

URLの情報指向型クラシフィケーション

西総一朗 * , 井上一成 (明石高専)

Information-oriented classification of URLs
Soichiro Nishi*, Kazunari Inoue (NIT, Akashi Collage)

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リーバ (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。

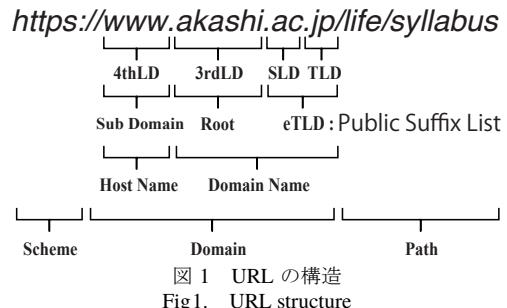
3. 本研究の目的

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。



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、パデ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。

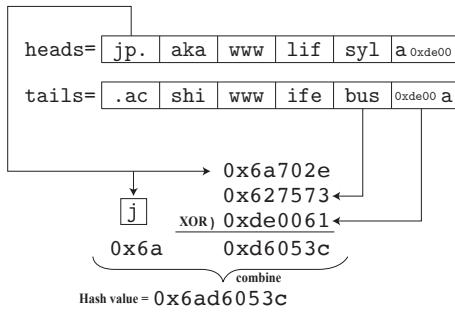


図2 ハッシュアルゴリズム
Fig2. Hash algorithm

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のポインターテーブル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

表1 解析データ中の上位2件のeTLDの出現確率
Table1. Top 2 of eTLD, URL count and probability in the sampled URLs.

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

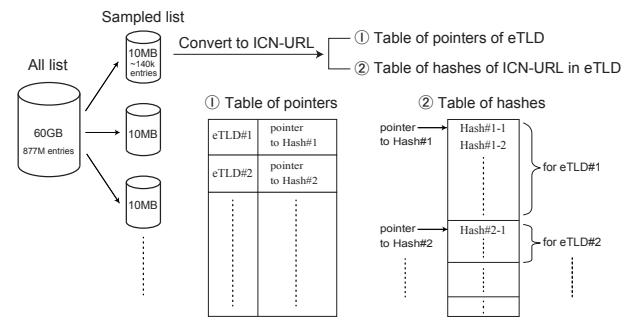


図3 評価手順
Fig3. Analysis procedure.

てだけハッシュ値の衝突率を求めるべきだ。しかし、表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eTLD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。

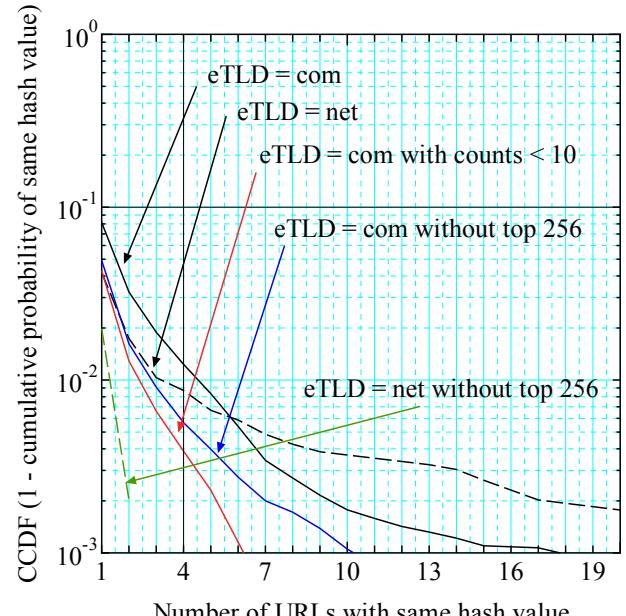


図4 eTLDごとのハッシュ値が同じURLの数の相補累積分布
Fig4. Complementary cumulative distribution function of same hash value for each eTLD.

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、ポインター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。

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).