URL の情報指向型クラシフィケーション

指導教員：井上 一成　発表者：E1533 西 総一朗

1　はじめに

近年，ネットワーク上を流通するトラフィックは増加の一途を辿っている．また Internet of Things (IoT) の普及によりネットワークに参加する機器も増加する．そのため，それぞれの機器を IP アドレスで管理するネットワークでは破綻が生じるという課題がある．

本来情報を取得する際，IP アドレスなどを意識する必要はなく，もし近くにある通信機器が当該コンテンツ (情報) を持っておりそこから情報を取得できるなら，それはより効率的であり将来の通信量増大にも対応できると考えられる．そこで，情報を効率的に取得するために情報指向ネットワーク: Information-Centric-Network (ICN)[1] というプロトコル体系が提案された [2].

2　ICN の課題

情報指向ネットワーク (ICN) においてユーザはサーバの IP アドレスではなくコンテンツ名を指定してコンテンツ取得要求を行うプロトコル体系である．また，情報を保持している者をパブリッシャ (Publisher)，情報の取得要求を出すものをサブスクライバ (Subscriber) と呼ぶ．Subscriber はコンテンツ取得要求である INTEREST パケットを発行して，Publisher からの DATA パケットによりコンテンツを取得する．すべてのパケットは Content Router (CR) によって転送される．各 CR はルーティングテーブルが存在し，コンテンツ名と宛先インターフェイス名の対応を管理している．コンテンツ名の命名規則は階層構造になっており，現在のインターネットで流通している識別子である Uniform-Resource-Locator (URL) に似ている．CR ではコンテンツ名を既存のハッシュアルゴリズムでハッシュ化してルーティングテーブルに登録しているが，このアルゴリズムをハードウェアで実装するには処理が複雑である．このことが ICN の実現化に向けた大きな課題の一つとなっている．

3　本研究の目的

本研究では，上記課題を解決するために新たなデータ構造による検索手法と高速で軽量なハッシュアルゴリズムを提案，検証することである．コンテンツ名はランダムな文字列ではなくある程度自然言語的な規則があるのでそれを用いることで軽量化を図る．

4　URL の構造

現在使用されている URL は図 1 のような構造である．URL を分類するために Top-Level Domain(TLD) を用いる手法がよく使われるが, 一つの分類に含まれる範囲が広すぎる． そこで, 今回提案する手法は eTLD(effective TLD) あるいは Public Suffix[3] と呼ばれる実質的に TLD として機能する指標を用いて分類する．

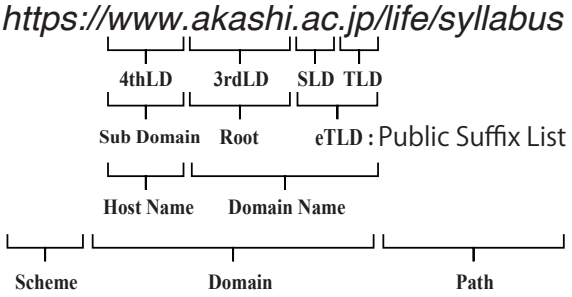

図 1　URL の構造

5　ICN のコンテンツ名

ICN におけるコンテンツ名を本研究では

```
icn:/<reTLD>/<Root>/<rHostName>/<Path>
```

のように定義し，ICN-URL と呼ぶ．`reTLD` (reverse-eTLD) と `rHostName` (reverse-HostName) はそれぞれ `eTLD` と `HostName` を”.”を区切りとして逆順に配置したものである．すなわち，`eTLD` が `ab.cd.ef` なら `reTLD` は `ef.cd.ab` となる．

6　ハッシュアルゴリズム

まず，ICN-URL を”/”で分割する．それぞれをセクションと呼ぶ．そのセクションの文字数が 3 文字未満の場合はセクションの文字数に応じて次のように 3 文字にする (パディング).

セクションが 1 文字のとき　ICN-URL の長さとスラッシュの数を掛けたものを uint16 型で 2 バイト付加する

セクションが 2 文字のとき　ICN-URL のスラッシュの数を byte 型として 1 バイト付加する

次に，各セクションから前 3 文字を抜き出して配列 heads とする．同様に後 3 文字を抜き出して配列 tails

とするが，パティングが含まれているセクションはパディングと元の文字との順序を入れ替える．

最後に，先程の heads と tails を用いてハッシュ値を求める．heads の先頭要素 3 バイト，tails の末尾要素 3 バイト，末尾から 3 つ目の要素 3 バイトの各 3 バイト，計 9 バイトをそれぞれのバイトごとに XOR を計算して 3 バイトにする．heads の先頭 1 文字と先程の 3 バイトを連結したものを 4 バイトのハッシュ値とする．ICN-URL を `icn:/jp.ac/akashi/www/life/syllabus/a` としたときの具体例を図 2 に示す．

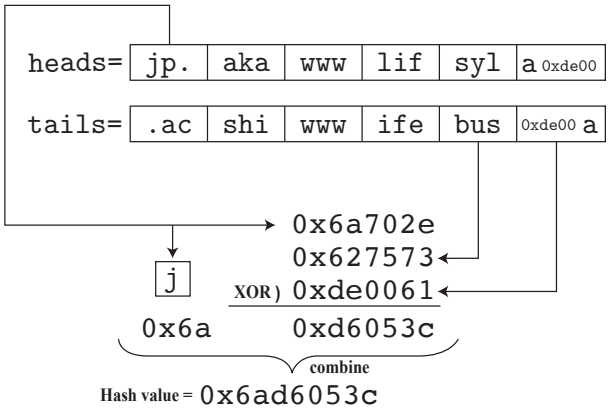

図 2　ハッシュアルゴリズム

7　性能評価

図 3 のようにバーゼル大学で公開されている 21 億件の URL のリスト [4] から重複を取り除いたものを全 URL リストと呼ぶ．これは 60GB ほどのサイズで約 8.8 億件の URL を含む．各 CR での実際的な URL は 10MB ほどであると仮定し，ランダムな 10MB 分を抽出し，解析データとした．これには約 14 万件の URL が含まれる．この解析データ中の各 URL を ICN-URL に変換する．

解析データから同じ eTLD ごとにハッシュを計算してテーブルを作成し, それを eTLD の頻度順に並べたものをハッシュテーブル②とする．ハッシュテーブルの各 eTLD の始まりのアドレスをポインターとしてポインターテーブル①に記録する．

図 3 のポインタテーブル①を用いることで，異なる eTLD は区別されるので 1 つの eTLD に対するハッシュ値の衝突だけが問題となる．したがって各 eTLD についてだけハッシュ値の衝突率を求めればよい．しかし，表

表 1　解析データ中の上位 2 件の eTLD の出現確率

eTLD	URL count	Probability[%]
com	89399	60.6856
net	8285	5.6240

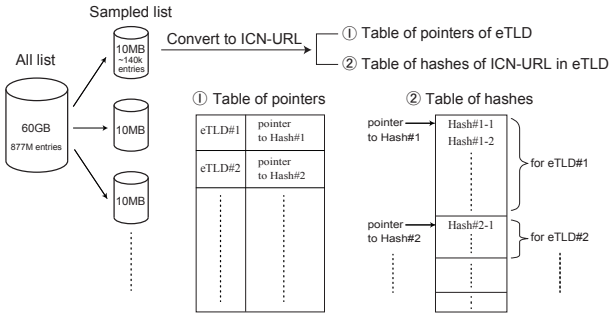

図 3　評価手順

1 に示すように，出現確率 1 位の com で 60% を占めており，極端に偏っていることがわかる．そのため，eTLD に Root を加えることを考える．

図 4 に eTDL に Root を加えたときと，加えてないときのハッシュ値の衝突率の比較を示す．グラフから `com`, `net` の衝突率が eTLD に Root を加えることで低減されていることがわかる．

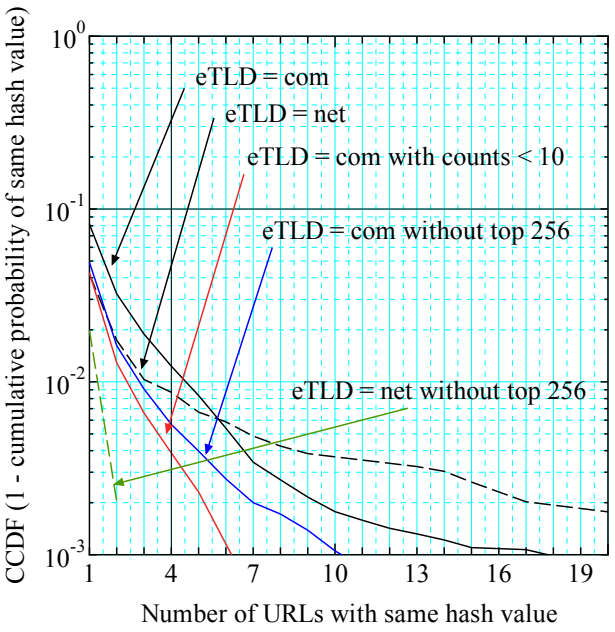

図 4　eTLD ごとのハッシュ値が同じ URL の数の相補累積分布

8　まとめ

このハッシュアルゴリズムのような 3 回の XOR を行うだけの軽量な計算負荷で求まる 4 バイトのハッシュ値でも，ポインタテーブルを併用することにより 5 回以上の衝突確率を 0.3% に抑えることができた．

参考文献

[1] Jacobson, Van et.al. : Networking Named Content (2009).

[2] 朝枝 仁, 松園 和久 : 情報指向ネットワーク技術におけるプロトタイプ実装と評価手法 (2016)

[3] Mozilla Foundation. : Public Suffix list , Feb. (2020).

[4] University of Basel. : The content name collection, Oct. (2019).