 科研費 17K00146 および公益財団法人放送文化基金の助成による成果である。

参考文献

- [1] S. Hodges, S. Taylor, N. Villar, J. Scott, D. Bial, and P. T. Fischer: Prototyping Connected Devices for the Internet of Things, IEEE Computer, Vol. 46, No. 2, pp. 26-34 (2013).
- [2] 第 6 章 2030 年の未来像 - ICT が創る未来のまち・ひと・しごと, 平成 27 年版 情報通信白書 第 2 部, 総務省, pp. 326-334, 2015. <http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h27/pdf/n6100000.pdf> (accessed July 1, 2018).
- [3] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan: Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications, IEEE/ACM Transactions on Networking, Vol. 11, No. 1, pp. 17-32 (2003).
- [4] S. Matsuura, K. Fujikawa, H. Sunahara: Mill: A Geographical Location Oriented Overlay Network Managing Data of Ubiquitous Sensors, IEICE Transactions on Communications, Vol. E90-B, No. 10, pp. 2720-2728 (2007).
- [5] J. Aspnes and G. Shah: Skip Graphs, ACM Transactions on Algorithms (TALG), Vol. 3, No. 4 (37), pp. 1-25 (2007).
- [6] A. González-Beltrán, P. Milligan, and P. Sage: Range Queries over Skip Tree Graphs, Computer Communications, Vol. 31, No. 2, pp. 358-374 (2008).
- [7] S. Alaei, M. Ghodsi, and M. Toossi: SkipTree: A New Scalable Distributed Data Structure on Multidimensional Data Supporting Range-Queries, Computer Communications, Vol. 33, No. 1, pp. 73-82 (2010).
- [8] S. Takeuchi, J. Shinomiya, T. Shiraki, Y. Ishi, Y. Teranishi, M. Yoshida, and S. Shimojo: A Large Scale Key-Value Store Based on Range-Key Skip Graph and Its Applications, Proc. of the 15th International Conference on Database Systems for Advanced Applications (DASFAA 2010), Vol. Part II, pp. 432-435 (2010).
- [9] R. Banno, T. Fujino, S. Takeuchi, and M. Takemoto: SFB: A Scalable Method for Handling Range Queries on Skip Graphs, IEICE Communications Express, Vol. 4, No. 1, pp. 14-19 (2015).
- [10] A. Mondal, Y. Lifu, and M. Kitsuregawa: P2PR-Tree: An R-Tree-Based Spatial Index for Peer-to-Peer Environments, Proc. of the International Workshop on Peer-to-Peer Computing and

Databases in Conj. with the 9th International Conference on Extending Database Technology (EDBT 2004), pp. 516-525 (2004).

[11] 金子 雄, 春本 要, 福村真哉, 下條真司, 西尾章治郎: ユビキタス環境における端末の位置情報に基づく P2P ネットワーク, 情報処理学会論文誌データベース, Vol. 46, No. SIG18(TOD28), pp. 1-15 (2005).

[12] E. Tanin, A. Harwood, and H. Samet: Using a Distributed Quadtree Index in Peer-to-Peer Networks, The International Journal on Very Large Data Bases (VLDB), Vol. 16, No. 2, pp. 165-178 (2007).

[13] M. Ohnishi, M. Inoue, and H. Harai: Incremental Distributed Construction Method of Delaunay Overlay Network on Detour Overlay Paths, Journal of Information Processing (JIP), Vol. 21, No. 2, pp. 216-224 (2013).

[14] Y. Teranishi, Susumu Takeuchi, and Kaname Harumoto: HDOV: An Overlay Network for Wide Area Spatial Data Collection, Proc. of the 26th ACM Symposium on Applied Computing (SAC 2011), pp. 506-513 (2011).

[15] 川上朋也: 複数の異なる時間間隔に基づく構造化オーバーレイネットワーク構築手法の検討, 2018 年度情報処理学会関西支部 支部大会, G-22, pp. 1-3 (2018).

[16] T. Kawakami: A Construction Method for Structured Overlay Networks Based on Multiple Different Time Intervals, Proc. of the 2019 World Congress on Information Technology Applications and Services (World IT Congress 2019 Jeju), pp. 81-86 (2019).

[17] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker: Search and Replication in Unstructured Peer-to-Peer Networks, Proc. of the 16th ACM International Conference on Supercomputing (ICS 2002), pp. 84-95 (2002).

[18] V. Gopalakrishnan, B. Silaghi, B. Bhattacharjee, and P. Keleher: Adaptive Replication in Peer-to-Peer Systems, Proc. of the 24th International Conference on Distributed Computing Systems (ICDCS 2004), pp. 360-369 (2004).

[19] H. Yamamoto, D. Maruta, and Y. Oie: Replication Methods for Load Balancing on Distributed Storages in P2P Networks, IEICE Transactions on Information and Systems, Vol. E89-D, No. 1, pp. 171-180 (2006).

[20] K. P. N. Puttaswamy, A. Sala, and B. Y. Zhao: Searching for Rare Objects Using Index Replication, Proc. of the 27th IEEE International Conference on Computer Communications (INFOCOM 2008), pp. 1723-1731 (2008).

[21] T. Schütt, F. Schintke, and A. Reinefeld, Scalaris: Reliable Transactional P2P Key/Value Store, Proc. of the 7th ACM SIGPLAN Workshop on ERLANG (ERLANG 2008), pp. 41-48 (2008).

[22] H. Shen and Y. Zhu: A Proactive Low-Overhead File Replication Scheme for Structured P2P Content Delivery Networks, Journal of Parallel and Distributed Computing, Vol. 69, No. 5, pp. 429-440 (2009).

[23] S. Legtchenko, S. Monnet, P. Sens, and G. Muller: RelaxDHT: A Churn-Resilient Replication Strategy for Peer-to-Peer Distributed Hash-Tables, ACM Transactions on Autonomous and Adaptive Systems (TAAS), Vol. 7, No. 2, article 28 (2012).

[24] A. Pace, V. Quéma, and V. Schiavoni: Scalaris: Exploiting Node Connection Regularity for DHT Replication, Proc. of the

30th IEEE International Symposium on Reliable Distributed Systems (SRDS 2011), pp. 111-120 (2011).

[25] 吉田 幹, 奥田 剛, 寺西裕一, 春本 要, 下條真司: マルチオーバーレイと分散エージェントの機構を統合した P2P プラットフォーム PIAX, 情報処理学会論文誌, Vol. 49, No. 1, pp. 402-413 (2008).