情報処理学会論文誌 Vol.63 No.2 526–538 (Feb. 2022)

複数の時間間隔に基づくオーバレイネットワークにおけるルーティング効率化手法

久保 達也[1,a)] 川上 朋也[1]

受付日 2021年5月10日, 採録日 2021年11月2日

概要：センサデバイスの低廉化が普及の背景にある IoT では，位置や時間と密接に関連付いた情報を大量に扱うという特徴がある．大量のセンサデータを高スケーラビリティで扱うための手法として，筆者らは特に時間の結びつきに着目してクエリを効率良く扱う手法を提案した．既存手法は一定の時間間隔でデータを要求されるケースにおいて効率的なデータ探索を実現するが，処理コンピュータ（ノード）やネットワークへの負荷が増大するという問題があった．そこで本論文では，ノードの仮想化およびルーティングのアルゴリズムを変更することにより効率化を図る手法を提案する．特定ノードへの負荷集中を回避するために仮想化を行ったネットワークについて，提案手法は従来考慮されなかった仮想化される前の物理的なレイヤーへもルーティングの処理対象を広げ，冗長なやりとりを削減する．シミュレーションによる結果から，提案手法はネットワークへの負荷や各ノードの処理回数を軽減できることを確認した．

キーワード：時間データ，間隔クエリ，仮想ノード，負荷分散，環状ネットワーク，分散データ管理

An Enhanced Routing Method for Overlay Networks Based on Multiple Different Time Intervals

Tatsuya Kubo[1,a)] Tomoya Kawakami[1]

Received: May 10, 2021, Accepted: November 2, 2021

Abstract: Due to the spread of sensor devices and improvement of network connection, IoT (Internet of Things) is rapidly growing these days. IoT treats massive data that deeply relate to time or location. Many peer-to-peer approaches have been researched to consider this characteristic factor. The authors also proposed a structured and virtualized overlay network that can efficiently handle time-specific interval queries. However, the existing method had a problem that causes more load to the processing computers (nodes) and the whole network. In this paper, the authors propose an enhanced routing method for an interval-queryable overlay network. To reduce redundant traffic, the authors expand the routing process to the physical node level. The simulation results show the proposed method can reduce the loads of nodes and the network.

Keywords: temporal data, interval query, virtual node, load balancing, ring-shaped overlay network, distributed data management

1. はじめに

近年の急激なデバイス数の増加そして回線品質向上にともない急速な成長を続けるインターネットは，その中でもとりわけセンサデバイスの低廉化や社会からのニーズが高まりを受けた Internet of Things（IoT）分野の発展が著しい．Cisco 社が行った予測によると，世界中のデバイスと接続数は世界の総人口やインターネット接続人口よりも速い年 10%のペースで増加を続けるとされている [1]．特に IoT の根幹である Machine to Machine（M2M，ユーザが介在しない機械どうしの通信）は最も著しい増加をみせる分野になり，接続数は 2018 年の 61 億に比べて 2024 年に

[1] 福井大学大学院工学研究科
Graduate School of Engineering, University of Fukui, Fukui 910–8507, Japan
[a)] mf210408@u-fukui.ac.jp

は 2.4 倍の 147 億になると予想されている．

上記のように大量のセンサデバイスおよび測定データからなるネットワークやサービスを提供するには，従来のクライアントサーバモデルでは負荷の上昇に耐えられなくなることが考えられる．負荷が高くなりすぎるとやがてサーバはクエリを処理しきれなくなり障害が発生し，サービスの可用性を低下させることとなる．対策としてはサーバの設備増強が行われるが，しかし年に十数%と予想されるクライアント側（IoT 機器）の増加に追いつくことは技術的にも経済的にも限界がある．

そこで，この問題の解決策として考えられるのが Peer to Peer（P2P）方式によるネットワークである [2]．P2P はネットワーク上で対等なノード（端末）が協調して処理を行うモデルであり，中央集権的なサーバを持たずにサービスを構築することができる．P2P のネットワークは構造化オーバレイネットワークと非構造化オーバレイネットワークの 2 種類に大きく分かれる．本研究で扱う構造化オーバレイネットワークは，データの分散処理を確実に実現するために分散ハッシュテーブル（Distributed Hash Table, DHT）の仕組みを用いているものが多い．DHT はネットワークに入力されたデータを確実に分散させる能力に優れている．しかし，センサデータは高頻度かつ位置，時間の情報に深く結びついているという特徴があり，DHT ではうまく扱えずに冗長な通信が発生したり負荷が集中したりする可能性がある．そのため，センサデータを効率良く扱うための DHT によらない構造化オーバレイネットワークが提案されている．既存研究は位置情報に基づいてネットワークを構築するもの [3] や，データの効率良い範囲探索を行うためのもの [4] が主であったが，本研究ではクエリの時間的間隔に着目した先行研究 [5], [6] のルーティングを効率化する手法を提案し，冗長な通信の削減とネットワークの負荷低減を目指す．

2. 関連研究

2.1 オーバレイネットワーク

インターネットは Internet Protocol（IP）によって経路が制御されるネットワークを ISP がクライアントに提供し，さらに ISP が ISP どうしあるいはより上位の ISP に接続することによって実現されている．この階層的な構造をたどることによってクライアントからサーバまでの経路が確保され，サービスを利用できる．一方で，サーバを持たない P2P では自立した各ノードが既存 IP ネットワーク上に論理的な異なる構造のネットワークを形成する．このようにアプリケーション層で作られるネットワークはそれより下層の IP ネットワークに覆いかぶさるように仮想化していることからオーバレイネットワーク（Overlay Network）と呼ばれる．オーバレイネットワークの概要を**図 1** に示す．既存のネットワークに制約を受けずに機能を付加することができるため，耐障害性やスケーラビリティを高めたりすることができる．

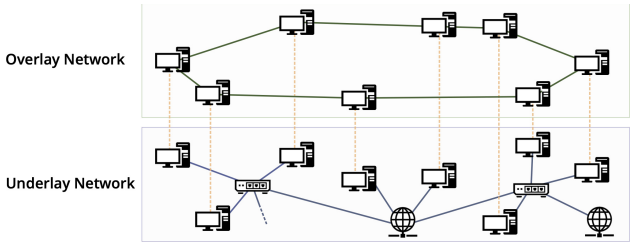

図 1 オーバレイネットワークの模式図
Fig. 1 Overlay and underlay networks.

オーバレイネットワーク上で行われた通信を実際に送信する際にはノードの IP アドレスに基づいてアンダレイネットワーク上で通常と同じルーティングを行う．そのため，オーバレイネットワークネットワーク上でのノード間の隣接関係がアンダレイネットワークと大きく異なることが起こりうる．そのような場合，オーバレイネットワーク上では最短経路の効率良い通信が行えていても，実際にはネットワーク上で遠く離れた場所への無駄なトラヒックを発生させてしまっている可能性がある．このようにオーバレイネットワークにはアンダレイネットワークとの齟齬が少なからず存在している [7]．そのため，アンダレイネットワークへの構造を考慮に入れてオーバレイネットワークを構築することは性能の向上に効果をもたらし，遅延時間やネットワーク層・データリンク層のトポロジといった様々な指標が存在している．

2.2 Chord

Chord [8] は，DHT を利用した構造化オーバレイネットワークのアルゴリズムである．構造化オーバレイネットワークは，ネットワークを生成する際に環状や木構造といった数学的な規則に基づいたトポロジに従ってネットワークを構築するオーバレイネットワークである．ノードの追加時等ネットワークの構造が変化した際にはトポロジを規則正しく保つためそのつど修正を行う必要があり，ネットワークの構築や維持のコストがかかる．またノードの障害といった突発的な事態には弱くなる．一方，ネットワークのトポロジが明確で見つけたいデータの場所を一意に特定することができるため N 個のノードに対してデータの検索が $O(\log N)$ ホップといった低コストで可能であり効率が高いという長所がある．

DHT はこの構造化オーバレイネットワークを実装する方法の 1 つである．DHT ではネットワーク上のノードと入力されるデータはすべて同じ空間上のキーを持つ．各ノードに SHA-1 等のハッシュ関数によってユニークなキー（識別子，ID）を割り当て，そのキーの値によってネットワークのどの位置に配置されどのノードと隣接関係を結ぶかが

定められる．さらにネットワークに入力されるデータにも同じハッシュ関数によってキーを算出し，そのキーとの比較によってどのノードに格納されるかが決定される．データを検索する際にはクエリからハッシュ値を算出し，そこから問い合わせるべきノードを特定する．基本的にハッシュ関数である SHA-1 を利用し，2^{160} の大きさのキー空間を分割する形で各ノードがネットワークを構成するが，それ以外の大きさのキー空間でも同様のアルゴリズムが利用可能である．DHT のアルゴリズムは Chord のほかには CAN [9]，Tapestry [10]，Kademlia [11] 等が代表的である．

Chord によるネットワークの例を**図 2** に示す．各ノードは Successor, Predecessor, Finger Table の 3 種類の経路情報を保持する．ネットワーク上のノードのうち，自身より ID が大きくかつ最も近い ID を持つノードを Successor とし，反対に ID が自身より小さい最初のノードを Predecessor とする．これに加えて離れた場所にあるノードの情報を Finger Table として保持する．この Finger Table には自身から 2^k（$0 < k < N$，N はキー空間のビット数）離れたキーを担当するノードへのリンクを保持する．このとき，ID 空間の終端に達した際には最初に戻って続行する．データを取り扱う際には，データのキーより ID が大きくかつ最も近いノードが担当となる．すなわちノードの ID と Predecessor の ID の間のキーを持つデータを格納する．

Chord のネットワーク上でデータやクエリを転送するときには必ず Successor への一方通行，すなわち ID の昇順の時計回りでのみで転送する．これに加えて Finger Table も通信に利用される．単純に Successor への転送を繰り返した場合に比べ，Finger Table を利用することでネットワーク上のどこに目的のノードがあってもつねに経路の長さを半分程度短縮することができる．これにより，N ノードを持つネットワークでのデータの検索に必要な平均ホップ数は $O(\log N)$ となる．このように，Finger Table は Chord の効率化に大きく寄与するものである．

Chord は環状のネットワークであるためつねに 1 方向にデータを送信すれば目的のノードに到達することができ，末端部等での特別な処理も不要で非常に簡素なアルゴリズムで実装できる．そのため，DHT の最も代表的なアルゴリズムとして知られており関連研究も多い [12], [13], [14]．たとえば，文献 [12] では四分木と，文献 [13] では非構造化オーバレイネットワークと組み合わせたハイブリッド P2P ネットワークが提案されている．

2.3 Chord#

Chord# [4] は Chord の関連研究の 1 つで，DHT の機能をなくしたものである．DHT はハッシュ関数を利用することでデータのキーを分散させ，特定のノードにデータが集中せず，なるべく均等な分布になるよう配慮したものである [15]．一方で，DHT では似たようなデータもまったく異なるキーとなり，ネットワーク上の離れたノードに配置される可能性がある．このことは，データの範囲検索を行いたいときに短所となる．ネットワーク上のあちこちのノードに同時にクエリを多数送信することになり，効率が悪くネットワークの負荷も高いためである．

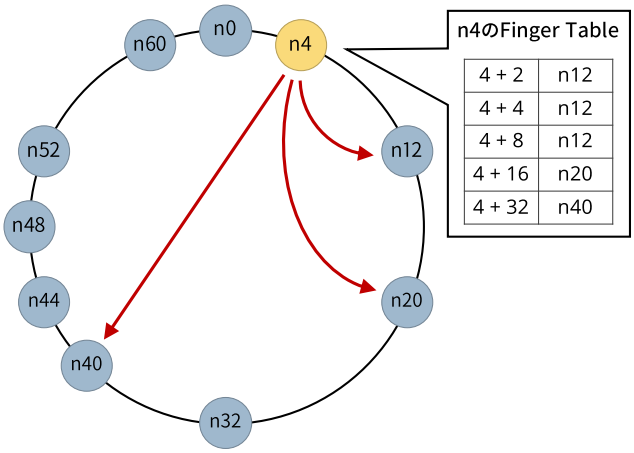

図 2 2^6 の ID 空間における Chord の模式図
Fig. 2 An example of Chord with a 2^6-bit keyspace.

そこで，Chord# ではデータの持つそのままの値をキーとしている．似たようなデータは似た場所に格納されることになるため，範囲検索を行う際にはノードを順番にたどっていけばすべてのデータが得られ，最小限のトラヒックに抑えることができる．Chord# には，Finger Table の更新コストを削減する手法も提案されている [16]．

しかし，Chord# のように負荷分散の仕組みであったハッシュ値を取り除くと，入力データの偏りがそのままネットワークに反映される．そのため，特定のキーのデータが多い場合，その担当ノードに負荷が集中する．Chord# ではその対策として，過負荷になっているノードの隣に動的に負荷が低いノードが入ってデータを分割する手法 [17] を利用している．

2.4 複数の時間間隔に基づくネットワーク

IoT において中心的存在となるセンサデータは高頻度で更新され位置・時間の情報に深く結びついているという特徴がある．よって，通常のオーバレイネットワークとは異なる特性を持たせる必要がある．

付加すべき機能には様々なものが考えられるが，ここでは本研究の基となるデータと時間との結びつきについて着目した研究 [18] について詳細に述べる．すなわち，センサデータの時間的な結びつきによりネットワークの利用者（またはシステム）はデータをある特定の時間間隔という条件で複数要求することが考えられる．例としては「去年までの 7 月の気温」や「毎週日曜日の電力消費量」といったものである．このような複数のクエリで構成されるクエリをまとめて間隔クエリと呼び，概要を**図 3** に示す．間隔

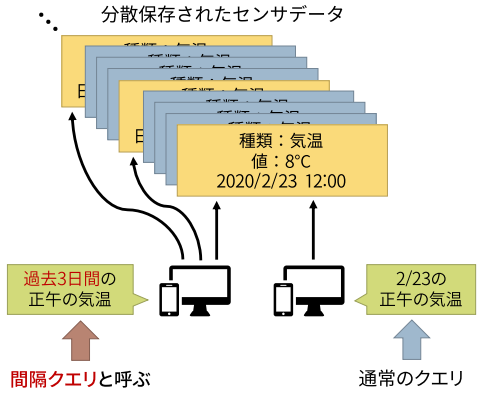

図 3 間隔クエリの概念図
Fig. 3 The concept of an interval query.

表 1 構築するショートカットの種類と数
Table 1 Types and quantities of the constructed shortcut links.

種類	間隔	数
年	2^0 年，2^1 年，2^2 年…$2^n - 1$ 年	n
月	6 カ月，3 カ月，1 カ月	3
日	15 日，7 日，3 日，1 日	4
時	12 時間，6 時間，3 時間，1 時間	4

クエリは過去のデータによる統計情報を得て，今後の予測を立てるといったビッグデータ等の用法に有用であると考えられる．

この研究では Chord と同じ環状のネットワークを想定し，データの記録された時間をキーとしてネットワークを構築する．ここでデータの記録された時間は UNIX 時間といった特定の基準日時からの相対的なものによる単純な一次元の値とする．Chord$^\#$ と同じく，連続したデータを効率良く扱うためにハッシュ値は使用せずに直接時間情報をキーとする．一方で間隔クエリは様々な単位で要求が行われ，ネットワーク上での単位時間より離れた間隔となるため単純な範囲クエリとは異なったものである．そこで，この手法では Chord の Finger Table に相当するものを間隔クエリを想定した複数の時間間隔に基づくものに差し替える．

テーブルの各要素が指すリンク先を 2 のべき乗とった単一のカテゴリではなく，週，月，年といった利用者から要求されることが想定される単位でショートカットを構築する．さらにたとえば月の場合は 1 カ月，3 カ月，6 カ月といったように，同じ単位の中でもさらに細分化したショートカットを構築する．先行研究では 1 時間を単位時間として年，月，日，時について考慮し，**表 1** に示すショートカットを構築した．ただし，表中の n はキー空間のサイズのビット数を示す．

この手法も DHT ではないため，Chord$^\#$ と同じ問題が起こる．すなわち，センサがデータを取得した時刻が偏っているとその時刻の担当ノードが多くのデータを保管しなければならず負荷が集中することになる．そのため，筆者らは仮想化を利用して負荷分散を行った [5], [6]．ここでの仮想化とは，あるマシン（以下，特定する必要があるときは物理ノードと呼ぶ）がネットワーク上に 1 つだけではなく複数のノード（仮想ノード）を持つことである．仮想ノードを持つことによって物理ノードがネットワークを担当する範囲が広くなり，1 つのデータがネットワークに与える影響は相対的に小さなものになる．仮想ノードの総数が十分なもので，物理ノードに均等かつ一様に割り当てられていれば確率的に負荷は均等になっていく．ハッシュ値によってネットワークの入力データを分散させるのではなく，ネットワークの側が分散することによって同様の効果をもたらすことを狙ったものといえる．

3. 提案手法

3.1 既存手法の問題点

2.4 節で述べた既存研究には，仮想化がない場合に比べ物理ノードの負荷は軽減されるがネットワーク上のトラヒックが増大するという解決されていない問題点がある．従来は p 個の物理ノードが単一のノードとしてネットワーク上に参加していたものを各物理ノードが v 個の仮想ノードを持った場合，ネットワーク上に存在するノードの数は p 個から pv 個へと増加することになる．そのため，何も対策をしなかった場合 Chord で示された法則に従い平均ホップ数はネットワーク上のノードの数に対して対数で増加していくこととなる．クエリを処理するのに必要なホップ数が増加することはネットワーク上でやりとりされるメッセージの総数も増加させる．メッセージ数が増えるとネットワークの帯域を過剰に消費するだけではなく各ノードがメッセージの転送や応答を行う回数も増加し，負荷分散の性能を低下させることになる．間隔クエリは必ず複数のクエリで構成されるためにメッセージ数増大の影響は単一クエリよりも大きなものとなり，特に対策が必要である．

3.2 冗長なトラヒックの削減

3.2.1 問題点の原因

前節の問題は，オーバレイネットワークで仮想化を行う場合に一般に起こりうる問題である．構造化オーバレイの 1 つである Skip Graph [19] に対して仮想化を適用した小西らの研究 [20] にも同様の問題が発生していた．

Skip Graph [19] は，線形リストを拡張したデータ構造の Skip List [21] を構造化オーバレイネットワークに適用したアルゴリズムである．Skip Graph は Chord と同じ平均 $O(\log N)$ ホップでの検索が行え，また範囲検索を効率良く行うことができる．しかし欠点として，構造上 1 つのノードにはただ 1 つのデータしか格納することができない．ノード仮想化を導入することでこの問題を解決したが，前節の問題が発生した．

小西らはこのメッセージ数増大の原因を物理ノード間での冗長なやりとりとした．そして，ネットワークのトポロジは異なるが同様の問題がこの複数の時間間隔に基づくネットワークでも発生していると考え，これへの対策が有効な手段であると考えた．

すなわち，同じ物理ノードが同一のクエリをルーティング中に何度も受け取ってしまうという点が原因の 1 つであると仮定した．仮想化されたネットワークにおいてはネットワーク上に仮想ノードのみが存在することとなる．よって通常の Chord と同じルーティングを行った場合には仮想ノード単位でのルーティングであり，物理ノードの存在は考慮されない．そのため，ネットワーク上では最短経路での最適な通信が行われているように見えても，実際には同じ物理ノードの間での通信が繰り返し行われていたり同じ場所を往復していたりといった冗長な通信が発生していることが考えられる．これは，2.1 節で述べたオーバレイネットワークの齟齬と似たような問題である．

図 4 にこの問題点を示した．この図はネットワークの一部を切り取って簡略化したものである．丸が仮想ノードを表し，書かれているアルファベットはどの物理ノードに関連付けられているかを示す．

また，アルファベット右下の添字は同じ物理ノードが持つ何番目の仮想ノードかを示すインデックスである．いま，この図の左端の仮想ノード A_1 から右端の仮想ノード B_2 までクエリが転送されていくとする．説明のため，ショートカットテーブルは省略しつねに Successor に対してのみ通信を行うとする．仮想ノード単位で見ると順番に重複せずネットワークを流れているように見えるが，たとえば物理ノード A に着目すると 2 度同じクエリを受信して転送していることが分かる．このように，図中の全体では物理ノード A と B に対して冗長な通信が発生している．

3.2.2 物理ノードを考慮したルーティング

この問題点への対処として，クエリ転送時に選択する転送先の候補を仮想ノードだけではなく物理ノード単位に拡大したルーティング手法を提案する．従来は仮想ノードはクエリを受け取ったら，そのままその仮想ノードが持つショートカットテーブルもしくは Successor から転送先を決定していた．それに対し提案手法は，以下の手順によってルーティングを行う．ただし，物理ノード P は仮想ノード $v_0, v_1, v_2, ..., v_n$ を持っているものとする．また仮想ノード v のキーを $K(v)$，クエリの宛先のキーを q とする．

手順 1　仮想ノード v_k（$0 \leq k \leq n$）に対して送られてきたクエリを，いったん物理ノード P が受け取る．

手順 2　P は配下の各仮想ノード v_k についてショートカットテーブルや Successor でルーティングを行い，v_k からの次の転送先 v'_k をそれぞれ収集する．

手順 3　収集された次ホップ候補 v' の中から，最も $K(v'_k)$ と q の距離が近く，かつ $K(v'_k)$ が q を超えない，つまり最もクエリの宛先に近い v_k から次の仮想ノードへ転送する．

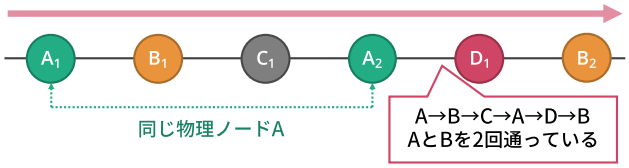

図 4 物理ノード間での冗長な通信の概要
Fig. 4 Redundant traffic among physical nodes.

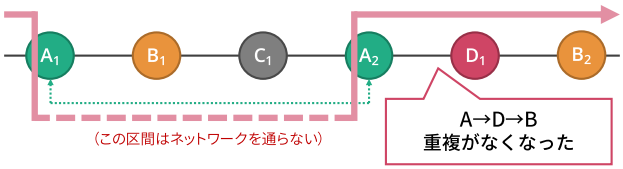

図 5 物理ノードを考慮したルーティングの提案手法
Fig. 5 The proposed routing method considering physical nodes.

すなわち，より低レベルである物理ノードからの視点もルーティングに加味する手法である．手順 2 では仮想ノードのルーティングテーブルを参照するのみであり，外部との通信は発生しない．ルーティングテーブルを参照する処理は増加するが，$O(n)$ の処理でありまたルーティングテーブルの要素数も決まっているため大きな負担とはならない．また，手順 1 でクエリを受信した仮想ノードと手順 3 で送信した仮想ノードが異なっていたときには，単に物理ノードのプログラム内で転送元を付け替えるだけで処理することができるため，完全にネットワークを迂回して経路を短縮できる．なお，手順 3 において基本的に $K(v'_k)$ と q の距離はキーの差により求め，$K(v'_k)$ が q を超えないかはキーの大小を比較して判断する．しかし q より $K(v'_k)$ が大きい場合は経路上でネットワークで最もキーが大きなノードに達してキーが最小のノードに戻る，すなわちキー 0 をまたぐ箇所が存在するためそれを考慮したうえで距離や大小関係を求める．

図 5 に提案手法によるルーティングを示した．図 4 と同じ状況で，仮想ノード A_1 から仮想ノード B_2 までクエリが転送されていく模式図であるが，ここでは最初にクエリを受信した物理ノードの A がネットワーク上に持っているもう 1 つの仮想ノード A_2 のほうがより目標に近いためショートカットを行っている．その結果，図 4 では物理ノード A，B が 2 回ずつクエリを受信していたが，重複がなくなり，経路長も 5 ホップから 2 ホップへ軽減された．なお，ここで仮にクエリの担当ノードが仮想ノード B_1 や C_1 だった場合は，仮想ノード A_2 にショートカットするとクエリの宛先のキーを超えてしまう．そのため，この動作は行われずにクエリを受信したのと同じ仮想ノード A_1 から次のノードへ転送する．次にクエリを受け取る物理ノード B も，同様の理由で仮想ノード B_1 から B_2 へのショー

トカットを行わない．

提案手法のネットワークは木構造のように途中で経路の枝分かれはなく，必ずキー順に一直線である．またクエリの伝送はキーの小さい方から大きい方への一方に限られている．1 度物理ノード内で最もクエリの宛先に近づける仮想ノードから転送を行えば，残りの仮想ノードは伝送経路ですでにクエリが伝送された範囲のキー，または宛先よりも大きなキーを持つことになる．これらのノードはクエリの伝送経路として選ばれないものであるため，この後に同じクエリを受信することはない．よって，この手法により 1 回のクエリ転送で特定の物理ノードを経由するのは 1 回のみとなり，物理ノード間での冗長なやりとりを解消することができる．

本研究では環状の Chord をベースとしたネットワークに提案手法を適用したが，それ以外にも以下の条件を満たすオーバレイネットワークに適用することが可能である．

- ノードが仮想化されていること
- ネットワークの構造が一意な構造化オーバレイネットワークであること
- ノードを識別するためのキーに一意な順序関係があること

この中でも，データに偏りが多く負荷分散のために物理ノードが多くの仮想ノードを持つ必要があることが，提案手法の効果が特に得やすい条件となる．

4. 評価

提案手法によってネットワークの複雑さが増し，ホップ数等の指標の定式化は難しいと考えられるため，ここでは仮想化を行った複数の時間間隔に基づくオーバレイネットワークのシミュレーションにより提案手法の有用性を示すための評価を行った．シミュレータは Java を用いて実装し，本章ではシミュレーション環境とその結果について述べる．

4.1 シミュレーション環境

ここでは，すべてのシミュレーションで共通した環境について述べる．本シミュレーションではキーの最小間隔を 1 時間に設定し，キー空間のサイズは 16 ビットとした．よってノードのストレージ容量が十分であれば最大で $2^{16} = 65{,}536$ 時間 ≈ 7.48 年 のデータを格納することができる．すべての仮想ノードは表 1 に示すショートカットを持ち，キーは空間内からランダムに選択される．すべての物理ノードはあらかじめ決められた数の仮想ノードを持ち，どの物理ノードがどの仮想ノードを担当するかはランダムに決定される．

ネットワークに入力される間隔クエリは，186 時間の間隔を開けた 52 個の連続した昇順のクエリで構成される．これは，1 週間間隔で 1 年分のデータを要求することと等しい．間隔クエリの起点となるキーは，一様分布，正規分布，Zipf 分布に従う．正規分布は平均 $\mu = 2^{15} = 32{,}768$，分散 $\sigma = \frac{2^{15}}{3} \approx 10{,}922.7$ とした．$\pm 3\sigma$ の範囲に ID 空間が収まり，正規分布の性質より 99.73%のクエリ起点がキー空間の中に収まる．Zipf 分布とは，一部のデータに要求が集中し処理の大半を占める一方で，これと比較してあまり要求されないデータが大量に存在するという特性を持つ分布である．Zipf 分布は Web やマルチメディアのキャッシュサーバ等の様々な領域で現れることが分かっており，重要な分布である [22]．具体的には「n 番目に頻度の高い事象の確率が最も頻度の高い事象の $\frac{1}{n^p}$ の確率で起きる分布」であるが，ここでは定数 p は 1 とした．

間隔クエリはネットワーク上を再帰的に転送されていく．各クエリの担当ノードは，受信した間隔クエリの中から自身が担当となるクエリを取り除き次のクエリの担当ノードへ向けて間隔クエリを転送する．52 個目である最後のクエリの担当ノードは単一のクエリを受け取ることとなり，その処理をもって間隔クエリ全体の処理が終了する．

比較手法として，従来の仮想ノード単位でのみのルーティングを同条件で行う．また，簡略化のために通信途中でのノードの参加や離脱は起こらないものとした．ここまでに述べた共通の実験条件を**表 2** に示す．

4.2 ネットワーク負荷に対する評価

この節では，間隔クエリをすべての担当ノードに転送するのにネットワーク全体への負荷がどう変化したかを評価するため，システム全体でのメッセージ数を指標として評価を行う．評価は 1 度に 1 つのみのクエリを送信して行うため，ここでのメッセージ数はホップ数と同じ指標と見なすこともできる．

4.2.1 この実験での追加条件

ネットワーク上の各物理ノードが 1，2，4，8，16，32 個の仮想ノードを持ったとき，物理ノード数を 250，500，750，1,000 個と変化させるとどのようにメッセージ数が変化していくかを確認した．なお，1 個の仮想ノードを持つときにはネットワーク上で仮想化は行われないことになる．間隔クエリを 20 回送信し，転送にかかったメッセージ数の平均を評価指標とする．

4.2.2 結果

クエリ起点を一様分布とした際の既存手法，提案手法それぞれの結果を**図 6** と**図 7** に示す．どちらの手法でも仮想ノードがない場合はルーティング方法は同じため結果はまったく同じになっている．

まず，どちらの手法でも仮想ノードがあるときはないときよりもメッセージ数は多くなり，メッセージ数は増大するという全体的な傾向は変わっていない．しかし既存手法（図 6）と比較すると提案手法（図 7）は仮想ノードを持つときの結果がすべて減少しており，提案手法は一定のメッ

表 2 共通の実験条件
Table 2 Common settings in simulation environments.

パラメータ	値
キーの単位時間	1 時間
キー空間の大きさ	16 ビット（65,536 時間）
各仮想ノードのキー	一様なランダム選択
仮想と物理の関係	一様なランダム選択
間隔クエリのパターン	168 時間間隔で 52 個
間隔クエリの起点	一様分布，正規分布，Zipf 分布

表 3 提案手法によるメッセージ数の削減率（一様分布）
Table 3 The message reduction rate under a uniform distribution.

物理ノード数	仮想ノード数				
	2	4	8	16	32
250	2.65	2.74	4.80	4.65	5.00
500	0.62	2.75	3.48	4.60	4.82
750	1.48	0.70	2.77	3.33	5.62
1,000	0.21	1.76	3.80	2.90	5.27

（単位：%）

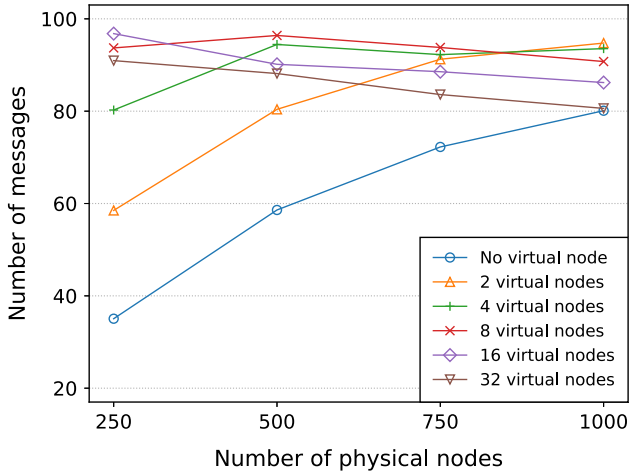

図 6 間隔クエリ起点が一様分布でのメッセージ数（既存手法）
Fig. 6 The number of messages under a uniform distribution (existing method).

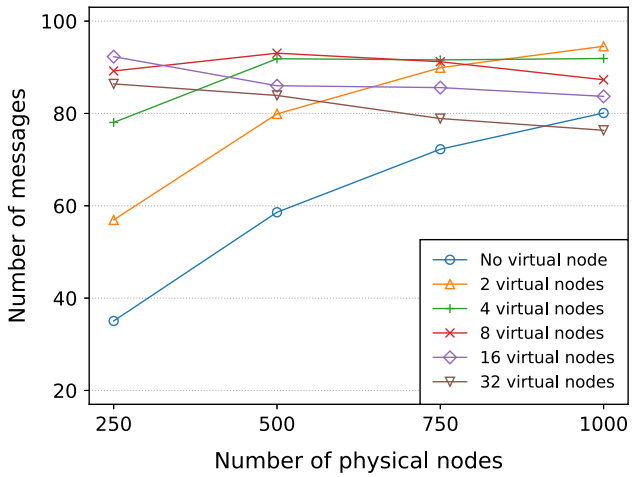

図 7 間隔クエリ起点が一様分布でのメッセージ数（提案手法）
Fig. 7 The number of messages under a uniform distribution (proposed method).

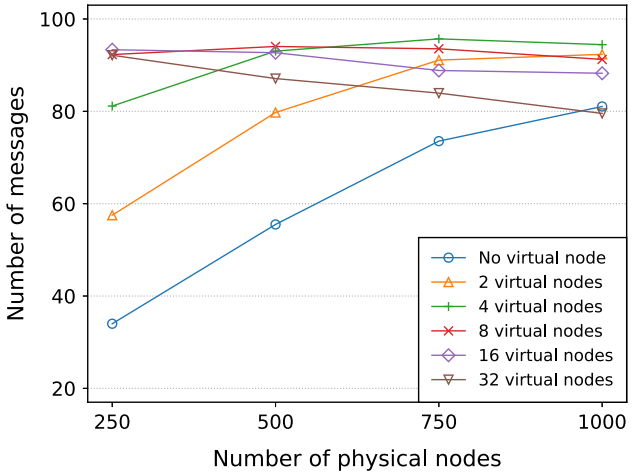

図 8 間隔クエリ起点が正規分布でのメッセージ数（既存手法）
Fig. 8 The number of messages under a normal distribution (existing method).

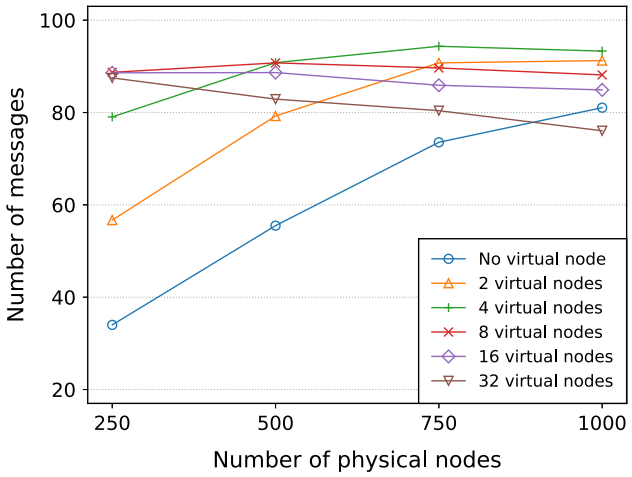

図 9 間隔クエリ起点が正規分布でのメッセージ数（提案手法）
Fig. 9 The number of messages under a normal distribution (proposed method).

セージ数の削減効果があることが示された．**表 3** に各条件について提案手法によって既存手法から削減されたメッセージ数のパーセンテージを示す．全体として仮想ノードの数が多いときほど提案手法の効果が大きくなり，多くの場合同じ物理ノード数のときには単純に仮想ノードが多いほうが削減量が大きくなっていることが分かる．また，同じ仮想ノードでは物理ノード数が少ないほうがわずかに削減率が上昇する傾向がある．これらの結果は仮想ノードが多くなるほど同一物理ノードの内でのショートカットが動作する機会が多くなり，また物理ノードが少ないほどそのショートカット効果を維持したままネットワークの規模が小さくなるためだと考えられる．

次に，クエリ起点を正規分布とした場合のそれぞれのメッセージ数を**図 8**，**図 9** に，既存手法と提案手法の差を

表 4 提案手法によるメッセージ数の削減率（正規分布）
Table 4 The message reduction rate under a normal distribution.

物理ノード数	仮想ノード数				
	2	4	8	16	32
250	1.39	2.59	3.90	5.09	5.05
500	0.63	2.42	3.51	4.37	4.82
750	0.38	1.41	4.17	3.32	4.23
1,000	1.19	1.22	3.40	3.80	4.40

（単位：%）

表 4 に示す．既存手法，提案手法とも平均値は一様分布のときとほとんど変わらずグラフの全体的な傾向も変わらない．しかし仮想ノードが 8，16 個では一様分布のときに比べ特に物理ノードが多いときに削減効果が大きくなっているのに対し，2，4 個ではほとんど変わらず，効果が若干小さくなっているところも存在する．このように，一様分布のときと比べて仮想ノードの数による削減効果の差が若干大きくなっていることが分かる．また，同一仮想ノード数のとき物理ノードの数が少ないほど削減率が向上する傾向が一様分布のときに比べて強くなっている．

最後に，クエリ起点を Zipf 分布とした場合の既存手法，提案手法それぞれのメッセージ数を**図 10**，**図 11** に，既存手法と提案手法の差を**表 5** に示す．結果はほとんど正規分布と同じ傾向になった．ショートカットリンクが期待されるとおり適切に機能しているため正規分布よりも偏りが大きい Zipf 分布においてもメッセージ数の増大が起きなかったと考えられる．一方で，提案手法によるメッセージ数の削減率については物理ノード数に依存する傾向が強まっているほか，ある程度の仮想ノード数があっても削減率が減少している箇所がある．これは，ネットワークの規模が拡大して 1 つのノードの担当範囲が狭くなってもノードへのリクエストが分散せず集中状態が解けないため，負荷分散が追いつきにくくなっているためと考えられる．

4.3 物理ノードの処理負荷に対する評価

この節ではメッセージ数の変化が各物理ノードの負荷にどのような影響をもたらすかを明らかにするため，間隔クエリの処理中に物理ノードが何回アクセスされるかを計測した．

4.3.1 本実験での追加条件

本実験では物理ノードのアクセス数の分布を集計し，どのような傾向があるかを示す．ここでアクセス数とは各ノードがクエリを送信，転送，受信した回数のことであり，クエリの転送経路上にあるノードすべてに 1 回ずつカウントアップされる．すなわち，物理ノードがクエリを処理した回数の総計である．ノードがデータを蓄えるストレージに対する負荷とは別に，ネットワークとの通信を処理する負荷を集計することを意図している．実際にはクエリに対する処理によって処理負荷は異なると考えられるが，ここでは純粋なカウントのみとし重み付けは行わない．

基本的な条件は前節の実験と変わらないが，いくつか変更した点がある．まず，前節の実験により物理ノードが少なく仮想ノード多いときに特に提案手法の効果があることが分かったため，物理ノード数を 300，仮想ノード数を 32 とした．また本実験ではネットワーク全体の物理ノードについて計測を行い，そこから分布を示すため試行回数が少

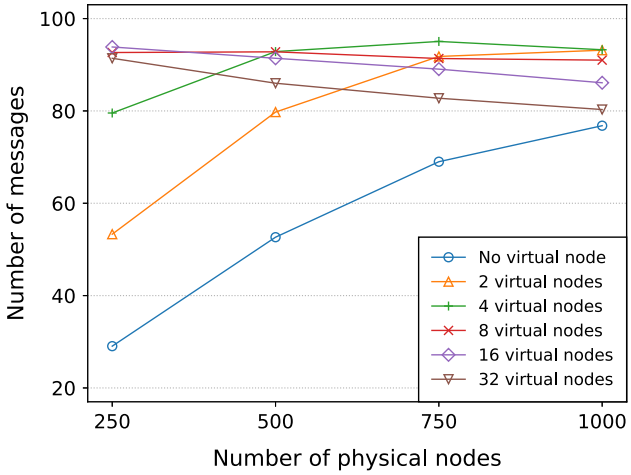

図 10 間隔クエリ起点が Zipf 分布でのメッセージ数（既存手法）
Fig. 10 The number of messages under a Zipf distribution (existing method).

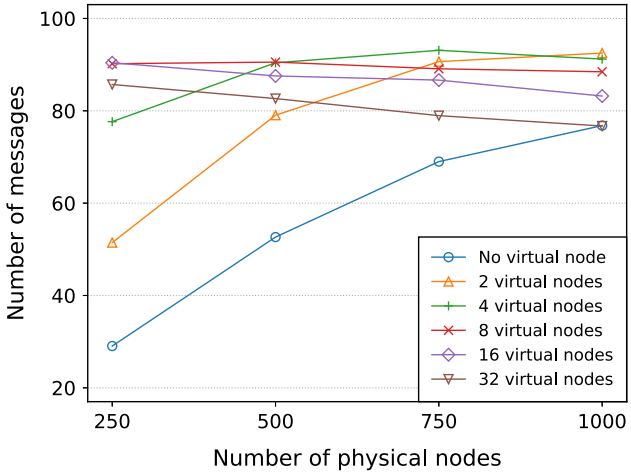

図 11 間隔クエリ起点が Zipf 分布でのメッセージ数（提案手法）
Fig. 11 The number of messages under a Zipf distribution (proposed method).

表 5 提案手法によるメッセージ数の削減率（Zipf 分布）
Table 5 The message reduction rate under a Zipf distribution.

物理ノード数	仮想ノード数				
	2	4	8	16	32
250	3.47	2.39	2.64	3.68	6.24
500	0.88	2.64	2.42	4.21	3.90
750	1.25	2.05	2.46	2.70	4.59
1,000	0.70	2.20	2.80	3.37	4.48

（単位：%）

ないとネットワーク全体にクエリが行き渡らずうまく結果を得られない．そのためこの実験では仮想クエリを 180 回送信し，その累計を結果とした．

4.3.2 結果

まず，クエリ起点を一様分布とした際の結果をヒストグラムとして**図 12** に，箱ひげ図として**図 13** に示す．箱ひげ図（図 13）を見ると最小値，最大値，四分位数がすべて提案手法は下がっていることが分かる．一方で四分位範囲の大きさはほとんど変化していない．また，ヒストグラム（図 12）では提案手法は全体的にアクセス数が少ない方へ 1 階級ずれていることが分かる．これらから，提案手法によって物理ノードへのアクセス数は偏りなく均等に削減され，どの物理ノードもルーティング処理の負荷が下がるといえることが示された．

次に，クエリ起点を正規分布とした際の結果を**図 14** および**図 15** に示す．ヒストグラム（図 14）は前項の図 12 のよりも中央値付近の尖りが低くなり，その分アクセス数の多い方に裾が広がっている．これはクエリの宛先が偏って分散が大きくなったことに加え，一部のノードにアクセスが集中していることを示している．そのため，一様分布のときよりも提案手法によって平均よりアクセス数の少ないノードが増加していない．またどちらも同程度の数の外れ値的な高負荷ノードが発生しているが，提案手法では 100 アクセス以上のノードは存在せず 90 アクセス程度が上限に抑えられている．

箱ひげ図（図 15）においては，図 13 に比べ四分位範囲が 3，4 割程大きくなっている．これに加えて第 3 四分位数から最大値，すなわちアクセス数が多い部分のひげの長さが顕著に伸びている．これはクエリの宛先が偏って分散が大きくなったことに加え，一部のノードにアクセスが集中していることを示している．一方で前項と同じく四分位数はすべて提案手法により下がっていることに加え，提案手法は既存手法に比べてアクセス数が多い部分のひげの長さが 2 割ほど短くなり最大値が大きく減少している．よって，提案手法は負荷が集中しているノードを減らすことにも効果があるといえる．

最後に，クエリ起点を Zipf 分布とした際の結果を**図 16** と**図 17** に示す．Zipf 分布は正規分布に比べさらに偏りが大きいため極端に一部のノードにアクセスが集中する結果となった．箱ひげ図（図 17）は前 2 つの結果よりも最大値が大幅に増大しているほか，四分位範囲もこれまでの倍程度になっておりアクセスの偏りがはっきりと現れる結果となった．四分位数については，いずれも提案手法は一様分布・正規分布の際と同じ程度の削減効果があったが，最大値の削減効果はあまり大きくならなかった．

ヒストグラム（図 16）においても Zipf 分布による負荷の偏りが明らかになっている．アクセス数が比較的少ない 100 アクセス以下の物理ノードについては，これまでと同様に分布を負荷の少ない方へ下げる傾向が見られた．しかし，最大値が非常に大きいため相対的に小さな削減効果しか得られなかったと考えられる．

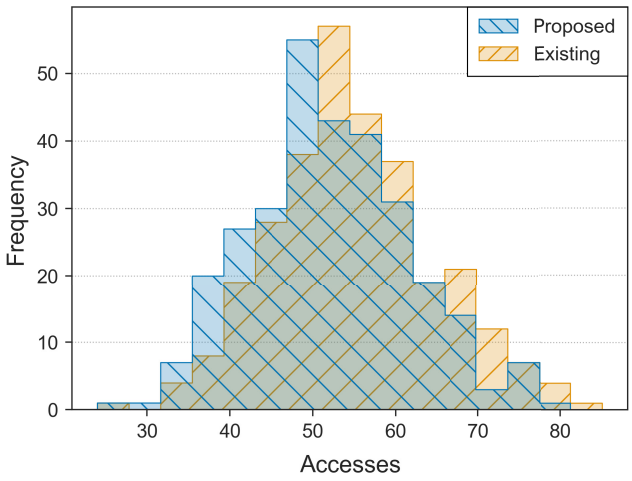

図 12 間隔クエリ起点が一様分布での物理ノードへのアクセス数分布

Fig. 12 A histogram of messages to physical nodes under a uniform distribution.

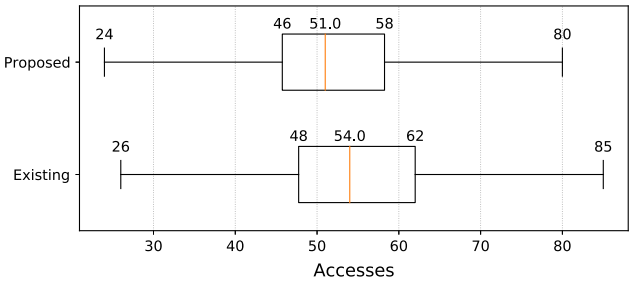

図 13 間隔クエリ起点が一様分布でのアクセス数分布

Fig. 13 A box plot of messages under a uniform distribution.

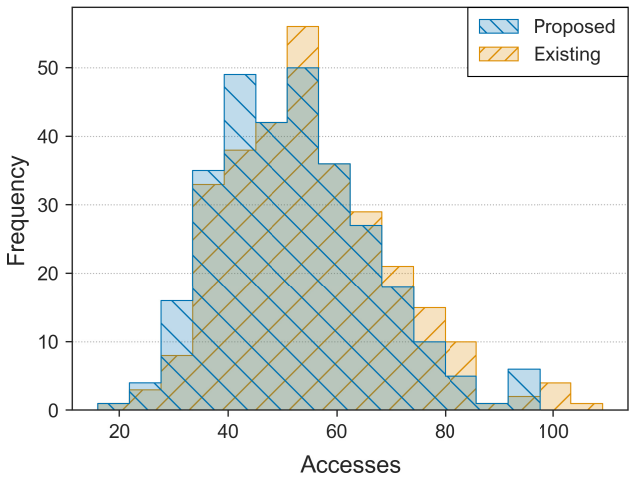

図 14 間隔クエリ起点が正規分布での物理ノードへのアクセス数分布

Fig. 14 A histogram of messages to physical nodes under a normal distribution.

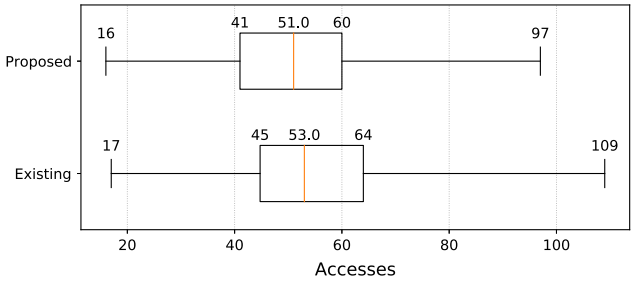

図 15 間隔クエリ起点が正規分布でのアクセス数分布
Fig. 15 A box plot of messages under a normal distribution.

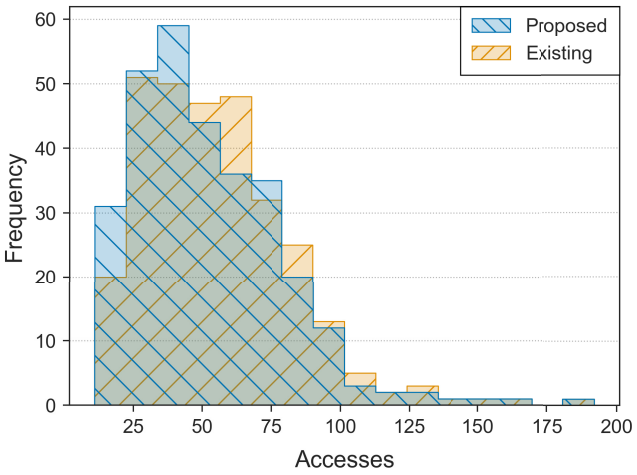

図 16 間隔クエリ起点が Zipf 分布での物理ノードへのアクセス数分布
Fig. 16 A histogram of messages to physical nodes under a Zipf distribution.

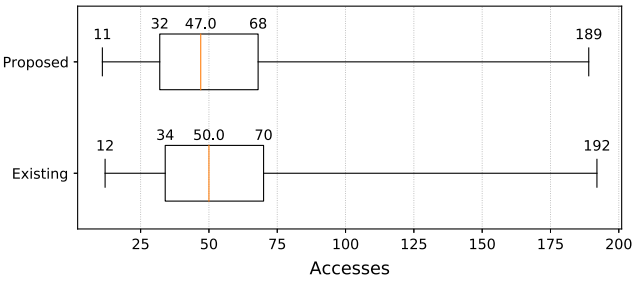

図 17 間隔クエリ起点が Zipf 分布でのアクセス数分布
Fig. 17 A box plot of messages under a Zipf distribution.

4.4 考察

4.4.1 メッセージ数の変化について

提案手法のメッセージ数への影響を検証した 4.2 節では，仮想ノードや物理ノードが少ないうちは多いほどメッセージ数が急速に増加していく傾向が見られるが，すべての結果で増加していくメッセージ数は一定で頭打ちとなっている．また仮想ノード 8 個以上では物理ノードが多いほどかえってメッセージ数が減少していく傾向が見られ，特に 32 仮想ノードでは顕著である．これは，間隔クエリの転送が再帰的であることが有効に働いているためと考えられる．図 7 と同じ条件で再帰的なルーティングを行わず，間隔

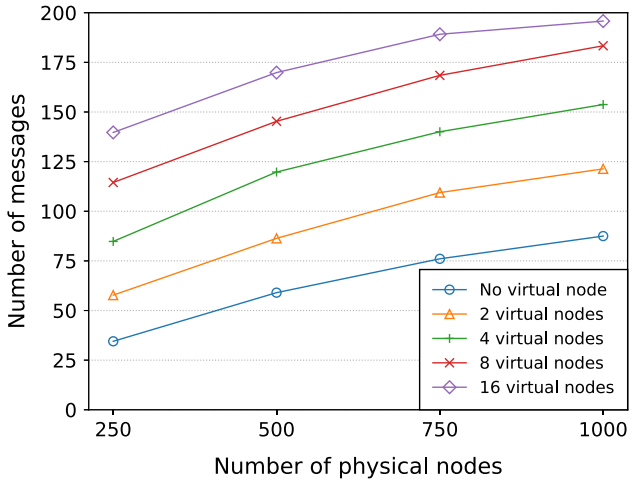

図 18 すべてのクエリが要求元から送信した際のメッセージ数
Fig. 18 The number of messages without recursive query forwarding.

クエリ内のすべてのクエリを要求元から送信するよう変更した追加実験の結果を**図 18** に示す．このときメッセージ数が一定で頭打ちになる傾向は見られず，仮想ノード数・物理ノード数の増加に合わせてつねにメッセージ数は増加していった．間隔クエリは要求元から最初のノードに到達したあとはキー空間のサイズに比して比較的短い一定間隔（本論文では 156 時間，キー空間の 0.24%）でのクエリが連続することとなるため再帰的な転送ではほとんどのクエリの転送が短いメッセージ数に収めることができ，それが平均を収束させることにつながっていると考えられる．

4.4.2 提案手法の効果

表 3 および表 4 は提案手法によって既存手法からどれだけメッセージ数を削減できたかの割合を示す．この表によれば，提案手法はつねに既存手法よりもメッセージ数が減少し増加することはなかった．この点より提案手法は当初の目的を満たしたが，一方で削減量は最大でも 5%程度である．

ルーティングの経路上で各物理ノードがどの仮想ノードを選択したかを詳細に確認した結果，多くの場合クエリを受け取った仮想ノードがそのまま次の転送に使用するノードに選ばれていることが分かった．すなわち，多くの場合既存手法と変わらないルーティングが行われているということである．

このことを詳細に調べるため，物理ノード数 250 で仮想ノード数 16 のネットワークに一様，正規，Zipf の各分布で間隔クエリを 20 回送信したときに，何ホップ目でどれだけの割合で物理ノード単位でのショートカットが作動したかを集計した．その結果を**図 19** に示す．始まりの方，特に最初の 1 ホップではどの分布でも高い確率で提案手法が働いているものの，その後は時折 10%程度の確率で作動するにとどまっている．各分布でどれだけの割合で提案手法が動作したのかを**表 6** に示す．一様分布では 5%のホッ

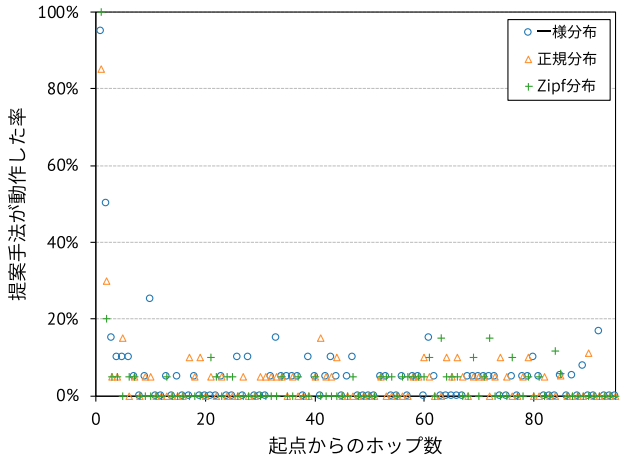

図 19 ホップごとに提案手法によるショートカットが動作した割合
Fig. 19 The usage rate of shortcut links on each hop.

表 6 ショートカットが動作したホップの全体割合（20 回平均）
Table 6 The rate of hops that the shortcut links have worked (average for 20 examinations).

	ホップ数	動作回数	動作割合
一様分布	90.15	4.8	5.33%
正規分布	89.55	3.9	4.38%
Zipf 分布	89.9	3.35	3.72%

プで提案手法が動作しているが，正規分布，Zipf 分布と偏りが大きくなるにつれて低下している．

提案手法が働いた回数が数%程度にとどまった理由として，前節でも触れた間隔クエリが比較的狭い間隔であることが影響している可能性がある．最初の 1 ホップは担当ノードまで遠く離れている可能性があるが，それから後は一定間隔となるため長距離の転送がほとんどなくなると考えられる．この後の区間でショートカットがときおり機能しているのは，担当区域の狭い仮想ノードが間に挟まっていた等が理由として考えられる．

また，4.3 節でも述べたとおりクエリに偏りが大きいときにはネットワーク上のノード数を増やしても偏りが解消されず，ネットワークの規模が増大してノードが遠くなるという副作用が大きくなった結果，ショートカットが動きにくくなったとも考えられる．

4.4.3 各ノードでの処理負荷の低減について

4.3 節では，提案手法は各物理ノードでのアクセス数を軽減させることが示された．外れ値を出すことなく全体で均等な負荷軽減が行えたうえ，リクエストに偏りがあるときは外れ値的に負荷の高いノードを抑えることができた．

一方で，提案手法ではクエリ処理時に物理ノードで行うべき処理が既存手法よりも多い．既存手法ではなかった同じ物理ノードの他の仮想ノードのルーティングテーブルをすべて調べるという動作が入る．仮想ノードでのルーティングは，現状ルーティングテーブルにある候補ノードのキーを for 文で線形探索することで実装している．ルーティングテーブルの要素数を T とすると，表 1 より間隔クエリを効率良く扱うためのルーティングテーブルにおいては $T = 11 + \log_2 N$（N はキー空間の大きさ）である．よって仮想ノード数を V とすると探索のコストは既存手法の $O(\log_2 N)$ に比べて提案手法は $O(V \log_2 N)$ となる．

この変化は物理ノードの中で行われる処理についてであり，クエリ転送のために必要な外部から追加の情報が必要になるようなことはない．よって，他のノードに問い合わせて情報を得ることが ms 単位の待ち時間がかかる処理となることと比較すれば，物理ノード内部で完結するこの処理は負荷や処理時間に無視できるほど小さな影響しか与えないといえる．

各物理ノードへのアクセス数，すなわち処理を行う頻度としては，先述のとおり提案手法によって均等に軽減させることができ，リクエストに偏りがある際は外れ値的にアクセスの多いノードの発生を抑えることができた．このことから，データのアクセスに偏りがあって負荷分散が必要となるネットワーク環境で，その中でも特に間隔クエリや範囲検索を利用するために DHT ではなく仮想化を用いて負荷を分散させる必要がある環境において，提案手法は有効であるといえる．

5. おわりに

5.1 まとめ

近年広まる IoT は，時間や位置に深く結びついたデータを高頻度かつ大量に扱うという今までのネットワークとは異なる特徴を持つ．このため，従来のクライアントサーバモデルに代わる分散ネットワークの仕組みが必要である．既存手法である複数の時間間隔に基づくネットワークは，時間の結びつきに着目して複数のデータの検索効率を向上させた．この手法ではノードを仮想化することでノードへの負荷の偏りを軽減していたが，それによりネットワークへの負荷が高くなるという問題が残されていた．

本論文では，この問題を解決するために新たなルーティングの手法を提案した．仮想ノードだけではなく，物理ノードレベルへルーティングの対象を広げることで仮想ノードどうしのネットワークでは見えない同じ物理ノード間での冗長な通信をなくすことができる．この手法をシミュレーションで実験した結果，既存手法よりメッセージを削減できることを確認した．仮想ノードが多いときに効果が大きく，またクエリの傾向によっては物理ノード数との関係も見られた．メッセージ数の減少にともない各物理ノードへのアクセス回数も均等に減少し，特に外れ値的にアクセスの多いノードを減らすことができた．これにより，提案手法によって各物理ノードの負荷を軽減する効果も期待できる．

5.2 今後の課題

今後の課題として最も大きいものが，メッセージ数削減効果のさらなる向上である．本論文では最大でも 5%程度にとどまり，最も理想的な結果であると考えられる仮想ノードを使用しないときの結果とは隔たりが大きい箇所があった．この問題の解決を目指し，以下のことが具体的にあげられる．

まず，本論文の提案手法はあくまで単一のクエリの転送経路内での重複を防ぐことを目指したものであった．しかし間隔クエリは複数の連続したクエリで構成されるものである．よって，複数のクエリにまたがって内容を先読みするといった間隔クエリ全体で重複を防ぐ仕組みが必要となる．

また，クエリの分布に偏りが大きくなると提案手法が動きにくくなる傾向も見られた．クエリの偏りは負荷の分散には悪条件となるが，図 19 からは一様分布では前半部分で動作率が高い一方で Zipf 分布では後半のほうが動作率が高いといった傾向も見られており，分布の性質によって最適化することも今後の展望として望まれる．

Chord の平均メッセージ数は $O(\log N)$ であり，対数規模ではあるがネットワーク上のノード数が直接メッセージ数に影響する．よって，ネットワーク上にノードを増やす仮想化だけによってメッセージ数を削減することには限界がある可能性も考えられる．ハッシュ値以外での仮想化以外のノードの負荷を軽減するための方法には，動的なノード再割当てがある．これには Chord# で使用されている負荷の高いノードに割り込む形のもの [17] や，仮想化されたネットワークで平時はデータを持たずに負荷が増加した際には受け皿となる仮想ノードを用意する手法 [23] 等が提案されている．こういった他の手法を複数の時間間隔に基づくネットワークに適用してメッセージ数の削減等につなげられるかの検討についても今後の課題としたい．

謝辞 本研究の一部は G-7 奨学財団研究開発助成事業の助成による成果である．

参考文献

[1] Cisco: Cisco Annual Internet Report（2018～2023 年），White paper, Cicso VNI (2020).
[2] Zarrin, J., Aguiar, R.L. and Barraca, J.P.: Resource Discovery for Distributed Computing Systems: A Comprehensive Survey, *Journal of Parallel and Distributed Computing*, Vol.113, pp.127–166 (2018).
[3] 北條真史，長尾洋也，宮尾武裕，首藤一幸：柔軟な経路表に基づく二次元平面上の構造化オーバレイ，情報処理学会論文誌，Vol.56, No.2, pp.439–447 (2015).
[4] Schütt, T., Schintke, F. and Reinefeld, A.: Range Queries on Structured Overlay Networks, *Computer Communications*, Vol.31, No.2, pp.280–291 (2008).
[5] 久保達也，川上朋也：複数の異なる時間間隔に基づく構造化オーバレイネットワークでのノード仮想化の検討，2020 年度情報処理学会関西支部 支部大会，G-42 (2020).
[6] Kawakami, T.: A Node Virtualization Scheme for Structured Overlay Networks Based on Multiple Different Time Intervals, *Applied Sciences*, Vol.10 (2020).
[7] 藤樫淳平：オーバレイネットワーク構築のためのネットワークトポロジ提供機構に関する研究，修士論文，奈良先端科学技術大学院大学 (2009).
[8] Stoica, I., Morris, R., Liben-Nowell, D., Karger, D.R., Kaashoek, M.F., Dabek, F. and Balakrishnan, H.: Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications, *IEEE/ACM Trans. Networking*, Vol.11, No.1, pp.17–32 (2003).
[9] Ratnasamy, S., Francis, P., Handley, M., Karp, R. and Shenker, S.: A Scalable Content-Addressable Network, *ACM SIGCOMM Computer Communication Review*, Vol.31, No.4, pp.161–172 (2001).
[10] Zhao, B.Y., Huang, L., Stribling, J., Rhea, S.C., Joseph, A.D. and Kubiatowicz, J.D.: Tapestry: A Resilient Global-Scale Overlay for Service Deployment, *IEEE Journal on Selected Areas in Communications*, Vol.22, No.1, pp.41–53 (2004).
[11] Maymounkov, P. and Mazières, D.: Kademlia: A Peer-to-Peer Information System Based on the XOR Metric, *Proc. IPTPS 2001*, pp.53–65 (2002).
[12] Wu, J.-G., Jiang, N., Zou, Z.-Q., Hu, B., Huang, L. and Feng, J.-L.: HPSIN: A New Hybrid P2P Spatial Indexing Network, *Journal of China Universities of Posts and Telecom*, Vol.17, No.3, pp.66–72 (2010).
[13] Duan, Z., Tian, C., Zhou, M., Wang, X., Zhang, N., Du, H. and Wang, L.: Two-Layer Hybrid Peer-to-Peer Networks, *Peer-to-Peer Networking and Applications*, Vol.10, pp.1304–1322 (2017).
[14] Tetarave, S.K., Tripathy, S. and Ghosh, R.K.: V-Chord: An Efficient File Sharing on LTE/GSM Network, *Proc. ICDCN 2018* (2018).
[15] Karger, D., Lehman, E., Leighton, T., Panigrahy, R., Levine, M. and Lewin, D.: Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web, *Proc. ACM STOC '97*, pp.654–663 (1997).
[16] 呉 承彦ほか：Chord#における経路表の維持管理コスト削減手法の提案とその評価，情報処理学会論文誌，Vol.53, No.12, pp.2752–2761 (2012).
[17] Karger, D.R. and Ruhl, M.: Simple Efficient Load Balancing Algorithms for Peer-to-Peer Systems, *Proc. SPAA 2004*, pp.36–43 (2004).
[18] Kawakami, T.: A Structured Overlay Network Scheme Based on Multiple Different Time Intervals, *Journal of Information Processing Systems*, Vol.16, No.6, pp.1447–1458 (2020).
[19] Aspnes, J. and Shah, G.: Skip Graphs, *ACM Transactions on Algorithms*, Vol.3, No.4, pp.1–37 (2007).
[20] 小西佑治ほか：単一ノードに複数キーを保持可能とするSkip Graph 拡張，情報処理学会論文誌，Vol.49, No.9, pp.3223–3233 (2008).
[21] Pugh, W.: Skip Lists: A Probabilistic Alternative to Balanced Trees, *Comm. ACM*, Vol.33, No.6, pp.668–676 (1990).
[22] 山下高生，栗田弘之，高田直樹：多様なデータサイズ分布を持つ Zipf 分布型処理要求に対する負荷分散とインメモリデータ・サイズ近似的最小化，情報処理学会論文誌，Vol.58, No.12, pp.1977–1992 (2017).
[23] Shao, X., Jibiki, M., Teranishi, Y. and Nishinaga, N.: An Efficient Load-Balancing Mechanism for Heterogeneous Range-Queriable Cloud Storage, *Future Generation Computer Systems*, Vol.78, pp.920–930 (2018).

久保 達也

2021 年福井大学工学部電気電子情報工学科卒業．現在，同大学大学院工学研究科博士前期課程に在学中．分散システムに関する研究に従事．

川上 朋也 （正会員）

2005 年近畿大学理工学部経営工学科卒業．2007 年大阪大学大学院情報科学研究科博士前期課程修了．同研究科特任研究員等を経て，2020 年より福井大学大学院工学研究科講師，現在に至る．情報科学博士（2013 年 3 月，大阪大学）．分散システム，ルールベースシステム，ストリームデータ処理に関する研究に従事．IEEE 会員．