NAIST-IS-MT1651044

修士論文

情報指向ネットワークにおける パケット転送テーブルを利用した キャッシュ利用効率向上手法の提案

木村 一統

2018 年 3 月 15 日

奈良先端科学技術大学院大学
情報科学研究科

本論文は奈良先端科学技術大学院大学情報科学研究科に
修士 (工学) 授与の要件として提出した修士論文である。

木村 一統

審査委員：

藤川 和利 教授	（主指導教員）
門林 雄基 教授	（副指導教員）
新井 イスマイル 准教授	（副指導教員）

情報指向ネットワークにおける パケット転送テーブルを利用した キャッシュ利用効率向上手法の提案*

木村 一統

内容梗概

インターネットの利用目的は、初期ではホスト間でのデータ送受であったが、近年では音楽、画像、動画などのコンテンツ配信システムへと変化してきている。また、YouTube や動画配信サービスの普及に伴って、配信されるコンテンツの容量も年々増加傾向にある。こういったインターネットの利用形態の変化に伴い、よりコンテンツ配信に適したネットワーク技術として ICN(Information Centric Networking) の研究が盛んに行われている。ICN ではネットワーク上に存在するノードがコンテンツのキャッシュを持つことによって、サーバの位置に依存しないネットワークを構築する。これによりレスポンス時間の短縮や負荷分散などのメリットが生じる。

本研究ではネットワーク上に存在するキャッシュをホップ数に従って効率良く利用し、キャッシュヒット率を向上させることを目的とする。先行研究で用いられた手法に加え、キャッシュの存在を示すテーブルの改良、新たなパケットの定義を行った手法を提案する。さらにこれらを実装しシミュレーション実験を行い評価を行った。その結果、先行研究と比較してキャッシュヒット率、レスポンス時間をそれぞれ 11.5%、22.3%向上させることが可能となった。

キーワード

情報指向ネットワーク, インターネット基盤技術, コンピュータネットワーク

*奈良先端科学技術大学院大学 情報科学研究科 修士論文, NAIST-IS-MT1651044, 2018 年 3 月 15 日.

Improved cache hit ratio using packet forwarding table in Information Centric Networking*

Itto Kimura

Abstract

The purpose of using the Internet was initially to send and receive data between hosts. In recent years, it has changed to a content distribution system such as music, images moving pictures and the like. Also, with the spread of YouTube and video distribution services, the volume of contents to be delivered is increasing every year. Along with this change in the usage form of the Internet, ICN (Information Centric Networking) is being researched as a network technology suitable for the change. In ICN, the network do not depend on server location by the nodes which have caches. It can reduce the response time and distribute load. In this research, the purpose of improving cache hit ratio using distributed caches. We extended tables showing the existence of caches which be used in previous study and define a new packet. Furthermore, we implemented and simulate these in the simulator.

Keywords:

Information Centric Networking, Internet infrastructure technology, Computer Network

*Master's Thesis, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-MT1651044, March 15, 2018.

目 次

図 目 次

表 目 次

1. 序論

本章では本研究における解決すべき問題点を明確にし、本研究での目的を説明する。まず、現代から将来へのネットワークの利用形態の変化について説明する。次に、現在主流である IP ネットワークでの問題点を説明し、その上で情報指向ネットワークの有用性について述べる。次に、情報指向ネットワークでの課題について説明し、その解決策について議論した後、研究目的を明確化する。最後に情報指向ネットワークの課題と研究目的をまとめ、論文構成を示す。

1.1 ネットワーク利用形態の変化

設計当初、インターネットはホスト間でのデータの送受信に利用されることを想定していたホスト指向なものであった。しかし、近年では動画像、音楽などのコンテンツを配信する目的で利用されることが増えてきているためコンテンツ指向なものとなってきている。従来ではこの様なコンテンツ配信のためのサービスでは IP マルチキャスト配信を利用しているものが多かった [2]。しかし、ユーザの嗜好が多様化するに連れてオンデマンド配信の需要が高まっており [3][4]、この様な状況下では必ずしもマルチキャスト配信が適しているとは言えない。また、2017 年 9 月の Cisco Systems 社の発表 [5] によると全世界の IP トラフィック量は 2016 年には年間 1.2zettabytess であったものが、2021 年には年間 3.2zettabytes になると予測されている。Cisco Systems 社の 2016–2021 年のコンシューマインターネットトラフィックに関する予測を表 1 に示す。全世界のコンシューマインターネットトラフィックに占めるビデオトラフィックの割合は、2016 年には 71%であったが 2022 年までには 82%になる見込みであるとされている。また 2017 年 9 月に Ericsson 社が出したモバイルトラフィック分析のレポート [6] によると、モバイルトラフィックにおけるビデオトラフィックの年増加率は約 50%におよび、2023 年にはモバイルトラフィックの約 75%がビデオトラフィックになるとされている。さらに日本国内においてもブロードバンド契約者の総トラフィック量はダウンロードトラフィック・アップロードトラフィック共に年々増加しており [7]、インターネットトラフィックのうち 2020 年で 7 割、2030 年で 7.5 割程度がビデオトラフィッ

クになることが予想されている [8]。以上より、現在から将来に渡ってインターネットで送受信されるトラフィックの総量は年々増加していく見込みであり、中でも一般的に大容量とされるビデオトラフィックが占める割合は大きくなっていくことが分かる。

表 1 コンシューマインターネットトラフィック 2016–2021 年 (petabyte/月)

種別	2016 年	2017 年	2018 年	2019 年	2020 年	2021 年
インターネットビデオ [1]	42,029	57,116	75,109	98,182	125,853	159,161
Web, E メール, データ [2]	9,059	10,681	12,864	15,120	17,502	19,538
オンラインゲーム [3]	915	1,818	2,857	4,396	6,753	10,147
ファイル共有 [4]	6,628	6,810	6,717	6,554	6,388	6,595
総トラフィック量	58,630	76,426	97,547	124,252	156,496	195,440

1.2 IP ネットワーク

本節では 1.2.1 節で IP ネットワークとその問題を説明する。次の 1.2.2 節で IP ネットワークの問題を部分的に解決している Content Delivery Network (CDN) について説明する。

[1]短編インターネットビデオ（YouTube など）、長編インターネットビデオ（Hulu など）、ライブインターネットビデオ、テレビ視聴のインターネットビデオ（Roku 経由の Netflix など）、オンラインビデオ購入とレンタル、Web カメラ視聴、Web ベースのビデオ監視（P2P ビデオファイルダウンロードを除く）

[2]Web、E メール、インスタントメッセージング、およびその他のデータトラフィック（ファイル共有を除く）

[3]カジュアルオンラインゲーム、ネットワークに接続されたコンソールでのゲーム、マルチプレイヤーによる仮想世界でのゲーム

[4]BitTorrent、eDonkey など、認識されているすべてのピアツーピア（P2P）システムによるピアツーピアトラフィック、および Web ベースのファイル共有システムによるトラフィック

1.2.1 Internet Protocol (IP)

Internet Protocol (IP) はインターネットの設計当初から利用されてきたネットワークプロトコルであり、現在でも広く利用されている。IP を用いた通信の様子を図 1 に示す。IP ネットワークではクライアントがコンテンツを要求する度

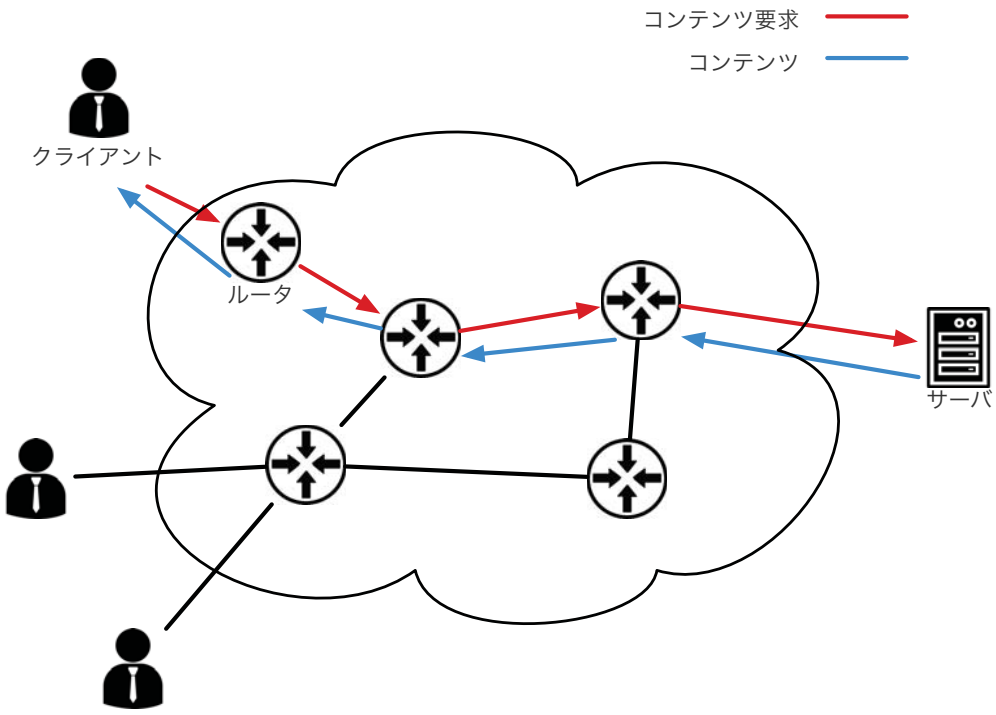

図 1 IP ネットワークでの通信

にサーバへと通信が行われ、コンテンツはサーバからクライアントへと提供される。このため、クライアントに対してサーバがネットワーク的に遠方にある場合であっても、必ずサーバとの通信が発生し非効率なものとなっていた。また IP アドレス (位置) を用いた通信を行っているため、要求を行うクライアントは必ず IP アドレスを調べてからサーバへの通信を行う必要がある。現状ではホスト名を指定し Domain Name System (DNS) サーバへと問い合わせることで、IP アドレスを検索するシステムが一般的である。しかし、クライアントにとってはコンテン

ツを取得することが目的であり、ネットワーク上のどのサーバと通信しているのかということは本質的ではない。

1.2.2 Content Delivery Network (CDN)

1.2.1 節で説明した IP ネットワークの欠点を補うアイデアとして Content Delivery Network (CDN) が存在している。CDN では図 2 のようにオリジナルのコンテンツをキャッシュした CDN サーバをインターネット上の各所に配置する。クライアントからサーバへのコンテンツリクエストが行われた時、もしオリジナルのコンテンツ配信サーバよりもキャッシュサーバの方がネットワーク上の距離が近い場合にはリクエストがキャッシュサーバへと転送され、キャッシュサーバがレスポンスを返す。オリジナルサーバが日本に存在し、キャッシュサーバが日本、

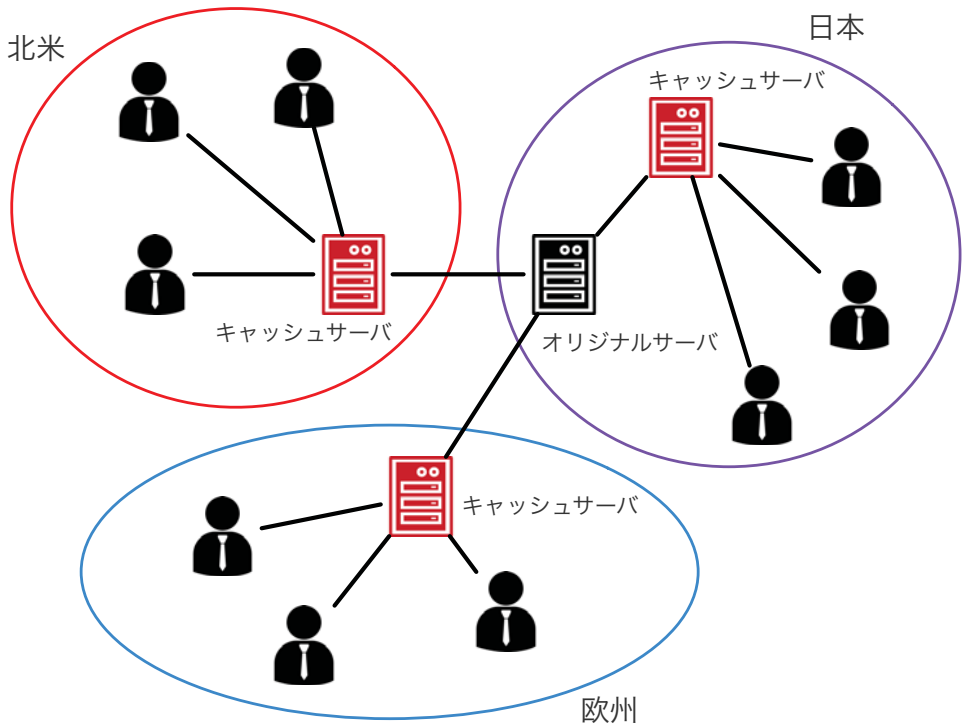

図 2 CDN イメージ図

北米、欧州にそれぞれ存在する場合を考える。今、北米にいるクライアントがコンテンツを取得しようとした場合、レスポンスを返すのは日本にあるオリジナルサーバではなく、距離の近い北米のキャッシュサーバが返すことになる。このようにコンテンツを配信することにより、クライアントは必ずしもオリジナルのコンテンツ配信サーバに通信を行う必要がなくなり、コンテンツを効率的に取得することができるようになる。

1.3 マルチキャスト配信

IP ネットワークにおいてコンテンツを配信する場合、マルチキャスト配信が従来より用いられてきた。ユニキャストが 1 対 1 の通信を実現するのに対して、マルチキャストでは 1 対多の通信を実現する。ユニキャストでコンテンツを配信する場合を図 3 に示す。ユニキャストでコンテンツを配信する場合、コンテンツを

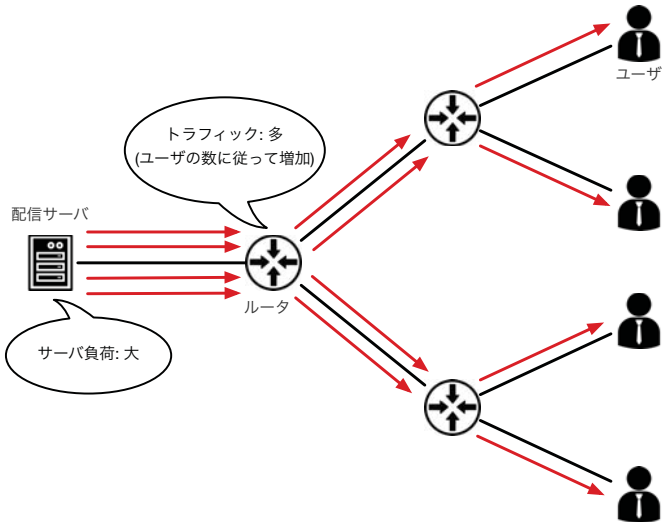

図 3 ユニキャストによるコンテンツ配信

視聴したいユーザの数に従ってトラフィックが増加していく。このため配信サーバや経路上のルータへとかかる負荷が大きくなる。マルチキャストでコンテンツを配信する場合を図 4 に示す。マルチキャストでコンテンツを配信する場合、経

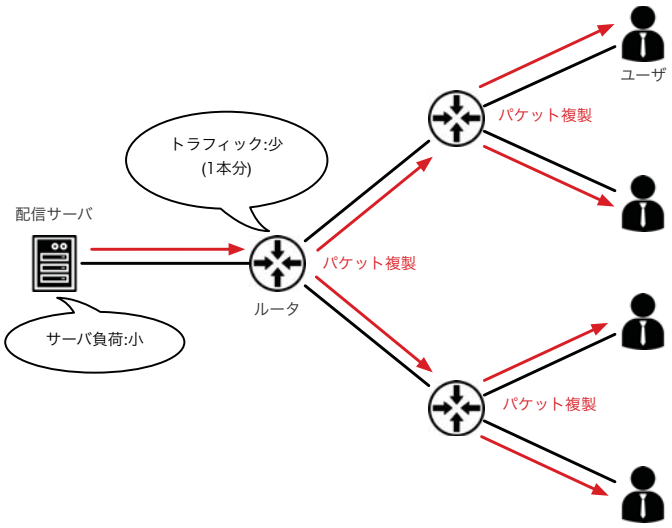

図 4　マルチキャストによるコンテンツ配信

路上のルータでパケットが複製されるため、経路途中のトラフィック増大を防ぐことができる。また 1 本のトラフィックでコンテンツを視聴したいユーザ全員へコンテンツの配信が行えるため、配信サーバの負荷も小さく済む。以上のことからコンテンツ配信に適した通信形態としてマルチキャスト配信が利用されてきた。しかし、ユーザの嗜好が多様化しコンテンツをオンデマンドに要求されるようになると、ユーザの需要を満たすためには配信サーバで全ユーザが要求するコンテンツを配信する必要が出てくるため、マルチキャストによる放送型のコンテンツ配信方法は適さない。このため現在のビデオオンデマンドな動画配信サービス等では、配信サーバを多重化しつつユニキャスト配信を行っている [9]。

1.4　情報指向ネットワーク

近年、IP ネットワークの問題点を解決するための CDN 以外のアプローチとして、情報指向ネットワークと呼ばれるシステムが注目を集めている。情報指向ネットワークではその開発プロジェクトや発表論文によって、Content Centric Networking (CCN) [1], Distributed Object-oriented Network Architecture (DONA)

[10], Publish-Subscribe Internetworking Routing Paradigm (PSIRP) [11], Network of Information (NetInf) [12], Named Data Networking (NDN) [13] 等の名称で呼称されているが、それらを総称して情報指向ネットワークまたは Information Centric Networking (ICN) と呼ぶ。本論文では主に CCN に基いて論じる。詳細は 2.1 で解説するものとし、以下には情報指向ネットワークの概要を示す。情報指向ネットワークでの通信の様子を図 5 に示す。情報指向ネットワークではネッ

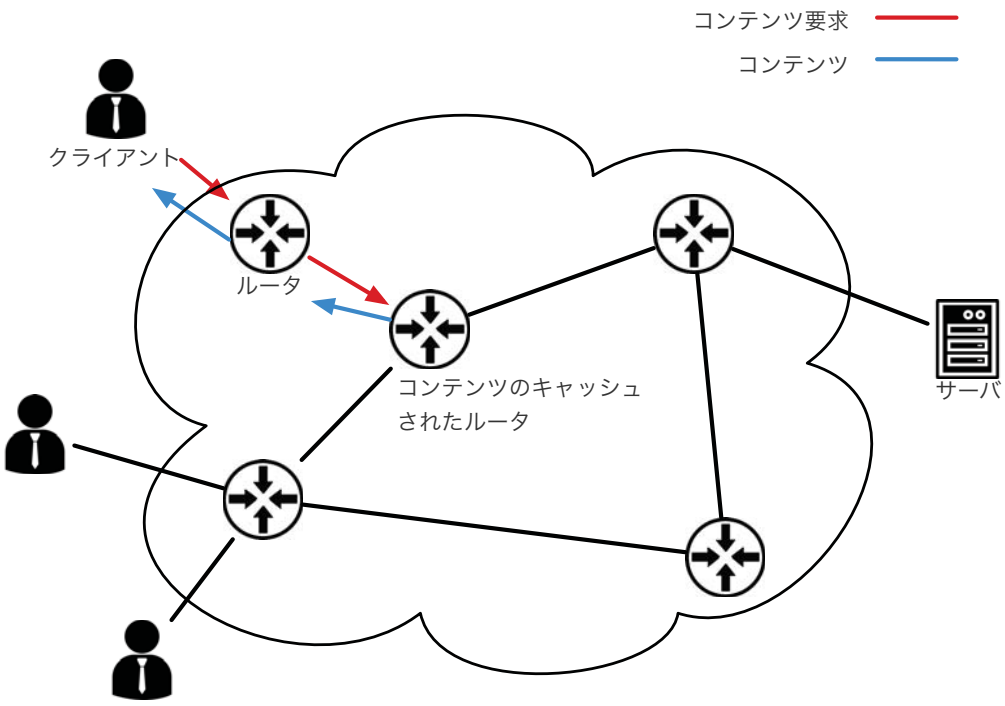

図 5 情報指向ネットワークでの通信

トワーク上のノードがオリジナルコンテンツのキャッシュを持つ。このことにより CDN と同様にクライアントは必ずしもサーバとの通信を行う必要はなくなる。またコンテンツを要求する時には IP アドレスではなく、コンテンツを一意に識別するためのコンテンツ名を用いて行う。ネットワーク上のキャッシュ、コンテンツ名を用いた要求の 2 つの特徴により、情報指向ネットワークではどのサーバと通

信するのかといったクライアントから見て本質的ではない事柄に依らずコンテンツを取得することが可能となる。またキャッシングをネットワークの機能として組み込むことでキャッシュを持つノードへコンテンツ要求を転送するといった能動的なキャッシュの利用が可能である。さらに IP ネットワークではセキュリティの要求があった場合、Secure Sockets Layer (SSL) や Internet Protocol Security (IPSec) 等で見られるようにセキュリティレイヤを追加して対応しているが、情報指向ネットワークではコンテンツパケットそのものにセキュリティ機構を組み込んでいる。IP ネットワークと情報指向ネットワークの特徴 [14] を表 2 にそれぞれ示す。

表 2 IP ネットワークと情報指向ネットワークの特徴

	IP ネットワーク	**情報指向ネットワーク**
通信形態	ホスト指向	コンテンツ指向
アドレシング	IP アドレス	コンテンツ名
セキュリティ	レイヤの追加	コンテンツに組み込み
キャッシュ	CDN サーバ	ネットワーク上のノード (ルータ等)

情報指向ネットワークの実用化に当たってはコンテンツ名の命名手法やキャッシュ利用手法、Internet of Things への適用等といった検討するべき問題が数多く存在している。中でもキャッシュ利用手法、キャッシュの管理手法は情報指向ネットワークにおいてコンテンツ取得までの時間を大きく左右する課題であり、情報指向ネットワークの分野が確立する前の類似の研究も含めると、キャッシュの管理をランダムに行う手法 [15]、チャンクの独自性に着目した手法 [16]、コンテンツの人気度に着目した手法 [17][18][19]、ヒューリスティックに行う手法 [20] 等、多くの研究者により研究されている。

1.5 研究目的

キャッシュの利用効率の向上は情報指向ネットワークにおいて、コンテンツ取得時間の短縮やサーバの負荷の分散といったネットワークの品質向上に繋がる重

要な課題である。そこで本研究の目的を、キャッシュ利用手法の提案によるキャッシュ利用効率の向上とする。キャッシュ利用効率を向上させるための既存研究として Cache Location and Searching[21] と呼ばれる手法が存在している。しかし、この既存研究ではキャッシュの一部を上手く利用できていないといった問題点がある。そこで本論文では全てのキャッシュを利用するために、既存研究で用いられていたパケット誘導テーブルを拡張し、さらに新規パケットを定義することによって既存研究の問題点を解決する。

1.6 論文構成

本論文の構成について説明する。第 2 章では情報指向ネットワークの詳細な解説として情報指向ネットワークが提案された論文である Networking Named Data[1] について説明する。さらに本研究の先行研究として Cache Location and Searching[21] について説明する。第 3 章では情報指向ネットワークにおけるキャッシュ利用効率向上のための提案を行う。第 4 章では提案手法をシミュレータ上で動作させ評価実験を行う。第 5 章では実験結果について考察を行い、シミュレータ上でない実環境での適用についても考察する。最後に第 6 章で結論を述べる。

2. 関連研究

本章では本研究の関連研究について述べる。最初に 2.1 節で情報指向ネットワークとは何かを理解するために情報指向ネットワークが提案された論文である Jacobson ら [1] の論文を説明する。次に 2.2 節で Yang ら [21] によるパケット転送テーブルを利用したネットワーキング手法の既存研究を説明する。最後に 2.3 節での関連研究の問題点をまとめる。

2.1 Networking Named Data

本節では情報指向ネットワークが提案された Jacobson ら [1] の研究について説明する。最初に情報指向ネットワークにおいて情報を送受信するためのパケットとして定義されている Interest および Data の 2 種類のパケットについて説明する。次にネットワーク内のノードが持つべき基本的な要素である Face、FIB (Forwarding Information Base)、PIT (Pending Interest Table)、CS (Content Store) の 4 つについて説明する。

Interest パケットは IP ネットワークにおけるリクエストパケットに等しい。Interest パケットは Consumer (情報指向ネットワークにおけるクライアント) から送信される。Interest パケットのフォーマットを図 6 に示す。コンテンツ名は Consumer が要求するコンテンツのネットワーク内での名前である。セレクタは要求するコンテンツをコンテンツの提供者や提供された時間などによって選別した場合に用いられ、セレクタの情報を元にコンテンツのフィルタリングが行われる。Nonce 値はコンテンツの暗号化や署名をする場合に用いられる使い捨てのランダムな値である。

Data パケットは IP ネットワークにおけるレスポンスパケットに等しい。Data パケットは Publisher (情報指向ネットワークにおけるサーバ) から送信される。Data パケットのフォーマットを図 7 に示す。コンテンツ名は Interest パケットのコンテンツ名と同様のものであ、Publisher やネットワーク上のキャッシュを持つノードが返信するコンテンツのネットワーク上での名前である。シグネチャはコンテンツが暗号化や署名された場合の暗号文や署名である。署名情報は暗号化や

署名を行った Publisher の識別情報である。データは Consumer が要求してきたデータである。

図 6 Interest パケットのフォーマット

図 7 Data パケットのフォーマット

情報指向ネットワークにおいてノードが持つネットワークインターフェースのことを Face と呼ぶ。情報指向ネットワークでのノードの概念図を図 8 に示す。

図 8 のノードはネットワークインターフェースとして Face1 から Face3 までの 3 つの Face を持っているということになる。

FIB(Forwarding Interest Base) はコンテンツ名と Face を対応付けたテーブルであり、Interest パケットの転送に使用される。図 8 ではそれぞれコンテンツ

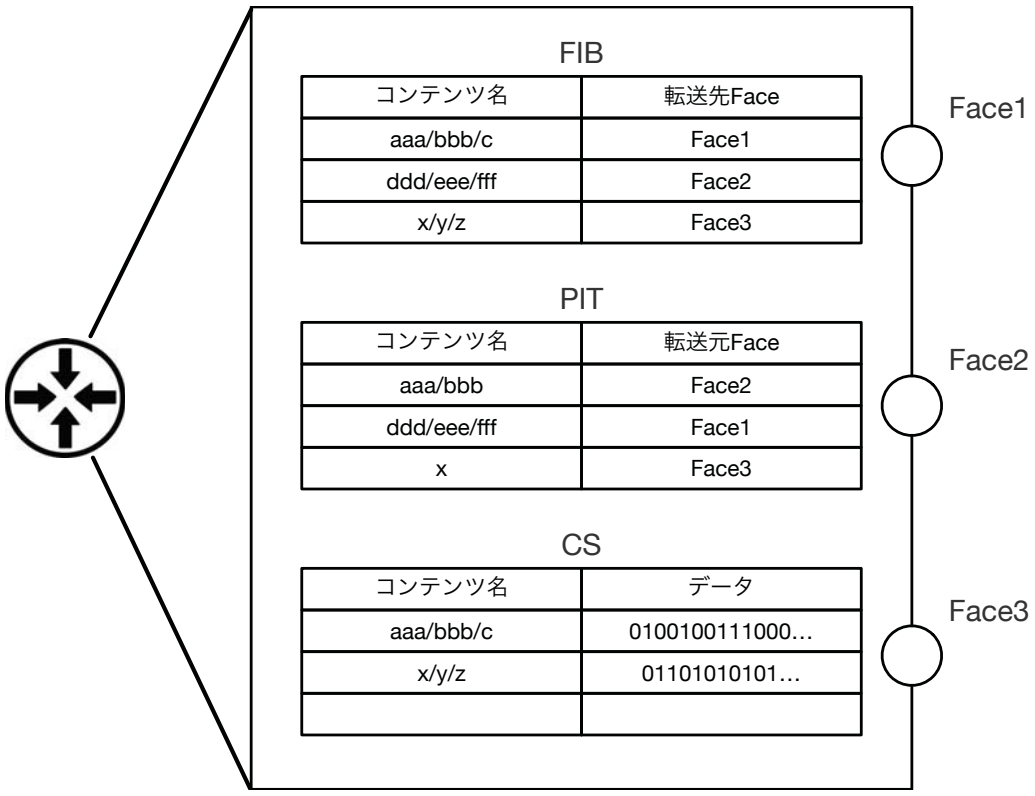
FIB
コンテンツ名
転送先Face
aaa/bbb/c
Face1
ddd/eee/fff
Face2
x/y/z
Face3
PIT
コンテンツ名
転送元Face
aaa/bbb
Face2
ddd/eee/fff
Face1
x
Face3
CS
コンテンツ名
データ
aaa/bbb/c
0100100111000…
x/y/z
01101010101…
Face1
Face2
Face3

図 8　ノードの概念図

aaa/bbb/c を Face1 と、コンテンツ ddd/eee/fff を Face2 と、コンテンツ x/y/z を Face3 と対応付けている。今、図 8 のノードにコンテンツ x/y/z への Interest パケットが到着したとすると、このノードは Interest パケットを Face3 へと転送する。

PIT(Pending Interest Table) は FIB と同様にコンテンツ名と Face を対応付けたテーブルであり、Data パケットの転送に使用される。PIT は Data パケットが対応する Interest パケットが通過してきた経路の逆を辿れるように、ノードが Interest パケットを転送してから Data パケットが返ってくるまでの間、Interest パケット送信元の Face をコンテンツ名と Face の組として記憶する。図 8 ではそれぞれコンテンツ aaa/bbb への Interest パケットが Face2 から、コンテンツ ddd/eee/fff への Interest パケットが Face1 から、コンテンツ x への Interest パケットが Face3 から転送されてきたことを示しており、同時にまだ返答の Data パケットが到着していないことを示している。

CS(Content Store) はコンテンツ名とコンテンツを対応付けたテーブルであり、ノード内でのキャッシュ管理に使用される。今、図 8 のノードにコンテンツ aaa/bbb/c への Interest パケットが到着した場合、このノードは FIB を参照して Interest パケットを Face1 へ転送することをせずに、直接このノードからコンテンツを格納した Data パケットを送信する。

情報指向ネットワークでは FIB, PIT, CS のテーブルをネットワーク内のルータ等のノードに実装し、Publisher が配信するコンテンツをノード上でキャッシングすることによって IP ネットワークよりも効率の良い通信を実現している。

この論文ではキャッシングにより、IP ネットワーク上で TCP を用いて通信を行った場合と比較して帯域幅が約 22%削減できたとしている。また、図 9 に示すようなネットワーク構成下においてコンテンツを Publisher へと要求する Consumer の数を 1 から 6 まで増加させていき、その時のダウンロード時間を計測する実験も行っている。この実験では IP ネットワーク上で TCP を用いて実験した場合と比較して、Publisher と ICN ノード間の通信を削減でき狭帯域である部分の影響を受けないため、ノードが増加していった場合であってもそれぞれの Consumer がコンテンツをダウンロードする時間がほぼ一定となるという結果が出ている。

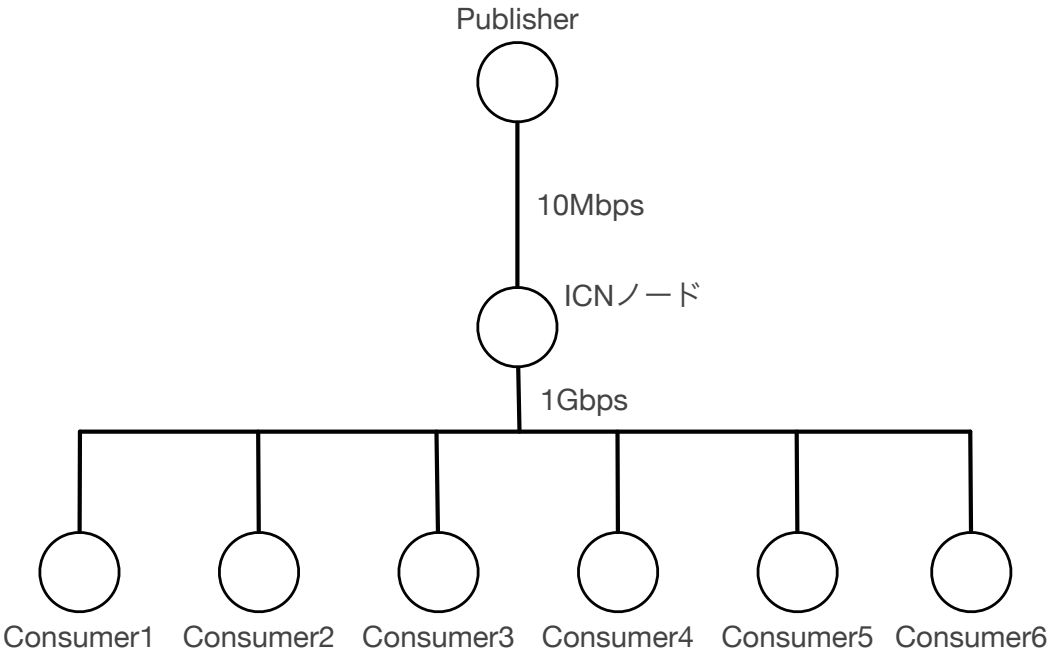

図 9 先行研究 [1] での実験トポロジー

2.2 Cache Location and Searching

本節では本論文の提案手法の直接の先行研究である Cache Location and Searching (以下、CLS) について説明する。CLS は Yang ら [21] によって提案された、情報指向ネットワークにおけるキャッシュ利用効率向上手法である。

2.2.1 CLS テーブル

CLS ではネットワーク上の各ノードが Interest パケットを転送するためのテーブル (以下、このテーブルを CLS テーブルと呼ぶ) を持つ。CLS テーブルエントリは 4 つのフィールドを持ち、それぞれのフィールドの役割は表 3 のようになっている。各ノードはこの CLS テーブルの情報を元に Interest パケットをキャッシュを持つノードの方向へと転送する。

表 3 CLS テーブルエントリ フィールド

フィールド名	役割
$ContentID$	コンテンツ名と同様
in	キャッシュが送信されてきた元のノード
out	キャッシュを送信した先のノード
h	Publisher からのホップ数

2.2.2 キャッシュ管理アルゴリズムと CLS テーブルの更新

CLS のキャッシュ管理アルゴリズムと CLS テーブル更新の様子を図 10 に示すネットワークトポロジーを用いて説明する。図 10 はまだどこからも Interest パケットが来ていない初期状態である。今、ノード C 以下にいる Consumer から Content ID が C1 のコンテンツに対しての Interest パケットが送信されてくる場合を考える (図 11)。

1. Interest パケットが Publisher へ到着 (→C→B→A→P)

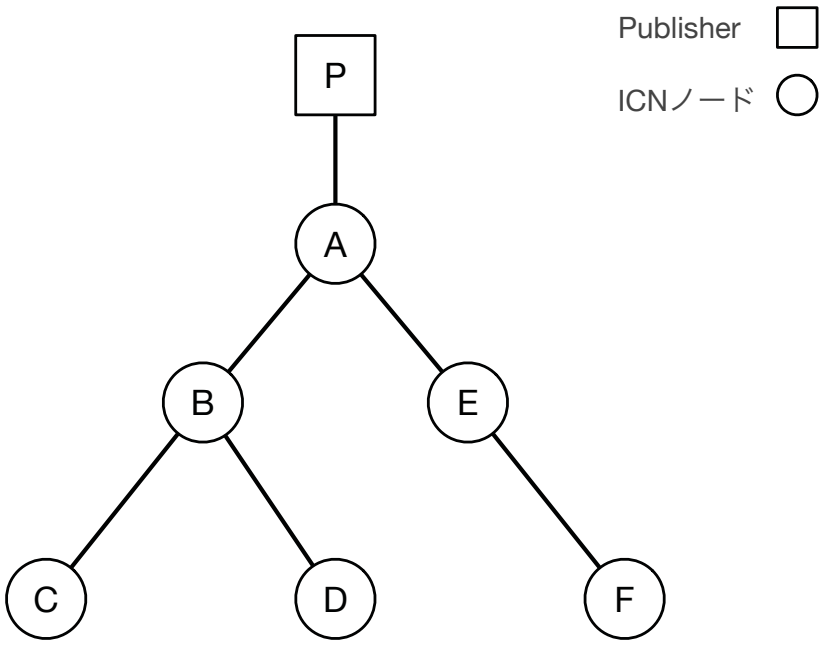

図 10 CLS トポロジー

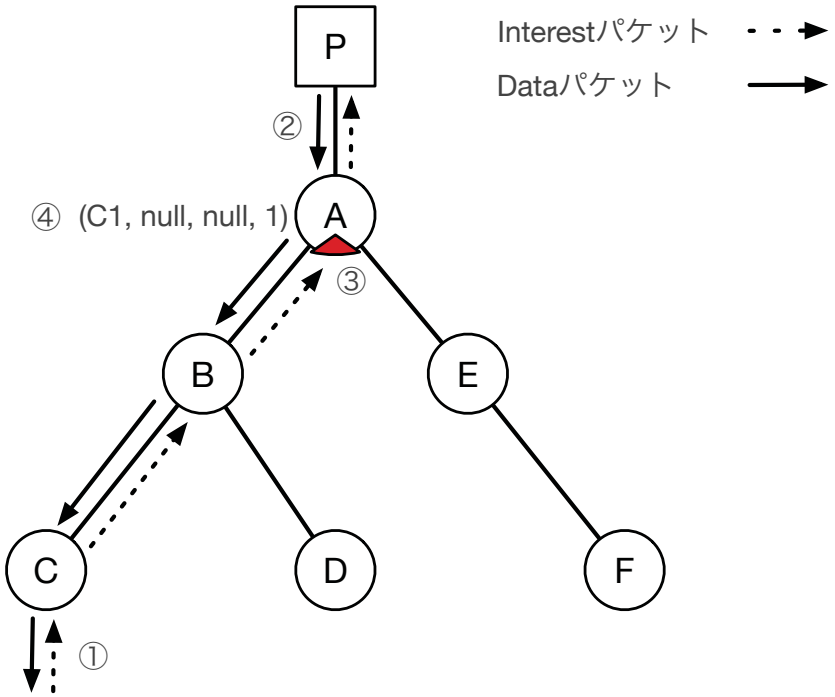

図 11 ノード A にキャッシュが追加された時 (CLS)

2. Publisher から Interest パケットの送信元 Consumer へ向けて Data パケットが返される (P→A→B→C→)

3. Publisher の下にあるノード A へとコンテンツがキャッシングされる (図 11 の赤色箇所)

4. コンテンツがキャッシングされたことによりノード A に CLS テーブルエントリが作成される

図 11 中では CLS テーブルエントリをタプルの形式で表し、タプルの内容は ($ContentID$, in, out, h) となっている。図 11 のタプルは Content ID が C1 のコンテンツを提供する Publisher はノード A からホップ数 1 の場所にあることを示している。エントリの in の値はノード A がキャッシングを行った初めてのノードであることから Null 値であり、out の値はまだキャッシュをどのノードに対しても移動させていないことから Null 値となる。

続いてネットワークの状態が図 11 である時に、ノード C 以下にいる Consumer から Content ID が C1 のコンテンツに対しての Interest パケットが送られてくる場合を考える (図 12)。

1. Interest パケットがノード A へ到着 (→C→B→A)

2. キャッシュをもつノード A からパケットの送信元 Consumer へ向けて Data パケットが返される (A→B→C→)

3. ノード A の下にあるノード B へとキャッシュが移動する

4. キャッシュを移動させたことによりノード A からノード B へキャッシュを移動させたことを示すようにノード A の CLS テーブルの out の値が更新される

5. コンテンツがキャッシングされたことによりノード B に CLS テーブルエントリが作成される

3 の動作のようにキャッシュが元々あったノードから次のノードへと移動し、なおかつ元のノードにキャッシュを残さないキャッシュ管理アルゴリズムを Move Copy

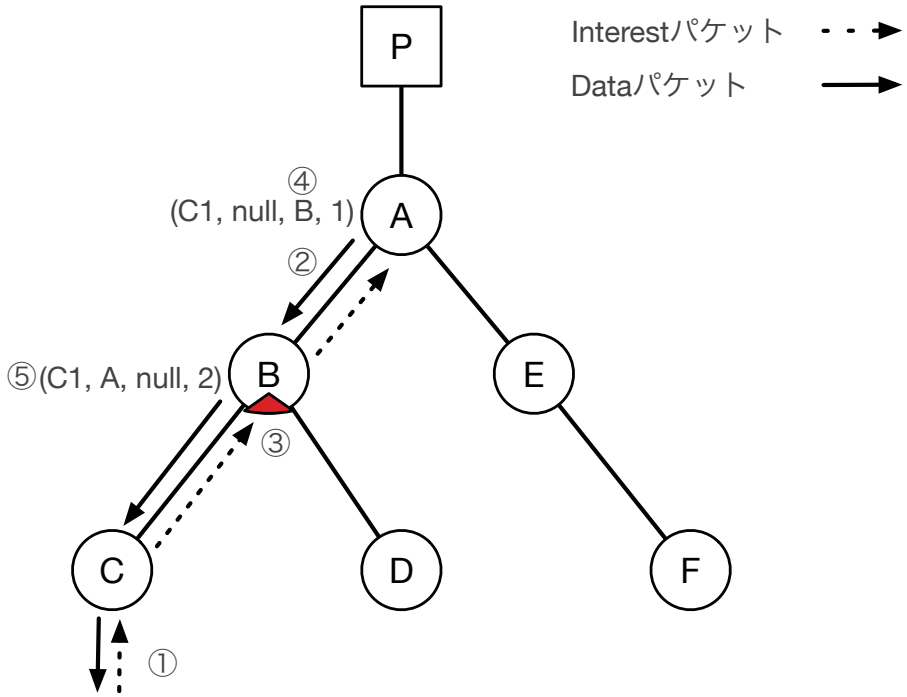

図 12　ノード A からノード B にキャッシュが移動された時 (CLS)

Down (MCD) と呼ぶ。図 12 のノード B のタプルは Content ID が C1 のキャッシュはノード A から送られてきたこと、C1 を提供する Publisher はノード B からホップ数 2 の場所にあることを示している。エントリの out の値はまだキャッシュをどのノードに対しても移動させていないことから Null 値となる。

さらに続いてネットワークの状態が図 12 である時に、ノード C 以下にいる Consumer から Content ID が C1 のコンテンツに対しての Interest パケットが送られてくる場合を考える (図 13)。

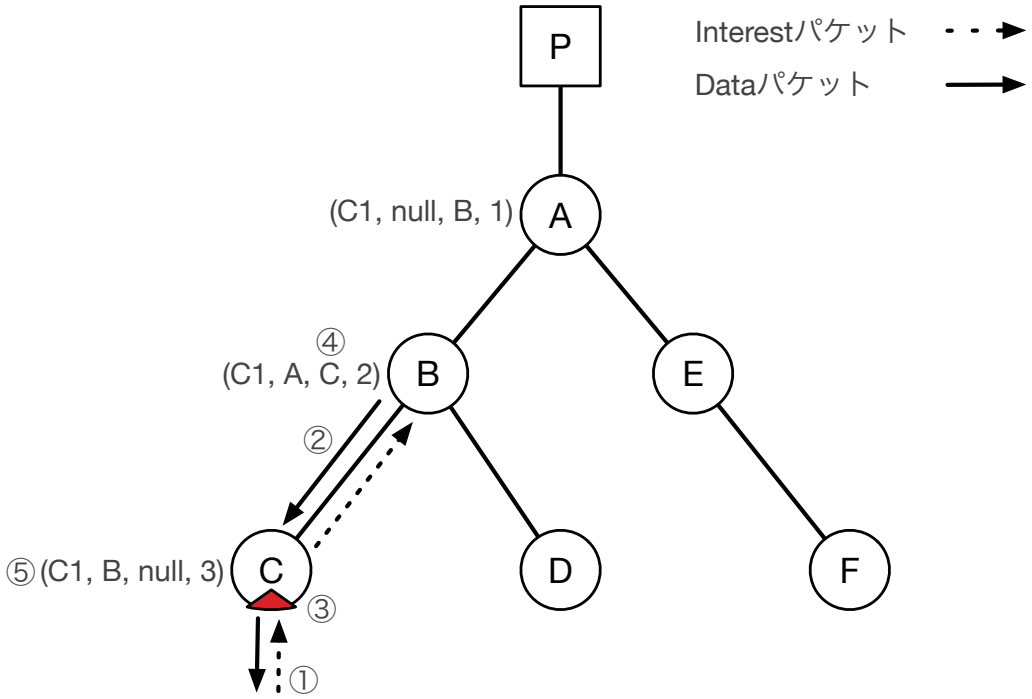

図 13 ノード B からノード C にキャッシュが移動された時 (CLS)

1. Interest パケットがノード B へ到着 (→C→B)
2. キャッシュを持つノード B からパケットの送信元 Consumer へ向けて Data パケットが返される (B→C→)

3. MCD のアルゴリズムに従いノード B からノード C へのキャッシュが移動する

4. キャッシュを移動させたことによりノード B からノード C へキャッシュを移動させたことを示すようにノード B の CLS テーブルエントリの *out* の値が更新される

5. コンテンツがキャッシングされたことによりノード C にも CLS テーブルエントリが作成される

図 13 のノード C のタプルは Content ID が C1 のキャッシュはノード B から送られてきたこと、C1 を提供する Publisher はノード C からホップ数 3 の場所にあることを示している。エントリの out の値は前回同様にまだキャッシュをどのノードに対しても移動させていないことから Null 値となる。

最後にネットワークの状態が図 13 である時に、ノード C のキャッシュ用のメモリ容量が埋まり、キャッシュが溢れて Content ID が C1 のキャッシュをノード C の CS から削除しなれけばならない場合を考える (図 14)。

1. ノード C が持つ CLS テーブルの in が示すノード B に対して Return-back パケットが送信される (C→B)

2. Return-back パケットを受信したノード B へキャッシュが戻される

3. Return-back パケットを受信したノード B の CLS テーブルエントリの out の値が Null 値となる

4. Return-back パケットを送信したノード C から CLS テーブルエントリが削除される

1 から 4 までの動作によりノードからキャッシュが溢れた場合であっても、キャッシュがネットワーク内から完全に削除されることを防いでいる。

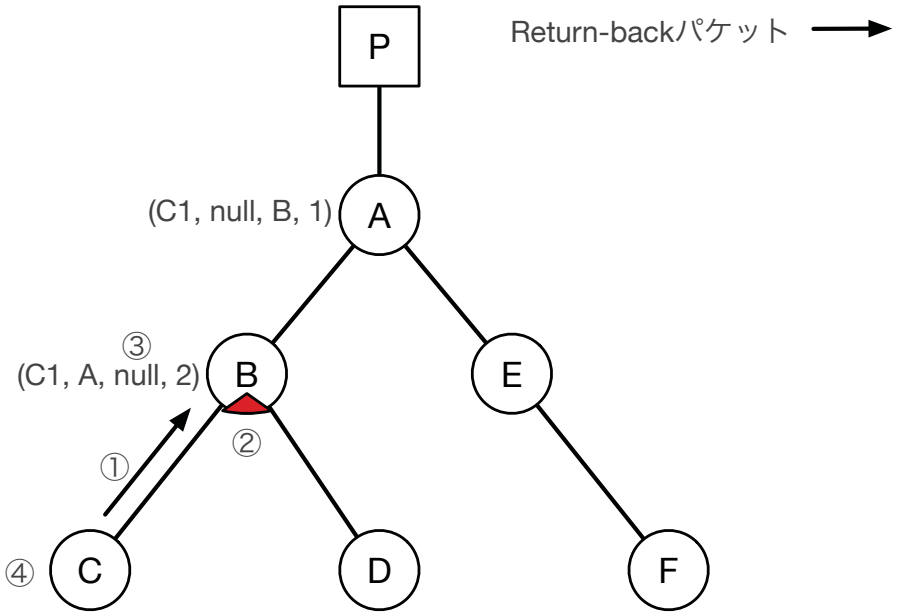

図 14　キャッシュが溢れた時 (CLS)

2.2.3 パケットの転送とキャッシュ保持のルール

前述の通り CLS では CLS テーブルの情報に従って Interest パケットをキャッシュを持つノードへと転送するが、全ての Interest パケットをキャッシュの方向へ転送するのではない。閾値によって Publisher へ転送するのか、キャッシュを持つノードの方向へ転送するのかを判断し、よりホップ数が小さい方向へ転送する。この閾値を H_{th} として CLS では以下のように定義している。

$$H_{th} = (\text{サーバとリーフノード間の高さ})/2 \tag{1}$$

この値と各ノードの持つ Publisher へのホップ数 (CLS テーブルの h の値) を比較して、どちらへ Interest パケットを転送するのかを判断する。また CLS にはキャッシュ保持に関して、あるコンテンツのキャッシュは同一経路上で 1 つのみ保持することができるという制約がある。

以下では Interest パケットを Publisher へ転送する場合と、キャッシュを持つノードへ転送する場合を例を挙げてパケットの転送とキャッシュ保持のルールについて説明する。

キャッシュを持つノードへ転送する場合

Interest パケットを転送するルータの h の値と H_{th} の値を比較し、以下の式が成り立つ場合にキャッシュを持つノードへ Interest パケットが転送される。

$$h \geq H_{th} \tag{2}$$

今、ネットワークの状態が図 13 の時にノード D 以下にいる Consumer から Content ID が C1 のコンテンツに対して Interest パケットが送信されてきた場合を考える (図 15)。

1. Interest パケットがノード B に到着 (→D→B)
2. ノード B は Interest パケットをキャッシュを持つノード C へと転送する (B→C)

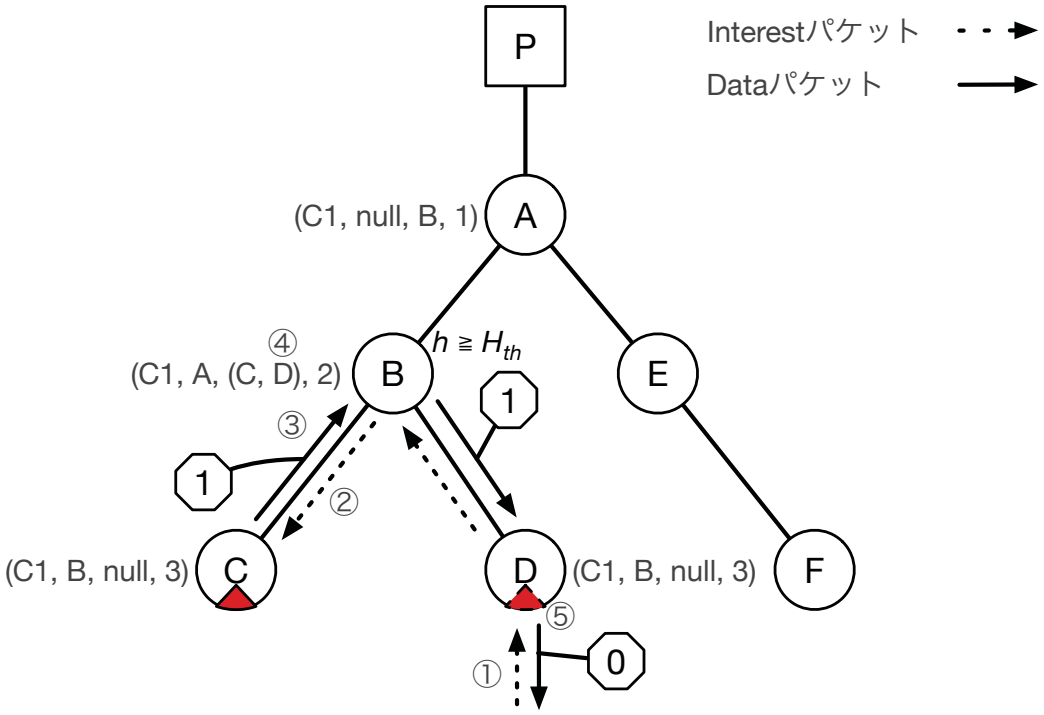

図 15　キャッシュをもつノードへ Interest パケットが転送される場合

3. Interestパケットを受信したノードCはDataパケットを返す (→C→B→D→)

4. ノードBから見てノードDの方向でキャッシュが追加されたことを示すようにノードBのCLSテーブルエントリの *out* の値が更新される

5. ノードDにキャッシュが追加される

2の時にノードBはInterestパケットをPublisherへ転送するか、キャッシュをもつノードCへ転送するかを判断する。この時、ノードBのCLSテーブルエントリの h の値は2である。一方、このネットワークトポロジーにおける H_{th} の値は式 (1) より1.5である。よって式 (2) が成り立つためノードBはキャッシュを持つノードCへInterestパケットを転送した。また4の時、Dataパケットがキャッシュを持っていないノードBを通過するが、ノードBにはキャッシュを残さない。これは経路P→A→B→Cにおいては既にノードCがキャッシュを持っているためである。このためノードBにはキャッシュが保持されず、Interestパケットがノードに転送された時に、まだノードDを通る経路にてキャッシュを保持していないことから、ノードDにてキャッシュを保持する。さらに今回の場合はノードCへ到着したInterestパケットがCLSテーブルの *in* の方向から来たため、MCDアルゴリズムは働かずノードCからはキャッシュが削除されない。

Publisherへ転送する場合

Interestパケットを転送するルータの h の値と H_{th} の値を比較し、以下の式が成り立つ場合にPublisherへInterestパケットが転送される。

$$h < H_{th} \tag{3}$$

今、ネットワークの状態が図13の時にノードF以下にいるConsumerからContent IDがC1のコンテンツに対してInterestパケットが送信されてきた場合を考える (図16)。

1. Interestパケットがノードに到着 (→F→E→A)

2. ノードはInterestパケットをPublisherへと転送する (A→P)

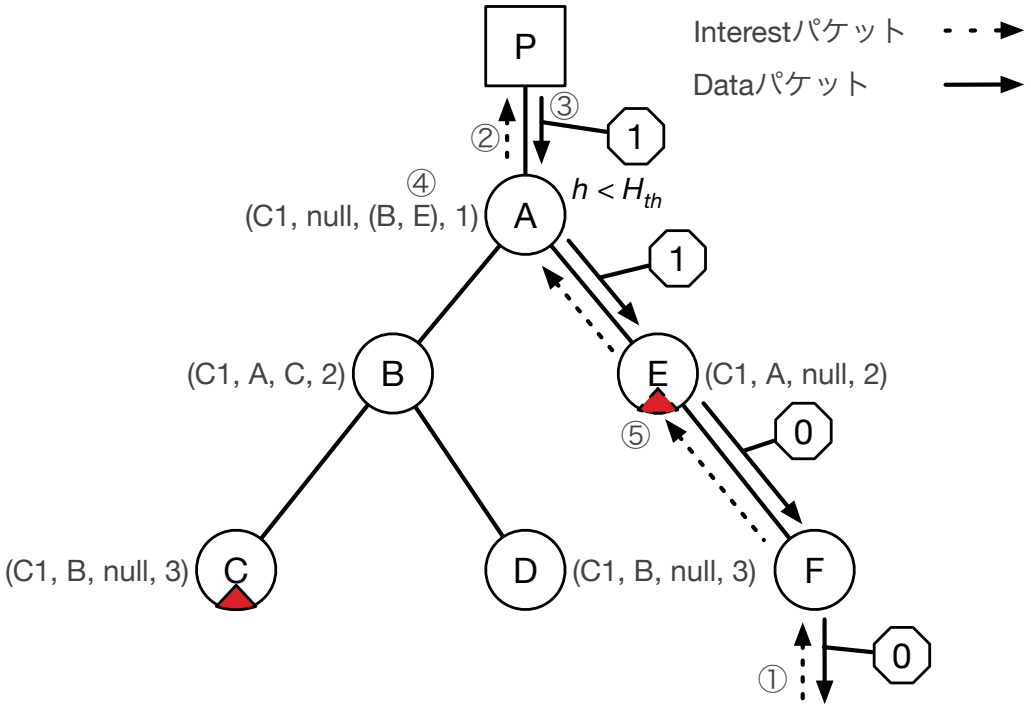

図 16　Publisher へ Interest パケットが転送される場合

3. Interestパケットを受信したPublisherはDataパケットを返す (P→A→E→F→)

4. ノードAから見てノードEの方向でキャッシュが追加されたことを示すようにノードAのCLSテーブルエントリの*out*の値が更新される

5. ノードEにキャッシュが追加される

2の時にノードAはInterestパケットをPublisherへ転送するか、キャッシュをもつノードCへ転送するかを判断する。この時、ノードAのCLSテーブルエントリのhの値は1である。一方、このネットワークトポロジーにおけるH_{th}の値は式 (1) より1.5である。よって、式 (3) が成り立つためノードAはInterestパケットをPublisherへと転送した。また4の時、Dataパケットがキャッシュを持っていないノードAを通過するが、ノードAにはキャッシュを残さない。これは経路P→A→E→Cにおいては既にノードCがキャッシュを持っているためである。このためノードBにはキャッシュが保持されず、Interestパケットがノード Eに転送された時に、まだノードEを通る経路にてキャッシュを保持していないことからノードEにてキャッシュが保持される。

2.2.4 CLSの問題点

CLSの問題点として以下の3つの点が挙げられる。

- $h < H_{th}$の場合に一部のキャッシュが活用されない
- CLSテーブルの情報に誤りが生じる問題
- 各ノードに対してH_{th}の値を入力する必要がある

以下では、この3つの問題点について説明する。

$h < H_{th}$の場合に一部のキャッシュが活用されない問題

図17に示すネットワークの状態について考える。図17ではノードCがContent IDがC1のコンテンツのキャッシュを持っている。また、ノードAとノードBが

それぞれノード C の持っているキャッシュに対する CLS テーブルエントリを持っている。今、ノード F 以下にいる Consumer から Content ID が C1 のコンテンツ

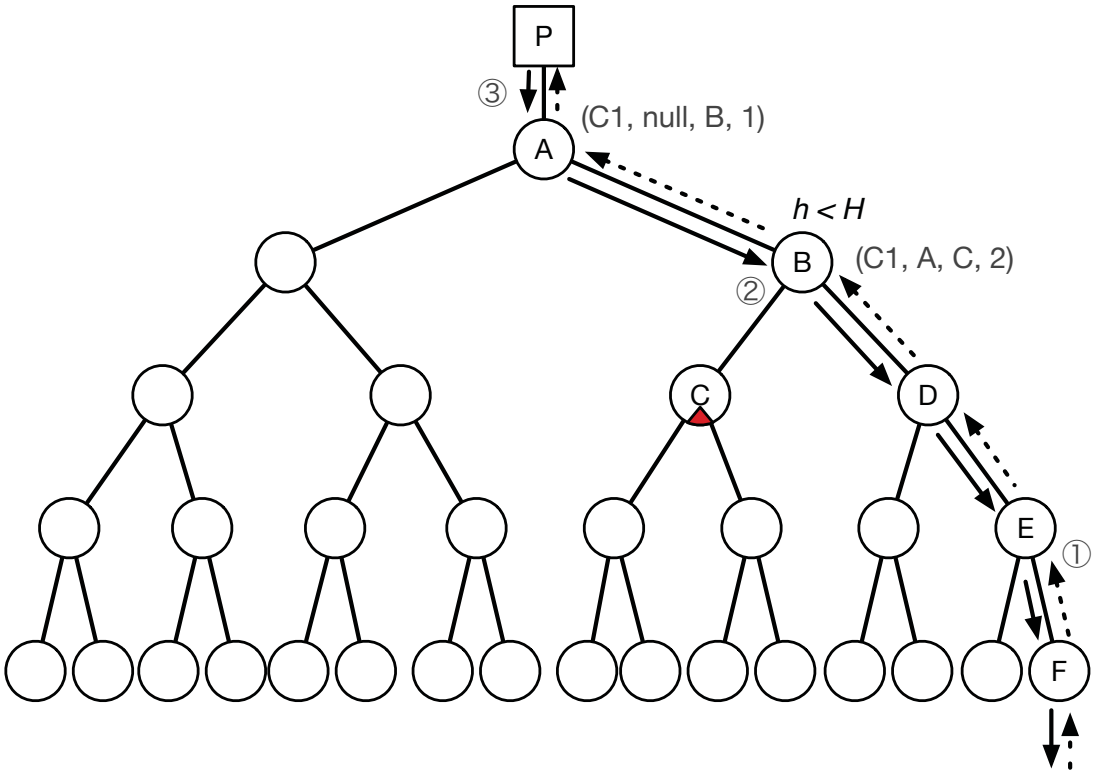

図 17　一部のキャッシュが活用されてない

に対して Interest パケットが送信されてきた場合を考える。

1. Interest パケットがノード B に到着 (→F→E→D→B)
2. ノード B は Interest パケットを Publisher へと転送する (B→A→P)
3. Publisher から Interest パケットの送信元 Consumer へ向けて Data パケットが返される

2 の時にノード B は Interest パケットを Publisher へ転送するか、キャッシュをもつノード C へ転送するかを判断する。この時、ノード B の CLS テーブルエント

リの h の値は 2 である。一方、このネットワークトポロジーにおける H_{th} の値は式 (1) より 2.5 である。よって、式 (3) が成り立つためノード B は Interest パケットを Publisher へと転送する (B→A→P)。しかしこの時、ノード B から考えたときの Publisher とノード C でのホップ数では、ノード C へのホップ数の方が小さい。このように $h < H_{th}$ の時、ノードが持つキャッシュの方へと Interest パケットを転送した方が効率的にも関わらず Publisher の方へと Interest パケットが転送されてしまう問題がある。

CLS テーブルの情報に誤りが生じる問題

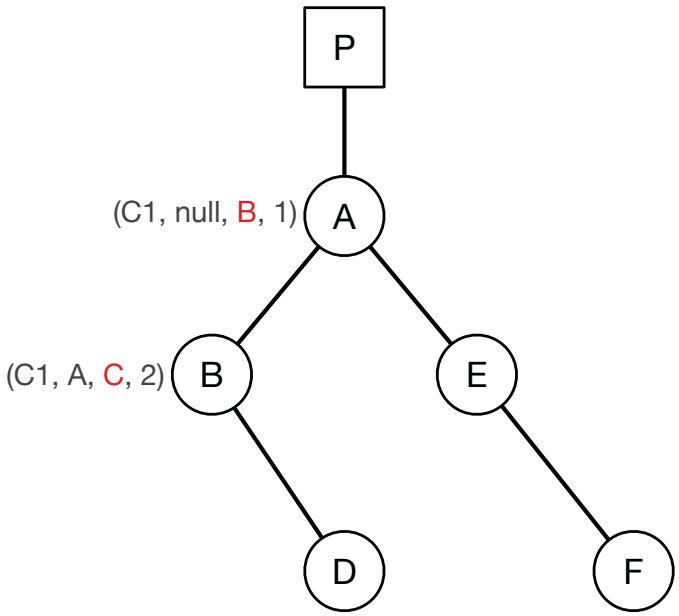

図 18　BC 間のリンクが切れた時 (CLS)

CLS では障害が発生しノード間のリンクが切断されキャッシュがネットワーク上から失われてしまった場合であっても、CLS テーブルは存在しないキャッシュ

の方向を指し続けてしまう問題がある。例えばネットワークの状態が 13 の時に、ノード B とノード C 間で障害が発生しリンクが切れてしまう場合を考える (図 18)。この時ノード C がネットワーク上から切り離されるため、大元のネットワーク上からはキャッシュが失われる。しかしキャッシュが失われてしまうにも関わらずノード A とノード B に存在する CLS テーブルはノード C にあったキャッシュを指し続けてしまうため (図 18 赤文字部)、CLS テーブルとキャッシュの整合性が取れなくなる。また図 15 の 4 の場合において必ずしもノード D の方向でコンテンツがキャッシングされるとも限らないにも関わらずノード B の CLS テーブルにノード D 方向にキャッシュがあることを示す情報が作成されてしまうことがある。これには例えばノード D がキャッシュを持たない Consumer である場合が考えられる。これはキャッシュされたか否かを確認せず、Interest パケットを転送した時に情報が追加されてしまうために起こる。以上の場合には Interest パケットが存在しないノード C に対して延々と転送され続けることが起こり得る。この問題を修正するには管理者がキャッシュがどこに存在するのかを把握した上で手動で CLS テーブルを更新するしかないが、それは現実的ではない。

全てのノードに対して H_{th} の値を入力する必要がある問題

これまでの説明で述べた通り、各ノードは自らの持つ CLS テーブルエントリと H_{th} の値を比較することで Interest パケットの転送方向を決定する。このため各ノードに対して事前に H_{th} の値を入力しておく必要がある。この入力作業はネットワークの管理者が行うことになると想定されるが、ネットワークが大規模になればなるほど、その管理コストは膨大していくと考えられる。

2.3 関連研究のまとめ

本章では 2.1 節で情報指向ネットワークが提案された論文である [1] を説明し、情報指向ネットワークがどのようなものであるかを説明した。また、2.2 節でパケット転送テーブルを用いたキャッシュ利用効率向上手法の既存研究である [21] を説明し、その問題点を指摘した。問題点をまとめると以下の 3 つになる。

- $h < H_{th}$ の場合に一部のキャッシュが活用されない
- CLS テーブルの情報に誤りが生じる
- ネットワーク上の全てのノードに対して H_{th} の値を入力する必要がある

これら 3 つの問題を解決し、CLS よりもキャッシュヒット率を向上させることを本論文で提案する手法の目的とする。

3. テーブル拡張と新規パケットの定義によるキャッシング手法の提案

前章でCLSについての3つの問題点を指摘した。これらの問題点のうち2つはCLSの各ノードがInterestパケットを転送するための判断にH_{th}の値を用いていることに起因している。ネットワーク上の各ノードがPublisherへのホップ数とキャッシュを持つノードへのホップ数の両方を持っていれば、ネットワーク上のどのノードであってもH_{th}の値に依存せず転送の判断が可能となる。このためH_{th}の値を各ノードへ入力する必要もなくなる。本章では、H_{th}の値へ依存しないパケット転送手法としてCLSテーブルの拡張とパケットの定義によるキャッシュ利用効率向上手法を提案する。最初に3.1節でCLSテーブルの拡張について説明し、続く3.2で新たに定義するHopCountパケットについて説明する。最後に3.3節で提案手法の例を挙げて説明する。

3.1 拡張CLSテーブル

キャッシュを持つノードへのホップ数を保持するためにCLSテーブルエントリにそれを記憶しておくフィールドが必要となる。このため本提案手法では第2.2節の表3で説明したCLSテーブルを以下の表4のように拡張した(以下では表4のテーブルを拡張CLSテーブルと呼ぶ)。フィールド*out*を*out-h*へと変更し、

表4 拡張CLSテーブルエントリ フィールド

フィールド名	役割
ContentID	コンテンツ名と同様
in	キャッシュが送信されてきた元のノード
out-h	キャッシュを送信した先のノードとそのノードまでのホップ数の組
h	Publisherからのホップ数

キャッシュを送信した先のノードとそのノードまでのホップ数の組となっている。

out-h は表 5 の様な組であり、CLS テーブルにも存在したキャッシュを送信した先のノードを示す *out* と、新しく追加されたキャッシュを持つノードまでのホップ数を示す h_{cache} をまとめて管理する。

表 5　*out-h* の組

フィールド名	役割
out	キャッシュを送信した先のノード
h_{cache}	キャッシュを持っているノードまでのホップ数

提案手法では CLS で用いた H_{th} の代わりに表 5 の h_{cache} の値を、h の値と比較することで、Interest パケットを Publisher の方向へ転送するか、キャッシュを持つノードの方向へ転送するかを判断する。

3.2 HopCount パケット

CLS テーブルに新しく追加した h_{cache} フィールドを設定するためのパケットとして HopCount パケットを定義する。HopCount パケットはノードにキャッシュが追加される度にキャッシュを受け取ったノードから送信されるパケットである。HopCount パケットは受け取ったキャッシュの $ContentID$ と更にホップ数をカウントするためのフィールドとして hc フィールドを持つ。hc フィールドには整数値が格納され、HopCount パケットを送信したノードからは $hc = 1$ として送信される。HopCount パケットを受け取ったノードは hc の値で自らの持つ拡張 CLS テーブルエントリの h_{cache} の値を更新する。さらに HopCount パケットを受け取ったノードは hc の値を 1 だけインクリメントして CLS テーブルの in の値が示すノードへ HopCount パケットを転送する。このようにすることでネットワーク上の各ノードがキャッシュを持つノードへのホップ数を正しく把握できる。

3.3 キャッシュ管理アルゴリズムと拡張 CLS テーブルの更新

提案手法がどのようにキャッシュを管理し、拡張 CLS テーブルを更新するのかを前章で CLS の説明に用いたものと同じネットワークトポロジー (図 19) を用いて説明する。図 19 はまだどこからも Interest パケットが来ていない初期状態であ

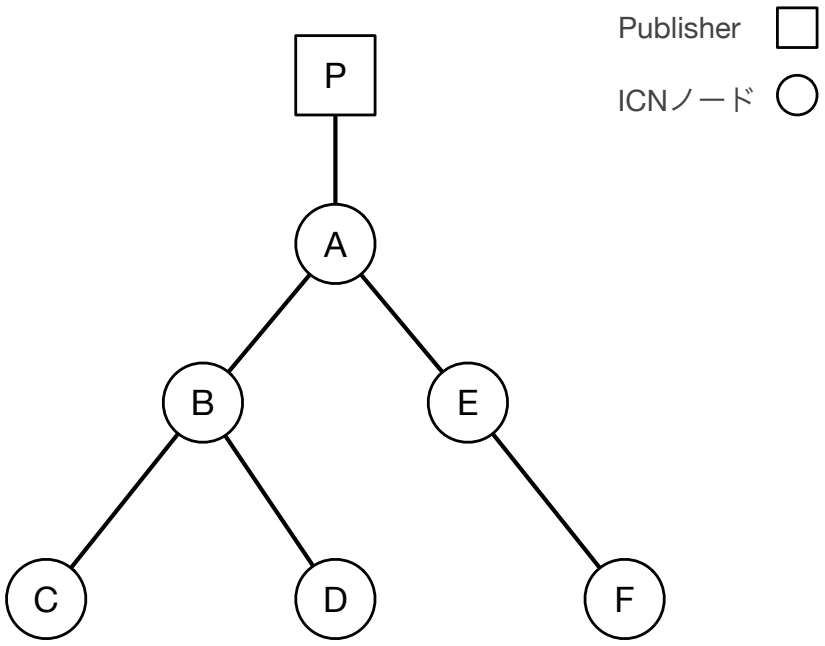

図 19 CLS トポロジー (再掲)

る。今、ノード C 以下にいる Consumer から Content ID が C1 のコンテンツに対しての Interest パケットが送信されてくる場合を考える (図 20)。

1. Interest パケットが Publisher へ到着 (→C→B→A→P)
2. Publisher から Interest パケットの送信元 Consumer へ向けて Data パケットが返される (P→A→B→C→)
3. Publisher の下にあるノード A へとコンテンツがキャッシングされる

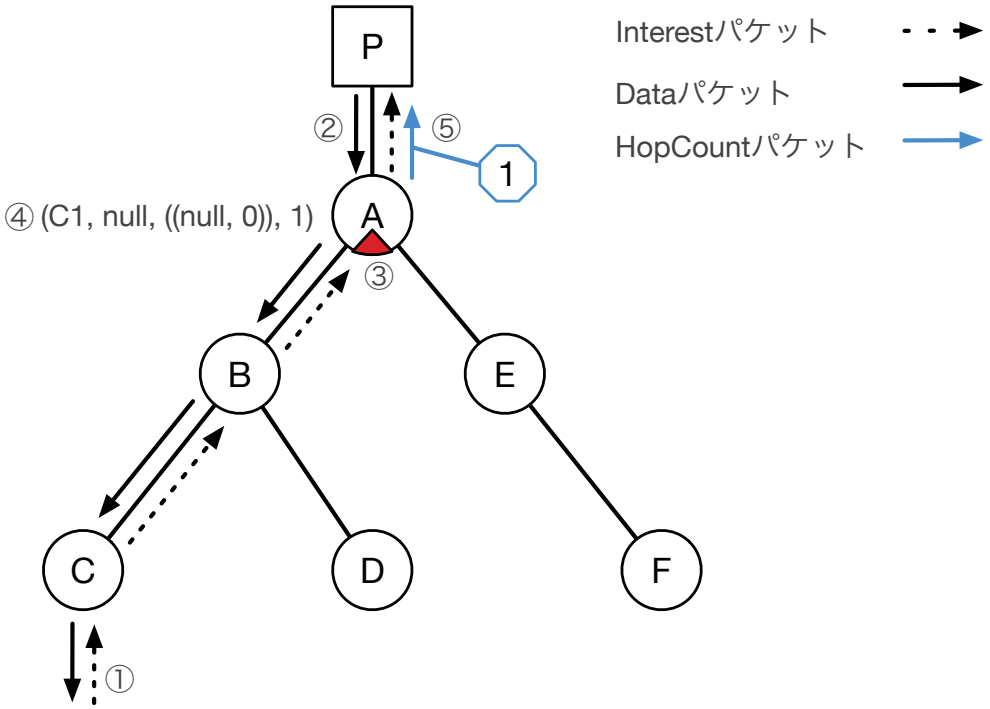

図 20　ノード A にキャッシュが追加された時 (提案手法)

4. コンテンツがキャッシングされたことによりノード A に拡張 CLS テーブルエントリが作成される

5. 拡張 CLS テーブルエントリの out の値が示す方向へノード A から HopCount パケットが $hc = 1$ で送信される (A→P)

図 20 中では拡張 CLS テーブルエントリを入れ子になったタプルの形式で表し、タプルの内容は $(ContentID, in, ((out, h_{cache})), h)$ となっている。今、自らがキャッシュを持っていることから h_{cache} の値は 0 である。拡張 CLS テーブルエントリの他のフィールドが示す意味は CLS の場合と同様である。

続いてネットワークの状態が図 20 である時に、ノード C 以下にいる Consumer から Content ID が C1 のコンテンツに対しての Interest パケットが送られてくる場合を考える (図 21)。

1. Interest パケットがノード A へ到着 (→C→B→A)

2. キャッシュをもつノード A からパケットの送信元 Consumer へ向けて Data パケットが返される (A→B→C→)

3. Data パケットを受け取ったノード B はコンテンツをキャッシングする

4. コンテンツがキャッシングされたことにより、ノード B にも拡張 CLS テーブルエントリが作成される

5. 拡張 CLS テーブルエントリの in の値が示す方向へノード B から HopCount パケットが $hc = 1$ で送信される (B→A)

6. HopCount パケットを受信したノード A は hc の値 1 とパケットを受信した Face 情報より自らの拡張 CLS テーブルエントリの out フィールドと h_{cache} フィールドを更新し、対応するコンテンツのキャッシュを削除する

7. ノード A は HopCount パケットの hc の値を 1 だけインクリメントし、拡張 CLS テーブルエントリの示す方向へ転送する (A→P)

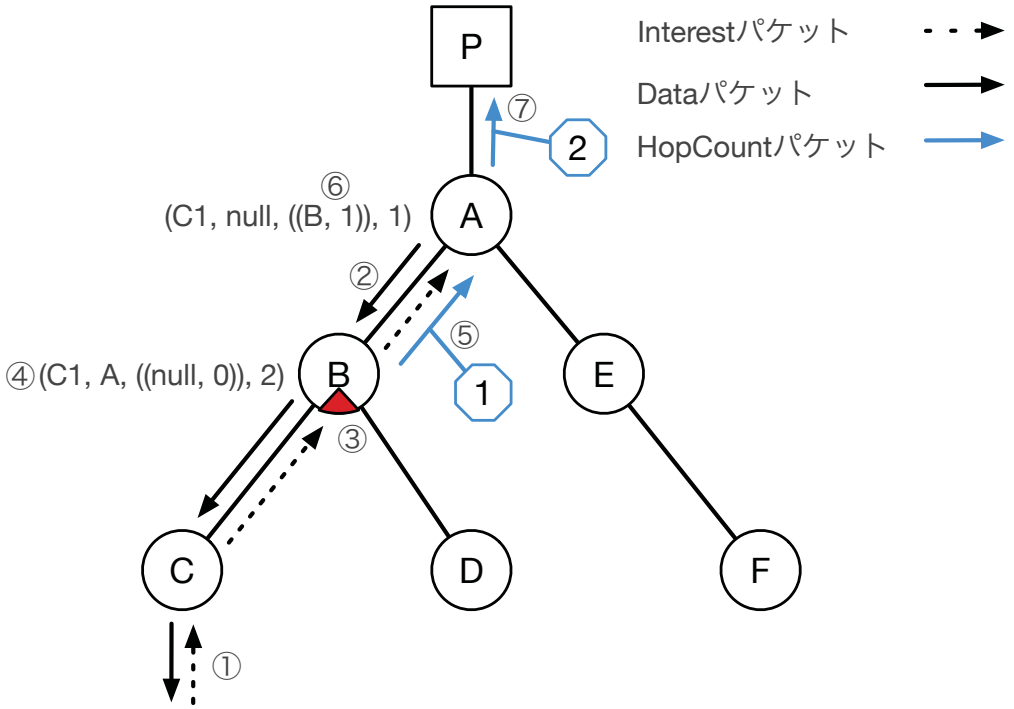

図 21　ノード A からノード B にキャッシュが移動された時 (提案手法)

4 の時、ノード B の拡張 CLS テーブルエントリの h_{cache} の値は自らがキャッシュを持っていることにより 0 となる。また提案手法では $hc = 1$ の HopCount パケットを受信した場合に、受信したノードは out の値と h_{cache} の値を更新し、キャッシュを削除する。hc が 1 以外の値の HopCount パケットを受信した場合は h_{cache} のみを更新する。このルールより、6 の時に h_{cache} と out が共に更新された。

さらに続いてネットワークの状態が図 21 である時に、ノード C 以下にいる Consumer から Content ID が C1 のコンテンツに対しての Interest パケットが送られてくる場合を考える (図 22)。

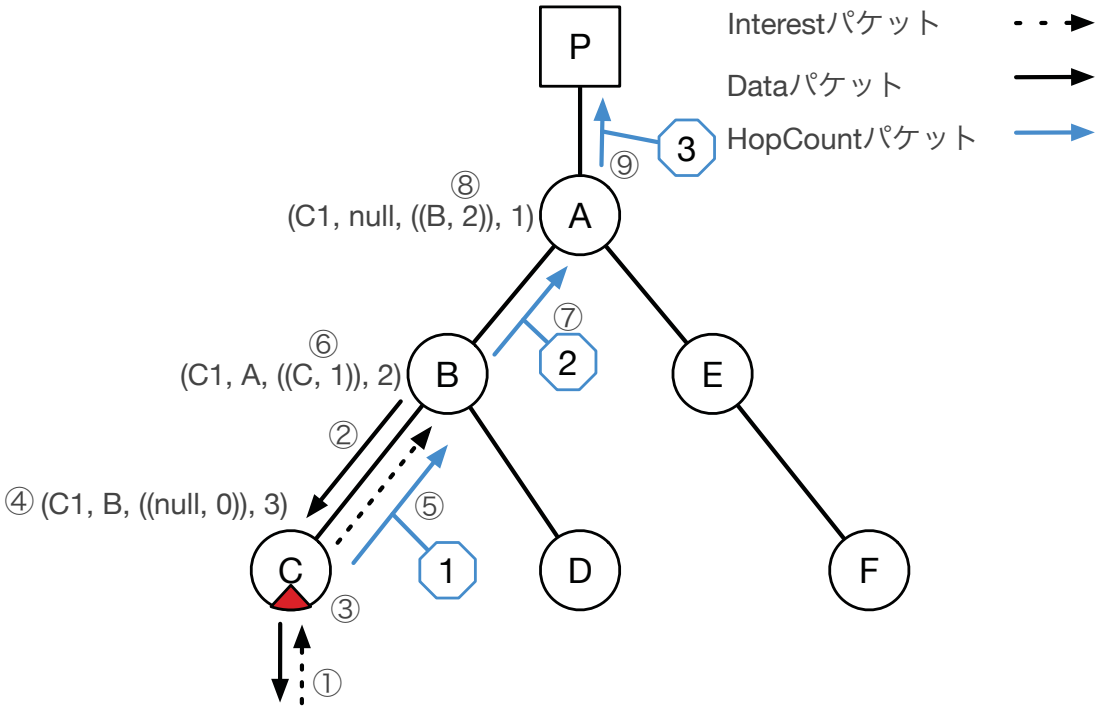

図 22　ノード B からノード C にキャッシュが移動された時 (提案手法)

1. Interest パケットがノード B へ到着 (→C→B)
2. キャッシュを持つノード B からパケットの送信元 Consumer へ向けて Data パケットが返される (B→C→)

3. Data パケットを受け取ったノード C はコンテンツをキャッシングする

4. コンテンツがキャッシングされたことにより、ノード C にも拡張 CLS テーブルエントリが作成される

5. 拡張 CLS テーブルエントリの in の値が示す方向へノード C から HopCount パケットが $hc = 1$ で送信される (C→B)

6. HopCount パケットを受信したノード B は hc の値 1 とパケットを受信した Face 情報より自らの拡張 CLS テーブルエントリの out フィールドと h_{cache} のフィールドを更新し、対応するコンテンツのキャッシュを削除する

7. ノード B は HopCount の hc の値を 1 だけインクリメントし、拡張 CLS テーブルエントリの in の値が示す方向へ転送する (B→A)

8. HopCount パケットを受信したノード A は hc の値 2 で自らの拡張 CLS テーブルエントリの h_{cache} のフィールドを更新する

9. ノード A は HopCount パケットの hc の値を 1 だけインクリメントし、拡張 CLS テーブルエントリの in の値が示す方向へ転送する (A→P)

4 の時、ノード C の拡張 CLS テーブルエントリの h_{cache} の値は自らがキャッシュをもっていることにより 0 となる。6 の動作によりノード B の out と h_{cache} の値が更新され、キャッシュを持つノード C がノード B からホップ数 1 の場所にあることが示される。また 8 の動作によりキャッシュを持つノード C がノード A からノード B の方向へホップ数 2 の場所にあることが示される。8 の時は hc の値が 1 以外であるため、out の値は更新しない。

次にネットワークの状態が 22 である時に、ノード C のキャッシュ用のメモリ容量が埋まり、キャッシュが溢れて Content ID が C1 のキャッシュを削除しなければならない場合を考える (図 23)。

1. ノード C が持つ拡張 CLS テーブルの in が示すノード B に対して Return-back パケットが送信される

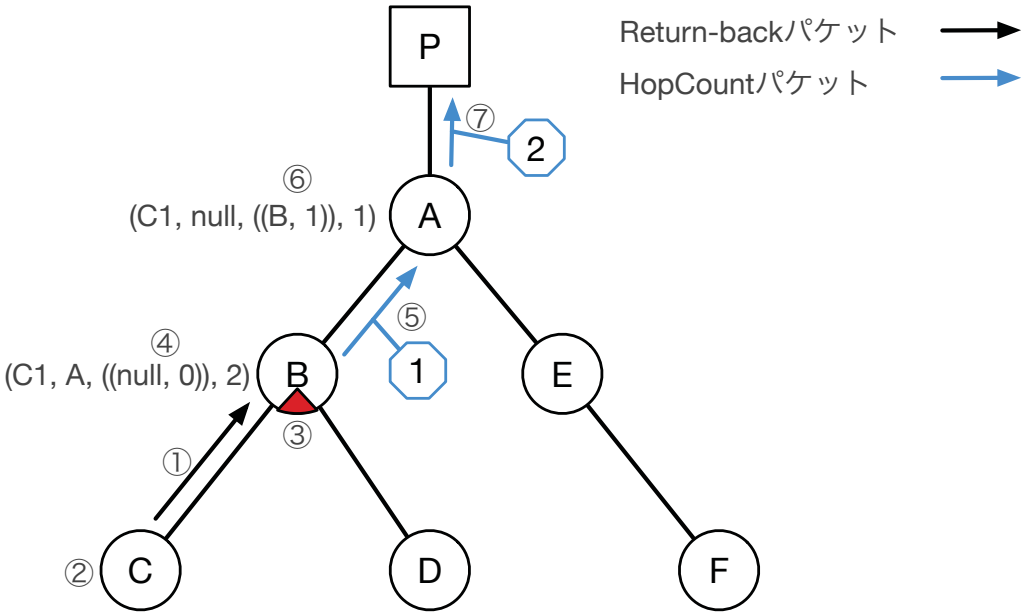

図 23　キャッシュが溢れた時 (提案手法)

2. Return-back パケットを送信したノード C から拡張 CLS テーブルエントリが削除される

3. Return-back パケットを受信したノード B へキャッシュが戻される Return-back パケットを受信すると受信したノード B へキャッシュが戻され、

4. ノード B の拡張 CLS テーブルエントリの out の値が null 値、h_{cache} の値が 0 となる

5. ノード B から拡張 CLS テーブルエントリの in の値が示す方向へ HopCount パケットが $hc = 1$ で送信される (B→A)。

6. HopCount パケットを受信したノード A は hc の値 1 で自らの拡張 CLS テーブルエントリの h_{cache} のフィールドを更新する

7. 最後にノード A は HopCount パケットの hc の値を 1 だけインクリメントし、拡張 CLS テーブルエントリの in の値が示す方向へ転送する (A→P)

6 の動作によりキャッシュを持つノード C が、ノード A からノード B の方向へホップ数 1 の場所にあることが示される。

最後にネットワークの状態が図 22 である時に、ノード B とノード C 間でネットワーク障害が発生し BC 間のリンクが切れる場合を考える (図 24)。この時、C がネットワークから切り離されるため、大元のネットワークからはキャッシュが存在しなくなる。このためノード A とノード B の拡張 CLS テーブルエントリの内容に誤りが生じる。この誤りを修正するために HopCount において $hc = -1$ の場合を、誤り修正のための特殊なパケットとした。

1. ノード B はノード C がネットワークから切り離されたことを検知すると誤りが生じている拡張 CLS テーブルエントリを削除する

2. ノード B は HopCount パケットを $hc = -1$ で拡張 CLS テーブルエントリの in の値が示す方向へ送信する (B→A)

3. $hc = -1$ の HopCount パケットを受信したノード A は誤りが生じている拡張 CLS テーブルエントリを削除

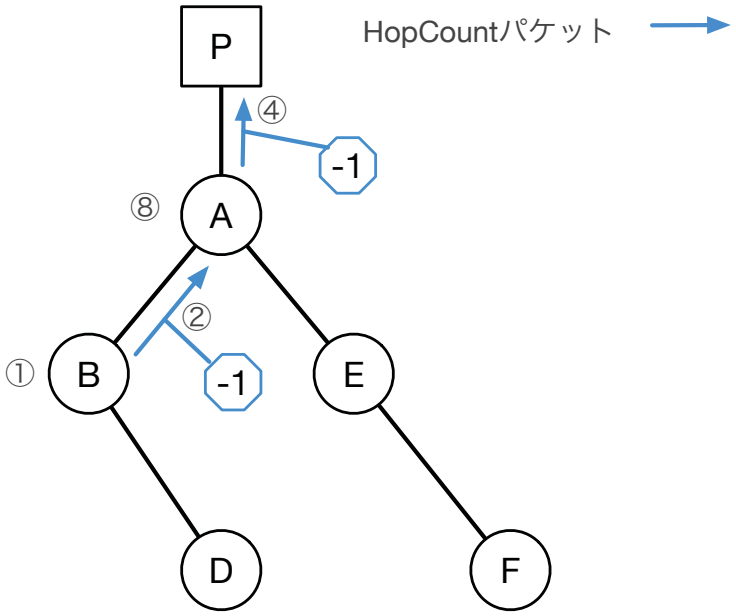

図 24　BC 間のリンクが切れた時 (提案手法)

4. ノード A は拡張 CLS テーブルエントリの in の値が指す方向へ HopCount パケットを転送する (A→P)

4 においてノード A は HopCount を転送しているが、$hc = -1$ の場合は例外として hc の値をインクリメントしない。

3.4 提案手法でのパケットの転送

提案手法ではパケットの転送を各ノードが持つ h の値と h_{cache} の値を比較して行う。パケット転送の規則は以下のようになっている。

キャッシュをもつノードへ転送する場合

$$h \geq h_{cache} \tag{4}$$

Publisher へ転送する場合

$$h < h_{cache} \tag{5}$$

3.5 提案手法の利点

拡張 CLS テーブルと HopCount パケットの定義により、CLS が問題としていた 3 つの問題が解決する。図 17 が示す CLS で問題が生じていたネットワークの状態について考える。図 17 を提案手法に置き換えたネットワークの状態を図 25 に示す。今、ノード F 以下にいる Consumer から Content ID が C1 のコンテンツに対して Interest パケットが送信されてきた場合を考える。

1. Interest パケットがノード B に到着 (→F→E→D→B)

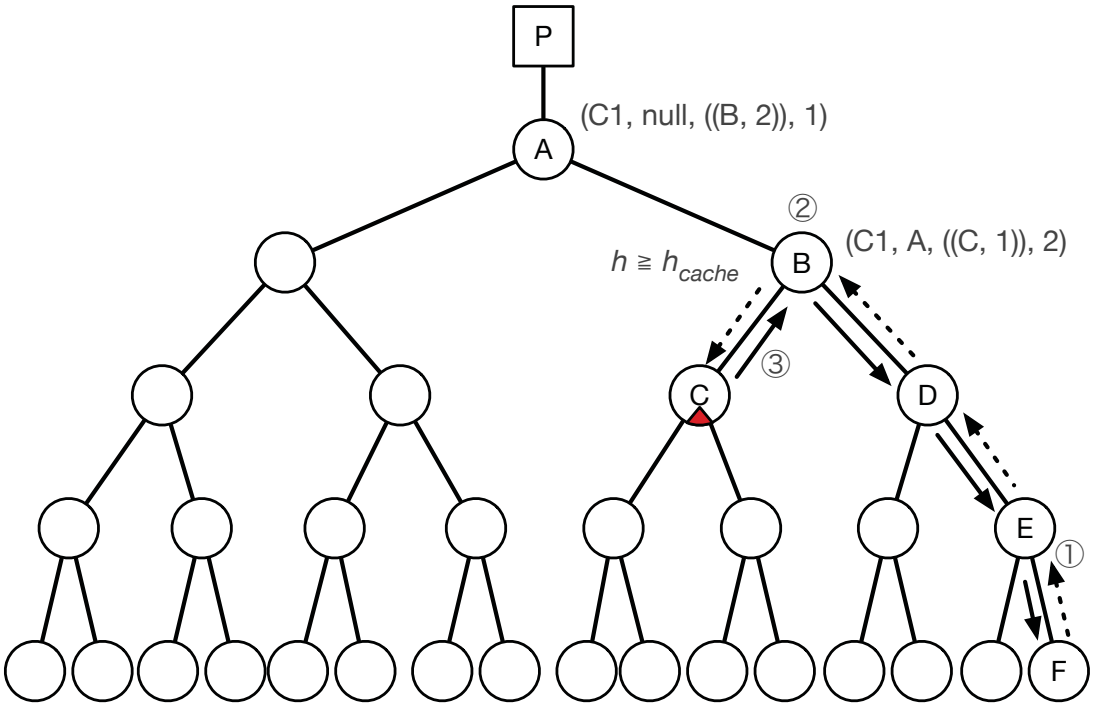

図 25 提案手法では全てのキャッシュが利用されている

2. ノード B は Interest パケットをキャッシュを持つノード C へと転送する (B→C)

3. ノード C から Interest パケットの送信元 Consumer へ向けて Data パケットが返される

2 の時にノード B は Interest パケットを Publisher へ転送するか、キャッシュをもつノード C へ転送するかを判断する。この時、ノード B の拡張 CLS テーブルエントリの h の値は 2 であり、h_{cache} の値は 1 である。よって式 (4) が成り立つためノード B は Interest パケットをキャッシュを持つノード C へと転送する (B→C)。このように提案手法では CLS の問題点であったノードが持つキャッシュの方へと Interest パケットを転送したほうが効率的にも関わらず Publisher の方へと Interest パケットが転送されてしまう問題が解決している。

更に提案手法では HopCount パケットの $hc = -1$ の場合を CLS テーブルの誤りを修正する特殊なパケットとして定義している。このためネットワーク障害が発生しリンクが切れた場合であっても CLS テーブルとキャッシュ間で整合性がなくなる問題が解決している。また HopCount パケットを受信して初めて out の値を更新するため、out の値を更新したものの、その先でキャッシングされない問題が解決しており、CLS テーブルの整合性が保たれる。

また提案手法では H_{th} の値の代わりに h_{cache} の値を用いて転送の判断を行っているため、各ノードへ入力する必要がなくなっている。

3.6 提案手法のまとめ

本章では拡張 CLS と HopCount パケットの定義により、CLS が抱えていた 3 つの問題点を解決できることを示した。第 4 章では実際に提案手法をシミュレータ上に実装し、提案手法と CLS を比較実験する。

4. 評価実験

本章では前章で述べた提案手法について評価実験をする。最初に 4.1 節で評価項目であるキャッシュヒット率、レスポンス時間、パケット数、メモリ消費量について説明し、各評価項目の必要性を述べる。次に 4.2 節で本実験の実験環境について述べる。続いて 4.3 節でキャッシュヒット率の評価、4.4 節でレスポンス時間の評価、4.5 節でパケット数の評価、4.6 節でメモリの使用量について評価結果を示す。

4.1 評価項目

本節では本提案で必要な評価項目について述べる。本論文で評価する項目は以下の 4 つである。

- キャッシュヒット率
- レスポンス時間
- パケット数
- メモリ消費量

キャッシュヒット率の評価によりネットワーク上に存在しているキャッシュの有用性を定量的に評価することができる。またキャッシュヒット率が向上すると Interest パケットの転送先が Publisher に偏ることを防ぐことができるため、負荷分散ができていることの指標にもなり得る。

レスポンス時間を評価することで Consumer から見た場合の提案手法の有用性を計測できる。レスポンス時間が短いほど Consumer は欲しいコンテンツを早く入手できるため、ユーザエクスペリエンスが向上すると言える。

パケット数を評価することでネットワークに掛かる負荷を計測できる。一般的にパケット数が多いほど、ネットワークの負荷は大きいと言える。

メモリ使用量を評価することでネットワーク上の各ノードの負荷を計測できる。CLS や提案手法では CLS テーブルや拡張 CLS テーブルをメモリ上に保持するた

め、実際のノードではメモリ量が有限であることを考えると、メモリ使用量が少ないほどより多くの CLS テーブルエントリを保持できる。

4.2 実験環境

実験を行った環境を表 6 にまとめる。実験環境として ndnSIM version 2.4.0[22][23] の上で提案手法を実装し実験を行った。ndnSIM は ns-3[24] と呼ばれるシミュレータのモジュールの 1 つとして動作する。

表 6 実験環境

シミュレータ	ns-3
シミュレータモジュール	ndnSIM version 2.4.0
シミュレーション時間 (シミュレータ時間)	1000 秒
トポロジー	6 段階の 2 分木
各ノード間接続の帯域	100Mbps
Publisher の提供するコンテンツ数	1000
1 コンテンツあたりの平均バイト数	100
Interest パケット送信頻度	10 パケット/秒
Interest パケット送信モデル	Zipf-Mandelbrot 法則 ($q = 0.7, s = 0.7$)

また Interest パケットの送信モデルに以下の式で表される Zipf-Mandelbrot 法則を適用している。

$$f(k; N, q, s) = \frac{1/(k+q)^s}{H_{N,q,s}}$$

$$H_{N,q,s} = \sum_{i=1}^{N} \frac{1}{(i+q)^s}$$

N : 要素数

k : ランク

s, q : 分散変数

Zipf-Mandelbrot の法則は言語学の分野で提唱された法則で、N 個の単語のうち出現頻度が k 番目に多い単語が文章全体に占める割合が $\frac{1}{k}$ に比例するというものである。この法則は Web のアクセス数に関しても適用できることが確認されている [25][26][27]。Web のアクセス数に対してこの法則を適用した場合、N 個のコンテンツのうち k 番目に多くアクセスされる Web サイトが全体のアクセス数に占める割合は $\frac{1}{k}$ であると言うことができる。Zipf-Mandelbrot 法則に基づいて Interest パケットを送信することで、コンテンツの要求頻度の偏りを再現した実験を行うことが可能となる。またトポロジーとして 6 段階の 2 分木を使用する。木のルートのノードを Publisher と、木の葉のノード群を Consumer として実験を行った。

4.3 キャッシュヒット率の評価

キャッシュヒット率を CCN (ndnSIM において標準的なキャッシュ管理アルゴリズムを用いたもの)、CLS、提案手法の 3 つで計測した。この結果、図 26 に示す評価結果が得られた。コンテンツ数に対するキャッシュ全体の大きさの割合に関わらず、提案手法のキャッシュヒット率が最も高く、次いで CLS、CCN となった。また、提案手法を適用することで CLS と比較して平均約 8.0%のキャッシュヒット率向上が可能となることが分かる。

4.4 レスポンス時間の評価

レスポンス時間を CCN、CLS、提案手法の 3 つで計測した。この結果、図 27 に示す評価結果が得られた。コンテンツ数に対するキャッシュ全体の大きさの割合に関わらず、提案手法のレスポンス時間が最も短く、次いで CLS、CCN となった。また、提案手法を適用することで CLS と比較してレスポンス時間が平均約 76.1%に短縮されることが分かる。

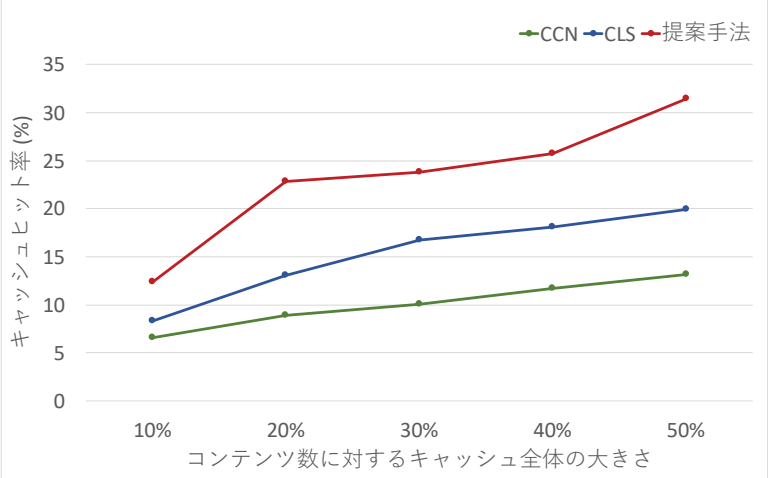

図 26　キャッシュヒット率の評価結果

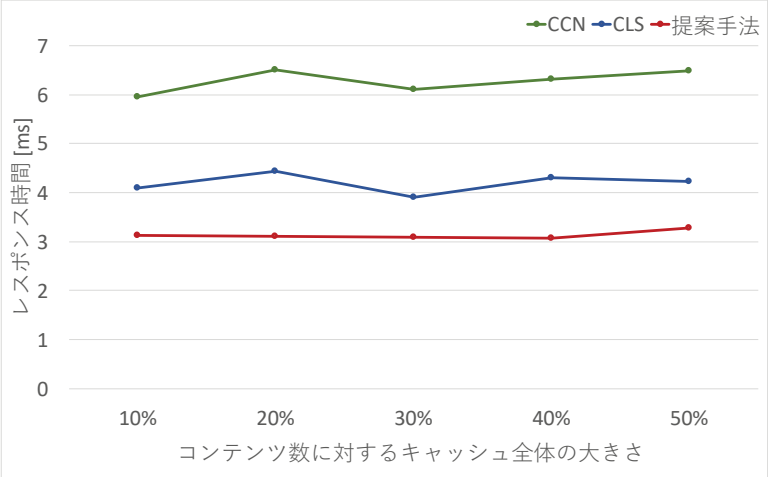

図 27　レスポンス時間の評価結果

4.5 送信されるパケット数の評価

提案手法で新しく定義したHopCountパケットが送信されるタイミングは以下の2通りである。

1. Interestパケットが送信され、キャッシュが移動した時

2. ノードが持つキャッシュが溢れ、キャッシュが移動した時

これらのうち1についてはInterestパケットが1個送信される度に起こるため、Interestパケットが1個送信される度にHopCountパケットも1個送信される。よって、Interestパケットが送信される回数をiとすると、HopCountパケットが送信される回数もiと表される。また2に関してはキャッシュが溢れる回数をoとすると、HopCountパケットが送信される回数もoと表される。以上より、HopCountパケットが送信される回数は$i+o$と表され、提案手法ではCLSと比較して$i+o$個だけ多くパケットが送信される。

4.6 メモリ消費量の評価

提案手法の拡張CLSテーブルはC++言語の構造体、連結リストとして管理している。拡張CLSテーブルを管理することによる、あるコンテンツに対する1ノードあたりの最大のメモリの使用量 (bytes) は以下の式で表される。

$$l + f + (f + i) \times p + i$$

l : $ContentID$ の長さ

f : $Face$ 構造体へのポインタの大きさ

i : 整数型の大きさ

p : 自ノードを通る $Publisher$ と $Consumer$ を繋ぐ経路の数

今回、提案手法を実装した環境では整数型の大きさiが4bytes、Face構造体へのポインタの大きさfが4bytesであるため、上の式は以下のように直される。

$$l + 8p + 8$$

よって、コンテンツ数が n の時に 1 個のノードあたりのメモリ使用量は以下のようになる。

$$nl + 8pn + 8n \tag{6}$$

また CLS の CLS テーブルは C++言語の構造体として管理している。CLS テーブルのメモリ使用量 (bytes) は次の式で表される。

$$l + f + f \times p + i$$

f と i については拡張 CLS テーブルと同様に 4bytes なので、上の式は次のように直される。

$$l + 4p + 8$$

よって、コンテンツ数が n の時に 1 個のノードあたりのメモリ使用量は以下のようになる。

$$nl + 4pn + 8n \tag{7}$$

2 つの式 6、7 の比較により、提案手法は CLS と比較して最大で $4pn$bytes だけメモリ使用量が増加することが分かる。

5. 考察

本章では4章で示した結果について考察し、本提案の有用性について議論する。

5.1 キャッシュヒット率の考察

コンテンツ数に対するキャッシュ全体の大きさの割合に関わらず提案手法、CLS、CNNの順にキャッシュヒット率が高くなり、平均で約8.0%のキャッシュヒット率向上が可能となった。これは$h < H_{th}$の場合に一部のキャッシュが活用されない問題が解決したためであると考えられる。この問題が解決しInterestパケットがPublisherに向け転送するよりも効率の良いノードへと転送されることになったことから、レスポンス時間の短縮が期待できる。またInterestパケットがPublisherに集中しなくなり、キャッシュをもつノード群へと分散するようになったことによりパケットを処理するための負荷を分散することにも繋がると考えられる。

5.2 レスポンス時間の考察

コンテンツ数に対するキャッシュ全体の大きさの割合に関わらず提案手法、CLS、CNNの順にレスポンス時間が短くなり、平均で約76.1%のレスポンス時間短縮が可能となった。これはキャッシュヒット率が向上し、より効率の良いノードへとInterestパケットが転送されるようになったためと考えられる。今回の実験は1コンテンツあたり約100bytesであるとして行ったが、現実の環境では1個1Mbytesのコンテンツが10個100bytesのチャンクに分割される場合等があり得る。この場合は100bytesのチャンクを10個取得して1コンテンツが取得できたことになるため、1チャンクあたりのレスポンス時間が短縮されるほど、1コンテンツあたりのレスポンス時間は大きく短縮される。特に大容量のコンテンツは多くのチャンクに分割されると考えられるため、より大容量のコンテンツを要求する時にレスポンス時間の短縮が有利に働くと考えられる。レスポンス時間が短縮されたことによりConsumerはより早くコンテンツを入手することができるようになるため、Webページ等を閲覧する時のユーザエクスペリエンスの向上が期待される。

Web ページの読み込みに掛かるまでの時間はユーザエクスペリエンスに対して秒単位で影響する [28]。このためサイズの大きい Web ページを読み込む際に提案手法を用いることでユーザエクスペリエンスを向上させることが可能となる。

5.3 送信されるパケット数の考察

提案手法では CLS と比較して $i + o$ 個 (i: Interest パケットの送信される回数、o: ノードからキャッシュが溢れる回数) だけ多くパケットが送受信される。ここで o の値の変化を考察するため、キャッシュ溢れが増大する要因を考える。キャッシュ溢れはキャッシュが入れ替わる時に起こり得る。キャッシュの入れ替わりは、コンテンツ要求がある 1 つのコンテンツに偏るほど起き難くなり、逆にコンテンツ要求が全てのコンテンツに対して一様に分布するほど起き易くなる。よって o の値はコンテンツの要求頻度の偏りに依存し、偏りが大きいほど o の値は小さくなり偏りが小さいほど o の値も大きくなると考えられる。以上より、送信されるパケット数はコンテンツ要求頻度の偏りに依存すると考えられる。

5.4 メモリ使用量の考察

提案手法では CLS と比較して最大 $4pn$bytes (p: 自ノードを通る Publisher と Consumer を繋ぐ経路の数、n: コンテンツ数) だけ 1 ノードあたりのメモリ使用量が増加する。p の値は自ノードより下流のトポロジーでの分岐点の数に依存する。分岐点が多いトポロジーの場合 p の値が大きくなり、分岐点が少ないトポロジーの場合 p の値が小さくなる。これは自ノードより下流のノードで保持しているキャッシュの情報を全て把握するためである。下流のノードで保持しているキャッシュのうち、もっともネットワーク上の距離が近いもの x 個のみを把握するようにするといった方法で p の値を定数 x に固定することは可能であるものの、x 個のキャッシュが全て失われると、存在はしているにも関わらず CLS テーブルから参照されないキャッシュが発生するといったデメリットがあり、メモリ使用量とキャッシュ利用効率のトレードオフと言うことが出来る。自ノードから見た時に下流に存在できるキャッシュの数についても x 個で制限することで、キャッ

シュが参照されないことを防ぐことができるが、これはそもそもネットワーク上に存在できるキャッシュの数を制限することであるため、キャッシュの利用効率を保ちつつメモリ使用量の増加を防ぐためにはネットワークのトポロジーによって最適な x の値を設定する必要がある。最適な x の値を設定することが可能であれば、メモリ使用量を抑えつつ、キャッシュの利用効率を保つことが可能となると考えられる。

5.5 今後の課題

提案手法の木構造以外のネットワークトポロジーへの適用と実ネットワークへの適用が今後の課題である。今回の実験においては先行研究と同様に 2 分木型トポロジーで実験を行ったが、メッシュ型トポロジーを始めとした多様なトポロジーでの実験も行う必要がある。メッシュ型などの複雑なトポロジーにおいては、提案手法や先行研究で用いたような CLS テーブルではキャッシュの位置を正確に把握することが困難となる可能性があるため、CLS テーブルに対して更なる情報を加えていく必要性があると考えられる。また今回の実験においては CLS テーブルの保持可能なエントリ数に制限を設けなかったが、実ネットワークに適用する場合はメモリ容量の制約が存在するため、メモリ使用量の制限値を検討する必要がある。情報指向ネットワークの実ネットワークへの適用は Cisco Systems 社や Palo Alto Research Center 社が取り組んでいる [29][30] が、ネットワーク機器などが対応できていないことから実例は少ない。今後、ネットワーク機器ベンダーによって情報指向ネットワークに対応したネットワーク機器が提供されることが期待されるが、こうした機器のスペックに応じてメモリ使用量の制限値を考えていく必要がある。

6. 結論

インターネットの利用形態の変化に伴い、既存の IP ネットワークを置き換える技術として情報指向ネットワークが注目され始めている。しかし情報指向ネットワークは未だ研究段階の技術であり、実用化にあたっては解決しなければならない問題が数多く存在している。

情報指向ネットワークにおけるキャッシュ利用効率向上手法である CLS には、一部のキャッシュが活用できない、パケットを転送するためのテーブルに不整合が発生してしまう、ネットワークの管理コストが大きいといった問題があった。

そこで本論文では CLS で用いられたテーブルを拡張し、新しいパケット Hop-Count パケットを定義することで CLS の問題点を解決する方法を提案した。さらにシミュレータ上で提案手法を実装と数値計算により、キャッシュヒット率、レスポンス時間、送受信されるパケットの数、メモリ使用量について評価を行った。その結果、キャッシュヒット率が平均約 8.0%の向上、レスポンス時間は平均約 76.1%にまで短縮が可能となった。今後の課題として提案手法の実ネットワークへの適用が考えられる。

謝辞

主指導教員であり、適切な研究指導をしていただくなど様々な側面から研究のサポートをしてくださいました本学情報基盤システム学研究室の藤川和利教授に心から感謝致します。副指導教員であり、研究の方向性について的確な助言をくださいました本学サイバーレジリエンス構成学研究室の門林雄基教授に心から感謝致します。副指導教員であり、研究指導、論文執筆に当たり熱いご指導をしていただきました本学情報基盤システム学研究室の新井イスマイル准教授に心から感謝致します。研究方針や論文執筆に関して数々のご助言をくださいました本学情報基盤システム学研究室の垣内正年助教に心から感謝致します。研究活動や論文執筆に関してご指導をいただきました本学情報基盤システム学研究室の油谷曉助教に心から感謝致します。研究面や生活面において多くの支援をしていただいた本学博士後期課程修了生であり現東京工業大学情報理工学院の石井将大特任助教に心から感謝致します。様々な面から研究活動を支援してくださいました本学総合情報基盤センターの辻元理恵女史と中野彩子女史、元本学総合情報基盤センターの鷲尾多佳子女史に心より感謝致します。様々な面から研究活動を支えてくださり、共に切磋琢磨し合った本学情報基盤システム学および本学サイバーレジリエンス構成学研究室の皆様に心より感謝致します。最後に経済面や生活面において支援していただきました家族に心より感謝致します。

参考文献

[1] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. Networking named content. In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '09, pp. 1–12, New York, NY, USA, 2009. ACM.

[2] 電気通信役務放送法に基づいた IPTV サービスの動向. https://iwparchives.jp/files/pdf/iwp2009/iwp2009-ch03-02-p112.pdf, 2018 年 1 月 29 日アクセス.

[3] 総務省. 諸外国の動向等について ～動画配信サービスを中心に～. http://www.soumu.go.jp/main_content/000449239.pdf, 2018 年 1 月 29 日アクセス.

[4] TV 局系 VOD のスマホからの利用者数は 500 万人超、TVer は約 250 万人～ニールセン、日本のビデオオンデマンドのスマホ、PC からの利用状況を発表～. http://www.netratings.co.jp/news_release/2016/01/Newsrelease20160126.html, 2018 年 1 月 29 日アクセス.

[5] Cisco visual networking index: Forecast and methodology, 2016–2021. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html, 2018 年 1 月 30 日アクセス.

[6] Mobile traffic analysis by application. https://www.ericsson.com/en/mobility-report/reports/november-2017/mobile-traffic-analysis-by-application, 2018 年 1 月 30 日アクセス.

[7] 我が国のインターネットにおけるトラヒックの集計結果 (2016 年 11 月分). http://www.soumu.go.jp/main_content/000462459.pdf, 2018 年 1 月 30

日アクセス.

[8] 国内外のトラヒック予測及びその周辺動向のご紹介. http://www.soumu.go.jp/main_content/000467644.pdf, 2018 年 1 月 30 日アクセス.

[9] IPTV の現状と課題. http://www.soumu.go.jp/main_content/000455126.pdf, 2018 年 1 月 29 日アクセス.

[10] Teemu Koponen, Mohit Chawla, Byung-Gon Chun, Andrey Ermolinskiy, Kye Hyun Kim, Scott Shenker, and Ion Stoica. A data-oriented (and beyond) network architecture. In *Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '07, pp. 181–192, New York, NY, USA, 2007. ACM.

[11] Psirp. http://www.psirp.org/, 2018 年 1 月 30 日アクセス.

[12] Netinf. http://www.netinf.org/, 2018 年 1 月 30 日アクセス.

[13] Named data networking. https://named-data.net/, 2018 年 1 月 30 日アクセス.

[14] 阿多信吾. 現実味を帯びてきた ICN/CCN の研究動向. http://www.ieice.org/~ia/archives/20130906-ICNCCN.pdf, 2018 年 1 月 30 日アクセス.

[15] Ioannis Psaras, Wei Koong Chai, and George Pavlou. Probabilistic in-network caching for information-centric networks. In *Proceedings of the Second Edition of the ICN Workshop on Information-centric Networking*, ICN '12, pp. 55–60, New York, NY, USA, 2012. ACM.

[16] Z. Li and G. Simon. Time-shifted tv in content centric networks: The case for cooperative in-network caching. In *2011 IEEE International Conference on Communications (ICC)*, pp. 1–6, June 2011.

[17] Nikolaos Laoutaris, Hao Che, and Ioannis Stavrakakis. The lcd interconnection of lru caches and its analysis. *Perform. Eval.*, Vol. 63, No. 7, pp. 609–634, July 2006.

[18] K. Cho, M. Lee, K. Park, T. T. Kwon, Y. Choi, and Sangheon Pack. Wave: Popularity-based and collaborative in-network caching for content-oriented networks. In *2012 Proceedings IEEE INFOCOM Workshops*, pp. 316–321, March 2012.

[19] Theodore M. Wong and John Wilkes. My cache or yours? making storage more exclusive. In *Proceedings of the General Track of the Annual Conference on USENIX Annual Technical Conference*, ATEC '02, pp. 161–175, Berkeley, CA, USA, 2002. USENIX Association.

[20] Jason Min Wang, Jun Zhang, and Brahim Bensaou. Intra-as cooperative caching for content-centric networks. In *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric Networking*, ICN '13, pp. 61–66, New York, NY, USA, 2013. ACM.

[21] Y. Li, T. Lin, H. Tang, and P. Sun. A chunk caching location and searching scheme in content centric networking. In *2012 IEEE International Conference on Communications (ICC)*, pp. 2655–2659, June 2012.

[22] ndnsim. `https://github.com/named-data-ndnSIM/ndnSIM`, 2018 年 1 月 30 日アクセス.

[23] ndnsim documentation. `http://ndnsim.net/2.4/`, 2018 年 1 月 30 日アクセス.

[24] ns-3. `https://www.nsnam.org/`, 2018 年 1 月 30 日アクセス.

[25] Monika Jauhari, Anurag Saxena, and J. N. Gautam. Zipf's law and number of hits on the world wide web. *Annals of Library and Information Studies*, Vol. 54, pp. 81–84, June 2007.

[26] Lada A. Adamic and Bernardo A. Huberman. Zipf's law and the internet. *Glottometrics 3.1*, pp. 143–150, 2002.

[27] Bernardo A Huberman. *The laws of the Web: Patterns in the ecology of information*. MIT Press, 2011.

[28] Does Page Load Time Really Affect Bounce Rate? `http://royal.pingdom.com/2018/01/18/page-load-time-really-affect-bounce-rate/`, 2018年1月30日アクセス.

[29] ネットワークインフラ技術の進展 -digital時代のネットワークアーキテクチャ. `http://www.soumu.go.jp/main_content/000477966.pdf`, 2018年1月30日アクセス.

[30] The CCNx Project. `https://blogs.parc.com/ccnx/`, 2018年1月30日アクセス.

発表リスト

国内研究会

木村一統、新井イスマイル、藤川和利、情報指向ネットワークにおけるキャッシュ利用効率向上のための位置検討、第 10 回 ICN 研究会 2017