SIG-SWO-A801-03

URI Context Databaseの提案

Proposal of URI Context Database

亀田 尭宙[*1] 大向 一輝[*2*3] 武田 英明[*1*2]
KAMEDA Akihiro OHMUKAI Ikki TAKEDA Hideaki

[*1]東京大学 The University of Tokyo
[*2]国立情報学研究所 National Institute of Infomatics
[*3]総合研究大学院大学 The Graduate University for Advanced Studies

Abstract: URL は現在のウェブにおいて使用される URI として重要な役割を担っているが，URL が示す「どこ」は常に同じコンテンツを指すわけではない．このことは URL のような URI を識別子として扱うメタデータとその応用技術の機能にも不完全さをもたらす．そこで URI に「いつ」「だれが」「どのように」アクセスしたかというコンテキストを付随させ，URI とコンテンツをハッシュ関数によって一対一対応に結びつけて記録し，その情報をデータベースとして提供することで，リソースの確かな同定および異なった場合の処理の支援を行うことを提案する．

1. はじめに

機械可読なかたちで記されたウェブリソースなどの情報はデータに関するデータという意味でメタデータと呼ばれ，情報検索の効率化やコンテンツ合成履歴の記録手段，権利関係の記述手段などさまざまな形での利用が期待されている [1]．また，Resource Description Framework(RDF)[2] といったメタデータ記述のための枠組みが作られており，これに基づいて実際に記述することができる．RDF は主語 (Subject) と述語 (Predicate) と目的語 (Object) の三つ組みで主語と目的語の間に述語の関係があることを表現するが，それぞれの要素は URI 参照と呼ばれる形で記述される[*1]．URI 参照は rfc3986[3] に示されているように，特定の種類の文字列の並びでリソースを指示する Uniform Resource Identifier(URI) においてフラグメント識別子#を使った表現を含むものであり，現在最もよく使われている URI 参照には Uniform Resource Locator(URL) がある．URI の指示するものはウェブページだけでなく概念や実在の物などでも良いが，Berners-Lee による Linked Data の構想では，URL を URI として使うことでユーザがリソースについて有用な情報が得られるようにすることを勧告している [4].

本論文では，このような状況の中で，URI とそれが示すコンテンツが一対一対応していないことによる問題を２章で述べ，その解決策としての URI Context Database を３章で提言し，ハッシュ関数を用いることを３章で提言し，４章で有効性について述べた後，５章で結ぶ．またこれ以降は URI という語を用いるにあたって，以上のような背景に基づいて，HTTP URIs（つまり URL）のように，指示されたウェブ上のリソースにアクセスしコンテンツを取得できる URI を想定する．

2. URIによる指示の問題点

URI によってリソースを指示する一般的なメタデータの利用例とその問題点を以下に３つ挙げる．

1. A，B，C，D，Eの５人がリソースXに対してA，B，Cは「信頼できない」とD，Eは「信頼できる」と評価を下していたので，PはXをあまり信頼できないと判断した．

   しかし，実はA，B，Cが評価した後にXの情報は訂正され，訂正された情報に対してD，Eが評価を下していた．その後，Pが見た時には十分に正しい内容だった．

2. A，B，C，D，Eの５人がリソースYに対してA，B，Cは「有用である」とD，Eは「有用でない」と評価を下していたので，PはYをそれなりに有用であると判断してアクセスした．

   しかし，実はアプリケーション内の２つのグループによって表示されている情報の範囲が異なっていたため，D，Eと同じグループに属していたPは全く有用でない情報を見て落胆した．

3. Pはメタデータを利用した検索エンジンでリソース Z_0 に辿り着いたが，求めていた情報はなかった．

   それは，実はリソース $Z_1 \rightarrow Z_0$ と辿ることによって得られる情報が検索エンジンに登録されていたためであって，この経路でリソースにアクセスしないとサーバ側に溜め込まれたセッション情報が異なってしまうために求めていた情報を得ることができなかったのである．

以上に述べたような問題を解決するには URI とコンテンツが一意に対応していることを保障する必要がある．しかし，コンテンツそのもの URI に対応させてメタデータ内に逐一保存するのは記録する際のスケーラビリティの面から非現実的である．また，著作権の問題や個人情報保護の問題などがあるため全ての情報についてコンテンツ自身を保存できるわけではない．

スケーラビリティに関しては，記録の網羅性を諦めれば可能であり実際にそういったサービスは存在する．ウェブ魚拓[*2]やInternet Archive[*3]のようにコンテンツそのものを保存するサービスも存在し，それらでは過去のコンテンツの状態を参照することができる．それぞれ記録するコンテンツを前者はユーザによる登録，後者は自動的なクロールという形で限っており，著作権や個人情報の問題については人手で対応している．

連絡先: 亀田 尭宙, 東京大学大学院新領域創成科学研究科, 千葉県柏市柏の葉 5-1-5, 04-7136-4275, kameda@envd.org

∗1 但し目的語だけは文字列自身（リテラル）も許される

∗2 http://megalodon.jp/

∗3 http://www.archive.org/web/web.php

しかし，ある URI の範囲内においては記録に網羅性があることにも重要な意義がある．例えば 2 章の 1 つ目のケースにおいて「A，B，Cが評価した後にXの情報は訂正され，訂正された情報に対してD，Eが評価を下していた」わけではなくそれぞれの人の評価の間に訂正が入っていた場合はまた信頼性の推量が異なってくることになるだろう．

3. URI Context Database の提案

本論文では，URI へのアクセスのコンテクストをデータベースとして記録し，そのコンテクストでアクセスした際のコンテンツのハッシュ値を共に記録し同定に用いることで，前章で述べた URI とコンテンツの一対一対応を実現することを提案する．そして，このデータベースを URI Context Database と呼ぶ．ユーザとサービスと別の第三者が URI Context Database を保持し，記録する URI とドメインを同じくするアプリケーションから送られてきた情報に対してハッシュ値を記録し，以下に述べるようなコンテクストと共に記録することで，URI Context Database 自身についての一定の信頼性を保障する．外部のアプリケーションには情報の投稿（Create）と読み出し（Read）のみが許されており，修正（Update）や削除（Delete）は認められていない．必要に応じて，アプリケーションがユーザにも同じ情報を送り，ユーザ側で URI Context Database との照合を行うことでより一層信頼性を増すこともできる．

また，アプリケーション側は先に述べた網羅性を高めるために，アプリケーション内のデータベースに変更を加えるようなリクエストに際し，変更を加えられたリソースについて URI Context Database に登録するようにする，もしくはユーザのアクセスごとに URI Context Database に登録するようにする，といった仕組みをアプリケーションに組み込むことが期待される．

以下，コンテンツのハッシュ値とアクセスする際のコンテクストについて詳しく述べる．

3.1 URI と コンテンツ

コンテンツのハッシュ値を得るためのハッシュ関数としては，具体的には Secure Hash Algorithm（SHA）[5] のようなアルゴリズムを想定している．インターネット上で情報を暗号化して送受信するプロトコルである Secure Sockets Layer（SSL）などにも使われている関数で，元のデータ列とそれをハッシュ関数にかけて得られたハッシュ値が一対一対応になることが強く求められている．また，ハッシュ関数から元のデータ列を得られてはならない．同じハッシュ値を持つ 2 つのデータ列が発見されることを強衝突耐性の突破と呼び，Message Digest Algorithm 5（MD5）[*4]は強衝突耐性が突破されている [6] が SHA のうち SHA256 や SHA512 は現在のところ強衝突耐性は突破されていない．このような強力な関数を用いて一対一対応を保障することで，非可逆性によって著作権や個人情報などの権利に抵触せずに，コンテンツを同定可能にすることができる．

また，対象のデータは HTML や XML のようなテキストコンテンツだけではなく PDF や JPEG などのバイナリデータでも扱え，非常に汎用性がある．

3.2 URI と コンテクスト

同じ URI にもかかわらずコンテンツが異なる要因には以下のようなものが考えられる．

1. アクセスした時間 (**When**) が異なり，その間にコンテンツが変更されている．
2. ユーザ (**Who**) でコンテンツの内容が制限もしくはパーソナライズされている．
3. URI のみではアクセスできず，どのように (**How**) 操作したら同じ情報に辿りつけるかという情報がさらに必要である．

よって，URI のコンテクストとして，これら **When**，**Who**，**How** を記録する[*5]．以下，それぞれの要素について詳しく述べる．

When

時間による変更は最も一般的である．コンテンツの更新や追加などによって異なる時間で異なるコンテンツをユーザは得ることになる．グローバルに一意な時間を示すために世界標準時などを用いるのが望ましいと考えられる．

Who

多くのウェブサービスではユーザごとにパーソナライズされたページを同じ URI で指示している（e.g. Twitter（ミニブログ）http://twitter.com/home，Mixi（SNS）http://mixi.jp/home.pl）．これは，同じ時間に接続しても人によって違うコンテンツが見えることを意味する．ユーザ登録が必要な情報の利用において，コンテンツのハッシュ値と組み合わせることによって利用者と利用したコンテンツの証明をしたり，**How** と組み合わせることによって自分の作ったコンテンツの権利主張をしたりすることが考えられる．

形式としてはユーザ名ドメインで一意にすることができるほか，OpenID[*6]のようなクロスドメインな認証手段をサイト側が提供することで様々な URI におけるユーザの活動をまとめて扱うことができ，名寄せの作業なしに人と人，人と情報の関係をウェブ全体から抽出することが可能になる．OpenID はサービス内におけるユーザのアカウントに対応した URL をユーザの ID とし他のサービスでも使えるように認証を代行する仕組みであり，ソーシャルメディアが増えてきている現在においては，「人」をウェブの中に位置づけることは重要であり，複数のサービスを透過的に利用する手段として普及してきている [7]．また，**Who** はログイン操作やその後のリクエストで受け渡されるセッション情報などの形で **How** の一部とみなすこともできるが，こういった背景から分けて扱っている．

How

URI の書式としてクエリの書き方が rfc3986[3] に定義されており，
http://example.com:8042/over/there?name=ferret#nose
においては”?”に続く name=ferret の部分がクエリとなる．HTTP URIs において，こういったクエリによってクライアントはサーバーに要求を伝え，要求したリソースを表示するが，この URI のクエリおよび HTTP ヘッダに付加された要

∗4 http://tools.ietf.org/html/rfc1321

∗5 URI とコンテンツの対応関係を変化させる要因としては他に，サーバー側でランダムに背景画像などのコンテンツを変化させるような実装を行っていた場合なども考えられるが，今回はいかなるコンテクストを用意しても完全には再現不可能な場合は考えずに，上記に述べたような URI のコンテクストについてのみ利用することを考える．

∗6 http://openid.net/

求のみで一意にサーバーのリソースを指せるアプリケーションを，サーバー側がクライアントのセッション情報を保持しないという意味でサーバーとってステートレスであると表現する．こういったアプリケーションにおいては，URI 以外の情報を記録しておいて呼び出すことによって再び同じリソースを指すことが可能になるので，URI とセットで保存しておきたいというのが **How** の保存に相当する．また，サーバにとってステートレスでないアプリケーションでも，サーバに状態を与えリソースを呼び出すフローを指示するようなインタフェースが用意されているならば，保存された **How** の情報を利用して一意にリソースにアクセスすることができるため，実質的にサーバにとってステートレスなアプリケーションとみなすことができる．また，このステートレス性には可視性・信頼性・スケーラビリティが確保されるというアプリケーション側のメリットもある [8]．可視性は，リソースを特定する情報が全て一回のリクエストの中に含まれていることによって確保される．信頼性は，途中でリソースの要求に失敗しても，操作をどこからやり直すかなどの問題を抱えずに要求しなおすことができることによって確保される．スケーラビリティは，サーバが状態を保持する必要がないのでクライアントの数が増えても保持する状態の管理の問題がおこらないために確保される．

書式の統一が難しく扱いにくいという問題があるが，アプリケーション側が自身が投稿した書式を認識して一意のリソースを再現できるならば問題がないと考える．また，ユーザの認証を必要とするようなサイトにおいて，ユーザの認証情報を省くことが必要であり，その点は注意が必要である．

4. URI Context Database の有効性

本章では，提案が２章で提示した問題に有効な解決策となっているかを確認し，URI Context Database の対象となるリソースの範囲について述べることで有効性を確認する．また，最後に，課題点についても述べる．

4.1 ２章における問題の解決

２章で提示した問題の解決シナリオを順に示す．

1. A，B，C，D，Eの５人がリソースXに対してA，B，Cは「信頼できない」とD，Eは「信頼できる」と評価を下していた場合に，URI Context Database を用いたアプリケーションは，実はA，B，Cが評価した後にXの情報は訂正され，訂正された情報に対してD，Eが評価を下していたということを **When** とコンテンツのハッシュ値の対応から判断し，Pがアクセスしたコンテンツは D，Eと同じであることを確認することで，Pに「リソースXは十分信頼できるのではないか」と示した．

2. A，B，C，D，Eの５人がリソースYに対してA，B，Cは「有用である」とD，Eは「有用でない」と評価を下していた場合に，URI Context Database を用いたアプリケーションは，実はA，B，CとD，Eでは見ていたコンテンツが異なることをコンテンツのハッシュ値から判断し，また，過去のコンテンツ閲覧履歴の **Who** とコンテンツのハッシュ値の共起情報から，PはD，Eと同じクラスターに属することが分かったので，リソースYは推薦せずにPの属するクラスターで「有用である」と評価されているリソースWを推薦した．

3. Pはメタデータを利用したある検索エンジンでリソース Z_0 に辿り着いたが，求めていた情報はなかった．しかし，URI Context Database を用いた検索エンジンではリソースのメタデータと共にリソース $Z_1 \rightarrow Z_0$ と辿ることで目的の情報が得られることが **How** の情報として提供されており，その部分のリクエストをユーザに代わってアプリケーションに行うことで，適切な状態でリソース Z_0 を表示し，ユーザの求めていた情報を提示した．

以上のように，URI Context Database を利用することで，２章で提示した問題を解決することが可能になる．

4.2 URI Context Database の対象

ハッシュ関数がバイナリを含むデータ列に適用可能であることから，HTTP URIs に限らず幅広い URI での利用が期待できる．但し，URI Context Database が対象としない URI も存在する．例えば，１章で断ったように，概念だけを指し説明のためのコンテンツを持たないような URI には無意味である．また，ある時点でコンテンツを有限に規定できない URI も対象にすることはできない．例えば，リアルタイムで映像などを流すライブストリーミングにおける rtsp プロトコル[*7]による URI の場合は，ハッシュ関数を適用するリソースのコンテンツ全体を取り出すことができないので対象にすることができない．このようなものを除くと，多くの URI に適用することができ，データサイズの面からも，SHA256 で $2^{64} - 1$bits，SHA512 で $2^{128} - 1$bits と十分に大きいサイズのリソースまで適用が可能である [5]．

4.3 URI Context Database の課題

課題としては，本質的でない部分がコンテクストによって常に変わるため同定が意味を成さない場合がある．例えば，ログイン時にログインしているユーザー名を明示するために「ようこそ，○○さん」といった表示が多くのウェブサービスサイトで見られるが，この部分が人によって違うため，他のコンテンツが同じでも違うコンテンツだと判別されてしまう．他には，ページを表示した時間を表示するサイトや，他サイトの RSS を表示しているサイトも同様である．このような場合においては，URI Context Database に登録する URI を本質的なコンテンツの部分に限るために適切にフラグメント識別子で領域を示すなどの工夫をサイト側に求めるなどの解決策が考えられるが，どの部分を本質的とするかについては，利用法によるので曖昧な場合もあり，サイト側の裁量に任される．

5. むすび

本論文では，URI とコンテンツの対応情報がないことによる問題を提示し，URI に「いつ」「だれが」「どのように」アクセスしたかというコンテキストを付随させ，URI とコンテンツをハッシュ関数によって一対一対応に結びつけて記録し，その情報をデータベースとして提供する URI Context Database を提案した．

また，URI Context Database をメタデータと共に用いることで，提示した問題を有効に解決できることをシナリオで示すことができた．

参考文献

[1] 安田浩, 阪本秀樹: メタデータの役割と世界標準化の動向 (＜小特集＞情報検索のためのメタデータ標準化), 社団法人映像情報メディア学会, Vol.55, No.3, pp.328-331, 2001.

∗7 http://tools.ietf.org/html/rfc2326

[2] Resource Description Framework (RDF): Concepts and Abstract Syntax W3C Recommendation 10, http://www.w3.org/TR/rdf-concepts/. 2004.

[3] Uniform Resource Identifier (URI): Generic Syntax, Request for Comments:3986, http://tools.ietf.org/html/rfc3986, 2005.

[4] Berners-Lee, T.:Design Issues: Linked Data. http://www.w3.org/DesignIssues/LinkedData.html, 2006.

[5] National Institute of Standards and Technology(NIST): Announcing the secure hash standard. Federal Information Processing Satndards Publication 180-2, 2002.

[6] Xiaoyun Wang and Dengguo Feng and Xuejia Lai and Hongbo Yu:Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD, Cryptology ePrint Archive, 2004.

[7] 大向一輝: SNS の現在と展望 -コミュニケーションツールから情報流通の基盤へ-, 情報処理, Vol.47, No.9, pp.993-1000, 2006.

[8] Fielding, R. T.: Architectural styles and the design of network-based software architectures, PhD Thesis, University of California, 2000.