ICNにおけるDataの通過情報を考慮したキャッシュへのルーティング手法の検討

佐藤 和也[1] 神本 崇史[1] 重野 寛[1]

概要：Information-centric Networking (ICN) において，ユーザはルータの持つキャッシュからコンテンツを取得することができる．キャッシュを取得する手法に，ルータ間のキャッシュの重複を減らすアルゴリズムを用い，コンテンツの通過情報をもとにキャッシュを発見するものがある．しかし，この手法は一部のルータがキャッシュの取得を行わない，キャッシュの取得に ISP からの情報を必要とするといった問題点がある．本稿では，Cache-aware Routing by Comparing Distance for Contents for ICN (CCD) を提案する．本提案手法はサーバまでのホップ数とキャッシュまでのホップ数を通過情報として記録し，比較を行い転送方向を決定することですべてのルータが ISP の情報に依存することなく最近傍コンテンツを取得する．また，通過情報のキャッシュまでのホップ数を更新するためにコンテンツ要求パケットを拡張する．エントリを更新するための新たなフィールドを加え，場合に応じて挙動を変える．有効性を確認するため CCD をシミュレータ上に実装しキャッシュヒット率と平均コンテンツ取得時間について評価する．

Cache-aware Routing Scheme for ICN Considering Passing Information of Data

KAZUYA SATO[1] TAKASHI KAMIMOTO[1] HIROSHI SHIGENO[1]

1. はじめに

Information-centric Networking (ICN) [1] においてユーザは IP アドレスではなくコンテンツ名を用いてコンテンツを取得する．現行の Internet Protocol (IP) はノードそれぞれに割り当てられてた IP アドレスに基づいてパケットのやり取りを行う．一方で，ICN はコンテンツ名に基づいてコンテンツの要求と転送を行う．ユーザはコンテンツ要求パケット Interest を生成し，サーバへ転送する．サーバが Interest を受信すると，コンテンツパケット Data が生成され，Interest のたどった経路を戻ってユーザまで転送される．また，ルータはコンテンツをキャッシュすることができ，該当する Interest を受け取るとサーバに代わってユーザに返信する．ユーザがキャッシュを取得することは，サーバの負荷軽減やコンテンツ取得時間の短縮といった利点をもたらす．

キャッシュを取得するための様々な手法が提案されている．Chunk Caching Location and Searching Scheme (CLS) [2] は，キャッシュへのルーティングとキャッシュするコンテンツを決定するキャッシングアルゴリズムに着目した手法である．CLS は，各ルータ間でキャッシュの重複を減らすキャッシングアルゴリズムを用い，コンテンツキャッシュ時に生成される通過情報 Breadcrumbs エントリによって Interest をキャッシュへルーティングする．しかし，CLS では，サーバに近い一部のルータが必ずサーバからコンテンツを取得する．結果，キャッシュヒット率が低下し，さらにサーバよりキャッシュが近い場合もサーバからコンテンツを取得する場合もある．また，CLS は転送方向の決定の際に ISP からの情報を必要とする．しかし，サーバの数が増加すれば ISP の負荷の増大につながるため，キャッシュへのルーティングを行う手法は ISP に頼るべきではない．

本稿では Cache-aware Routing by Comparing Distance for Contents for ICN (CCD) を提案する．CCD は Breadcrumbs エントリにサーバ及びキャッシュまでのホップ数

[1] 慶應義塾大学大学院理工学研究科
Graduate School of Science and Technology, Keio University

を加えて記録し，比較を行って転送方向を決定することでISP からの情報に依存せず，すべてのルータが最近傍コンテンツを取得可能にする手法である．エントリを持つルータはキャッシュまでのホップ数とサーバまでのホップ数を比較し，近い方へ Interest を転送する．CCD は，各コンテンツがサーバーリーフルータ間で最大でも 1 つのルータでしかキャッシュされないキャッシングアルゴリズムを用いる．Breadcrumbs エントリのキャッシュまでのホップ数を更新するため，Interest を拡張する．Interest には Data を返信する必要性を判断するフラグと Breadcrumbs エントリを更新する値のフィールドを加える．さらに，キャッシュの移動に応じて Interest の挙動を変えることで Breadcrumbs エントリの更新，追加を行う．

以降では，2 章で ICN におけるコンテンツの要求と転送の流れとキャッシュについての関連研究を述べ，3 章で提案手法を説明し，4 章でシミュレーションによる評価，5 章で結論を述べる．

2. 関連研究

本章ではまずはじめに ICN におけるコンテンツの要求と転送について説明し，その後，キャッシュを取得するうえで重要な役割を担うルーティング手法とキャッシングアルゴリズムに関する関連研究，そして両者を合わせることによるキャッシュへのルーティング手法について述べる．

2.1 Information-centric Networking の概要

ICN はコンテンツ名によってコンテンツの要求を行う．コンテンツにのみ着目し取得する場所を限定しないコンテンツ指向型ネットワーク [3] の 1 種であり，CCN [4] や NDN [5] といったプロジェクトがある．

ICN は Interest パケットと Data パケットによってコンテンツの要求と転送を行う．ユーザによって生成されるコンテンツ要求パケット Interest はコンテンツ名を含み，ルータの持つ経路表に従ってサーバまで転送される．Interest を受け取ったサーバはコンテンツパケット Data を送り返す．その後，Data は Interest のたどった経路を戻ってユーザに到達する．その際経路上のルータは，通過した Data を Content Store (CS) に保存することができる．受信した Interest に対応するコンテンツをキャッシュしていた場合，サーバに代わってユーザに Data を送り返す．

キャッシュを取得することによって，サーバが送り返す Data 数が減少し，負荷が軽減される．また，ユーザのコンテンツ取得時間が短縮されるという利点がある．負荷分散に関する研究は多数の分野で取り組まれている [6]．

2.2 キャッシュへのルーティング手法

キャッシュを効率的に取得するためには各ルータの持つキャッシュの情報が必要である．ルータ間でキャッシュの情報交換を行わず，一定範囲への Interest のフラッディングによってキャッシュを取得する手法 [7] もあるが，ルータやリクエスト数が増加に伴って Interest の複製と処理にかかるコストが増大する．

ルータ間でキャッシュの情報を共有する方法として，メッセージパケットなどを用いて明示的に行うものとメッセージパケットを用いずに間接的に行うものがある．Potential Based Routing (PBR) [8] は，キャッシュしているコンテンツをメッセージパケットを用いて明示的に情報を通知するが，新たなメッセージパケットを定義する必要があり，ICN に加える変更が大きい．

一方，Breadcrumbs [9] はコンテンツが通過した際に通過方向などを示す Breadcrumbs エントリをルータに生成し，後続の同コンテンツへの Interest を通過方向へ転送しキャッシュへ誘導する．新たに定義が必要なものはエントリのみであり ICN に加える変更は小さい．Breadcrumbs エントリは以下にある要素から構成されている．

- コンテンツ ID: コンテンツ名などのコンテンツを一意的に特定するもの
- *in*: Data 転送元インタフェース
- *out*: Data の転送先インタフェース
- t_{req}: Interest 通過時刻
- t_{out}: Data 通過時刻

Data の通過方向 (*out*) へ Interest を転送することでキャッシュを取得できる可能性がある．しかし，Breadcrumbs エントリの示す方向にあるルータにおけるキャッシュの有無を知ることができないため，t_{req} や t_{out} を用いて推測するが正確ではない．さらに，遠く離れたキャッシュでも取得するため，キャッシュを取得してもコンテンツ取得遅延時間が増大することがある．

2.3 キャッシングアルゴリズム

キャッシングアルゴリズムもキャッシュを取得するうえで重要な役割を担う．ネットワーク内に存在するコンテンツに対してルータがキャッシュできるコンテンツ数は限られている．そのため，各ルータのキャッシュの重複を減らしできる限り多くの種類のコンテンツをキャッシュすることでネットワーク内の CS 空間を活かすことができる．

キャッシュするコンテンツまたは破棄するコンテンツを決定するアルゴリズムは，協調型キャッシングと分散型キャッシングに分類される．協調型キャッシングはルータ間でキャッシュの情報を交換することによってキャッシュを決定する．一方，分散型キャッシングは，通過するコンテンツから得られる情報だけをもとにしてキャッシュするコンテンツを決定し，また，ルータは所持しているキャッシュを周囲に広告しない．LRU (Least Recently Used) や LFU (Least Frequently Used)，確率に基づいてコンテンツのキャッシュを決定する Probcache [10] などが分類され

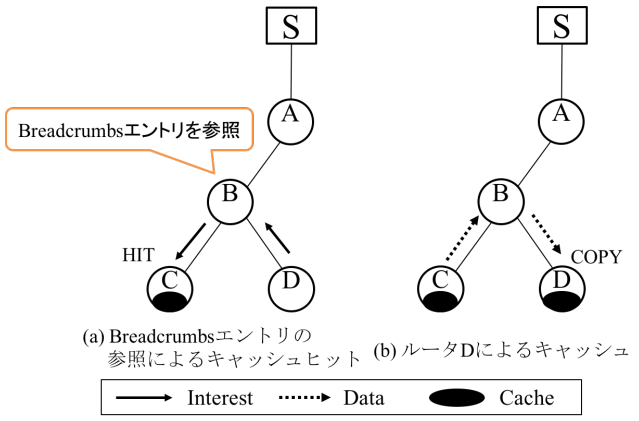

図 1　CLS におけるキャッシング

る．協調型キャッシングはルータ間のキャッシュの重複を避けられるが，コンテンツ数の増加に伴って情報交換のシグナリングコストが増加するため，スケーラビリティの観点で分散型キャッシングが優れている [11]．

Leave Copy Down (LCD) と Move Copy Down (MCD) は分散型キャッシングの 1 種であるが，ルータ間のキャッシュの重複を減らすことができる．LCD と MCD は，Data パケットが持つキャッシュ済みであるか否かを示すフラグによってキャッシュを決定する．LCD [12] は，キャッシュヒットが発生するとその 1 つ下のルータにキャッシュがコピーされ，それ以降コンテンツが通過したルータはこのコンテンツをキャッシュしない．MCD [12] は，キャッシュヒット後キャッシュがユーザ方向に 1 ホップ分移動し，それ以降のルータはコンテンツをキャッシュしない．

2.4　CLS

Chunk Caching Location and Searching Scheme (CLS) [2] は，キャッシングアルゴリズム MCD と Breadcrumbs を組み合わせることで Breadcrumbs の問題点に対処した手法である．

CLS ではキャッシュヒットが発生するとそのキャッシュはユーザ方向に 1 ホップ移動し，キャッシュしたルータは Breadcrumbs エントリを生成する．CS が溢れるとキャッシュを 1 つ選択し，サーバ方向にあるルータに移動させる．このとき Breadcrumbs エントリを削除する．したがって，Breadcrumbs エントリが示すコンテンツの通過方向にあるルータはいずれかが必ずキャッシュを持つ．

また，CLS において同じコンテンツはサーバ－リーフルータ間で最大で 1 つのルータでしかキャッシュされない．図 1 に Breadcrumbs エントリによるルーティングでキャッシュを取得した場合の動作をする．ルータ B が Breadcrumbs エントリを参照することでルータ C でキャッシュを取得した時，ルータ B がキャッシュするとサーバ S －ルータ C 間で複数のキャッシュが存在してしまうためルータ D がキャッシュを行う．

さらに，サーバまでのホップ数を Breadcrumbs エントリに加えて転送条件として利用することで，コンテンツ取得時間の増大を抑制している．CLS における Breadcrumbs エントリは，

- コンテンツ ID: コンテンツ名などのコンテンツを一意的に特定するもの
- *in*: Data の転送元（サーバ方向の）インタフェース
- *out*: Data の転送先インタフェース
- H_{src}: サーバまでのホップ数

によって構成され，以下の条件で転送方向を決定する．

$$\begin{cases} \text{サーバ方向へ転送} & H_{src} < T \\ \text{キャッシュ方向へ転送} & H_{src} \geq T \end{cases} \quad (1)$$

T はサーバ－リーフルータ間のホップ数の 1/2 から定められる値である．

しかし CLS は，$H_{src} < T$ を満たすノードが Breadcrumbs エントリを参照したキャッシュへのフォワーディングを行わない．結果，キャッシュヒット率が低下し，さらにキャッシュが近い場合でもサーバからコンテンツを取得する．次に，閾値 T を定めるためにサーバ－リーフルータ間のホップ数を得る必要があり，ISP の情報に依存している．各ルータは Interest が到達するたびに ISP から閾値の情報を得るため，ISP のサービスエリア内にサーバが増加すれば ISP にかかる負荷が増大する．

3.　CCD

本稿では，すべてのルータが ISP に依らずに最近傍コンテンツを取得するルーティング手法 Cache-aware Routing by Comparing Distance for Contents for ICN　(CCD) を提案する．

3.1　概要

CCD は Breadcrumbs エントリにサーバまでのホップ数とキャッシュまでのホップ数を記録し，比較によって転送方向を決定する．すべてのルータが自身の持つ Breadcrumbs エントリから最近傍コンテンツを判断することができ，転送方向を決定する際に ISP からの情報に依存しない．また，キャッシュの重複を減らすためキャッシングアルゴリズムは CLS と同様のものを用いる．Interest によってキャッシュが移動するため，拡張した Interest を用いて Breadcrumbs エントリを更新する．Interest パケットに新たなフィールドを加え，ルータが Interest をどのようにフォワーディングするかによって挙動を変える．

3.2　Breadcrumbs エントリの拡張

CCD は，CLS における Breadcrumbs エントリの要素に加えて，キャッシュまでのホップ数を記録する．したがって，CCD における Breadcrumbs エントリの構成は以下の

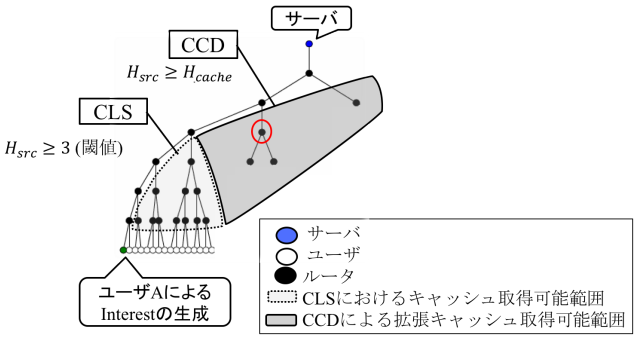

図 2　CLS と CCD のキャッシュ取得可能ルータの相違

通りである．

- コンテンツ ID: コンテンツ名などのコンテンツを一意的に特定するもの
- in: Data の転送元（サーバ方向の）インタフェース
- H_{src}: サーバまでのホップ数
- out: Data の転送先インタフェース
- H_{cache}: out 方向の最近接キャッシュまでのホップ数

各ルータは H_{cache} と H_{src} を比較し次の条件でサーバまたはキャッシュへの転送を決定する．

$$\begin{cases} \text{サーバ方向へ転送} & H_{src} < H_{cache} \\ \text{キャッシュ方向へ転送} & H_{src} \geq H_{cache} \end{cases} \quad (2)$$

図 2 は深さ 7（サーバ－リーフルータ間のホップ数が 6）のトポロジにおいて，両手法が Breadcrumbs エントリを参照することによるキャッシュの取得が可能なノードの違いを表している．図中のユーザ A があるコンテンツをリクエストしたとき，CLS における閾値は 3 であるのでサーバから 3 ホップ以上離れたルータはキャッシュへのフォワーディングをする．したがって，点線で囲まれたルータからキャッシュを取得できるが，2 ホップ以内のルータはサーバからコンテンツの取得を行う．一方 CCD は H_{src} と H_{cache} の比較によって実線で囲まれた範囲のルータからキャッシュを取得できる．そのため，丸で囲まれたルータがキャッシュを保持していた場合，CLS はこれを取得することができないが CCD は取得が可能である．

以下では，Breadcrumbs エントリを (コンテンツ名, in, H_{src}, out, H_{cache}) のように表すものとする．なお，out が Null であるとき，そのルータ自身がキャッシュを持っている状態であることを示す．

3.3　Interest の拡張

CCD は，キャッシュの移動を上流のルータに通知して Breadcrumbs エントリを更新するため Interest を用いる．通常の ICN において，ルータは受信した Interest のコンテンツをキャッシュしていた場合，キャッシュを返し Interest をそれ以上転送しない．CCD はキャッシュを発見した場合もサーバ方向へ Interest のフォワーディングを行い，キャッシュの移動を通知する．

表 1　Interest に対する動作

$returnedflag$	$false$	Data の返信が必要
	$true$	Data の返信不要
キャッシュヒット	あり	$returnedflag$ を $true$ にセット
Interest の到達方向	サーバ方向	エントリの更新をしない
	ユーザ方向	エントリを更新する
Breadcrumbs エントリ	あり	Interest の V をインクリメント

また，Interest パケットに Data が返信済みであるかを示すフラグ $returnedflag$ とエントリの更新値 V を加える．ルータは Interest を受け取ると $returnedflag$ から Data を返信する必要があるかを判断する．続けて，Interest の到達した方向から Breadcrumbs エントリを更新する必要があるか判断する．ユーザ方向から来たものである場合にはキャッシュが移動するので V を用いて Breadcrumbs エントリの更新を行う．Interest がサーバ方向から来たものであった場合にはキャッシュは移動しないので（図 1 参照），Breadcrumbs エントリの更新を行わない．

さらに，サーバ方向のフォワーディング（シンプルフォワーディングと呼ぶ）の途中でキャッシュヒットがあった場合と Breadcrumbs エントリの参照によってフォワーディングする（Breadcrumbs フォワーディングと呼ぶ）場合で Interest の挙動を変更する．

以下では，Interest $(T/F, j)$ は $returnedflag = true/false$，更新値 $V = j$ の Interest を表すものとする．

returnedflag

$returnedflag$ は Data が返信済みであるかを示し，$false$ は Data がまだ返信されていないので返信する必要があることを意味し，$true$ はすでに Data を返信済みでありデータを返す必要がないことを意味する．ユーザは Interest を生成したとき，フラグを $false$ にセットする．その後，キャッシュヒットがあったルータと Breadcrumbs エントリによって Interest をキャッシュ方向へフォワーディングするルータによって $true$ にセットされる．$returnedflag$ が $true$ の Interest を受け取ったノードは Data を返信しない．

更新値 *V*

V は，キャッシュヒットによって移動するキャッシュまでのホップ数を表す．ルータは，Breadcrumbs エントリの H_{cache} に V を代入することでエントリを更新できる．ユーザが Interest を生成したとき V には 1 が代入され，その後通過したルータで目的のコンテンツの Breadcrumbs エントリを発見するたびにインクリメントされる．ただし，前述した通り，Breadcrumbs エントリの更新を行うのはユーザ方向からの Interest に対してのみである．

ここまでの Interest に対するルータの動作をまとめると表 1 のようになる．

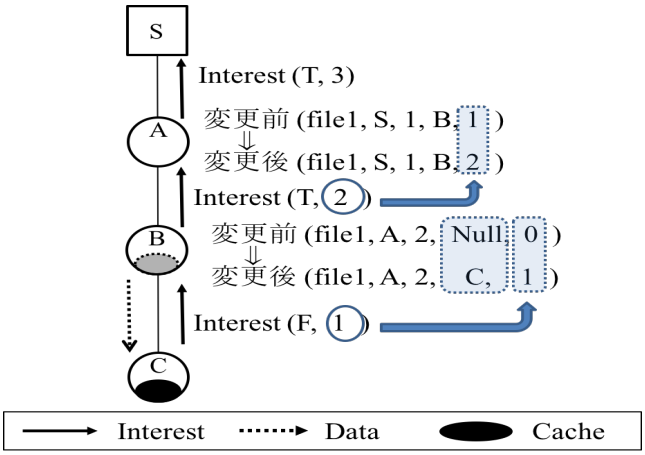

図 3　シンプルフォワーディングにおけるエントリの更新手順

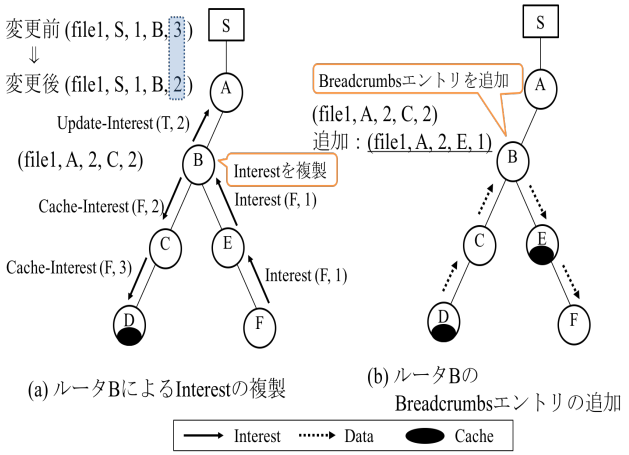

図 4　Breadcrumbs フォワーディングにおけるエントリの更新手順

3.4　シンプルフォワーディングにおけるエントリの更新

サーバ方向へのフォワーディングのみを行うシンプルフォワーディングにおいてキャッシュヒットした場合の Interest の動作について説明する．キャッシュヒットしたルータは Interest の転送を止めずにサーバ方向へ転送を続ける．キャッシュヒットしたルータはキャッシュを返したあと，次の動作を行う．

(1)　Interest の *returnedflag* を *true* にする．

(2)　Breadcrumbs エントリの Data の通過方向をユーザ方向に書き換える．

(3)　Breadcrumbs エントリの H_{cache} に Interest の V を代入して更新を行う．

(4)　Interest の更新値 V をインクリメントし，サーバ方向へ転送を続ける．

以上の動作によってキャッシュヒットしたルータは Breadcrumbs を更新することができる．そして，さらにサーバ方向にあるルータは (3) 及び (4) の動作を行いサーバへ転送する．サーバは Interest の *returnedflag* を参照して *true* であるため Data を返す必要がないと判断する．

具体例を図 3 に示す．キャッシュヒットしたルータ B はキャッシュを返したあと (1) から (4) の動作を行い，自身の Breadcrumbs エントリを更新しサーバ方向へ転送を続ける．ルータ A はユーザ方向からの Interest に対して (3), (4) の動作に従いエントリを更新する．最後に，Interest を受け取ったサーバは *returnedflag* が *true* であることを確認し Data が返信済みであると判断し返信を行わない．

3.5　Breadcrumbs フォワーディングにおけるエントリの更新

次に，Breadcrumbs エントリを参照したフォワーディングを行う場合について記す．Breadcrumbs エントリの参照によってサーバからコンテンツを取得する場合とキャッシュからコンテンツを取得する場合がある．

キャッシュを取得する場合，最初に Breadcrums フォワーディングを行うルータは Interest を複製し，サーバ方向とキャッシュ方向へ Interest をフォワーディングする．このとき，キャッシュ方向へ向かう Interest はキャッシュを取得するための Interest であり，Cache-Interest と呼ぶものとする．一方，サーバ方向へ向かう Interest は Breadcrumbs エントリを更新するための Interest であり，Update-Interest と呼ぶものとする．Cache-Interest はキャッシュの取得を行うので，*returnedflag* は *false* にセットされている．キャッシュ方向にある各ルータは Cache-Interest がサーバ方向から到達するため Breadcrumbs エントリの更新を行わない．一方，Update-Interest はコンテンツの返信を必要としないため *returnedflag* は *true* にセットされている．サーバ方向にある各ルータは Update-Interest がユーザ方向から到達するため Breadcrumbs エントリの更新を行う．また，最初に Breadcrumbs フォワーディングをしたルータは Breadcrumbs エントリを追加する必要があるが，Interest のフォワーディング時ではなく Data 通過時にエントリの追加を行う．したがって，キャッシュの取得を行う場合，最初に Breadcrumbs フォワーディングを行うルータの動作をまとめると次の通りになる．

(i)　Interest の更新値 V をインクリメントし複製する．(Cache-Interest と Update-Interest)

(ii)　Cache-Interest の *returnedflag* を *false* にしたままキャッシュ方向へ転送する．

(iii)　Update-Interest の *returnedflag* を *true* にし，サーバ方向へ転送する．

(iv)　Data 通過時に Breadcrumbs エントリを追加する．

また，Breadcrumbs フォワーディングによってサーバからコンテンツを取得する場合，Interest を複製せずにサーバ方向にのみフォワーディングする．サーバ方向の各ルータはユーザ方向からの Interst に対して Breadcrumbs エントリを更新するが，最初に Breadcrumbs フォワーディングを

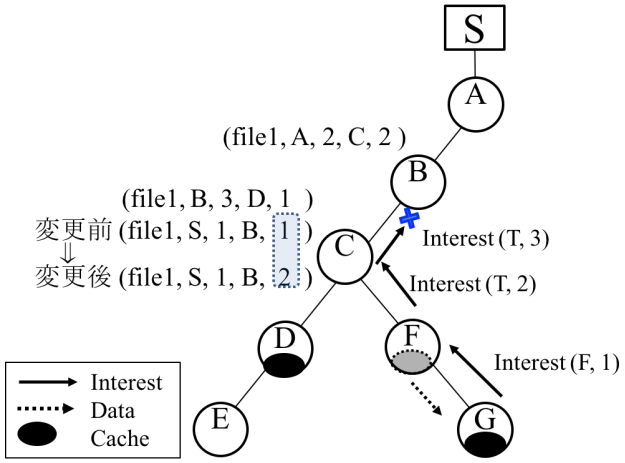

図 5　エントリの更新のために Interest を転送しない場合

行ったルータは，キャッシュを取得する場合と同様に Data 通過時に Breadcrumbs エントリを追加する．

Breadcumbs フォワーディングによってキャッシュの取得を行う場合について図 4 に具体例を示す．(a) は Interest 転送時の動作を表し，(b) は Data 通過時の動作を表している．まず，(a) においてルータ B は Breadcrumbs フォワーディングによってキャッシュからコンテンツの取得を行う判断をする．1. から 4. の動作を行い，Update-Interest をサーバ方向へ，Cache-Interest をキャッシュ方向へフォワーディングする．ルータ A はユーザ方向からの Interest によって Breadcrumbs エントリを更新し，ルータ C，D はサーバ方向からの Interest に対してエントリを更新しない．そして，(b) に示す Data 通過時にルータ B は Breadcrumbs エントリを追加する．

3.6　エントリの更新を行わない場合

上流ルータの Breadcrumbs エントリを更新するための Interest をサーバ方向へ転送してはいけない場合もある．CCD において，ルータはユーザ方向のインタフェース 1 つに対して所持する Breadcrumbs エントリは 1 つだけであるため，実際にはある 1 つのインタフェース方向に複数のキャッシュがある場合でもルータはそれぞれのキャッシュまでのホップ数を知ることができない．したがって H_{cache} はその方向にある最近傍のキャッシュまでのホップ数を示している必要がある．しかし，3.4，3.5 で示したように，サーバ方向に必ず Interest をフォワーディングして Breadcrumbs エントリの更新を行うと，最近傍キャッシュを示さなくなることがある．

Breadcrumbs エントリを複数持ち，更新される方のエントリの H_{cache} が他の H_{cache} より小さい値でない場合，サーバ方向への Interest の転送を止め，サーバ方向ルータの Breadcrumbs エントリが更新されるのを防ぐ．あるルータから見て複数方向にキャッシュがあり両者の距離が

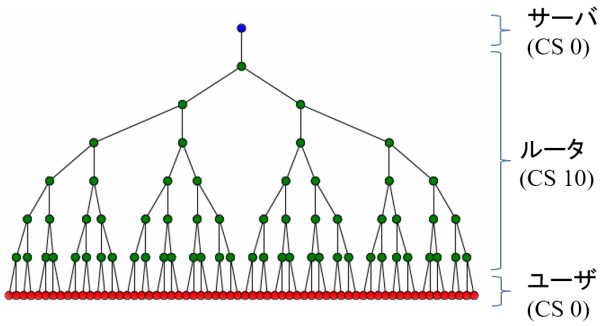

図 6　シミュレーションで用いたトポロジ

異なる場合，その 1 つ上流にあるルータ（親ルータと呼ぶ）が持つ Breadcrumbs エントリは近い方のキャッシュを示している．このとき遠い方のキャッシュがユーザからのリクエストによってさらに遠ざかることになっても親ルータにとっての最近傍キャッシュまでのホップ数は変わらない．しかし，この親ルータがユーザ方向からの Interest を受け取り，Breadcrumbs エントリを更新すると親ルータの Breadcrumbs エントリが示すキャッシュは，さらに遠ざかったキャッシュを示すことになる．複数方向にあるキャッシュが等距離にあった場合でも同様である．

上流ルータの持つ Breadcrumbs エントリが更新されるのを防ぐ例を図 5 に示す．ルータ C から見てルータ D とルータ F の複数方向にキャッシュがあり，ルータ E のキャッシュがユーザ方向に遠ざかる．ルータ C は Breadcrumbs エントリを更新するが，複数方向の Breadcrumbs エントリを持ち，更新された方の H_{cache} は他方のエントリの H_{cache} と等しい．したがって，ルータ C はサーバ方向への Interest のフォワーディングを止め，サーバ方向ルータの Breadcrumbs エントリが更新されるのを防ぐ．もしフォワーディングを続ければルータ B の持つ Breadcrumbs エントリは (file1, A, 2, C, 3) に更新され，最近傍キャッシュまでのホップ数を示さない．ルータ B の最近傍キャッシュはサーバまでと同じ 2 ホップであるのでキャッシュの取得を行うべきであるが，Breadcrumbs エントリが (file1, A, 2, C, 3) である場合サーバからコンテンツの取得を行ってしまう．結果，本来取得すべきキャッシュが取得できなくなってしまう．

4.　評価

本章では，提案手法を評価するために実装したシミュレーションと，その評価結果，考察を述べる．

4.1　シミュレーション環境

CCD を実装するにあたって，ネットワークシミュレータとして ns-3 [14]，ICN モジュールとして ndnSIM1.0 [15] を利用した．使用したトポロジについては図 6 に示した深さ 7 のバイナリツリーである．またシミュレーションのパ

表 2　シミュレーションパラメータ

シミュレータ	ns-3.21
ICN モジュール	ndnSIM1.0
シミュレーション時間	200.0 秒
リンク遅延	2.0 ミリ秒
(サーバールータ間)	(10.0 ミリ秒)
トポロジ	二分木 (深さ 7)
リクエスト頻度	10 リクエスト/秒
コンテンツ数	1000
コンテンツの人気度分布	Zipf の法則 ($\alpha = 0.7$) [13]
ルータ CS サイズ	10

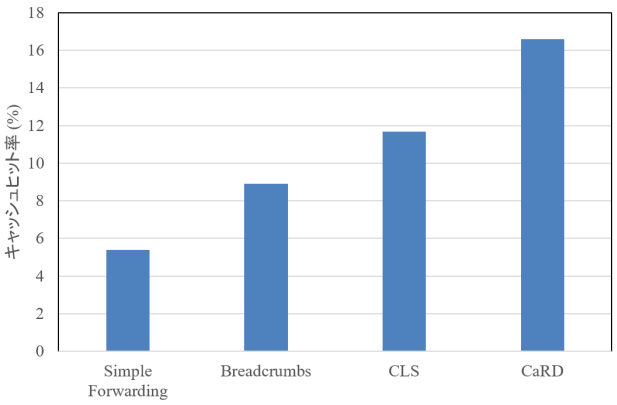

図 7　コンテンツ数 1000 のときのキャッシュヒット率

ラメータを表 2 に示す．

4.2　評価項目

キャッシュヒット率と平均コンテンツ取得時間について評価した．キャッシュヒット率は，ユーザが受信した Data 数に対するキャッシュから取得された Data 数の割合と定義し，コンテンツ数を 1000 から 6000 まで変化させて数値を測った．平均コンテンツ取得時間については，ユーザがリクエストを出してから取得するまでの時間と定義した．

評価の対象手法として，サーバ方向へのフォワーディングのみを行うシンプルフォワーディング，Breadcrumbs，CLS を用いた．

4.3　キャッシュヒット率

図 7 にコンテンツ数が 1000 の場合での各手法のキャッシュヒット率を示した．キャッシュヒット率の向上はサーバにかかる負荷の削減を意味する．キャッシュを取得できなかった Interest に対してサーバはコンテンツを返さなくてはならない．キャッシュヒット率が向上するとサーバが処理しなければならない Interest 数，すなわち負荷が削減されることになる．

Breadcrumbs 手法は，Breadcrumbs エントリによるフォワーディングのためにシンプルフォワーディングよりも高いキャッシュヒット率を示したが，エントリが転送方向として示すルータがキャッシュの破棄を行っている場合があるために CLS に比べて低いヒット率を示している．

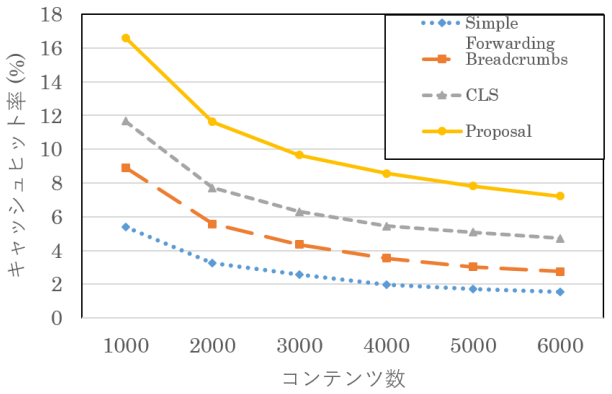

図 8　コンテンツ数を変化させた場合のキャッシュヒット率の変化

CCD は CLS よりもさらに高いキャッシュヒット率を示した．CLS では Breadcrumbs エントリによるキャッシュへのフォワーディングを行っていないルータが，CCD ではフォワーディングを行っているためだと考えられる．

また，コンテンツ数を 1000 から 6000 まで変化させたときのキャッシュヒット率を図 8 に示した．コンテンツ数が増加すると，コンテンツ数に対するトポロジ全体でキャッシュできるコンテンツの種類数の割合が減少するためキャッシュヒット率はいずれの手法についても低下する．CCD についてもヒット率が低下し，コンテンツ数 1000 の時は CLS と比較して約 5%向上していたのに対し，2000 から 6000 の間ではその差は約 3%になった．しかし，コンテンツ数が増加すれば，コンテンツを管理するサーバにかかる負荷は大きくなる．したがって，キャッシュヒット率の向上がサーバにかかる負荷の軽減に対する意味は大きくなる．コンテンツ数が増加してもサーバの負荷を軽減できている CCD は有効性があると言える．

4.4　平均コンテンツ取得時間

図 9 に各手法における平均コンテンツ取得時間を示した．Breadcrumbs はシンプルフォワーディングに比べて大きい取得時間を示した．Breadcrumbs は最近傍コンテンツを判断することができず，キャッシュが遠いルータにあった場合でもキャッシュを取得しているためだと考えられる．

一方，CLS はサーバまでのホップ数を Breadcrumbs エントリに加えて転送条件として用いたことでコンテンツ取得時間を削減している．

そして，CCD は CLS とほぼ同等の取得時間を示した．取得時間に差が表れなかったのはキャッシュヒット率の差がおよそ 5%程度とユーザの総リクエスト数に対して大きな値ではなかったために影響を及ぼさなかったのだと考え

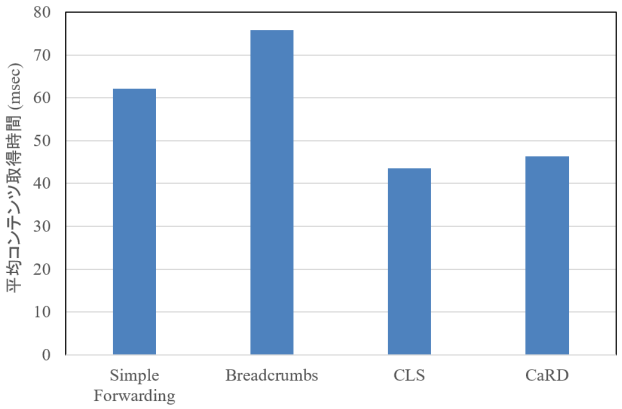

図 9 平均コンテンツ取得時間

られる．しかし，CCD は CLS と異なり ISP に依存しない手法である．CLS において各ルータは Interest が到達するたびに ISP からサーバーリーフルータ間のホップ数を得る必要がある．実際のネットワーク環境においては，サーバの増加によって ISP からの応答が遅くなった場合，取得時間が増加する可能性がある．したがって，ISP に依らない CCD は有効性があると考えられる．

5. おわりに

ICN におけるキャッシュへのルーティング手法として，CLS はルーティング手法 Breadcrumbs とキャッシングアルゴリズム MCD を組み合わせた手法であるが，一部のルータが Breadcrumbs エントリを参照したキャッシュへのフォワーディングを行わない，フォワーディング条件を ISP からの情報に依存しているといった問題点があった．

そこで本稿は Cache-aware Routing by Comparing Distance for Contents for ICN (CCD) を提案した．CCD は Breadcrumbs エントリにサーバまでのホップ数とキャッシュまでのホップ数を記録し，比較することによってすべてのルータが ISP からのトポロジ情報に依存することなく最近傍コンテンツへ Interest をフォワーディングする．キャッシュまでのホップ数を更新するため，Interest に新たなフィールドを加えて拡張し，キャッシュの移動に応じて挙動を変えた．

CCD をシミュレータ上に実装し評価した結果，キャッシュヒット率は既存手法に比べて約 5%以上の改善を示し，コンテンツ数が増加しても 3%程度維持されたことから有効性が示された．また，コンテンツ取得時間については ISP に依存することなく CLS とほぼ同等の値を示した．CLS は実際にはサーバの増加による ISP の応答遅延の増加によってさらにコンテンツ取得時間が増大する可能性があり，提案手法 CCD は有効性があると言える．

謝辞 本研究は JSPS 科研費 16H02811 の助成を受けたものです．

参考文献

[1] Ahlgren, B., Dannewitz, C., Imbrenda, C., Kutscher, D. and Ohlman, B.: A survey of information-centric networking, *IEEE Communications Magazine*, Vol. 50, No. 7, pp. 26–36 (2012).
[2] Li, Y., Lin, T., Tang, H. and Sun, P.: A chunk caching location and searching scheme in Content Centric Networking, *2012 IEEE International Conference on Communications (ICC)*, pp. 2655–2659 (2012).
[3] 山本　幹：コンテンツオリエンテッドネットワーク，電子情報通信学会誌， Vol. 95, No. 4 (2012).
[4] Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H. and Braynard, R. L.: Networking Named Content, *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '09, pp. 1–12 (2009).
[5] Zhang, L., Afanasyev, A., Burke, J., Jacobson, V., claffy, k., Crowley, P., Papadopoulos, C., Wang, L. and Zhang, B.: Named Data Networking, *SIGCOMM Comput. Commun. Rev.*, Vol. 44, No. 3, pp. 66–73 (2014).
[6] Mori, K., Hatakeyama, S. and Shigeno, H.: DCLA: Distributed Chunk Loss Avoidance Method for Cooperative Mobile Live Streaming, *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*, pp. 837–843 (2015).
[7] Lee, M., Cho, K., Park, K., Kwon, T. and Choi, Y.: SCAN: Scalable Content Routing for Content-Aware Networking, 2011 IEEE International Conference on Communications (ICC)，pp. 1–5 (2011).
[8] Eum, S., Nakauchi, K., Murata, M., Shoji, Y. and Nishinaga, N.: Potential based routing as a secondary best-effort routing for Information Centric Networking (ICN), *Computer Networks*, Vol. 57, No. 16, pp. 3154 – 3164 (2013).
[9] Rosensweig, E. and Kurose, J.: Breadcrumbs: Efficient, Best-Effort Content Location in Cache Networks, *IEEE 2009 INFOCOM*, pp. 2631–2635 (2009).
[10] Psaras, I., Chai, W. K. and Pavlou, G.: Probabilistic In-network Caching for Information-centric Networks, *Proceedings of the Second Edition of the ICN Workshop on Information-centric Networking*, ICN '12, New York, USA, ACM, pp. 55–60 (2012).
[11] Mori, K., Kamimoto, T. and Shigeno, H.: Push-Based Traffic-Aware Cache Management in Named Data Networking, *2015 18th International Conference on Network-Based Information Systems (NBiS)*, pp. 309–316 (2015).
[12] Laoutaris, N., Che, H. and Stavrakakis, I.: The {LCD} interconnection of {LRU} caches and its analysis, *Performance Evaluation*, Vol. 63, No. 7, pp. 609 – 634 (2006).
[13] Breslau, L., Cao, P., Fan, L., Phillips, G. and Shenker, S.: Web caching and Zipf-like distributions: evidence and implications, *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, Vol. 1, pp. 126–134 vol.1 (1999).
[14] ns 3: https://www.nsnam.org/, Retr. May 2016.
[15] ndnSIM1.0: http://ndnsim.net/1.0/, Retr. May 2016.